UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MICROELETRÔNICA

FABIO BENEVENUTI

Enhancements on Fault Injection for Xilinx 7 Series and Ultrascale+ SRAM-Based FPGAs

Thesis presented in partial fulfillment of the requirements for the degree of Doctor of Microelectronics in the field of test of electronic circuits and systems.

Advisor: Dr. Fernanda Gusmão de Lima Kastensmidt

Porto Alegre
2022

CIP — Cataloging-in-Publication

Benevenuti, Fabio

Enhancements on Fault Injection for Xilinx 7 Series and Ultrascale+ SRAM-Based FPGAs / Fabio Benevenuti. – 2022.

234 f.: il.

Advisor: Fernanda Gusmão de Lima Kastensmidt.

Thesis (Doctoral) – Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Microeletrônica, Porto Alegre, Brazil, 2022.

1. SRAM-based FPGA. 2. Radiation. 3. Single-event effects. 4. Reliability. 5. Fault injection. I. Kastensmidt, Fernanda Gusmão de Lima (advisor). II. Title.

ACKNOWLEDGMENT

The research for this thesis was supported, in part, by scholarship granted by the Brazilian funding agency Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

The research for this thesis relied on sample preparation and decapsulation at the Núcleo de Qualificação e Análise de Produtos Eletrônicos (NAPE) from Centro de Tecnologia da Informação Renato Archer (CTI), and at the Analytical Laboratory from the Centro Nacional de Tecnologia Eletrônica Avançada S. A. (CEITEC), both entities under the Brazilian Ministry of Science, Technology and Innovations (MCTI).

This thesis used data obtained from experiments performed at the Pelletron 8 MV heavy ions accelerator from the Laboratório Aberto de Física Nuclear (LAFN) and at the IEA-R1 research nuclear reactor from the Instituto de Pesquisas Energéticas e Nucleares (IPEN), both facilities on the premises of the University of São Paulo (USP), Brazil, at the fast neutrons generator facility from the Laboratório de Radiação Ionizante (LRI) of the Instituto de Estudos Avançados (IEAv), a research unit under the Brazilian Air Force (FAB), at the Thermal and Epi-thermal Neutron Irradiation Station (TENIS) from Institut Laue-Langevin (ILL), France, and at the single- and two-photon absorption laser fault injection facilities at the Institut D'Électronique et des Systèmes at Université de Montpellier, France.

I am very grateful to the support of Fernanda Gusmão de Lima Kastensmidt, which went beyond the obligations as my advisor, to the support and encouragement of Reinaldo de Bernardi and Cristiano Krug, my immediate supervisors while working at CEITEC, to Vitor Ângelo Paulino de Aguiar, for sharing the dataset from static tests experiments on FPGA board under heavy ions, and to Augusto Gouvêa Weber, for the support during the coordinated effort in the development and integration of the INPE-SMDH/UFSM-UFRGS payload board for the NanosatC-BR2 mission.

Finally, I am very grateful to colleagues in Laboratory 230 who allowed the use of their designs as test specimen for fault injection, mainly André Flores dos Santos and Juan Suzano da Fonseca with benchmark designs based on high-level synthesis (HLS), Ádria Barros de Oliveira and Luis Alberto Contreras Benites with softcore microprocessors, Marcio Macedo Gonçalves with softcore graphics processing unit (GPU), and Fabiano Libano and Israel da Costa Lopes with neural network experimental designs.

ABSTRACT

Commercial grade SRAM-based FPGAs are susceptible to radiation effects that can affect safety- and mission-critical cyber-physical systems. Emulated fault injection is a test strategy based on provoking failures in a controlled manner, which can be applied in assessing the reliability and fault tolerance of such systems. The use of fault injection inside the engineering process of complex systems, such as systems on chip and applications of neural networks, with a higher problem dimensionality and combining several fault mitigation techniques, requires a higher consistency between the results of fault injection and the behavior under radiation to steer the engineering process in right direction. Fault injection must also be fast enough to not hinder the engineering productivity. The main claims of this thesis are that a higher consistency is achieved, when related to accelerated radiation experiments, by the use of synchronous and asynchronous cumulative-randomized fault injection emulating single and multiple bits upsets and that the sampling by cumulative-randomized fault injection improves the fault injection campaign productivity to be used inside design space exploration engineering processes. The originality and main contributions of this thesis resides on injecting faults clustered together on the Xilinx 7 Series FPGA memory emulating patterns generated by a single radiation particle, injecting multiple faults cumulatively over time to allow better comparison with data from radiation experiments, injecting faults in the presence of Xilinx 7 Series memory scrubber, injecting faults on user memory holding persistent data, and, finally, porting the fault injection to the newer family of Xilinx UltraScale+ devices. The consistency with radiation experiments and fault injection campaign speedup are evaluated in comparison with the legacy methodology that consisted on exhaustive and synchronous fault injection emulating single bit upsets only in the persistent configuration memory of the FPGA. Dynamic tests of study-case designs are used for comparison between fault injection and radiation. The methodology adopted on this thesis consists in bibliographic review, development of the study-case designs for comparative experiments, laboratory experiments for characterization of the target devices and the study-case designs, enhancement of the fault injection tool and development of fault injection campaign scripts realizing the proposed methodologies.

Keywords: SRAM-based FPGA. Radiation. Single-event effects. Reliability. Fault injection.

Aperfeiçoamentos em Injeção de Falhas para FPGAs SRAM Xilinx 7 Series e UltraScale+

RESUMO

Os arranjos de portas lógicas configuráveis em campo (FPGAs) de classe comercial baseados em SRAM são suscetíveis aos afeitos de radiação que podem afetar sistemas de segurança crítica e de missão crítica. A injeção de falhas baseada em emulação é uma estratégia de testes que consiste em provocar falhas de forma controlada, a qual pode ser aplicada na avaliação de confiabilidade e tolerância a falhas destes sistemas críticos. O uso de injeção de falhas dentro do processo de engenharia de sistemas complexos, com uma dimensionalidade de problema maior e combinando diversas técnicas de mitigação de falhas, requer maior fidelidade entre a injeção de falhas e o comportamento sob radiação para que o processo seja conduzido na direção correta. A injeção de falhas também precisa ser rápida para manter a produtividade do processo de engenharia. As principais hipóteses de trabalho nesta tese são de que uma consistência maior é obtida, quando comparado com experimentos de radiação, pelo uso de injeção de falhas acumulada-aleatória sincronamente e assincronamente, e de que a amostragem pela injeção acumulada-aleatória acelera e melhora a produtividade da campanha de injeção de falhas a ser utilizada na exploração do espaço de projeto. A originalidade e principais contribuições desta tese estão na injeção de falhas agrupadas para emular o efeito de radiação, na injeção de falhas acumuladas, na injeção de falhas na presença do mecanismo de deteção e correção de erros de memória, na injeção de falhas em memória do usuário, e no porte da injeção de falhas para a família de FPGAs Xilinx UltraScale+. A consistência com radiação e a aceleração da injeção de falhas são avaliadas em comparação com a metodologia de injeção síncrona e exaustiva emulando falhas simples na memória de configuração do FPGA. Testes dinâmicos de estudos de caso são utilizados para a comparação entre injeção de falhas e radiação. A metodologia adotada nesta tese consiste em revisão bibliográfica, desenvolvimento das aplicações de estudo de caso para experimentos comparativos, realização de experimentos em laboratório para a caracterização dos dispositivos alvo e aplicações de estudo de caso, aperfeiçoamentos na ferramenta de injeção de falhas e desenvolvimento de scripts de automação da campanha que realizam as metodologias propostas.

Palavras-chave: FPGAs SRAM, Radiação, Efeitos de eventos singulares, Confiabilidade, Injeção de falhas.

LIST OF ABBREVIATIONS AND ACRONYMS

BRAM Block RAM; memory on FPGA dedicated to storage of user data.

CRAM Configuration RAM; memory layer on FPGA dedicated to storage of configuration for logic resources, signal fabric and, occasionally, a small amount of user data.

DUT Design under test; refers to the entire FPGA design or selected modules implemented in a designated region in the FPGA floorplan; not to be confused with the DUT abbreviation present in the literature when referring to *device* under test.

FPGA Field programmable logic device, historically were based on gate arrays while modern devices have logical functions defined by look-up tables.

ICAP Internal Configuration Access Port, proprietary communication interface available on Xilinx FPGA devices.

IES Institut d'Electronique et des Systèmes of the Université de Montpellier.

IEAv Instituto de Estudos Avançados (Institute for Advanced Studies) under the Brazilian Air Force.

LAFN Laboratório Aberto de Física Nuclear (Nuclear Physics Open Laboratory) at University of São Paulo (USP).

LET Linear energy transfer. Measure of energy deposited by distance unit along the particle trajectory through material.

LUT Look-up table; FPGA configuration memory storing logic equation or truth table.

MBU Multiple bits upset, a SEU affecting multiple memory bits, may refer to bits in a same logical word.

MCU Multiple cells upset, a SEU affecting multiple memory bits, may refer to bits in different logical words or be irrespective to the logical word organization.

MPSoC Multiprocessor system-on-chip; system-on-chip featuring multiple homogeneous or heterogeneous microprocessors, may include also programmable logic.

RAR Reset after reconfiguration; attribute of a physical block in placement constraint file that causes the configuration of the global reset mask in Xilinx 7 Series FPGAs.

SAFIIRA Sistema de Feixes Iônicos para IRradiações e Aplicações (ion beam system for irradiation and applications), test facilty at the LAFN 0° beamline.

SBU Single bit upset, a SEU affecting a single memory bit.

SCU Single cell upset, equivalent to SBU.

SET Single event transient, a short pulsed transient caused by a single particle strike.

SEU Single event upset, a memory upset caused by a single particle strike.

SoC System-on-chip, refer to combinations of microprocessors, memory resources and peripherals on a single chip, typically interconnected by an open or general purpose communication subsystem such as a NoC or AMBA.

SRAM Static random-access memory.

TID Total ionizing dose; associated with effects of long-term exposition to radiation.

UFRGS Universidade Federal do Rio Grande do Sul (Federal University of Rio Grande do Sul).

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

1.1 Applications of SRAM-based FPGA in critical systems

Recent advances on computing and electronics made feasible new embedded real-time and computing-intensive applications in safety- or mission-critical fields like transportation, autonomous vehicles, driver assistance systems, industrial automation, medical diagnosis, implantable medical devices, biology research, earth observation, pervasive surveillance, military and defense, among others. Some of those applications include digital signal processing, sensor fusion, image processing, computer vision, pattern recognition and machine learning, which require high productivity, efficiency and adaptability of computing systems.

When we take into account these computing-intensive applications, the modern programmable logic devices may be a competitive alternative to application specific integrated circuits (ASIC) and general-purpose microprocessors in terms of design and production costs, power consumption, time to market, maintainability and capability of improvement to cope with changing environment or evolving requirements.

Commercial grade SRAM-based FPGAs are notably attractive for such applications due to its low cost given the technology in use and the large-scale production, its general availability, high density of resources and combination with multicore microprocessors allowing load distribution between hardware and software.

However, due to the huge amount of SRAM cells and its inherent susceptibility to radiation effects, the use commercial grade SRAM-based FPGAs in critical cyber-physical systems in radiation rich environments, or even on ground level, may raise some concerns in terms of reliability and fault tolerance. Although radiation hardened devices are also available on the market, they may have a higher cost or limited availability due to export restrictions given its potential use in military applications. These limitations of the radiation hardened devices may help in choosing a commercial grade device.

Consequently, the use of such devices in critical cyber-physical systems requires the characterization of the reliability of the devices and applications, and, occasionally, the use of proper mitigation techniques for fault tolerance coping with reliability requirements on the target environment.

Ideally, those critical systems and application could be tested and qualified in its operational environment subject to the effects of radiation. The reliability evaluation in operational conditional, however, have several disadvantages, including higher cost and longer test time. Another disadvantage is that it would be performed only in the later stages of development, increasing the correction costs.

As alternative, the fault injection, in laboratory environment, allows the evaluation since the earlier stages of proof of concept until later stages near the real environment and operational conditions.

The fault injection, in itself, is also a complex task, because it requires deep understanding of the target environment, the target device, and its behavior in that environment.

Notwithstanding, fault injection also enables new engineering approaches, for instance the reliability aware design space exploration for complex applications where reliability and fault tolerance compete with other engineering constraints and requirements such as area, power consumption and processing time. To cope with this higher dimensionality of the problem, design space exploration can use methods of heuristic search which, in turn, are fed with reliability metrics that can be fused in the multi-objective search for a satisfying solution.

Although accelerated irradiation and circuit simulation can provide such reliability metrics, its use inside design space exploration may not be feasible due to large cost of radiation facilities and large processing time of circuit simulation for complex systems. As alternative, the use of fault injection can emulate the radiation effects while the application runs on hardware at full speed, orders of magnitude faster than circuit simulation and cheaper than accelerated irradiation.

The challenges, however, are that adequate fault injection methodology must be devised to achieve the desired speed to be used inside the design space exploration process and that a higher level of consistency between fault injection and radiation effects must the achieved to avoid that the search process converges towards solutions that do not present the desired fault tolerance and reliability properties under radiation.

To tackle these challenges, this thesis implements improvements on an existing emulated fault injection tool based on Xilinx Internal Configuration Access Port (ICAP) and the associated fault injection methodologies, supported by study-case applications using high-level synthesis and softcore microprocessors for mathematical processing and flagship machine learning applications.

1.2 Motivation and innovative contributions

The emulated fault injection engine adopted in this thesis is built around the Xilinx ICAP interface and have a long line of development. Nazar et al. (2012) used this mechanism for fault injection of addressed bits on configuration memory of a Xilinx Virtex-5 device, being later adapted by Leipnitz et al. (2016) for other applications. Tarrillo et al. (2015) also used this mechanism on a Xilinx Virtex-5 device, injecting fault on randomized memory addresses of the configuration memory. Tonfat et al. (2016) used the mechanism on a Xilinx Artix-7 device to inject faults exhaustively on the whole configuration memory of the device.

While the fault injection tool evolved across the families of Xilinx devices, including Xilinx 7 Series FPGAs and Xilinx Zynq-7000 MPSoCs, the test methodology targeted only injection of single faults on the configuration memory, either on randomized locations or exhaustively and sequentially on contiguous locations. In either case, the test methodology provided an estimate on the number of memory bits that, alone, could cause a functional failure on the application implemented on the FPGA, also known as the critical bits.

The number of critical bits is relevant as a reliability metric, but it do not represent properly the cumulative behavior of the memory upsets on the FPGA configuration memory as we see under radiation environments. Another limitation of critical bits is that it does not consider temporal windows in which the information is relevant, for instance when healing by memory scrubbing is active on when faults affect memory storing application user data.

Also, although previous works injected faults on memory locations known as susceptible to memory upsets caused by radiation, the methodology does not cover the cases where a single radiation particle causes memory upsets on multiple memory cells.

The originality and the main contributions of this thesis resides on (1) injecting multiple faults clustered together on the FPGA memory, emulating arbitrary patterns such as those generated by a single radiation particle or by laser fault attacks, (2) injecting multiple faults cumulatively over time, emulating the effect of multiple radiation particles, (3) injecting faults synchronously and asynchronously enabling reliability evaluation in the presence of memory scrubber, (4) injecting faults on user memory holding persistent data, and, finally (5) porting the fault injection mechanism and the associated methodologies to the newer family of Xilinx UltraScale+ devices.

In this context, this investigation is motivated by questions such as:

1. what type of patterns of single and multiple bit upsets are dominant on the two main types of memory of the FPGA under different radiation particles?
2. what type of patterns and to which level of detail they need to be emulated to achieve adequate accuracy of the reliability metrics obtained from fault injection when compared to radiation experiments?
3. taking into account the interest in higher levels of reliability, how far the injected faults must be accumulated or, in other terms, where the sampling process can be interrupted to further increase the fault injection campaign productivity?
4. what are the fault injection campaign conditions, or the methodology simplifications, that allows for faster fault injection and, hence, higher productivity?

1.3 Thesis claim

The main claims of this thesis are that a higher consistency is achieved, when related to accelerated radiation experiments, by the use of synchronous and asynchronous cumulative-randomized fault injection emulating single and multiple bits upsets and that the sampling by cumulative-randomized fault injection improves the fault injection campaign productivity to be used in design space exploration processes. The consistency with radiation experiments and fault injection campaign speedup are evaluated in comparison with the legacy methodology that consisted on exhaustive and synchronous fault injection emulating single bit upsets only in the persistent configuration memory of the FPGA.

1.4 Thesis goals

The main objective of this thesis is to improve the fault injector for Xilinx SRAM-based FPGAs developed at UFRGS to enhance the consistency between emulated fault injection and radiation tests, accelerate the fault injection process to reduce campaign time, and extent the support of the fault injector to an additional family of Xilinx products.

The activities performed to achieve this objective include:

- Analyze the FPGA memory organization using laser to build a cartographic model in terms of memory rows, columns and direction of the columns;

- Investigate the multiple upset patterns occurring on configuration memory and user data memory under alpha particles, heavy ions, fast neutrons and thermal neutrons;
- Implement required changes on the fault injection engine to support asynchronous, cumulative fault injection and faults affecting multiple bits, both on configuration memory and user data;
- Port the fault injection engine to the newer UltraScale+ family of Xilinx SRAM-based FPGA devices;
- Develop reference procedures and scripting for fault injection campaigns implementing different test strategies;
- Demonstrate the use of the fault injection methodology with different study-case applications based on high-level synthesis and softcore microprocessors.

1.5 Description and organization of this report

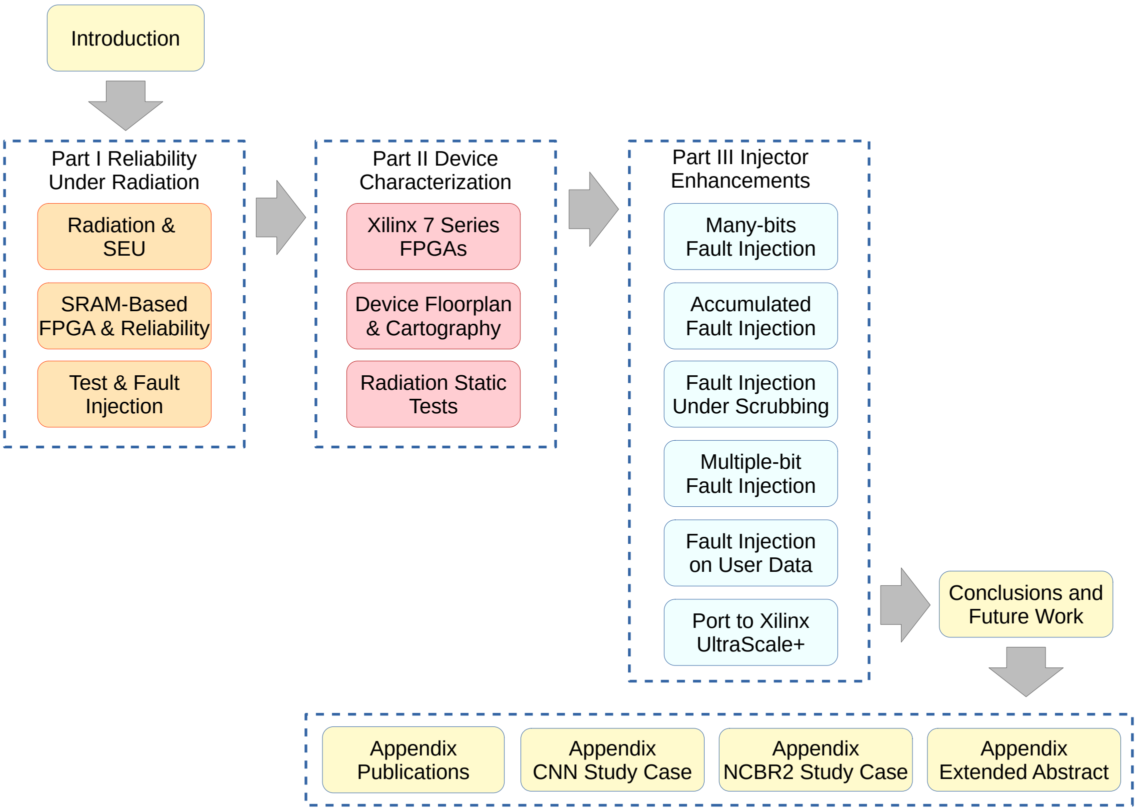
The main body of this thesis is organized in three parts with the main topics summarized in Figure 1.1.

The first part of this thesis addresses the theoretical aspects of radiation effects in electronic devices that are relevant for mission critical an safety critical applications, the characteristics of SRAM-based FPGAs, and the use of fault injection as a tool for reliability assessment. This part is based exclusively on bibliographic research in the fields of radiation effects, reliability and tests of electronics circuits and systems.

The second part of the thesis deepens in the description of the organization and addressing of the memory matrix in Xilinx FPGAs, including results from laser and radiation experiments performed to clarify the memory organization, the underlying static cross section, and the occurrence of multiple-bit memory upsets under radiation. This part is partly based on bibliographic research on Xilinx documentation for the products object of this thesis, but also includes results from floorplan mapping, laser experiments and radiation experiments for static tests required to understand and describe the behavior of the device under radiation.

The third part of the thesis addresses the improvements on the legacy fault injection engine adopted in this thesis as well as further improvements in the fault injection methodology. This part if mostly based on experimental results from fault injection and

Figure 1.1 – Report organization


Introduction
Part I Reliability Under Radiation
Radiation & SEU
SRAM-Based FPGA & Reliability
Test & Fault Injection
Part II Device Characterization
Xilinx 7 Series FPGAs
Device Floorplan & Cartography
Radiation Static Tests
Part III Injector Enhancements
Many-bits Fault Injection
Accumulated Fault Injection
Fault Injection Under Scrubbing
Multiple-bit Fault Injection
Fault Injection on User Data
Port to Xilinx UltraScale+
Conclusions and Future Work
Appendix Publications
Appendix CNN Study Case
Appendix NCBR2 Study Case
Appendix Extended Abstract

Source: The Author

irradiation of case-study and benchmark applications.

Finally, we present the main conclusions and proposals for future works and further development in this research theme.

An appendix provides a list of publications and knowledge transfer activities that incorporate the contributions of this thesis. Another appendix describes the study-case application of image classification based on machine learning and convolutional neural networks developed by a prototyped design space exploration framework based on generic algorithms. A third appendix describe the context of the study-case microbenchmark applications onboard the NanosatC-BR2 payload. An additional appendix presents an expanded abstract of this thesis.

Part I

SRAM-Based FPGAs and Reliability Under Radiation

2 RADIATION EFFECTS AND RADIATION-INDUCED MEMORY UPSETS

2.1 Semiconductor devices

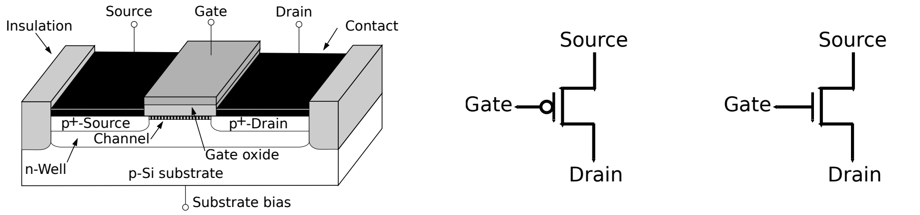
Semiconductor devices are fabricated and commercialized either as *discrete components*, such as single diodes or transistors, or combined with other elements composing an *integrated circuit*. In both cases, the technological manufacturing process consists of fabricating the devices over a substrate of semiconductor material, where silicon is the more commonly used material due to its electrical properties at room temperature, abundant availability in nature in the form of sand and quartz, among other advantages (KANO, 1998).

To be used in the fabrication of electronic devices, the silicon substrate must be in a very pure crystalline form that is obtained by refining quartz or sand from nature using chemical and metallurgical processes in high temperatures. In this process, the purified silicon is also carefully contaminated, or doped, with very specific impurities, such as atoms of boron (to obtain a P-type substrate) or phosphorous (to obtain a N-type substrate), in very specific concentrations, to leverage its semiconducting properties. The final result is typically a circular slice of monocrystalline semiconductor substrate with the silicon atoms in a very specific crystalline orientation.

One of the devices most commonly found in integrated circuits is the transistor or, more specifically in modern microelectronics, the field-effect transistor (FET). For historical reasons, this type of transistor is known as metal-oxide semiconductor (MOS) transistor (MOSFET). The electrical properties of transistors are decided by selecting the technological process of fabrication and scaling the transistor geometry, such as the channel width and length. A simplified view of a typical transistor is presented in Figure 2.1a. A transistor where the source and drain regions area heavily doped to be of P-type, as seen in Figure 2.1a, is named a PMOS transistor. Conversely, a transistor with source and drain regions of N-type is named as NMOS transistor. Figure 2.1b and Figure 2.1c present the schematic symbol for PMOS and NMOS transistors.

Combining several PMOS and NMOS transistors together we can build digital circuits such as logical functions and data storage elements. The storage elements are of particular interest to this thesis, notably the static random access memory (SRAM) found in commercially available off-the shelf (COTS) integrated circuits. A simplified view of a SRAM cell using six transistors (6T-SRAM) is depicted in Figure 2.2a comprising

Figure 2.1 – Schematic view of classical transistor



(a) P-channel transistor built on P-doped substrate

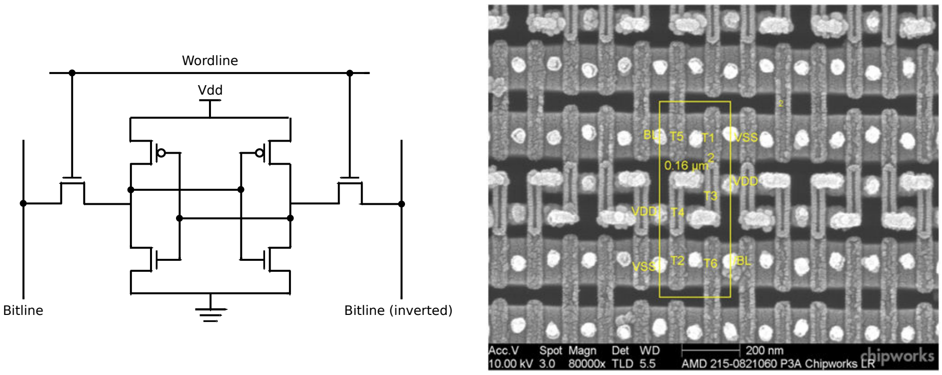
(b) Symbol for P-channel transistor

(c) Symbol for N-channel transistor

Source: Adapted from Sze et al. (2007)

two cross-coupled inverters in the center using four transistors where the information is latched. A scanning electron microscope image of such a memory cell measuring $0.16\,\mu\mathrm{m}^2$ is presented in Figure 2.2b.

Figure 2.2 – A 6T-SRAM cell



(a) Schematic view of memory cell

(b) Memory cell fabricated in TSMC 28 nm technology process

Source: (a) adapted from Weste et al. (2010); (b) extracted from Dixon-Warren (2012).

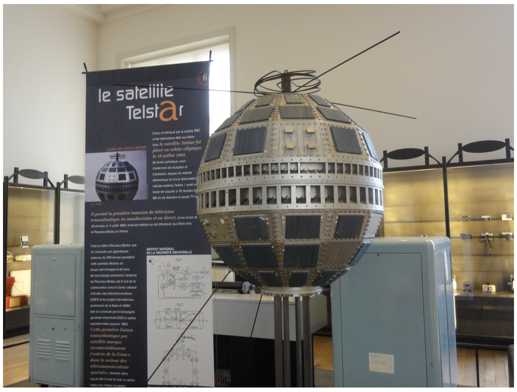
SRAM memory cells of this kind is found on several COTS products such as microprocessors and programmable logical devices (FPGA), holding critical information inside the processor registers, tightly coupled memory supporting high performance logic and configuration memory of logical elements.

2.2 Radiation effects on electronic devices

Since the invention of transistors in the 40's and the growth of semiconductor industry, the increasing use of semiconductor devices in computing and data storage systems intensified the research on the reliability of such devices. Starting in the 60's, several studies on how ionizing radiation could affect those devices were conducted influencing new applications of semiconductor devices, such as in aerospace technology and computer systems in environments with continuous exposition to ionizing radiation like nuclear reactors and high energy particle accelerators.

An iconic event of perturbations on electronic devices attributed to radiation is the degradation and loss of the Telstar communication satellite (model in Fig. 2.3) in 1963, only a few months after launched, in the aftermath of high-altitude nuclear tests (ECOFFET, 2007). Nevertheless, ionizing radiation is also an issue in mission critical and safety critical electronics operating at ground level (MAY et al., 1979; NORMAND, 1996; OLIVEIRA et al., 2020).

Figure 2.3 – Model of a Telstar satellite on display at the CNAM, Paris



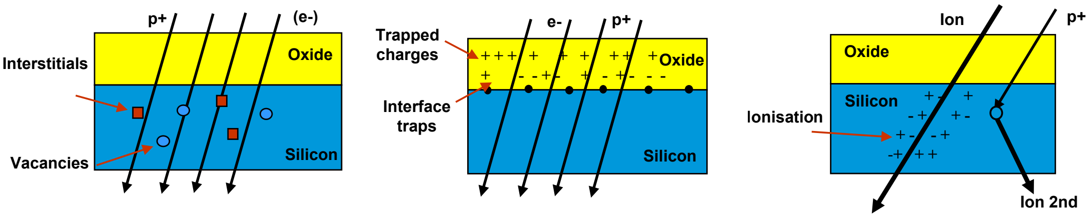
Source: The Author

Pacini (2017) notes that the pioneers on research of atmospheric ionization in the beginning of the 20th century did not recognize the corpuscular nature of cosmic sources and associated this phenomenon initially with electromagnetic rays (radiation), hence the term radiation is still used nowadays for both electromagnetic radiation and particles. In this thesis we consider the effect of rays, such as photons, but we are mostly concerned with the effects of particles, such as protons, alpha particles, other heavier ions and neutrons.

Radiation effects on electronics can be divided into long-term parametric degradation or transient changes on the device (SCHRIMPF, 2007). In the first class we have the displacement damage (DD) and effects associated with the total ionizing dose (TID). Among the transient effects we are interested in the effects caused by a single particle (SEE). These effects will be described briefly on the following while Bräunig et al. (1994, Table 4) summarize the sensitivity of different types of devices to the effects of radiation.

The displacement damages (DD) refer to the displacement of atoms in the crystalline lattice due to atomic collisions, characterized by a non-ionizing energy loss (NIEL), caused by particles such as protons, heavy ions, neutrons, nuclear recoils and high energy electrons. Displacement damages create defects in the crystalline lattice of the material, which may be vacancies in the lattice or displacement of atoms to interstitial positions (Figure 2.4a), among others, and acts as charge accumulation regions due to the trap effect. The displacement damages electrically active defects affects both insulating material and semiconductor material (SCHRIMPF, 2007), impacting several types of devices, such as bipolar junction transistors and image sensors, but the MOS transistor, specifically, are less affected by displacement damages.

Figure 2.4 – Radiation effects on semiconductor devices



(a) Displacement damage (DD) **(b)** Total ionizing dose (TID) **(c)** Single event effect (SEE)

Source: Extracted from Poivey (2017)

The total ionizing dose (TID) refers to the cumulative effects of deposited dose of different types of photons and particles, including protons, heavy ions, electrons and neutrons, affecting primarily insulating materials. The silicon dioxide used in semiconductors is usually amorphous and creates an interface region between the oxide and bulk silicon, as present in transistor gate oxide (Figure 2.1a), where there is a perturbation of the periodic potential of the lattice. Besides, oxygen atoms also perturb the periodic potential of the lattice creating traps. These traps may become centers of charge accumulation (Figure 2.4b) under the presence of electric field due to the different mobility of electrons and holes in the oxide. This effect is enhanced by the incidence of radiation during the device operation when the electrical field is present. The effect of trapped charges includes the degradation of electrical parameters of the devices, including increased leakage current and power consumption, reduced breakdown voltage, increased noise, changes in the characteristic curves and threshold voltage of transistors.

Single-event effects (SEE) refer to the immediate effects of a single ionizing particle strike on a electronic device. These effects are observed when a single ionizing particle is capable of generating an amount of electron-hole pairs enough to cause perturbations in the operation of the device (Figure 2.4c). Single-event effects can be observed by incidence of heavy ions, alpha particles, muons, pions, as well as byproducts of incidence of neutrons and protons. Scale reduction of devices also decreased the amount of charge required to trigger a single-event effect enabling the observation of such effects by direct ionization by electrons (TRIPPE et al., 2015) and protons (RODBELL et al., 2007; HEIDEL et al., 2008; SIERAWSKI et al., 2009). The amount of charge generated by the incidence of radiation can lead to different types of SEE, some of them being non-destructive creating momentary consequences (the soft errors) and others destructive making the device permanently defective (the hard errors). Examples of soft errors include the single-event transients (SET), single-event upsets (SEU) affecting storage elements, single-event functional interrupt (SEFI). As hard errors we can find single-event burn-out (SEB), single-event dielectric rupture (SEDR), among others.

A single-event upset (SEU) occurs when charges generated by the particle incidence are collected and interfere with some of the transistors in the storage element, seem in the example of Figure 2.2. Those transient charges can disturb the balance inside the storage element, causing the stored value to change, that is, a logical zero (0) value is changed to a logical one (1) or a logical one is changed to a logical zero, what is also named as a *bit-flip*. Occasionally, it can also be interpreted as a SEU when a single-event transient (SET) in the input of a the storage element occurs concomitantly with the clock edge and a false value is registered changing the stored value at the storage element (DODD et al., 2004). In this case, increase on clock frequency also increases the likelihood of capturing SETs, therefore increasing the sensitivity to transient effects. A similar situation occurs when a transient propagates through clock signal causing an invalid value to be captured by the storage element.

2.3 Sources and characteristics of space radiation environment

Radiation as a natural phenomenon has its sources in cosmic rays, solar flares, radiation belts and materials on Earth. It comes also from man-made sources such as nuclear reactors and particle accelerators for experimental physics and medical purposes.

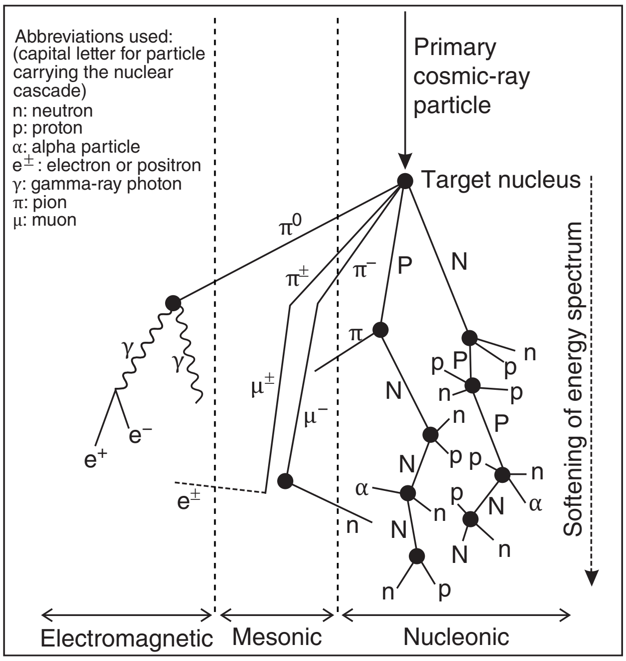
The main sources of radiation affecting electronic devices include neutrons, energetic particles such as electrons, protons, alpha and heavy ions, and electromagnetic radiation such as x-rays and gamma rays (STASSINOPOULOS et al., 1988; CLAEYS et al., 2002).

At space environment near Earth and low atmosphere, affecting airborne electronics on orbital stations, satellites and other spacecrafts, there are energetics particles such as protons and heavy ions from cosmic rays and solar flares, protons and electrons trapped in Van Allen belts and heavy ions trapped in the magnetosphere (CLAEYS et al., 2002;

BOUDENOT, 2007).

Additionally, the terrestrial atmosphere acts as a filter protecting the earth from radiation (BOUDENOT, 2007) with the exposition to radiation increasing with the altitude and, for lower altitudes, increasing in the proximity of polar regions. Cosmic rays entering the atmosphere will interact with other particles triggering a process of spallation and consequent chain reactions that produces more particles at higher altitudes The particle flux then decreases with altitude due to energy loss, absorption and decay, with thermal and high energy neutrons representing the more relevant contribution to single-event effects in electronic devices at sea level (O'GORMAN, 1994; NORMAND, 1996). This phenomenon is named air shower adopting a term coined by Blackett et al. (1933) after the works of Bruno Rossi in the 30's (KAMPERT et al., 2012). Figure 2.5 shows an schematic representation of such air shower with a possible development of secondary cosmic rays in the atmosphere, triggered by an incident high-energy cosmic particle.

Figure 2.5 – Air shower: secondary cosmic rays in the atmosphere



Source: Dunai (2010)

Overall, the effect of cosmic ray heavy ions and trapped protons on radiation belts are the dominant sources of single-event effects in altitudes higher than 18.000 m while neutron interaction is the dominant source for lower altitudes and ground level (TSAO et al., 1984; STASSINOPOULOS et al., 1988; HANDS et al., 2017). Consequently, heavy ions and protons are the main concern for electronics onboard satellites and other aircrafts on orbit while neutrons are the main concern for civil aviation, ground level high-performance computing and safety critical electronics such as autonomous vehicles.

2.4 Interactions of radiation with matter

Ionizing radiation, such as photons, protons, heavy ions, fast neutrons and thermal neutrons, transfer energy to the matter but present different interaction mechanisms.

Photons with sufficient energy can interact with the matter by three main processes. At the photoelectric effect all the photon energy is transferred to an atomic electron. At the Compton effect, only part of the energy is transferred to an electron and a photon is scattered in a given angle. Finally, at the production of pairs, photons are transformed in electron-positron pairs. Cross section for the occurrence of each of these effects depends on the energy of the photons and the nature of the matter.

Energetic particles, such as proton, alpha particles and heavy ions interacts more easily with matter because they are charged. Charged particles loses energy across the matter interacting with each atom and transferring small amounts of energy. This is a stochastic process where the mean energy transfer can be known and is named stopping power. In the case of light particles such as electrons and positrons, the energy loss occurs also by irradiation (*bremsstrahlung*). The gradual loss of energy across the matter defines a maximum range of the particle in the matter.

Neutrons, having no electrical charge, do not interact with electrons and ionizes indirectly by producing a recoil that will cause ionization or by a nuclear reaction where generated charged particles will cause ionization.

2.5 Single-event effects and memory upsets

The charge deposited by a single ionizing particle can produce a wide range of single-event effects, that can, in turn, be divided into hard errors, which induce destructive effects, and soft errors, which are non-destructive functional errors.

As hard errors we can find single-event burn-out (SEB), single-event dielectric rupture (SEDR), among others. Examples of soft errors include the single-event transients (SET), single-event upsets (SEU) affecting storage elements, single-event functional interrupt (SEFI) (IBE, 2015).

Due to the object of this thesis, which is the fault injection on FPGA memory, and the nature of the target device, which is a commercial grade COTS SRAM-based FPGA, we will be mostly concerned with the single-event upsets (SEU).

In a simplified form, a memory element in a digital circuit, that is, one data bit, is a storage cell whose value will be updated from its input upon a clock event and when that update is enabled. The storage cell current value can then be read at its output.

There are two main mechanisms by which the radiation effect can cause a memory upset changing stored value of the data bit from zero to one or from one to zero, that is a bit-flip. One mechanism is when a transient is occurring at the input signal of the storage cell at same time as the clock event and enable signal allowing the memory value to be updated with a potentially erroneous value. As the transient is a short pulse, the coincidence of the transient and the clock event may be seen as a rare event although increasing the operating frequency of digital circuits also increases the likelihood of such events.

Another mechanism is when the transient occurs by incidence of radiation inside the storage cell itself, disrupting its balance and allowing its value to change spontaneously. This type of mechanism is more likely to be seen in the case of unhardened commercial grade COTS devices.

In any case, minimal amount of charges must be generated to reach a critical current where the transient is observed as a logical value or the unbalanced storage cell inverts its value.

Alpha particles, heavy ions and neutrons will be used in this thesis for study of the static cross section of the target device and as benchmark for reliability comparison with fault injection.

We are also interested in the use of photons from laser to investigate the organization of memory cells on the target device (cartography). However, since photon radiation causes ionization by absorption and generation of electron-positron pairs, a single photon is not enough to reach the critical current for a memory upset. Hence, in the case on photon, it must be clarified that it is conceptually incorrect to name the process as a single-event since a burst of photons is required to cause the memory upset.

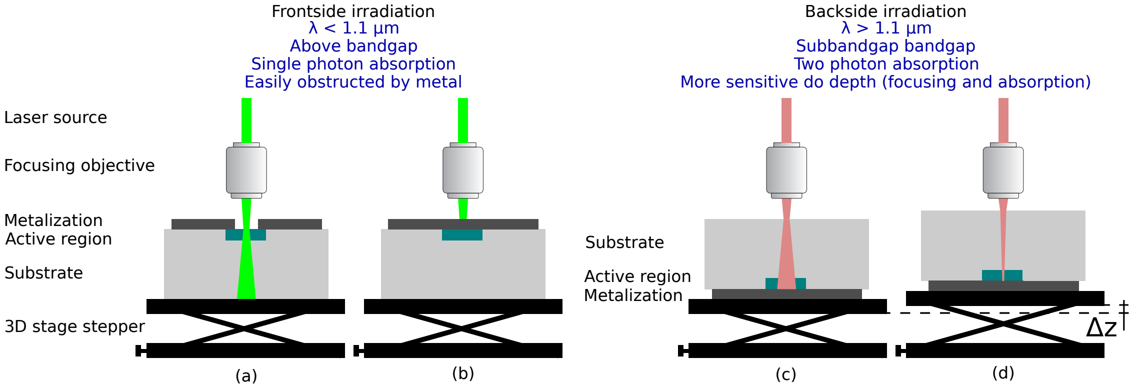
2.6 Laser induced memory upsets

Ionization and transients in semiconductor devices can be generated by irradiation with laser in an adequate wavelength (HABING, 1965). Besides the use of laser, other sources of photons can also be used to cause memory upsets. Skorobogatov et al. (2003), for instance, reports the occurrence of memory upsets in SRAM memory caused by a photographic flash attached to focusing lens. Most wavelengths are not suitable for simulating radiation-induced transients because of the absorption characteristics of the matter. Infrared laser, for instance, have low absorption coefficient in silicon but photon energy below the bandgap. Near-infrared laser may represent a compromise between absorption coefficient and photon energy above the bandgap. With green laser in silicon, for instance, we will have photon energy well above bandgap, but most energy will be converted in heat, presenting a small penetration depth that may not reach the memory cell active region. With infrared laser we may have a higher penetration range in silicon due to absorption coefficient but smaller photon energy below bandgap may not generate enough electron excitation.

There are two technical processes for simulating radiation induced transient effects in semiconductor devices using laser.

The first one is the single photon absorption (SPA) where the wavelength is chosen so that the photon energy is above bandgap. The small penetration range may suggest frontside irradiation (Figure 2.6a) but, in the case of complex devices such as FPGAs, it is obstructed by the dense metal interconnection (Figure 2.6b). Alternatively, backside irradiation could overcome the obstruction problem but the use of wavelengths for single photon absorption would require thinning of the device substrate.

Figure 2.6 – Schematic representation of frontside and backside laser irradiation



Source: The Author

The second process for laser testing of electronic devices is the two photon absorption (TPA) (GÖPPERT-MAYER, 2009; MCMORROW et al., 2002). In this case, wavelength in infrared is used to obtain higher penetration range allowing backside irradiation (Figure 2.6c) without substrate thinning but, since a single photon energy will be below bandgap, the ionization depends on the excitation of the electron by the absorption of two photons whose energies added will be above bandgap. However, the relatively rare occurrence of two photon absorption and the need of a minimal amount of electron positron pairs for creating a significant transient effect will require meticulous focusing of the laser beam in the region of the storage cell active region (Figure 2.6d) which, in turn, requires complex focusing optics, positioning and control of environment parameters such as air temperature to continuously compensate or minimize substrate undulations due to manufacturing process and thermal gradient along device body.

In brief, while single photon absorption (SPA) has lower cost, is easier to use and a more stable process, the two photon absorption (TPA) process overcomes the problem of obstruction and penetration depth and has a smaller laser spot size affecting less storage cells simultaneously.

Buchner et al. (1987) describe simulation of single-event upsets (SEU) in SRAM memory using single-photon absorption (SPA) in near-infrared from frontside obtaining a laser spot size of $2\,\mu\mathrm{m}$ that was of the same order of the size of the sensitive area of the transistor at the time. McMorrow et al. (2013) also uses single-photon absorption (SPA) for simulation of single-event upsets (SEU) in SRAM memory but using backside irradiation in a thinned substrate. In this case, the wavelength in the range of ultraviolet allowed the acquisition of a laser spot size of $0.32\,\mu\mathrm{m}$.

Towards the case of SRAM-based FPGAs, Bocquillon et al. (2007), Pouget et al. (2007) and Canivet et al. (2008) discuss the use of laser as a test tool in SRAM-based FPGAs and, more specifically, the Xilinx Virtex and Virtex II FPGAs using single photon absorption (SPA) with near-infrared laser and backside irradiation, both with and without substrate thinning, analyzing aspects of sensitivity to different levels of power in the laser pulse and the SRAM memory geometry and organization in the FPGA.

While testing a Virtex and Virtex II FPGAs fabricated with $150\,\mathrm{nm}$ technology, Pouget et al. (2007) noted that the laser spot size obtained was too big for an accurate simulation of single-event effects induced from radiation.

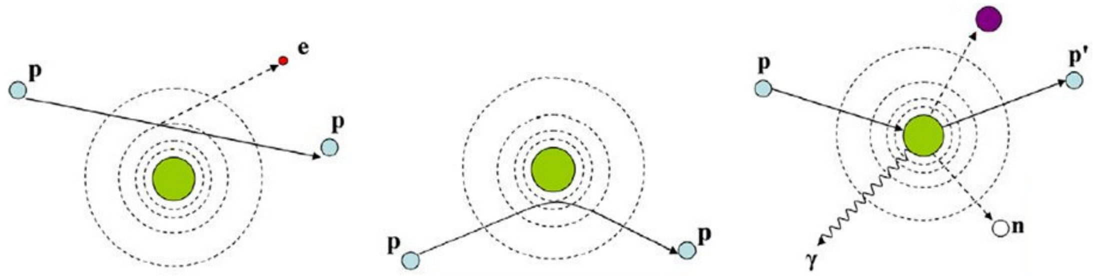
More recently, Kastensmidt et al. (2014) discuss laser testing in a Xilinx Virtex 5 SRAM-based FPGA fabricated with $65\,\mathrm{nm}$ technology using green laser with spot diameter of $1.2\,\mu\mathrm{m}$ and near-infrared laser with spot size of $2.2\,\mu\mathrm{m}$, while Pouget et al. (2017) uses laser on a Xilinx 7 Series / Zynq-7000 MPSoC device fabricated with $28\,\mathrm{nm}$ technology using two photon absorption with near-infrared laser and backside irradiation without substrate thinning obtaining a spot size of $1.5\,\mu\mathrm{m}$.

2.7 Protons, alpha particles and heavier ions induced memory upsets

A certain amount of electron-hole pairs must be produced to overcome the current threshold necessary to unbalance the memory cell causing it to change its value. A major difference between memory upsets induced by laser, as described previously, and induced by charged particles, is that in the case of laser a large number of photons must be absorbed in a short period of time to reach that current threshold while a single charged particle may produce the amount of electron-holes reaching the current threshold.

Charged particles interact easily and strongly with matter and may ionize directly. Taking as example the case of protons, the three main mechanisms of interaction are depicted in Figure 2.7.

Figure 2.7 – Interactions of proton with matter



(a) Coulomb interaction with atomic electrons

(b) Coulomb interaction with atomic nuclei

(c) Nuclear interaction with atomic nuclei

Source: Extracted from Newhauser et al. (2015)

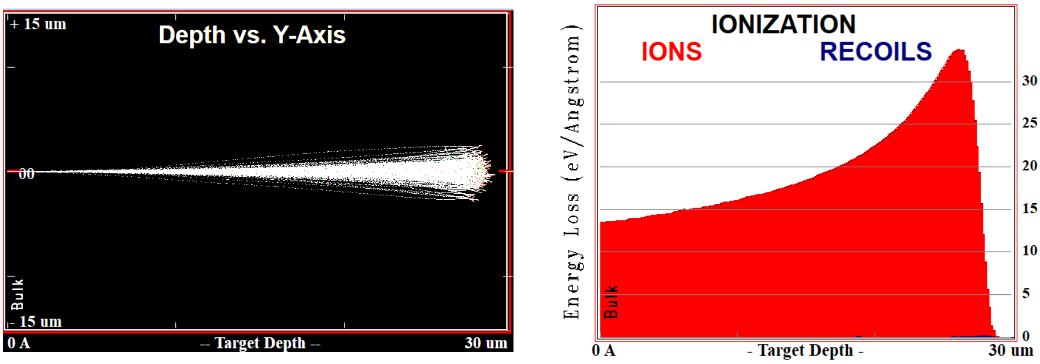
The ionization due to inelastic Coulomb interaction with electrons (Figure 2.7a) is the dominating effect, with a small energy loss per interaction. The gradual loss of energy limits the range of the particle in matter. Since the particle mass is significantly larger than the mass of electron there is no significant deflection of the incident particle that would impact its depth range in matter. Differently, in the elastic Coulomb scattering (Figure 2.7b) the lateral deflection due to nucleus repulsion will reduce the depth range of the particle in matter. Finally, in the nonelastic nuclear interaction (Figure 2.7c) the production of secondary particles, including neutrons, protons and heavier ions may lead to indirect ionization following further interactions of such secondary particles with matter.

Each single incident particle will have a large number of interactions, of different types, losing continuously small amounts of energy while interacting with each atom throughout its trajectory. The average energy loss by unit of length in the particle depth range in matter, known as stopping power, can be estimated using stochastic processes and extensive empirical data. Despite its complex formulation, the particle beam range and stopping power for protons and heavy ions can be conveniently calculated using the SRIM/TRIM software tool (ZIEGLER et al., 2010).

As mentioned earlier, to create a memory upset, the particle incidence must generate an amount of electron-hole pairs large enough to interfere with the memory cell. The amount of electron-hole pairs is proportional to the energy deposited by the particle in the matter, known as *linear energy transfer* (LET). In silicon, for instance, the average energy required to create an electron-hole pair is $3.6\,\mathrm{eV}$ (LUTZ, 2007).

As a first approximation, the energy deposited in the matter is the energy loss, hence we can approximate the LET by the value of stopping power, as estimated by SRIM/TRIM software. Figure 2.8 depicts a SRIM/TRIM simulation for alpha particles (He^{2+}) with energy of $5.48\,\mathrm{MeV}$, for instance from a ^{241}Am radioactive source, on bulk silicon giving an average depth range of $28\,\mu\mathrm{m}$ with the peak of the Bragg curve at the depth of $26\,\mu\mathrm{m}$ where the approximate LET is $33.6\,\mathrm{eV\mathring{A}}^{-1}$ (or equivalently $1.4\,\mathrm{MeV mg^{-1} cm^{-2}}$).

Figure 2.8 – Range and stopping power in silicon estimated by SRIM/TRIM



(a) Depth range of $5.48\,\mathrm{MeV}$ alpha particle

(b) Stopping power of $5.48\,\mathrm{MeV}$ alpha particle

Source: The Author

It is worth noting that this ionization mechanism is present both in the direct ionization, as in the example of Figure 2.7a, and in the indirect ionization by secondary particles after a nuclear interaction, as in the example of Figure 2.7c. That second case is particularly relevant in the case of protons.

There is direct ionization by protons and, in fact, most of the proton energy loss occurs by direct ionization while interacting with electrons. Energy loss, and direct ionization, is higher in low energy protons and can cause memory upsets in some classes of devices (SIERAWSKI et al., 2009). However, the LET in this case is too low to produce memory upsets in most electronic devices.

In devices with a higher LET threshold, it is more likely to observe memory upsets in consequence of nuclear interactions. With protons of medium and higher energies, the main mechanisms causing memory upsets are recoils from elastic scattering and byproducts of nuclear reactions such as emission of alpha particles (evaporation) and heavier particles (fission) (PETERSEN, 1981; REED et al., 2002; AKKERMAN et al., 2017). For these recoils and fission products on proton collision, in turn, the same interaction mechanisms of Figure 2.7 may occur, again possibly causing direct or indirect ionization.

Han et al. (2017) use simulation to study the effects of neutrons and protons in silicon and present the estimates of an equivalent LET considering also the effects of those recoils and nuclear fragments.

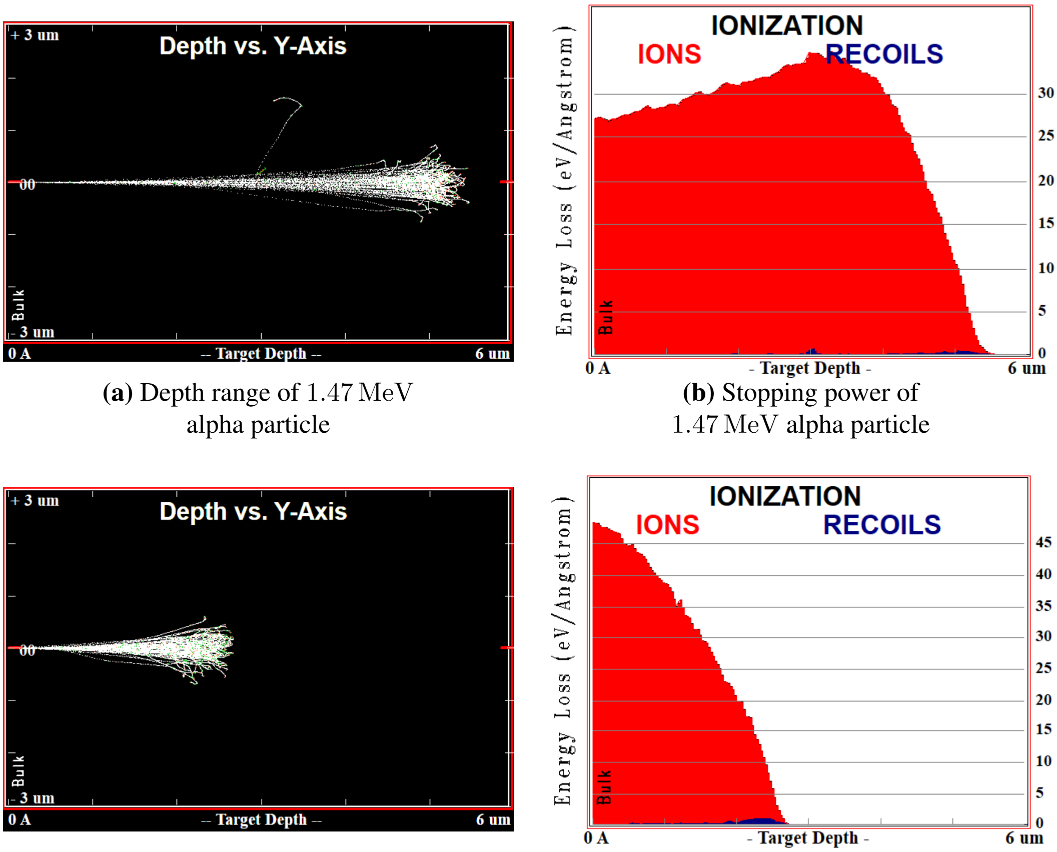
2.8 Thermal and fast neutrons induced memory upsets

Neutrons generated by incidence of high energy particles from cosmic radiation in outer atmosphere (Figure 2.5) are the main contribution to memory upsets in natural environment at ground level and low altitudes. Being electrically neutral, the interaction mechanisms for neutrons are mostly nuclear reactions similar to that of protons seen on Figure 2.7c. Also, having no energy loss by direct ionization, neutrons have a penetration depth typically larger than protons.

Neutrons that lost energy down to an average kinetic energy around $25\,\mathrm{meV}$, becoming in equilibrium with room temperature, are known as thermal neutrons. For instance, high energy neutrons generated by high energy particles in upper atmosphere may lose energy while going deeper in the atmosphere being reduced to thermal energies.

In its low energy state, the thermal neutrons present a random trajectory following elastic collisions with nuclei in the medium until they decay or are absorbed by one of the colliding nuclei. Only a few isotopes have relevant interaction with thermal neutrons causing memory upsets, one of those being the boron (^{10}B) with alpha particles and lithium (^{7}Li) as its fission byproducts causing memory upsets by indirect ionization. Figure 2.9 presents simulation results from SRIM/TRIM software tool for typical energies of alpha particles and lithium produced by thermal neutron capture reactions (BAUMANN et al., 1995), with ranges of approximately $5.1\,\mathrm{\mu m}$ and $2.4\,\mathrm{\mu m}$ in silicon, and peak LET of $32.2\,\mathrm{eV\mathring{A}}^{-1}$ $(1.5\,\mathrm{MeVmg^{-1}cm^{-2}})$ and $48\,\mathrm{eV\mathring{A}}^{-1}$ $(2.1\,\mathrm{MeVmg^{-1}cm^{-2}})$, respectively for alpha particles (^{4}He) and lithium (^{7}Li). In earlier generations of electronic devices, the ^{10}B isotope was present in insulating materials, for instance at the integrated circuit packaging, but in modern devices it is present mostly as dopant in semiconductors.

Figure 2.9 – Range and stopping power in silicon estimated by SRIM/TRIM



(a) Depth range of 1.47 MeV alpha particle

(b) Stopping power of 1.47 MeV alpha particle

(c) Depth range of 0.84 MeV ^{7}Li

(d) Stopping power of 0.84 MeV ^{7}Li

Source: The Author

Interactions of fast neutrons are different from thermal neutrons in many aspects. First, fast neutrons do not bounce around in a random path like thermal neutrons. Second, the fission products are scattered in a forward angle covering 180°. As a consequence, while testing electronic devices in neutron irradiation facilities, the difference between frontside and backside irradiation must be taken into account (HARADA et al., 2012; MILLER et al., 2013; KATO et al., 2019; KORKIAN et al., 2020). Finally, fast neutron interactions are not limited to the neutron capture with a few isotopes producing a few types of secondary particles as is the case for thermal neutrons.

In silicon, for instance, secondary charged particles generated by nuclear interaction of fast neutrons ranges from hydrogen (H) to phosphorus (P), including its isotopes, with varying energies, ranges and LETs (VIAL et al., 1998; MILLER et al., 2013). Besides silicon, depending on the foundry technology process, many materials can be found in the vicinity of sensitive region of transistors and can be target of neutron interactions, including, for instance, tungsten (W), aluminum (Al), tantalum (Ta), titanium (Ti), oxygen (O), hafnium (Hf), nickel (Ni), copper (Cu), nitrogen (N), carbon (C) and hydrogen

(H) (WROBEL et al., 2003; JAMES, 2012).

2.9 Qualification and reliability metrics for radiation

Static and dynamic cross-sections are metrics often applied in the qualification of devices and designs under radiation.

Dynamic cross-section and other metrics used in reliability analysis will be discussed later in Section 4.2, but for now we will focus on the static cross-section, which is an intrinsic parameter of the device or technology.

The static cross section (σ_{SEU}) is usually expressed in terms of area, typically in cm^2 or $\mathrm{cm}^2\mathrm{bit}^{-1}$, and is related to the minimum SEU susceptible area of the device for a particle species such as neutrons, protons and heavier ions.

The particle fluence (Φ), typically in cm^{-2}, can be estimated from average particle flux (ϕ), typically in $\mathrm{cm}^{-2}\mathrm{s}^{-1}$, and irradiation time ($t$) (Eq. 2.1). The static cross section can be obtained from particle fluence (Φ) and number of observed SEU (N_{SEU}) (Eq. 2.2)(JEDEC, 2006). For comparability among devices with different memory volumes, static cross section is usually normalized by the number of bits in the device (N_{bit}) (Eq. 2.3).

$$
\begin{aligned}
\Phi &= \phi \times t, && (2.1)\\
\sigma_{SEU} &= \frac{N_{SEU}}{\Phi}, && (2.2)\\
\sigma_{SEU,bit} &= \frac{N_{SEU}}{\Phi \times N_{bit}}. && (2.3)
\end{aligned}
$$

Overall, the static cross section conveys the sense of probability of a particle generating a memory upset (SEU).

Using the static cross section, we can estimate the soft error rate for the underlying device (SER_{SEU}) (Eq. 2.4) for the design in a targeted environment with a given the particle flux (ϕ_{env}).

$$
\begin{aligned}
SER_{SEU} &= \sigma_{SEU} \times \phi_{env}, && (2.4)\\
SER_{SEU,bit} &= \sigma_{SEU,bit} \times \phi_{env}.
\end{aligned}
$$

The soft error rate (SER) is usually expressed in Failure in Time (FIT) units, being defined as the expected number of errors per 10^9 hours of device operation in a determined radiation environment.

2.10 Radiation-induced susceptibility to memory upsets in modern devices

Schrimpf (2007) notes that commercial devices reduced its sensitivity to total-dose effects in recent years but, at the same time, the technology scaling and reduction of device dimensions increased its fabrication variability and sensitivity to transient effects. Operation at higher frequencies and lower voltages also contributes to sensitivity to transient effects (BARNABY et al., 2008; TRIPPE et al., 2015).

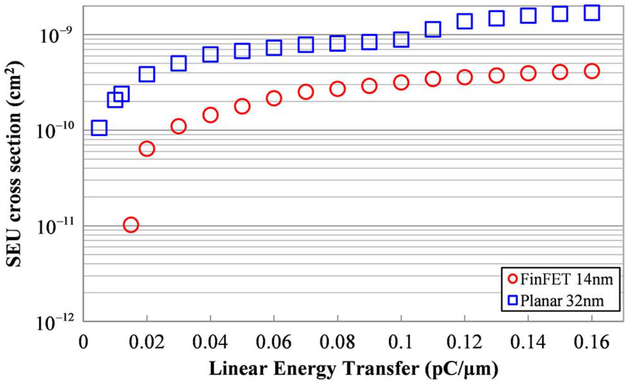
These historical trends on electronic devices were influenced by planar CMOS technology and the trends may be disrupted by the introduction of FinFET technology in modern devices presenting new profiles for TID and SET sensitivity (HUGHES et al., 2015; HAO et al., 2016).

As mentioned in Section 2.5, a key aspect in the SEUs is that a minimal amount of charges must be generated reaching a critical current to unbalance the storage cell and invert its value.

Using geometric modeling and simulations, Fang et al. (2011) compared the charge collection due to neutrons incidence on SRAM memory cells built in planar and FinFET technologies. The key finding was that the two technologies would present a similar critical current, but the charge collection would be lower in FinFET because of the constructive geometry of the transistors. Combining all the factors, the authors estimated a SER to be around $16\times$ lower in FinFET, compared to planar technologies.

Noh et al. (2015) use a similar approach of geometric modeling and simulations, comparing with results from irradiation experiments, for different technological nodes of $32\,\mathrm{nm}$ and $14\,\mathrm{nm}$, for planar and FinFET respectively. The authors address also the occurrences of single-events causing memory upsets in single memory cells (SBU, for single-bit upsets) and in multiple cells (MBU, for multiple-bit upsets, or MCU, for multiple-cell upsets). The authors report SEU, SBU e MCU rates lower for FinFET, besides FIT rates $10\times$ lower for FinFET, with the caveat that this comparison is not only between planar and FinFET, but also $32\,\mathrm{nm}$ and $14\,\mathrm{nm}$. After validating the simulations against irradiation results, the authors present the simulated static cross section (σ_{SEU}) for different energies, as presented in Fig. 2.10. As with Fang et al. (2011), Noh et al. (2015) also infer that the factor with higher impact to the better results of FinFET resides in the constructive geometry of the FinFET, more than doping profile, interconnection (back-end) or other electrical elements.

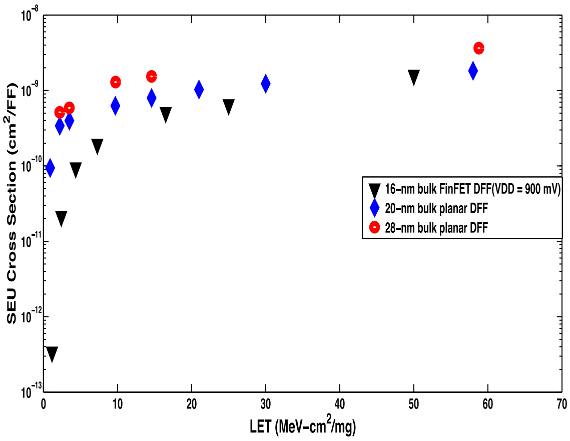
Figure 2.10 – Simulated neutron cross-section for planar and FinFET devices



Source: Noh et al. (2015)

Nsengiyumva et al. (2016) perform a similar comparative study between planar and FinFET for memory elements, but under heavy-ions with energies over $100\,\mathrm{MeV}$. The authors observe cross section orders of magnitude lower for FinFET, but with the remark that this difference is less prominent at higher LETs, as seen in Fig. 2.11.

Figure 2.11 – Experimental heavy-ions cross-section for planar and FinFET devices

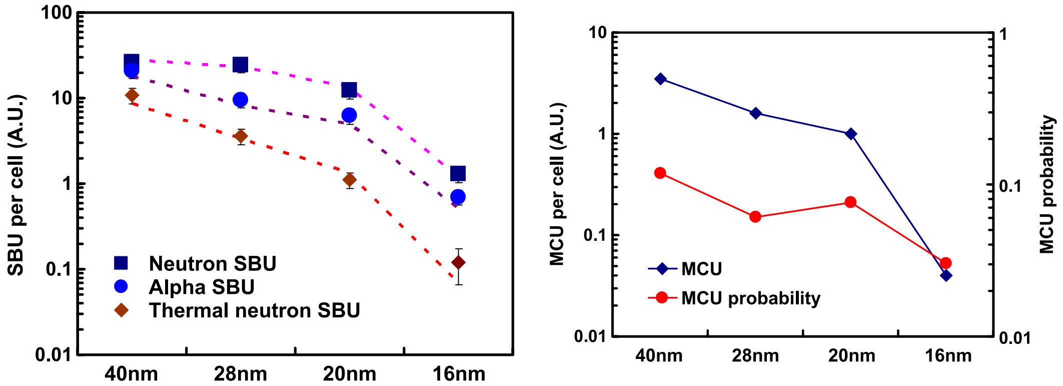


Source: Nsengiyumva et al. (2016)

In another work from Fang et al. (2016), different technology nodes of planar $40\,\mathrm{nm}$, $28\,\mathrm{nm}$ and $20\,\mathrm{nm}$ devices, and $16\,\mathrm{nm}$ FinFET, are compared with results from spallation neutrons, thermal neutrons and alpha particles from ^{241}Am. The authors present the single-bit SEU rates for different technologies irradiated with different particles (Fig. 2.12a), as well as the rate and probability of occurrence of multiple-cell SEU for high-energy neutrons (Fig. 2.12b).

It worths noting in Fig. 2.12 that the occurrence of MCUs, and the overall SEU rate, scales not only because the constructive geometry of planar and FinFET devices, but

Figure 2.12 – Experimental neutron and alpha particles SEU rates for planar and FinFET devices



(a) Measured neutrons and alpha particle SBU rates

(b) High-energy neutrons MCU rates

Source: Fang et al. (2016)

also with the technology node, although it seems to have a new trend starting with the FinFET technology.

The work from Fang et al. (2016) is specially relevant in the context of this study because it addresses the two technologies near the technologies used at the Xilinx 7 Series and Zynq-7000 devices, and the Xilinx UltraScale+ devices covered in this thesis, which are fabricated in planar $28\,\mathrm{nm}$ and FinFET $16\,\mathrm{nm}$ respectively.

Narasimham et al. (2018) also address the $16\,\mathrm{nm}$ FinFET, compare with planar technology in different technological nodes, and go beyond analyzing the $7\,\mathrm{nm}$ FinFET. The authors present results from irradiation experiments with neutrons and alpha particles from ^{241}Am.

Other works addressing planar and FinFET technologies, as well as scaling trends, are summarized in the survey from Bhuva (2018).

When we go into the world of SRAM-based FPGAs, more specifically, the presence of MBUs plays an important role in the reliability as it may interfere with fault tolerance techniques implemented in the circuit by the FPGA manufacturer, such as error correction codes in memory blocks and healing by memory scrubbing, as well as fault tolerance techniques implemented in the design or application by the end-user, such as fine-grained modular redundancy.

This is where some authors start to distinguish between the multiple-bit upsets (MBU) and multiple-cell upsets (MCU). There is no standardized nomenclature, but some authors prefer to call MCU the SEU occurring and the lower level of memory devices, and MBU whenever that MCU affects multiple bits in a data word at the higher level of circuit or application. For instance, for some authors working with microprocessors, a MCU is a MBU when multiple bit-flips are seen in the same memory byte. Such byte may be protected by mitigation techniques like Hamming codes that can correct only single-bits, hence the distinction between MBU, which would not be corrected by the Hamming code, and the MCU, where the bit-flips occurs in different bytes, prone to be corrected by the Hamming code.

As we will work mostly at an intermediate level of memory organization, with little or no access to the physical organization of the memory cells, throughout this thesis we will use preferably the term MBU irrespective of the bit-flips occurring or not in the same data word in some arbitrary level of memory organization.

3 PROGRAMMABLE GATE ARRAYS

Most integrated circuits in mass production are custom designed, for instance the application-specific integrated circuits (ASIC). In contrast, programmable devices, typically of general purpose, are used when low volume, quick design turnaround and upgrade capability are important (MENCER et al., 2020). The programmable logic devices of interest for this thesis are known as *field-programmable gate arrays* (FPGA).

The computing power of FPGA devices comes from basic components such as logic gates, registers, signal routers, arithmetic blocks and memory blocks.

The FPGA technology as we know today grew out of the programmable logic at the beginning of the decade of 1980s. At that time already existed gate arrays from several manufacturers, typically programmable structures of AND, OR and NOT gates. Distinctive characteristics introduced by FPGAs were the programmability of both the interconnects, which compose the signal fabric, and the logic gates, which are the combinational logic function typically with truth table encoded in the form of a look-up table (LUT) (XILINX, 1988b). The FPGAs also used volatile (RAM) memory to store the configuration bits instead of fuses and EPROM memory. Some of these characteristics are seem on earlier models of FPGA from major manufacturers like Altera Corporation, with its FLEX 8000 (ALTERA, 2003) device family, and Xilinx Inc., with its XC2000 family (XILINX, 1988a).

Market forecasts from Gartner analysts (BLANCO et al., 2017) suggested that by 2021 75% of the of new business volume for large-scale FPGAs would be driven by AI, mobile telecommunication networks (5G) and industrial Internet of things (IoT).

The interest for SRAM-based FPGAs in aerospace applications have also increased in recent years. For space applications this phenomenon is twofold. First, in classical space applications, there is an increase of lifetime for satellites, meaning that higher flexibility and reprogrammability are becoming requirements for future proof spacecrafts. Studies fostered by the European Space Agency (ESA) in the beginning the the decade of 2000s (HABINC, 2002) already tried to map opportunities and challenges of using SRAM-based FPGAs in space. Second, we have seem the emergence of the called "new-space" paradigm (PAIKOWSKY, 2017), with increased participation of private companies leading low cost small satellite missions adopting simplified industrial and engineering processes for reliability, conformance and quality assurance aiming at saving costs. The adoption of dual-use technology, commercial grade off-the-shelf (COTS) devices, and higher integration of microprocessors and reprogrammable logic in systems on chip (SoC) helps in reducing volume, mass and power consumption to fulfill the new-space requirements.

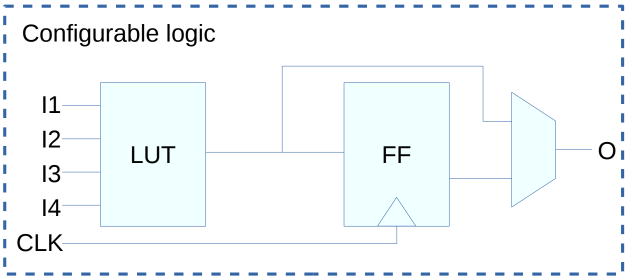
However, a well known characteristic of reprogrammable FPGAs based on SRAM is its higher susceptibility to SEUs, meaning that even in the more relaxed scenario of new-space the use of COTS SRAM-based FPGAs without the proper qualification, reliability evaluation and SEU mitigation may jeopardize the mission objectives.

To better understand the role of SRAM memory in reconfigurable FPGA applications and its relationship with the reliability under radiation, the architecture if the FPGA devices is described briefly on the following, with focus on devices from Xilinx.

3.1 Combinational and sequential logic

Combinational and sequential logic resources are some of the fundamental elements of FPGA devices. The architecture varies with manufacturers and device families, but usually these two classes of resources are tightly related into a configurable logic block as depicted in Figure 3.1.

Figure 3.1 – Simplified view of a logic block with 4 inputs and one memory element



Source: Adapted from Xilinx (2013)

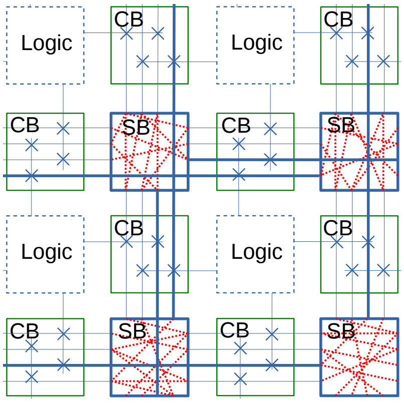
In earlier products from Xilinx the basic structure of LUT and flip-flop was named a *logic cell* (LC) (XILINX, 2013), organized in structures named *slice* which, in turn, was grouped in a structure named *Configurable Logic Block* (CLB). The number of LUT, flip-flop and slices by CLB, as well as the number of inputs in a LUT, varied along the various generations of Xilinx products. In Xilinx 7 Series product family each each slice includes four 6 input LUT and eight flip-flops. In these devices the CLB is composed of two slices (XILINX, 2016). In later UltraScale+ product family (XILINX, 2017a) the slice pairs were fused into a single slice, one CLB having one slice with eight 6 input LUT and sixteen flip-flops, hence the same resources density of 7 Series but potentially more efficient routing and logical mapping.

3.2 Signal fabric

The signal fabric is the second fundamental element of an FPGA. It is a key element because most of the FPGA area and configuration resources will be dedicated to the signal routing. The transport of signals throughout the FPGA is implemented with programmable switches to build different connections using the interconnect architecture available at the device. There are many forms of organization for the interconnection architecture, including sea of gates, row-based, hierarchical and island style (BETZ et al., 1999; GEORGE et al., 2001). Many FPGA devices use combinations of these different interconnection styles.

An organization similar to island is found in Xilinx FPGAs (XILINX, 2013) but other styles of interconnection are also present on the same devices. In the island interconnection (Fig. 3.2) the array of logic blocks is interwoven with vertical and horizontal wires connecting to the logic blocks through programmable switch boxes (SB) and connection boxes (CB) . Masud et al. (1999) describe also different styles of switch matrices to implement in a SB considering wires of different lengths.

Figure 3.2 – Simplified island interconnection architecture



Source: Adapted from Betz et al. (1999) e George et al. (2001)

Besides que island structure, direct interconnections between adjacent elements are also seen on Xilinx (XILINX, 2013) devices for high-speed connections between horizontally adjacent CLBs.

Another aspect of the signal fabric is that switch matrices may be unidirectional (single-driver), for instance using buffers, bidirectional (multiple drivers), for instance using pass transistors, or combination of both. Lewis et al. (2003) e Lemieux et al. (2004)

discuss advantages of unidirectional over bidirectional routing wires in aspects such as area and delay. Modern FPGAs use mostly unidirectional routing wires but bidirectional wires may still show useful for long wire segments.

In Xilinx FPGAs the signal routing is implemented by a structure named *Programmable Interconnection Point* (PIP), which is a connection multiplexer that can be programmed to connect one wire to another forming the routing required to implement a specific net in the design (XILINX, 2018c).

The PIP specifies a configurable connection between a switch matrix input and a switch matrix output, which is connected to a fixed unidirectional wire in turn connected to the input of another switch matrix or to a pin (BOZZOLI et al., 2017). These switch matrices can the interconnection blocks, which are similar to the SB in Fig. 3.2, the interface blocks, which are similar to the CB in Fig. 3.2, or the dedicated clock distribution blocks forming the clock tree or clock spine.

3.3 Memory blocks

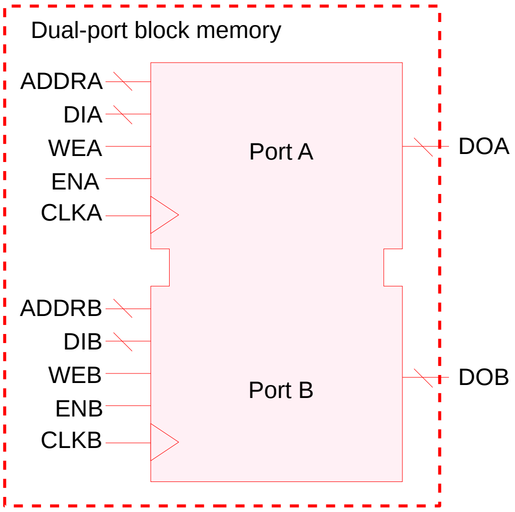
Data storage resources on FPGAs go beyond the one bit flip-flops available on the logic block. One of the first attempts to provide larger volumes of data memory inside the FPGA consisted in reorganizing the memory already available at the logic block, for instance the memory used to store the LUT, as an array to store user data.

In Xilinx products, the LUT memory inside the CLBs can be used as on-chip ROM, single-port or dual-port RAM, shift registers (SR), stacks (last-in first-out, LIFO) and buffers (first-in first out, FIFO) (XILINX, 1999; HSIEH et al., 1990). This resource, also known as *distributed* RAM or LUTRAM, allows the use of LUT memory to store end user data, but the memory available inside the logic block is limited to small arrays that must be cascaded to build larger memory structures. Another limitation is that only part of the device CLBs can be configured as LUTRAM.

Another kind of memory resource available in FPGAs is the RAM blocks, which are dedicated memory arrays for data storage (Fig. 3.3). In Xilinx products these resources are known as *block* RAM (BRAM) and UltraRAM (URAM).

In Xilinx 7 Series product family (XILINX, 2019) the BRAM size is 36×2^{10} bits that could also be configured as two independent blocks of 18×2^{10} bits (XILINX, 2012b). Xilinx UltraScale+ product family introduced a new type of memory block named UltraRAM (URAM) with 288 Kibit but these devices also have BRAM with the same size and

Figure 3.3 – Simplified view of dedicated memory array



Source: Adapted from Xilinx (2013)

functionality of the 7 Series product family (XILINX, 2021e).

3.4 Arithmetic

Arithmetic operators can be implemented using combinational logic resources in the logic block while propagating intermediate results through the general routing of the signal fabric. Notwithstanding, FPGAs may have also specialized structures supporting high-performance arithmetic computations.

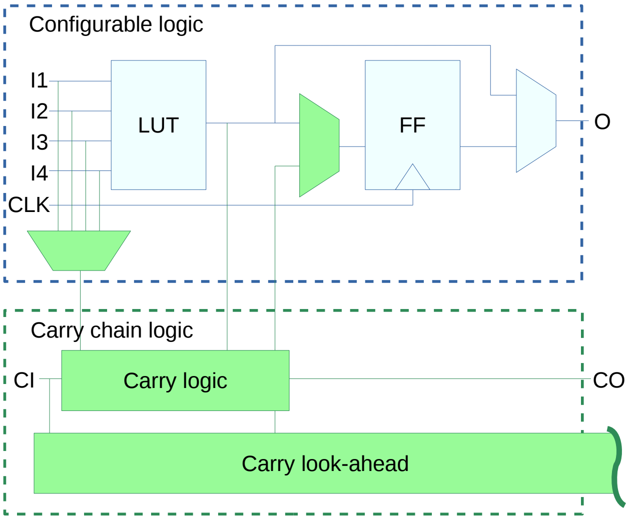
A hardware structure commonly used by arithmetic operators is the carry chain logic, passing information from one logic function (implemented in CLB) to another (Fig. 3.4). A typical use of carry chain is seen in addition operators but carry chain can also be used in other cases, for instance in comparison operators.

Despite a small increase in area, the presence of dedicated carry chain logic can improve significantly the computation performance. Also, there are different architectures of carry chain, such as ripple carry, carry select and carry look-ahead, that present different area requirements and performances.

Xilinx introduced dedicated carry logic circuit in its earlier products (XILINX, 1999; HSIEH et al., 1990). Xilinx 7 Series (XILINX, 2016) and UltraScale+ (XILINX, 2017a) product families supports fast carry look-ahead dedicated logic over LUTs in the same slice and high-speed carry cascading between slices.

Other important resources for arithmetic operations are the dedicated digital signal processing (DSP) blocks (Fig. 3.5) that increase performance in DSP applications with

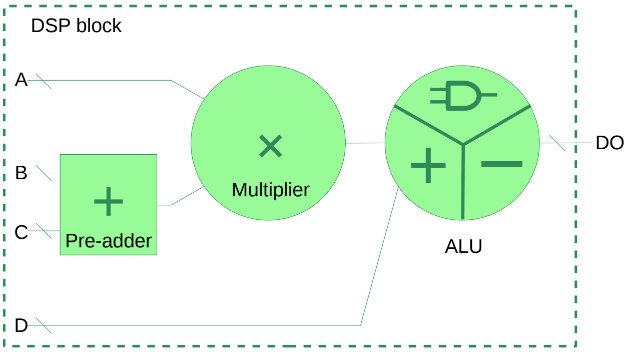
Figure 3.4 – Simplified view of carry logic integrated with the logic block



Source: The Author

high data throughput, such as radio and image processing, while saving resources from the logic blocks.

Figure 3.5 – Simplified view of dedicated DSP block



Source: Adapted from Xilinx (2011)

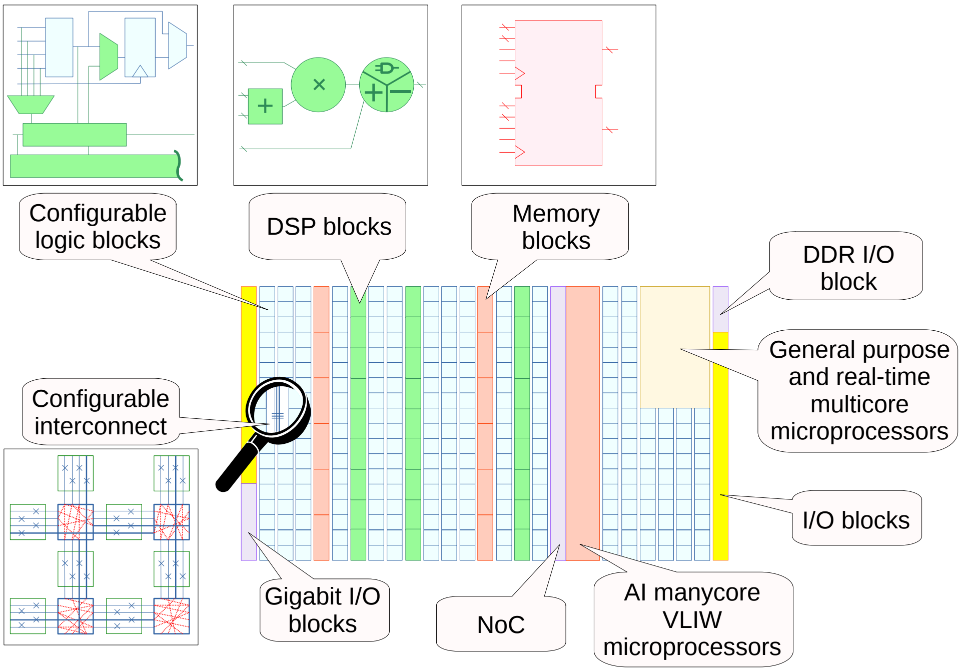
Earlier Xilinx products featured a dedicated multiplier block operating on two's-complement signed 18 × 18 bits with 36 bits output (XILINX, 2007). Using this multiplier block, the implementation of multiply-accumulate (MACC) operations commonly found in DSP required the addition stage to be implemented in the logic block. A major improvement was introduced later with a DSP block a 48 bits adder block that could be used independently, could be combined with the multiplier to implement fused MACC operation, or could be used as a pre-adder to another DSP block (XILINX, 2008). This DSP block could still operate in the legacy mode equivalent to the previous generation of multiplier block. Along the different product families the adder inside the DSP block became an arithmetic logic unit (ALU) extended with bit-wise logical operations like AND, OR and XOR (XILINX, 2017b). In Xilinx 7 Series device the DSP block features a 25 bits pre-adder, 25 × 18 bits multiplier and 48 bits ALU (XILINX, 2018a), while in

Xilinx UltraScale the DSP block was extended with a pre-adder of 27 bits and 27 × 18 bits multiplier, maintaining the 48 bits ALU (XILINX, 2021d). Another major improvement was introduced in Xilinx Versal product family with a 27 × 24 bits multiplier, 58 bits ALU, complex operations and floating-point operations in 16 bits and 32 bits (IEEE 754™ binary16 and binary32 representation) (XILINX, 2021f).

3.5 Other features in modern FPGA devices

Modern FPGA devices are also implemented as systems-on-chip (SoC) along with multicore or real-time microprocessors, typically a RISC (reduced instruction set computer) microprocessor, graphics processing unit (GPU), arrays of specialized processing architectures supporting very long instruction word (VLIW), and communication infrastructure such as network-on-chip (NoC), extensible interface for microcontroller bus architecture (AMBA AXI), external DDR interface and external gigabit I/O (Fig. 3.6). As stated by Mencer et al. (2020), everything is becoming an SoC.

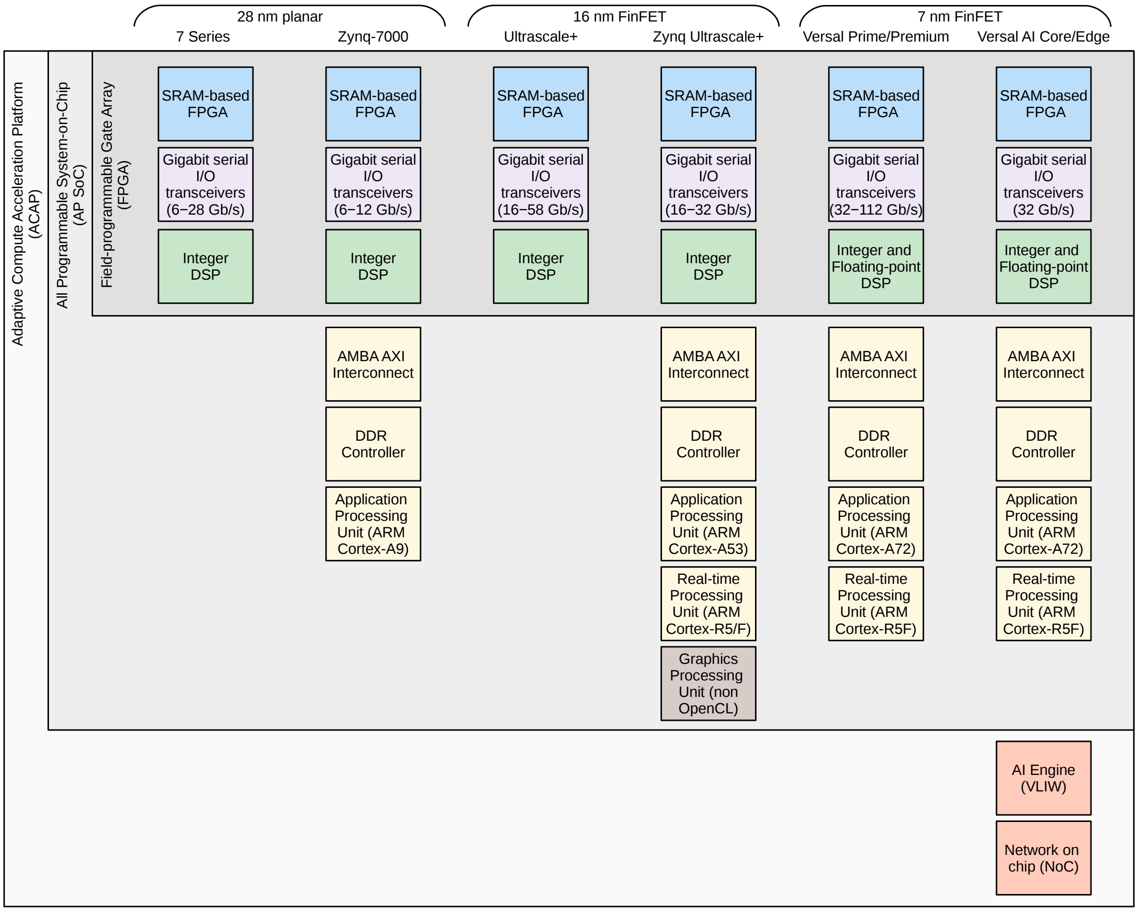
Figure 3.6 – Simplified view of FPGA elements


Configurable logic blocks
DSP blocks
Memory blocks
DDR I/O block
General purpose and real-time multicore microprocessors
Configurable interconnect
I/O blocks
Gigabit I/O blocks
NoC
AI manycore VLIW microprocessors

Source: The Author

As this works focus on Xilinx devices, Figure 3.7 summarizes main features available on recent devices families from this manufacturer.

Figure 3.7 – Recent Xilinx devices families


28 nm planar
7 Series
Zynq-7000
16 nm FinFET
Ultrascale+
Zynq Ultrascale+
7 nm FinFET
Versal Prime/Premium
Versal AI Core/Edge
Adaptive Compute Acceleration Platform (ACAP)
All Programmable System-on-Chip (AP SoC)
Field-programmable Gate Array (FPGA)
SRAM-based FPGA
Gigabit serial I/O transceivers (6−28 Gb/s)
Integer DSP
SRAM-based FPGA
Gigabit serial I/O transceivers (6−12 Gb/s)
Integer DSP
AMBA AXI Interconnect
DDR Controller
Application Processing Unit (ARM Cortex-A9)
SRAM-based FPGA
Gigabit serial I/O transceivers (16−58 Gb/s)
Integer DSP
SRAM-based FPGA
Gigabit serial I/O transceivers (16−32 Gb/s)
Integer DSP
AMBA AXI Interconnect
DDR Controller
Application Processing Unit (ARM Cortex-A53)
Real-time Processing Unit (ARM Cortex-R5/F)
Graphics Processing Unit (non OpenCL)
SRAM-based FPGA
Gigabit serial I/O transceivers (32−112 Gb/s)
Integer and Floating-point DSP
AMBA AXI Interconnect
DDR Controller
Application Processing Unit (ARM Cortex-A72)
Real-time Processing Unit (ARM Cortex-R5F)
SRAM-based FPGA
Gigabit serial I/O transceivers (32 Gb/s)
Integer and Floating-point DSP
AMBA AXI Interconnect
DDR Controller
Application Processing Unit (ARM Cortex-A72)
Real-time Processing Unit (ARM Cortex-R5F)
AI Engine (VLIW)
Network on chip (NoC)

Source: The Author

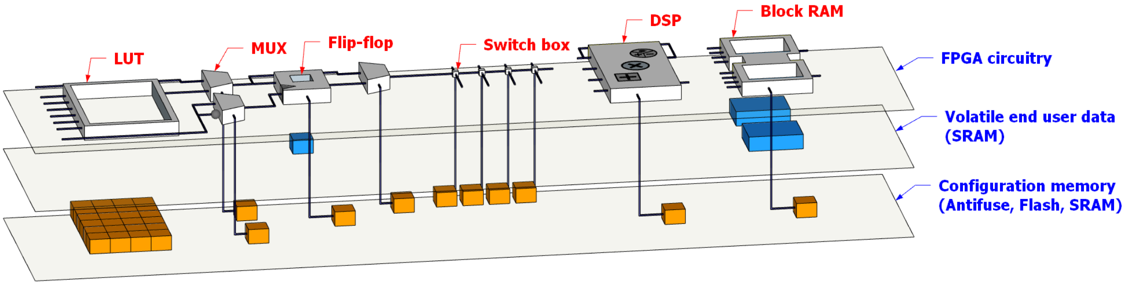
3.6 Configuration memory and bitstream

As mentioned previously, the FPGAs contain a set of basic elements such as logic gates, registers, signal routers, arithmetic blocks and memory blocks that can be programmed, in the field, into a specific application.

The FPGA device can be seen as organized different planes (Fig. 3.8). Behind the FPGA configurable elements there are bits, in the configuration memory, that programs the desired behavior of the FPGA for the targeted application. Inside the FPGA, that programming, or configuration, can be stored in different technologies, the most relevant being antifuse, Flash and static RAM (SRAM).

The behavior of antifuse is the opposite of the fuse. An antifuse is a device that initially does not conduct current (normally open) but can be programmed, or burned, to conduct current, for instance by breaking a dielectric and forming a low-impedance connection. As antifuses cannot be opened to its initial factory state, it can be programmed only once and the regions of an antifuse-based FPGA that were programmed cannot be

Figure 3.8 – FPGA circuitry and data planes


LUT
MUX
Flip-flop
Switch box
DSP
Block RAM
FPGA circuitry
Volatile end user data (SRAM)
Configuration memory (Antifuse, Flash, SRAM)

Source: The Author

reprogrammed anymore. One example of antifuse-based FPGA is the Actel, now Microchip, SX-A product family (ACTEL, 2007).

As design errors cannot be corrected in antifuse-based FPGAs by reprogramming the device, design verification and test procedures for antifuse-based FPGAs are more rigorous than on the other technologies, becoming closer to the ASIC design flow and potentially undermining the faster time-to-market and lower non-recurring engineering (NRE) cost advantages of the FPGAs. Being unable to reprogram also makes it difficult to upgrade and adapt to new functions, requirements or environment changes over the product lifetime.

Another type of FPGAs store the configuration data in non-volatile, but reprogrammable, Flash memory, therefore allowing upgrade and correction of errors in the design. Example of Flash-based FPGAs are the Actel, now Microchip, ProASIC and ProASIC3 product families (ACTEL, 2002; MICROSEMI, 2012).

Finally, in SRAM-based FPGAs the configuration data is stored in SRAM cells that, being volatile, must be programmed each time the device is powered on. FPGA products from Xilinx, Altera, now Intel, and NanoXplore are examples of SRAM-based FPGAs.

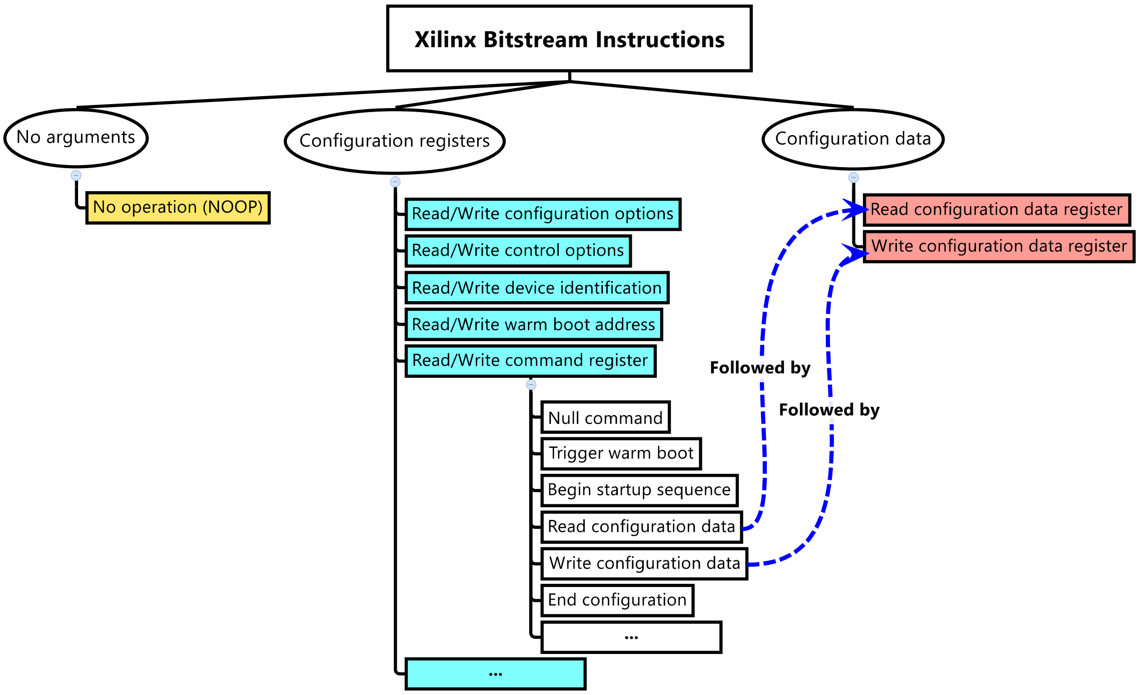
The two leading manufacturers of FPGA, Xilinx and Altera, now Intel, adopted the SRAM approach and, consequently, SRAM-based FPGAs had a prominent position in the market over the last decades, with antifuse-based and Flash-based FPGAs alternating in the second place in market share.

Flash-based FPGAs still may have an advantage over SRAM-based FPGAs in terms of power consumption, but the production scale of SRAM-based FPGAs make it competitive in costs. Another important advantage of SRAM-based FPGAs is the reconfiguration and partial reconfiguration that allows the programmed design to be changed quickly allowing the same hardware device to be retargeted to different tasks.

The configuration data is loaded into the SRAM-based FPGA in the form of a sequence of bits, the *bitstream*, generated by the synthesis or configuration tools. The bitstream does not contain only the memory image of configuration data for the FPGA basic elements but also other configuration commands and instructions for setup and maintenance of the FPGA device, such as to configure encryption keys or to reset the device by reloading the contents of flip-flips to its initial value.

In the case of Xilinx devices, the bitstream is a sequence of instructions and these instructions may of may not have data arguments (Fig. 3.9). The data present in the bitstream can be broadly classified into two types, which are the values for configuration registers and contents for the programming memory, or configuration data. The instructions to write the configuration registers will be followed by the new value of the register as its argument. The instruction to write the configuration data will be followed by the new contents of the programming memory, which can be a long sequence of several megabytes.

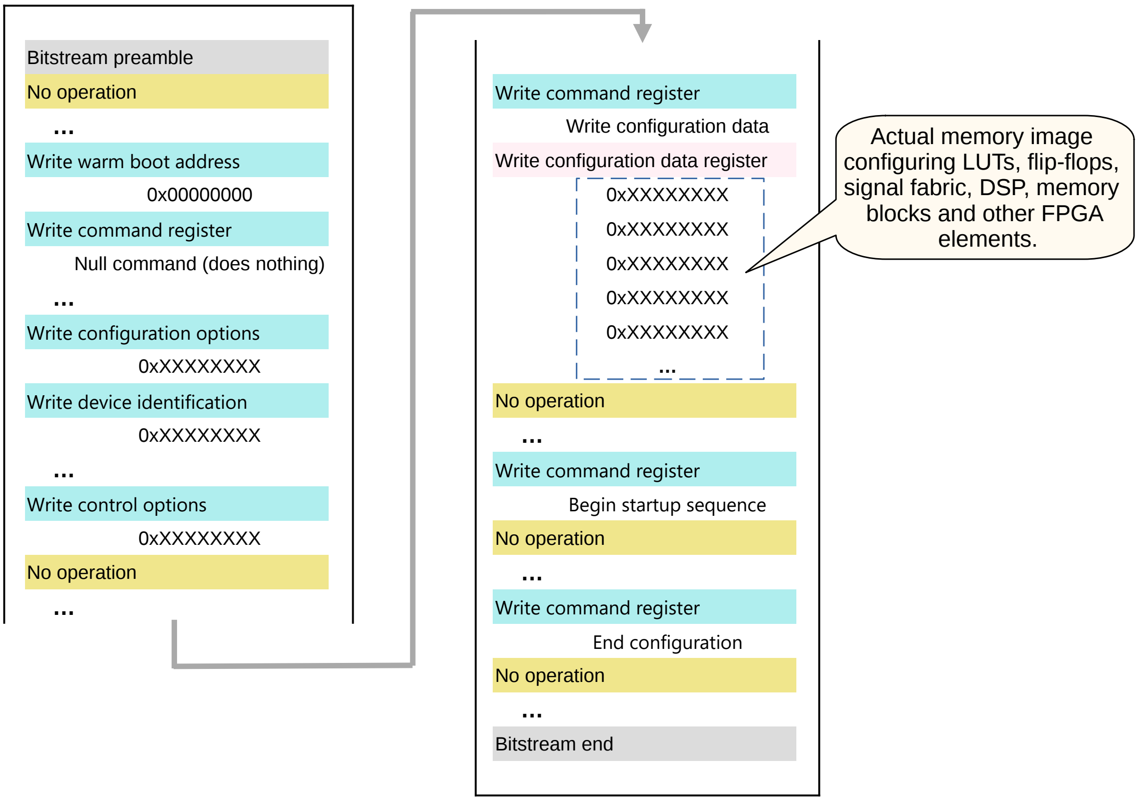
Figure 3.9 – Xilinx FPGA bitstream instructions


Xilinx Bitstream Instructions
No arguments
Configuration registers
Configuration data
No operation (NOOP)
Read/Write configuration options
Read/Write control options
Read/Write device identification
Read/Write warm boot address
Read/Write command register
Null command
Trigger warm boot
Begin startup sequence
Read configuration data
Write configuration data
End configuration
...
...
Read configuration data register
Write configuration data register
Followed by
Followed by

Source: The Author

The Xilinx bitstream to program the FPGA, simplified in Fig. 3.10, typically starts with a sequence of instructions to place the FPGA in a well known state and configuration mode, for instance setting the values of configuration registers, followed by an instruction to set the configuration data, which follows as an argument to that instruction, and then terminates with another sequence of instructions to complete the configuration, check if the bitstream was loaded correctly and begin the FPGA startup sequence to put the device in execution mode (XILINX, 2012b, 2018b, 2021c).

Figure 3.10 – Example Xilinx bitstream to program FPGA



Source: The Author

A bitstream or, in other words, a sequence of instructions, can also be loaded into the FPGA not to write a new value into a configuration register but to read the current value of the register. In the same way, a bitstream can be loaded into the FPGA not to write but to read back its current configuration data.

The configuration data loaded by the bitstream may apply to the whole device, for instance when the device is first powered on, or apply to only a region of the device, for instance in the case of partial reconfiguration where only a region of the FPGA is retargeted to another task or application. This configuration data also may contain not only configuration for the logical elements and signal fabric, but also the initial values for the memory blocks.

The Xilinx FPGA programming memory is divided logically in three types, which are:

- The configuration memory (CRAM) for BRAM blocks, DSP blocks, CLB blocks and signal fabric, among other elements,
- The initialization user data contents for BRAM blocks, and

- The special configuration for the CLB blocks.

It is worth noting that BRAM blocks may appear twice in the bitstream, once to configure the BRAM block, for instance to define address and data width and if it will operate as a single port or dual port memory, and again to load the initial user data content of the memory. In the first case the configuration data goes into the CRAM memory region for BRAM and in the second case the configuration data goes into the BRAM memory region for BRAM.

In the same way, CLB blocks may appear twice in the bitstream, once to configure the CLB block, for instance loading the content of the LUTs and defining the initial value and reset value of flip-flops, and once to configure the scope (or contour, fence, seal-ring, or bounding box) for global operations, for instance restricting the scope of a global reset to only a specific region of the FPGA that was subject to partial reconfiguration. In the first case the configuration data go into the CRAM memory region for CLB and in the second case the configuration data go into the special configuration memory region for CLB.

The Xilinx FPGA programming memory is also divided logically in rows, columns and frames. Each type of FPGA element, for instance I/O blocks, CLB blocks, DSP blocks, BRAM blocks defines a different type of column, as depicted in the simplified example of Fig. 3.6. The number of rows and columns on the device depends on the model or size of the part.

A frame is a sub-element into the column. The size of a frame is fixed and depends on the device family. For instance, in Virtex-5 product family the frame size is 1,312 bits, in 7 Series product family the frame size is 3,232 bits and in UltraScale+ product family the frame size is 2,976 bits. For each device family, the number of frames in a column depends of the type of the column but, for each type, it is also fixed.

One frame is the minimal unit of the programming memory that the instructions in a Xilinx bitstream can read or write, therefore the size of the argument to the instruction to write the configuration data into the FPGA will always be a multiple of the size of the frame.

4 QUALIFICATION AND RELIABILITY IN SRAM-BASED FPGAS

There are many parameters used to characterize a circuit, function or application implemented into SRAM-based FPGA under soft errors. Some of the terms and metrics useful in the scope of this thesis will be defined briefly in the following, as well as its specific uses in the context of SRAM-based FPGAs.

4.1 Qualification parameters

Three important parameters used to characterize a design are *computing performance*, *energy*, and *area*.

The computing performance of a design can be expressed in terms of the execution time, operational frequency and the processed workload. The execution time can be defined by the number of clock cycles to perform the operation. According to the FPGA and design architecture, a maximum clock frequency is achieved. Another important parameter is the workload processed by the design, being the amount of data computed in one execution.

In terms of reliability, the computing performance information is helpful to know how much time the design is exposed to radiation-induced errors during the execution of the implemented function.

Another dimension that is relevant to embedded environments with constrained power supply, such as small satellites and devices powered by batteries or energy harvesting, is energy consumption. This aspect also translates in heat that must be managed.

In SRAM-based FPGAs the power rating of a design is usually expressed in terms of static power, which does not depend on the electronics switching, and dynamic power, which depends on electronics switching and, therefore, in influenced by operating frequency and processed data. In SoC devices, discussed briefly in Section 3.5, the power consumption by other elements such as microprocessors and external DDR memory can easily surpass the power required by the FPGA elements.

The total power required by a device can be measured or estimated by synthesis tools. To obtain the total energy consumption one must take into account also the processing time. A design may be constrained by both the total energy budget an the instantaneous power that can be delivered or dissipated by the heat management.

In SRAM-based FPGAs, the area of an implemented design can be expressed in terms of the number of used resources such as the LUTs, flip-flops, BRAM blocks, DSP blocks, etc, that were defined along the Chapter 3. Is is also possible to express area in terms of number of frames in the region of the configuration memory used to implement the design. Since each memory frame is related to a specific row and column in the device floorplan, as described in Section 3.6, it is possible to calculate the number of configuration frames used by a design.

Another form of expressing area is in terms of number of essential bits that will be seen in Section 4.3.

It is also import to characterize the failure events. Functional failures can be further classified in terms of failure behavior. For instance, a functional failure can be classified as a silent data corruption (SDC) when the design reports a result but that result is found to be incorrect. Failures can also be classified as hang, timeout of functional interruption (SEFI) in the case the design does not report a result in the expected time.

Depending on the fault detection and mitigation techniques implemented on the design, other diagnostic status may also be reported, for instance a detected unrecoverable error (DUE).

Different grades of severity can also be attributed to the failures, for instance considering how many elements in the output vector are incorrect (BENEVENUTI et al., 2018c) or the error magnitude relative to an acceptance threshold (RODRIGUES et al., 2019; MAILLARD, Pierre et al., 2022).

4.2 Additional radiation and reliability metrics

While the static cross section, seen on Section 2.9 refers to the underlying device or technology susceptibility to SEU, the dynamic cross section refers to the susceptibility to functional failure in the end-user design or application. Dynamic cross section encompasses the underlying susceptibility to SEU, but also takes into account the logical and temporal masking characteristic of the design.

The dynamic cross section ($\sigma_{failure}$) conveys the probability of a particle generating a functional failure in the design, that is, chance of a particle generating a SEU in a relevant bit of memory to the design or application, in a relevant moment in the processing cycle.

Dynamic cross section ($\sigma_{failure}$) can be used to compare designs susceptibility for soft errors, computed as the ratio between the number of events of functional failure

(N_{events}) and total particles fluence (Φ), as presented at Eq. 4.1. Using the dynamic cross section, one can also estimate the soft error rate for failures ($SER_{failure}$) (Eq. 4.2) for the design in a targeted environment with a given the particle flux (ϕ_{env}).

$$\sigma_{failure} = \frac{N_{events}}{\Phi}, \tag{4.1}$$

$$SER_{failure} = \sigma_{failure} \times \phi_{env}. \tag{4.2}$$

Another form of expressing the area susceptible do SEU is in terms of number of critical bits as seen in Section 4.3.

Yet another metric often seen in reliability analysis is the mean time between failure (MTBF), which is composed by the mean time to failure (MTTF) and the mean time to repair (MTTR). However, for reparable systems with negligible MTTR, the MTBF can be simplified to MTTF as in Eq. 4.3, for a targeted environment with a given the particle flux (ϕ_{env}). Known the execution time (t_{exec}) of one application processing cycle, one can estimate the mean execution between failure (MEBF) as in Eq. 4.4. A metric for mean workload between failure (MWBF) evaluates the amount of data (workload, w) processed correctly by the design before the appearance of an output error (RECH et al., 2014), as seen in Eq. 4.5.

$$MTBF = \frac{1}{\sigma_{failure} \times \phi_{env}}, \tag{4.3}$$

$$MEBF = \frac{MTBF}{t_{exec}}, \tag{4.4}$$

$$MWBF = MEBF \times w. \tag{4.5}$$

The metrics presented above are useful to describe the design reliability but may be inadequate to compare different choices of implementation, especially in designs with the presence of parallelism, redundancy or mitigations techniques were the failure rate changes over time. A better alternative in this case may be to look at the reliability, or survival, curves of the design.

Given a set of N_{events} observed failure events i for which are known the time $t_{elapsed,i}$, or the fluence Φ_i, of occurrence of each failure, we can compute the empirical cumulative distribution function, or failure curve, as in Eq. 4.6.

$$
\begin{aligned}
F(t) &= \frac{1}{N_{events}} \sum_{i=1}^{N_{events}} \mathbf{1}_{\{t_{elapsed,i} \leq t\}}, \\
F(\Phi) &= \frac{1}{N_{events}} \sum_{i=1}^{N_{events}} \mathbf{1}_{\{\Phi_i \leq \Phi\}}, \\
\mathbf{1}_{\{P\}} &= \begin{cases} 1 & \text{if } P \\ 0 & \text{if } \neg P. \end{cases}
\end{aligned} \tag{4.6}
$$

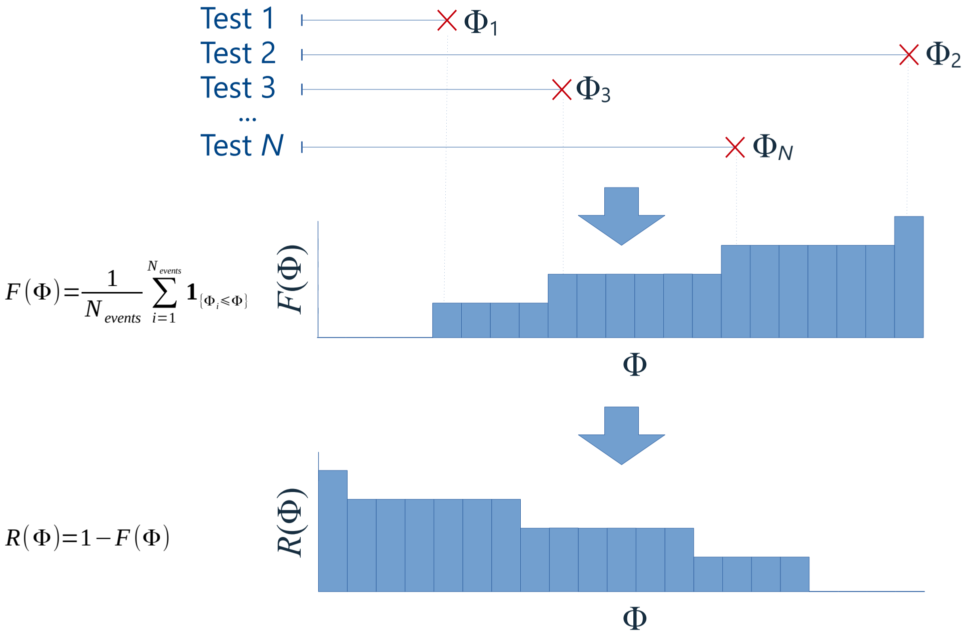
For a varying flux ϕ, the fluence Φ must be integrated over time and the Eq. 4.6 must be applied in terms of particle fluence $F(\Phi)$. Alternatively, for a stable flux in the experimental environment, known the average particle flux ϕ of the experiment we can simplify to $F(\Phi) = F(t) \times \phi$.

Being the reliability the reciprocal of failure, we can then compute the reliability curves $R(t)$ and $R(\Phi)$ as in Eq. 4.7.

$$
\begin{aligned}
R(t) &= 1 - F(t), \\
R(\Phi) &= 1 - F(\Phi).
\end{aligned} \tag{4.7}
$$

This is also depicted in the conceptual scheme of Fig. 4.1.

Figure 4.1 – Construction of the empirical reliability curve



Source: The Author

It is noteworthy that, known the particle flux ϕ_{env} in the targeted radiation environment, the reliability curve $R(\Phi)$ can be translated to $R(t)$ at that specific environment.

The analysis of the reliability curves allows the comparison of alternative implementations of the design in zones of higher reliability, for instance where $R(\Phi) \geq 90\%$ or $R(\Phi) \geq 99\%$, where the behavior of the design may be very different from its behavior at lower reliability zones characterized by the MTTF and its derived metrics.

Given a target reliability requirement, we can look at the reliability curves obtained from radiation experiments to identify the maximum time the design under test can be exposed to radiation while sustaining that target reliability. This metric we called as Mission Time (MT) for the given reliability requirement, where the time can also be expressed in terms of particle fluence.

The MT can be extracted directly from the empirical reliability curves $R(\Phi)$ or $R(t)$, obtained from radiation experiments, but it could also be obtained from the equation of a fitted model of probability distribution function. For instance, for a reliability curve described by a Weibull distribution parameterized by *scale* parameter α, a *shape* parameter β and an *offset* parameter γ in the form of Eq. 4.8, the MT sustaining a target reliability $R(t) \geq r$ could be estimated as in Eq. 4.9.

$$\begin{aligned} R(t) &= \exp\left[-\left(\frac{t-\gamma}{\alpha}\right)^{\beta}\right], && (4.8) \\ t &= \gamma + \alpha(-\ln r)^{\frac{1}{\beta}}. && (4.9) \end{aligned}$$

In the context of fault injection, Velazco et al. (2010) introduced the metric of error rate from fault injection $\tau_{failure}$ which, similar to the dynamic cross section, is computed as the ratio between the number of observed failure events N_{FI} and the number of faults injected Φ_{FI} (Eq. 4.10).

Known the device underlying static cross section σ_{SEU} (Eq. 2.2), and following the model from (VELAZCO et al., 2010), one can predict the dynamic cross section as $\sigma_{failure}$ as in Eq. 4.11. For generality we introduced a proportionality constant κ that accounts for other parameters such as fault injection density.

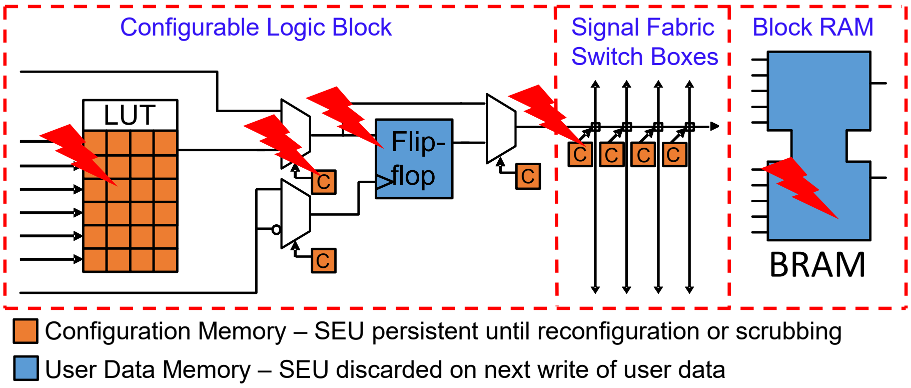
$$\begin{aligned} \tau_{failure} &= \frac{N_{FI}}{\Phi_{FI}}, && (4.10) \\ \sigma_{failure} &= \tau_{failure} \times \sigma_{SEU} \times \kappa. && (4.11) \end{aligned}$$

4.3 Single-event memory upsets in SRAM-based FPGAs

As seen along the Chapter 3, the configuration elements of the FPGA have many possible configurations and not all elements are required for every design. In modern SRAM-based FPGAs, the configuration memory ranges from a few megabits to over a gigabit and only the subset of these bits configuring the resources used by the specific application programmed into the FPGA is relevant.

When using standard grade commercial grade off-the-shelf (COTS) devices, the huge amount of SRAM cells behind every configurable elements in SRAM-based FPGAs makes it specially susceptible to SEUs (Fig. 4.2). In ASIC devices and other FPGA technologies, such as antifuse and Flash, only the end user data is under risk of memory upsets. Besides the user data, in SRAM-based FPGAs an upset in the configuration memory can effectively change the circuit implemented in the FPGA and cause a functional failure.

Figure 4.2 – SEUs on FPGA configuration memory and user data


Configurable Logic Block
Signal Fabric Switch Boxes
Block RAM
LUT
Flip-flop
C
BRAM
🟧 Configuration Memory – SEU persistent until reconfiguration or scrubbing
🟦 User Data Memory – SEU discarded on next write of user data

Source: Adapted from Kastensmidt et al. (2004)

The number of configuration bits effectively required to realize a design mapped in a certain FPGA device is called *essential bits* (XILINX, 2012a).

Disregarding eventual fabric short circuits on unused parts of the FPGA that could provoke over-current or other perturbations, an SEU in any non-essential bit has a lower likelihood of creating an error or a functional failure of the device.

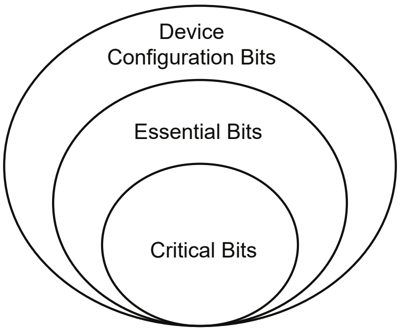
In fact, even a SEU in an essential bit may not cause a functional failure in the design implemented in the FPGA. This occurs for several reasons, including the logical masking, temporal masking, and the effective data input to the application.

For instance, some fault mitigation technique may be implemented in the design, such as triple modular redundancy (TMR) or Hamming codes, masking the effect of the SEU. A SEU can also damage some part of the design or some piece of user data after it was already used to compute the result. Also, some part of the design damaged by a SEU may not be activated for a given set of input data. In the case of machine learning applications, even errors on some output data element may not cause a functional failure. As an example, a data element in the output vector of a neural network may be incorrect but, as far as that error does not change the ranking of the true output element, the neural network classification may still be correct.

There may be, however, a few configuration memory bits for a specific application that will surely lead to a functional failure upon an SEU. These bits are known as *critical bits* and are an important factor for the design reliability.

For typical designs, it is expected that only 5% to 10% (XILINX, 2021b) of the essential bits will be critical bits. The critical bits can generally be seen as a subset of the essential bits, while the essential bits can be seen as a subset of the configuration memory bits. This relationship is represented in Fig. 4.3. However, there are situations where the cumulative effect of SEUs in non-critical bits can also cause functional failure.

Figure 4.3 – Relationship between configuration memory, essential and critical bits



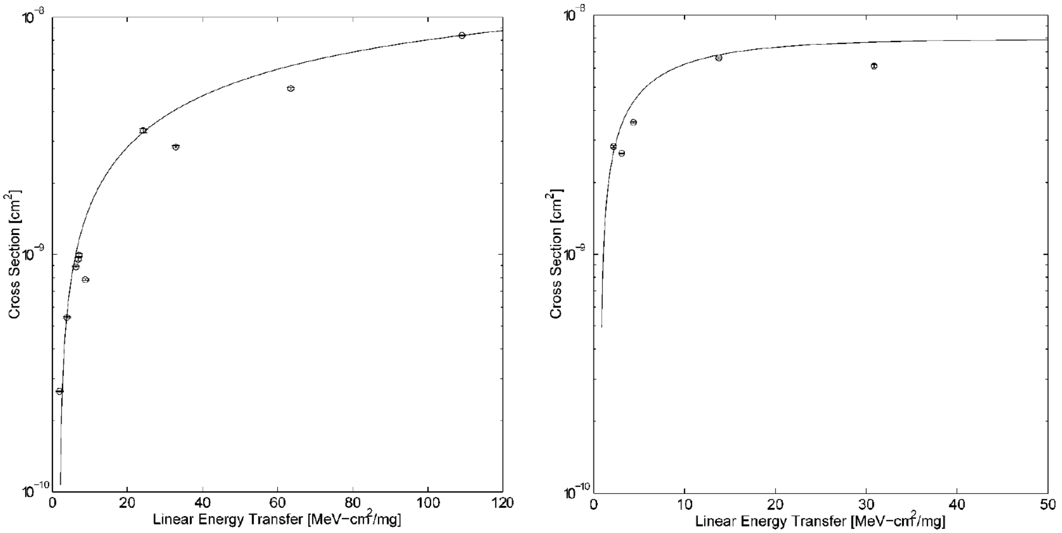
Source: Adapted from Xilinx (2012a)

As discussed in the previous section, the knowledge about the device underlying reliability metrics, such as the static cross section (σ_{SEU}), can be combined with other qualification and reliability metrics extracted at higher levels of the design to estimate the behavior in specific conditions of operation of specific environments.

There are in the literature several studies of the reliability of SRAM-based FPGA products from Xilinx, which are of main interest in this thesis.

Lee et al. (2014) present the static cross section for a Xilinx Kintex-7 FPGA. Although not exactly the same devices used here, the Kintex-7 is still inside the same family of Xilinx 7 Series and Zynq-7000 devices manufactured in $28\,\mathrm{nm}$ planar technology. The chart in Fig. 4.4 presents the static cross section for CRAM and BRAM under heavy ions at different LETs. The authors also demonstrate the occurrence of MCUs in the device. The same device is also tested by M. J. Wirthlin et al. (2014) in different facilities for neutrons, protons and heavy ions irradiation, presenting statistics about the occurrence of MBUs inside the same CRAM memory frame or in adjacent frames.

Figure 4.4 – Heavy-ions cross section for Xilinx Kintex-7

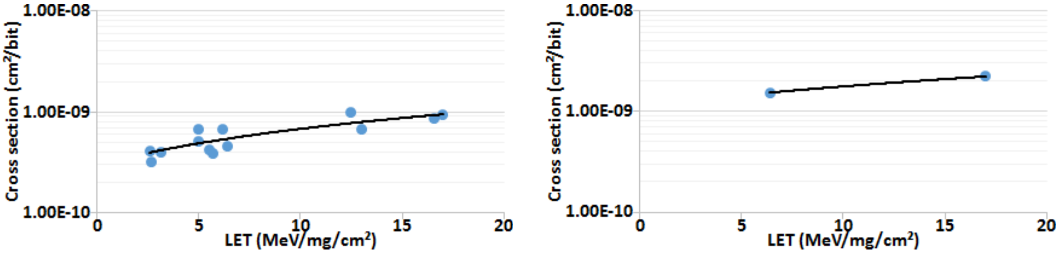


(a) SEU cross section at CRAM **(b)** SEU cross section at BRAM

Source: Lee et al. (2014)

The Xilinx Zynq-7000 (model Z020) was tested under heavy ions by Tambara et al. (2015a), presenting the static cross section for CRAM, BRAM and other FPGA elements. The authors also discuss the occurrences of MBUs. Figure 4.5 presents static cross section for CRAM and BRAM, which is comparable to Fig. 4.4. The same device is tested by Tambara et al. (2015b) under heavy ions and protons in another irradiation facility.

Figure 4.5 – Heavy ions cross section for Xilinx Zynq-7000 Z020

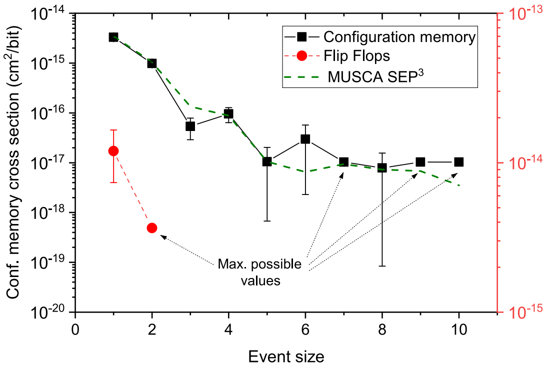


(a) SEU cross section at CRAM **(b)** SEU cross section at BRAM

Source: Tambara et al. (2015a)

Fabero et al. (2020) tested a Xilinx Artix-7 FPGA under fast neutrons, with detailed statistics and geometric description of MBU artifacts in the CRAM. Figure 4.6 presents the fast neutron cross section for SUB and MBU, with different number of bit-flips per event, aside static cross section for flip-flops elements in CLB blocks and predicted cross section by as modeling and simulation tool (MUSCA-SEP[3]).

Figure 4.6 – Neutron cross section for Xilinx Artix-7



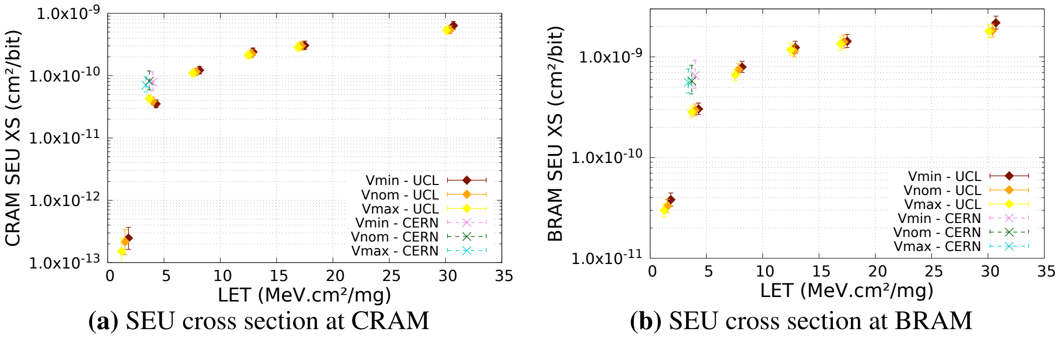
Source: Fabero et al. (2020)

Another device of interest for this thesis is the Xilinx UltraScale+, which is manufactured in $16\,\mathrm{nm}$ FinFET technology.

Pierre Maillard et al. (2017) tested a Xilinx UltraScale+ MPSoC device under proton and alpha particle, presenting static cross section and comparing with values from literature. The authors also compare the Xilinx UltraScale+ with the Xilinx Kintex-7 and, according to the authors, the UltraScale+ $16\,\mathrm{nm}$ FinFET device presents a reduction of around $40\times$ in the static cross section when compared with the previous 7 Series device $28\,\mathrm{nm}$ planar device. This improvement, however, is a consequence not only of the technology change but also of the modifications on the design by the manufacturer.

The Xilinx UltraScale+ MPSoC was also tested by Glorieux et al. (2018) under heavy ions. Fig. 4.7 presents static cross section for CRAM and BRAM under different conditions of operating voltage.

Figure 4.7 – Heavy ions cross section for Xilinx UltraScale+ MPSoC



(a) SEU cross section at CRAM

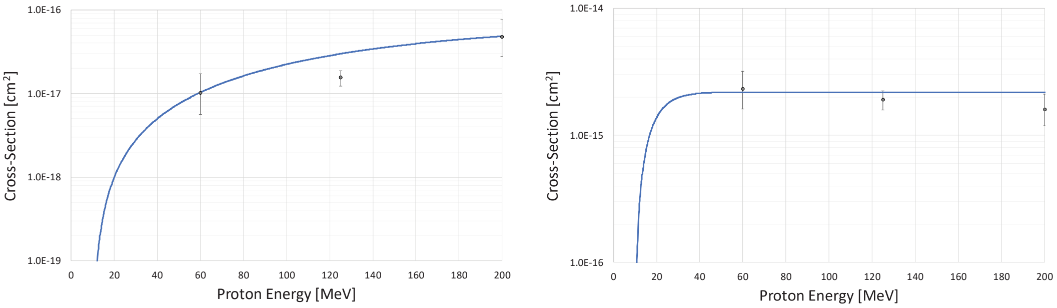
(b) SEU cross section at BRAM

Source: Glorieux et al. (2018)

Davis et al. (2019) present results from proton irradiation of a Xilinx UltraScale+ RFSoC device of the family. Fig. 4.8 shows static cross section for CRAM and BRAM in this device.

Azimi et al. (2022) also tested a Xilinx UltraScale+ MPSoC device under proton,

Figure 4.8 – Proton cross section for Xilinx UltraScale+ RFSoC

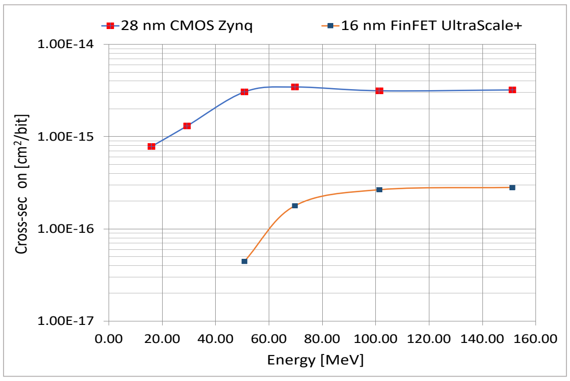


(a) SEU cross section at CRAM **(b)** SEU cross section at BRAM

Source: Davis et al. (2019)

together with a Xilinx Zynq-7000 Z020 for comparison. Figure 4.9 presents the CRAM static cross section for these two devices. Another contribution of Azimi et al. (2022) is to present an inventory and geometric description of MBU artifacts in the CRAM, although it still lacks statistical information about the frequency of occurrence of each MBU geometry.

Figure 4.9 – Proton cross section for UltraScale+ MPSoC and Xilinx Zynq-7000 Z020



Source: Azimi et al. (2022)

Finally, Chang Cai et al. (2019) presents results from tests in a UltraScale+ Kintex device using laser and heavy ions, with cross section for SBU and MBU with different numbers of bit-flips. The authors, however, do not provide the geometric description of the MBU artifacts and the frequency of occurrence of each artifact.

5 TEST AND FAULT INJECTION METHODOLOGY

5.1 Fault injection as a reliability assessment tool

The use of commercial grade SRAM-based FPGAs in critical cyber-physical requires the characterization of the reliability of the devices and applications, and, occasionally, the use of proper mitigation techniques for fault tolerance coping with reliability requirements on the target environment.

On the engineering process, that evaluation of reliability may be part of the test methodologies and one of the techniques used in this scenario is the fault injection.

Provoking faults in a controlled manner in memory cells of SRAM-based FPGAs to evaluate reliability can be done in several ways, such as accelerated irradiation on particle generators, accelerators and radiation sources, use of laser beams, or direct manipulation of the memory content emulating the radiation effects.

Fault injection can be used to verify the resilience against errors as well as safe behavior in case of errors and is required. For instance, fault injection is present in the IEC 61508 (IEC, 2010) international standard for functional safety on safety related systems and the ISO 26262 (ISO, 2018) international standard for functional safety on road vehicles (ISLAM et al., 2013; VEDDER, 2015).

For SRAM-based FPGAs, there are broadly three main branches of fault injection techniques:

- **Physical fault injection** use laser beams, radiation sources or particle accelerators to induce errors in real hardware; some drawbacks of this approach are high cost and availability of test facilities; this approach can also cause permanent damages to the device; depending on the radiation flux or device area this approach may require a long test time in controlled environments.

- **Emulation-based fault injection** also uses real hardware but explore debug structures and test pins to introduce errors in the device, for instance emulating the effects of radiation; emulation-based fault injection can overcome the cost and availability drawbacks of hardware-based fault injection because it does not require complex facilities; as the application run in the real hardware at normal speed, there is a small impact on test time while the fault rate can be controlled by the tester making this approach potentially faster than hardware-based.

- **Simulation-based fault injection** use a model of the hardware or circuit to simulate the influence of faults, offering detailed observability of its effects; as it can be executed in software only, simulation may have a lower cost than the previous approach and can be used even in very early stages of engineering in the absence of real hardware; however a major drawback of this approach is the very low speed.

In this thesis, we focus on the emulated fault injection on SRAM-based FPGAs, more specifically the Xilinx 7 Series and UltraScale+ families of devices, that allows the direct manipulation of SRAM memory through communication interfaces originally designed for test and configuration of the FPGA.

Another relevant aspect of fault injection is the definition of the fault space, can be seen as thee main dimensions:

- **Injection location** refers to the positions in the circuit, e.g. wires, or device memory, e.g. bits or bit groups, where the fault will be injected; in SRAM-based FPGAs it includes notably all the configuration memory bits behind the FPGA configurable elements and all the block memory block bits holding end user data.
- **Fault type** include the kind of effect, for example stuck-at zero, stuck-at one or bit-flip, if it is single or multiple bit-flips, or even representative abstraction of physical phenomena that can change the fault susceptibility or profile, such as operation at lower voltages or emulation of a particular radiation environment with specific particle types and energies.
- **Injection time** represents each point in time, during the application processing cycle, where the fault can occur and affect the device; this aspect is specially relevant in the case of end user data since faults may be subject to temporal masking; in SRAM-based FPGAs without any healing technique applied to the configuration memory the faults are mostly persistent and the injection time becomes less relevant as persistent latent faults may be evidenced in the next processing cycles.

Emulation and simulation-based techniques give a good control of these aspects but they are hard to control in hardware-based techniques, for instance using radiation.

Four desired properties are enumerated by Quinn et al. (2015) for emulation-based fault injection frameworks for SRAM-based FPGAs:

- Have access to the internal memory element where the faults will be injected.

- Capacity to stimulate or execute the application or circuit implemented in the device.
- Ability to observe the processing cycle results and detect the occurrence of errors or functional failures.
- Capacity to clear faults and return the device to a known good state.

5.2 Emulation-based fault injection for SRAM-based FPGAs

Antoni et al. (2000) report the implementation of fault injection in Xilinx XC4000 and Virtex devices by modifying the bitstream file before loading it into the FPGA. On Xilinx Virtex devices, where partial reconfiguration is supported, the bitstream file can contain the smaller memory image for partial reconfiguration of the regions of interest while on previous devices the whole FPGA must be reconfigured, hence with a larger bitstream file and longer communication time.

Lima et al. (2001) report results of fault injection on Xilinx Virtex devices using a fault injector developed by the Los Alamos National Laboratory. In this implementation faults were injected into the configuration memory of the FPGA from an external source using the Xilinx MultiLINX cable (JOHNSON et al., 2002).

A fault injection testbed for Xilinx Virtex devices was presented by Johnson et al. (2003), implemented by researchers from Brigham Young University (BYU) and Los Alamos National Laboratory. This implementation was designed to manipulate the bitstream introducing bit-flips to simulate the effect of SEUs in the configuration memory of the FPGA and explored the feature of partial reconfiguration of the FPGA by programming only the configuration memory frame modified by the fault injection (JOHNSON et al., 2002). In this implementation two FPGA devices were programmed with the same design, only the golden bitstream was programmed with the clean bitstream while the device under test was programmed with the faulty bitstream. The functional failure diagnosis was obtained by off-chip comparison of the results provided by the two devices given the same stimulus vector.

Another implementation of fault injector at BYU is the XRTC Virtex-5 Fault Injector (XRTC-V5FI) (HARWARD et al., 2015). This implementation used one FPGA as fault injector that manipulated the configuration memory of a second FPGA, the device under test, using the Xilinx SelectMAP configuration access parallel port. The authors reports an increase in the fault injection speed by using the SelectMAP port but it is not clear if this improvement is due to programming at a single frame level, instead of programming the whole bitstream, or only due to the higher speed of the parallel port.

One of the last implementations of fault injection on BYU is the TURTLE fault injection platform (THURLOW et al., 2019) that explores the same principle of partial reconfiguration and was implemented to Xilinx 7 Series product family, also using two FPGAs and off-chip diagnosis. Although partial reconfiguration could allow the implementation of both the design under test and golden design in the same physical device, the authors indicates that the implementation in two independent FPGA and using off-chip diagnosis have the advantage to avoid any risk of a fault injection in the design under test disrupt the test instrumentation and control logic.

Another approach is presented by Sterpone et al. (2007) exploring the Xilinx Internal Configuration Access Port (ICAP) of the Xilinx Virtex-II to modify the contents of the configuration memory from within the FPGA.

Villalta et al. (2014) follow a similar approach by using the Xilinx Processor Configuration Access Port (PCAP) of the Xilinx Zynq-7000 device to modify the contents of the configuration memory. The PCAP interface is also used by (GOMEZ-CORNEJO et al., 2017) to modify the contents of the memory blocks (BRAM). The PCAP interface, however, is available only in the Xilinx Zynq SoC devices, limiting the range of the devices where the fault injection can be used.

Napoles et al. (2007) presents the FT-UNSHADES system (Fault Tolerant - UNiversity of Sevilla HArdware DEbugging System) for fault injection in Xilinx Virtex-II devices using the Xilinx SelectMAP configuration access parallel port. Two FPGA devices are used in this implementation, one used to control the fault injection following commands from the campaign coordinating computer and to communicate at high-speed SelectMAP interface with the second FPGA where the design under test is implemented. This implementation used on-chip comparison to diagnosis of functional failures, with both the golden design and design under test implemented into the same FPGA. A characteristic reported to this fault injection is the synchronization and timing functions that supports injection of fault in specific points in time and also the injection of multiple bit-flips associated with the same SEU (NAPOLES et al., 2008). An improved version FT-UNSHADES2 is presented by Mogollon et al. (2011), targeting the Xilinx Virtex-5 devices and still using the SelectMAP interface in the two FPGA scheme.

Alderighi et al. (2007) present the FLIPPER fault injection platform targeting Xilinx Virtex II devices. This fault injection platform also uses two FPGAs, one managing the overall fault injection campaign and other implementing the design under test. The tool can inject faults sequentially, at random positions or at arbitrary positions indicated by the user. The tool can also inject single bit-flips or two adjacent bit-flips in a memory frame. A given set of test vectors is exercised after each fault injected. Faults injected can be accumulated allowing to obtain the probability distribution of the number of randomly injected faults to cause a functional failure.

5.3 UFRGS fault injection tool for Xilinx FPGAs

The emulation-based fault injection engine adopted in this thesis was developed in-house at the Universidade Federal do Rio Grande do Sul (UFRGS, Federal University of Rio Grande do Sul) and is built around the Xilinx Internal Configuration Access Port (ICAP).

This fault injector have a long line of development. One implementation of fault injector is presented by Nazar et al. (2012), targeting Xilinx Virtex-5 devices. Both the design under test and fault injection instrumentation are implemented inside the same FPGA with the system composed of five modules:

- The design under test, which interfaces with a test vector.
- The design controller that contains parameter for the design under test, the set of test vectors and the golden results.
- The reporting module that transmits diagnose information to the campaign coordinating computer.
- The fault injection module that communicates with the Xilinx ICAP interface to read and write the configuration memory frames.
- The system controller that coordinates the overall campaign process of fault injection, diagnosis of the design under test, reporting of diagnosis and fault clean-up.

To avoid risk of faults being injected over the fault injector itself, the placement of both the fault injection module, the design under test and other modules is controlled by floorplanning constraints at the synthesis tool. The author defines the *area under test*

(AUT) as floorplan region of interest for fault injection and the logical modules of the *circuit under test* (CUT) must be contained into the AUT.

This implementation injected single faults in random locations of the AUT, with the fault location randomized inside the fault injection instrumentation at the FPGA by a pseudo-random number generator.

The fault injection process executed autonomously for a desired amount of faults to be injected while reporting the processing results of the design under test. The faults injected were not accumulated over time and, consequently, the information obtained was a sample of the design sensitive bits and not a probability distribution for accumulated faults.

Leipnitz et al. (2016) adapted the Xilinx Virtex-5 fault injector from Nazar et al. (2012) in another configuration with the FPGA board as a PCI-Express peripheral of the campaign controlling computer, therefore achieving higher speeds for data transfer of configuration and diagnosis. This implementation simplified the design controller module and eliminated the reporting module with the host computer communicating directly to the design controller through the high speed of the PCI-Express interface to send the stimulus vectors and collect processing results.

Tarrillo et al. (2014) presented another implementation of the fault injector for Xilinx Virtex-5 devices with some of the fault injection instrumentation implemented by a Xilinx PicoBlaze microcontroller. In this implementation, instead of using a pseudo-random number generator to define the fault locations, the fault injection followed a database of predefined fault locations obtained from radiation experiments and stored in an external auxiliary memory. Another implementation from Tarrillo et al. (2015), still supported by the Xilinx PicoBlaze microcontroller, had as alternative both using the external fault location database or injecting faults on randomized memory addresses obtained from a pseudo-random number generator.

Tonfat et al. (2016) adapted the fault injection mechanism from Tarrillo et al. (2015) to a Xilinx Artix-7 device to inject faults exhaustively on the whole configuration memory of a region of interest in the device. In this implementation there is no system controller module inside the FPGA coordinating the fault injection campaign. Instead, the fault injection module communicates with a campaign coordinating computer through a serial communication interfaces and wait the user commands for fault injection containing the fault location.

This implementation used on-chip comparison to diagnose functional failures with the design under test and a design controller implemented in the same FPGA. For each fault injection command from the host computer, the fault injector module activated the design controller to exercise the design, collect the processing results and compare with the golden results to obtain a diagnose that was returned to the fault injector module and, ultimately, transmitted blindly to the host computer. It was up to the host computer to interpret the diagnosis information from the design controller.

The implementation from Tonfat et al. (2016), however, was used only to inject faults sequentially covering the whole region of interest in the FPGA floorplan with each fault injected being cleaned-up before injecting another fault. Another characteristic was that faults were injected before the activation of the design controller and, consequently, before the beginning of the processing cycle of the design under test.

Part II

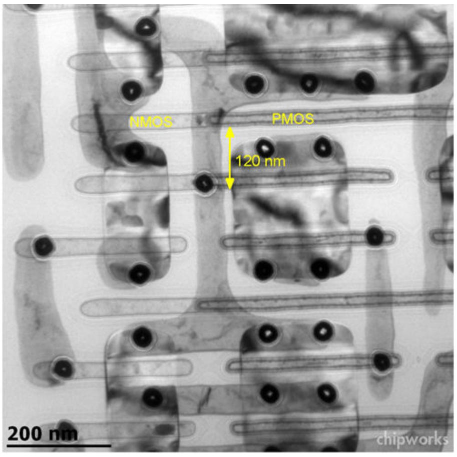
Characterization of the Xilinx 7 Series devices

6 DISCOVERING XILINX 7 SERIES SRAM-BASED FPGA

6.1 Constructive structure of Xilinx 7 Series devices

The FPGA devices investigated in this thesis are the Xilinx Zynq-7000 SoC and 7 Series manufactured in 28 nm technology by the TSMC HPL (high-performance, low-power) process (Fig. 6.1) that uses HKMG (high-k metal gate) planar transistors (XILINX, 2021h). This technology is relatively mature, available since 2011 (TSMC, 2018), with newer devices of this manufacturer using 14 nm (XILINX, 2015c) and 7 nm (XILINX, 2020b) technologies with FinFET transistors.

Figure 6.1 – Electronic microscopy of Xilinx 7 Series Kintex-7 device



Source: Dixon-Warren (2012)

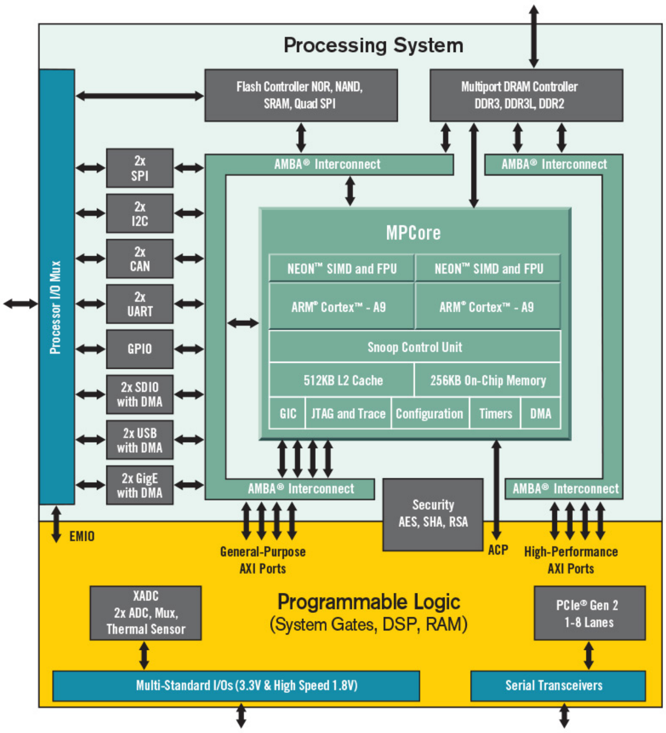
The Xilinx Zynq-7000 devices can be divided in two main parts (Fig. 6.2) of processing system (PS) and programmable logic (PL).

The processing system (PS) includes several communication peripherals, memory interfaces, and application processing units (APU), which are Arm Cortex-A9 processors based on ARMv7-A 32 bits core. The programmable logic is simply the SRAM-based FPGA as found in other devices of the Xilinx 7 Series product family.

The Zynq-7000 device models XC7Z020 and XC7Z030 are of special interest here because they were tested under radiation (Section 8.1, 8.2 and 8.3) and laser (Section 7.2), respectively. According to the manufacturer (XILINX, 2021h), the model XC7Z020 has the programmable logic (PL) based on the same logic of Xilinx Artix-7 FPGAs while the XC7Z030 is based on Xilinx Kintex-7.

Different devices based on the Xilinx Advanced Silicon Modular Block (ASMBL) technology share the implementation for the different FPGA columns (Fig. 6.3a), includ-

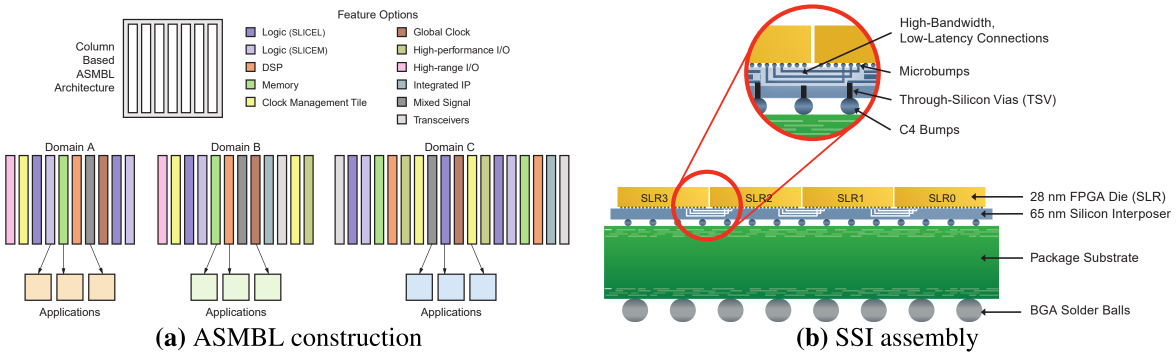
Figure 6.2 – Xilinx Zynq-7000



Source: Xilinx (2018d)

ing devices in the stacked silicon interconnect (SSI) assembly (Fig. 6.3b). Figure 6.4 shows a representation of the internal organization of some of these columns (note that the GRM block, general routing, refers to the switch matrices as discussed in Section 3.2).

Figure 6.3 – Xilinx 7 Series ASMBL construction and stacked SSI assembly

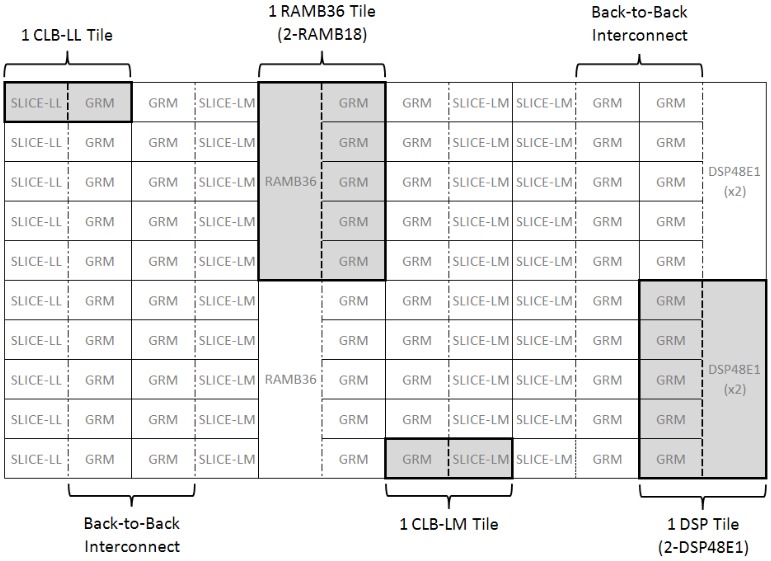


(a) ASMBL construction **(b)** SSI assembly

Source: Xilinx (2016, 2012c)

Based on these characteristics of the Xilinx ASMBL constructive technology, it is expected that geometry information for the different FPGA columns of CLB, DSP and BRAM extracted from Zynq-7000 XC7Z030 shall also apply to Zynq-7000 XC7Z020, Artix-7 and Kintex-7 devices.

Figure 6.4 – Different kinds of Xilinx 7 Series building blocks



Source: Xilinx (2020a)

6.2 Logical organization of the device

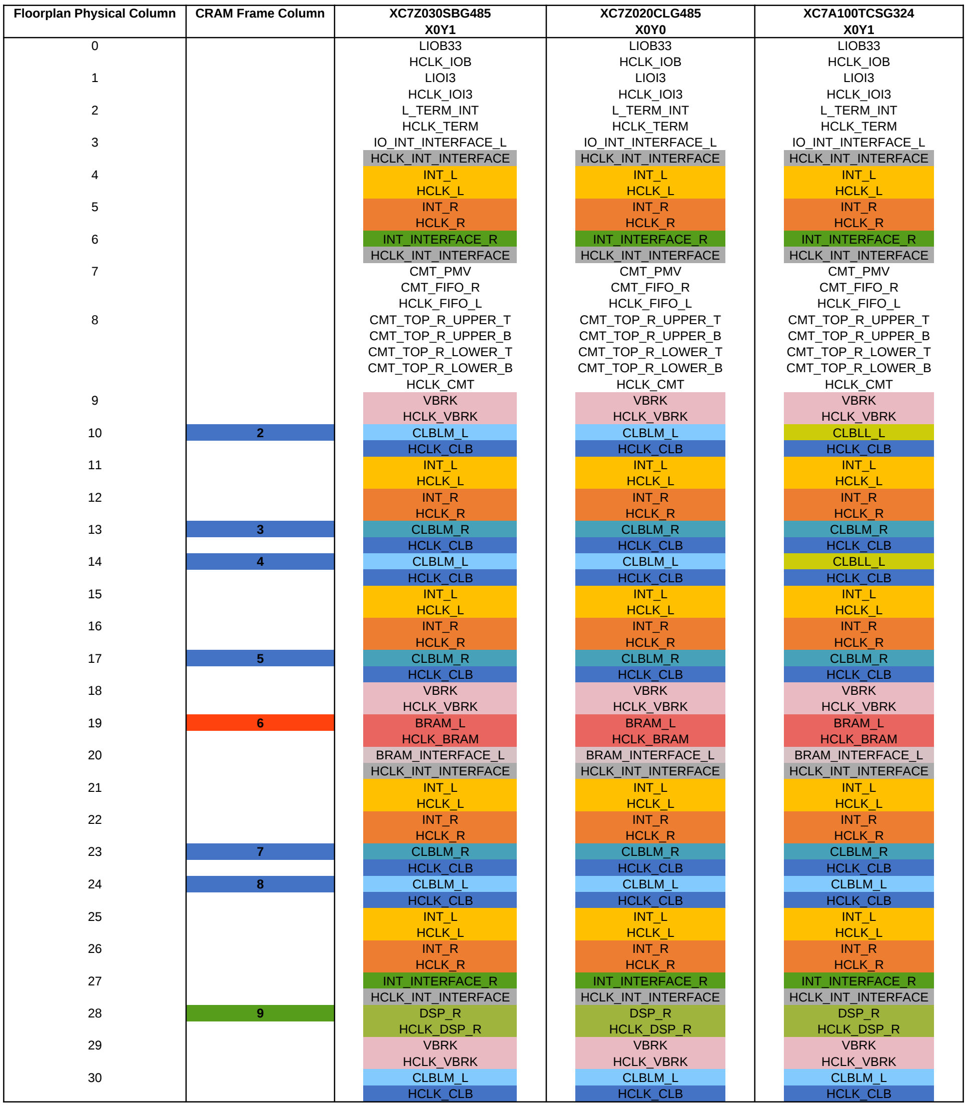
The information about Xilinx Zynq-7000 and 7 Series devices can be obtained from the Xilinx Vivado synthesis tool (XILINX, 2015b) while physical dimensions of the die can be obtained from the packaging documentation (XILINX, 2021a).

As an example, the table in Fig. 6.5 presents the content of the first physical columns for three devices in the same clock region. Despite two devices being a SoC, at the clock region presented we have only PL columns, including I/O block (IOB), clock management (CMT), clock distribution (HCLK), logic blocks (CLB), memory blocks (BRAM) and DSP blocks (DSP).

Although the floorplan fragment presented in the example of Fig. 6.5 is quite similar for the three devices, it is worth noting that for the Artix-7 XC7A100T device the floorplan columns 10 and 14 had the logic blocks holding memory slices (CLBLM, SLICEM) replaced by logic blocks for logic only slices (CLBLL, SLICEL), meaning that these Artix-7 XC7A100T columns cannot be used to implement LUTRAM memory blocks as discussed in Section 3.3 and may have a smaller circuit area. Besides that, the three devices have a spacer (VBRK) in columns 9, 18 and 29 suggesting that these spacer columns do not have, necessarily, the same physical width at the die.

The study of the Xilinx 7 Series FPGA structure can lead to information about the geometric proportions and physical location of the configuration memory (CRAM) and block memory (BRAM) in the die. With this, and considering the nature of the ASMBL

Figure 6.5 – Fragment of the floorplan content in different Xilinx devices



Floorplan Physical Column	CRAM Frame Column	XC7Z030SBG485 X0Y1	XC7Z020CLG485 X0Y0	XC7A100TCSG324 X0Y1
0		LIOB33 HCLK_IOB	LIOB33 HCLK_IOB	LIOB33 HCLK_IOB
1		LIOI3 HCLK_IOI3	LIOI3 HCLK_IOI3	LIOI3 HCLK_IOI3
2		L_TERM_INT HCLK_TERM	L_TERM_INT HCLK_TERM	L_TERM_INT HCLK_TERM
3		IO_INT_INTERFACE_L HCLK_INT_INTERFACE	IO_INT_INTERFACE_L HCLK_INT_INTERFACE	IO_INT_INTERFACE_L HCLK_INT_INTERFACE
4		INT_L HCLK_L	INT_L HCLK_L	INT_L HCLK_L
5		INT_R HCLK_R	INT_R HCLK_R	INT_R HCLK_R
6		INT_INTERFACE_R HCLK_INT_INTERFACE	INT_INTERFACE_R HCLK_INT_INTERFACE	INT_INTERFACE_R HCLK_INT_INTERFACE
7		CMT_PMV CMT_FIFO_R HCLK_FIFO_L	CMT_PMV CMT_FIFO_R HCLK_FIFO_L	CMT_PMV CMT_FIFO_R HCLK_FIFO_L
8		CMT_TOP_R_UPPER_T CMT_TOP_R_UPPER_B CMT_TOP_R_LOWER_T CMT_TOP_R_LOWER_B HCLK_CMT	CMT_TOP_R_UPPER_T CMT_TOP_R_UPPER_B CMT_TOP_R_LOWER_T CMT_TOP_R_LOWER_B HCLK_CMT	CMT_TOP_R_UPPER_T CMT_TOP_R_UPPER_B CMT_TOP_R_LOWER_T CMT_TOP_R_LOWER_B HCLK_CMT
9		VBRK HCLK_VBRK	VBRK HCLK_VBRK	VBRK HCLK_VBRK
10	2	CLBLM_L HCLK_CLB	CLBLM_L HCLK_CLB	CLBLL_L HCLK_CLB
11		INT_L HCLK_L	INT_L HCLK_L	INT_L HCLK_L
12		INT_R HCLK_R	INT_R HCLK_R	INT_R HCLK_R
13	3	CLBLM_R HCLK_CLB	CLBLM_R HCLK_CLB	CLBLM_R HCLK_CLB
14	4	CLBLM_L HCLK_CLB	CLBLM_L HCLK_CLB	CLBLL_L HCLK_CLB
15		INT_L HCLK_L	INT_L HCLK_L	INT_L HCLK_L
16		INT_R HCLK_R	INT_R HCLK_R	INT_R HCLK_R
17	5	CLBLM_R HCLK_CLB	CLBLM_R HCLK_CLB	CLBLM_R HCLK_CLB
18		VBRK HCLK_VBRK	VBRK HCLK_VBRK	VBRK HCLK_VBRK
19	6	BRAM_L HCLK_BRAM	BRAM_L HCLK_BRAM	BRAM_L HCLK_BRAM
20		BRAM_INTERFACE_L HCLK_INT_INTERFACE	BRAM_INTERFACE_L HCLK_INT_INTERFACE	BRAM_INTERFACE_L HCLK_INT_INTERFACE
21		INT_L HCLK_L	INT_L HCLK_L	INT_L HCLK_L
22		INT_R HCLK_R	INT_R HCLK_R	INT_R HCLK_R
23	7	CLBLM_R HCLK_CLB	CLBLM_R HCLK_CLB	CLBLM_R HCLK_CLB
24	8	CLBLM_L HCLK_CLB	CLBLM_L HCLK_CLB	CLBLM_L HCLK_CLB
25		INT_L HCLK_L	INT_L HCLK_L	INT_L HCLK_L
26		INT_R HCLK_R	INT_R HCLK_R	INT_R HCLK_R
27		INT_INTERFACE_R HCLK_INT_INTERFACE	INT_INTERFACE_R HCLK_INT_INTERFACE	INT_INTERFACE_R HCLK_INT_INTERFACE
28	9	DSP_R HCLK_DSP_R	DSP_R HCLK_DSP_R	DSP_R HCLK_DSP_R
29		VBRK HCLK_VBRK	VBRK HCLK_VBRK	VBRK HCLK_VBRK
30		CLBLM_L HCLK_CLB	CLBLM_L HCLK_CLB	CLBLM_L HCLK_CLB

Source: The Author

construction, we expect that gathering geometric information for a determined region of the die, for instance with laser or heavy ions microbeam, we could apply that dimensions and internal organization of the bits to all other columns of the same time at the FPGA floorplan.

Although the FPGA devices contains simple wiring columns (FEEDTHRU) and spacer columns (VBRK and VFRAME), with potentially variable widths, its is plausible that the knowledge about the memory array geometry, density and location of the bits in CLB, BRAM and DSP columns be enough to ensure a spatial distribution uniformity at the emulation-based fault injection and to verify if occasional differences in density of location of the bits have any impact at the reliability evaluation of the designs under fault injection.

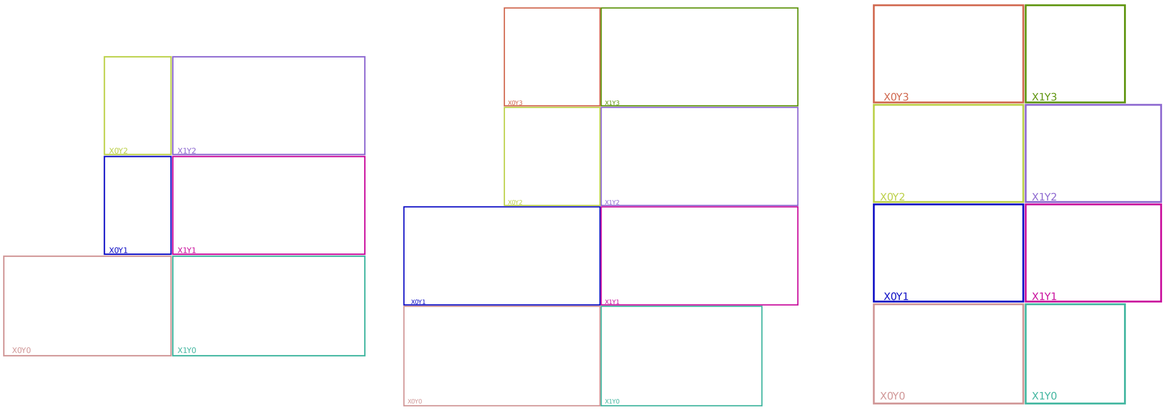
6.3 Organization of the configuration and user data memory

As mentioned in Section 3.6, the configuration (CRAM) and user data memory (BRAM) in Xilinx 7 Series devices are organized in frames of 3,232 bits grouped in rows and columns (XILINX, 2018b).

The frame is arranged vertically and have the same height of the row and the column. Different device model have different number of rows.

The Zynq-7000 XC7Z020 (Fig. 6.6a), for instance, has three rows while the Zynq-7000 XC7Z030 (Fig. 6.6a) and Artix-7 A100T (Fig. 6.6a) have four rows. Considering the physical dimensions of the die (XILINX, 2021a,g) we can estimate the height of the frame, column and row as $2.41\,\mathrm{mm}$.

Figure 6.6 – Organization of the PL in different Xilinx devices



(a) Zynq-7000 XC7Z020 **(b)** Zynq-7000 XC7Z030 **(c)** Artix-7 XC7A100T

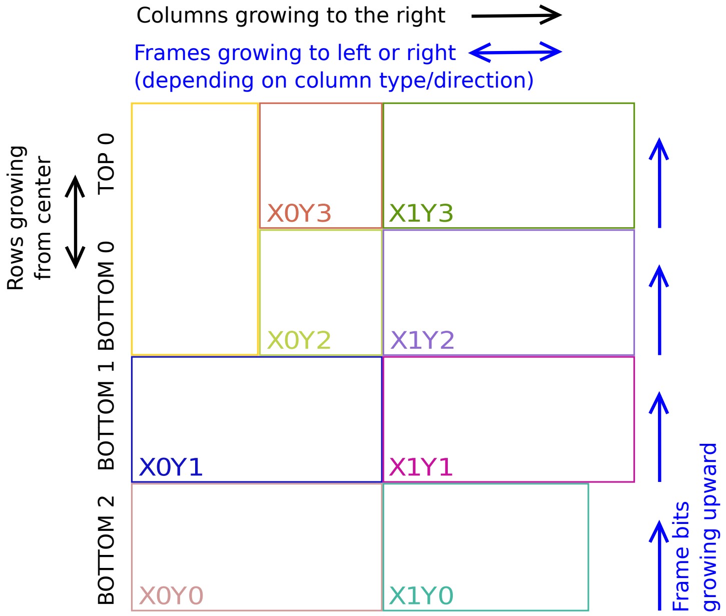
Source: Adapted from Xilinx Vivado synthesis tool

In Xilinx 7 Series devices the device rows are organized in two regions, TOP and BOTTOM. At the Artix-7 XC7A100T, for instance, the divider of TOP and BOTTOM occurs in the middle of the die, the two first rows being TOP and the two last rows being BOTTOM. On Zynq-7000 XC7Z020 and XC7Z030 devices this division does not occur in the middle of the die but in the middle of the PS block. For these two devices the first row is TOP and the remaining rows are BOTTOM (Fig. 6.7).

Also in Xilinx 7 Series the rows are numbered sequentially from the position of the divider between TOP and BOTTOM with TOP rows numbered upwards and BOTTOM rows numbered downwards. In Zynq-7000 XC7Z030 the row Y3 in the floorplan of Fig. 6.7 is the row TOP 0 in the frame address while rows Y2, Y1, and Y0 are, respectively, BOTTOM 0, 1 and 2.

Each frame in configuration (CRAM) and data (BRAM) memories is identified by

Figure 6.7 – Rows, columns, frames and bits in Zynq-7000 XC7Z030


Columns growing to the right
Frames growing to left or right
(depending on column type/direction)
Rows growing from center
TOP 0
BOTTOM 0
BOTTOM 1
BOTTOM 2
X0Y3
X1Y3
X0Y2
X1Y2
X0Y1
X1Y1
X0Y0
X1Y0
Frame bits growing upward

Source: The Author

a 26 bits address composed by the type of memory, the identification of the region (TOP or BOTTOM), the number of the row inside the region, the column number (major) and the frame number inside the column (minor), according to the Table 6.1. This same frame addressing is used in the UFRGS fault injector (Section 5.3).

Columns containing the signal fabric interconnects (INT) and configuration for CLB blocks are typically 36 frames wide. In this configuration memory columns contains routing configuration, LUT data, initial and reset values for flip-flops, among other configurations to the logic block. Columns containing routing and configuration for DSP and BRAM are typically 28 frames wide. The BRAM columns holding user data have typically 128 frames.

The frame is a vertical vector of 3,232 bits. In some devices it was found frames that do not have all the bits but even in those cases there was space reserved for the 3,232 bits in the bitstream.

From the static tests using laser fault injection presented in Section 7.2 it was observed that frame grows upward, with the bit position 0 is at the bottom of the floorplan row and the bit 3,231 at the top of the row, as indicated in Fig. 6.7.

Table 6.1 – Composition of the frame address in Xilinx 7 Series

Segment	***Address bits***	***Description***
Type of block	25:23	Typical values are 001b for BRAM content, 000b for CRAM including configuration and routing for BRAM, CLB and DSP, and 010b for special CLB configuration.
Region (TOP/BOTTOM)	22	TOP rows are identified by value 0 and BOTTOM by 1.
Row	21:17	Row number inside region, starting from 0, TOP rows numbered upwards and BOTTOM rows numbered downwards.
Column (major)	16:7	Column number inside the row, starting from 0 at the leftmost position and growing sequentially to the right.
Frame (minor)	6:0	Frame number inside the column, starting from 0 with growing direction depending on the column direction (see Section 7.3).

Source: Adapted from Xilinx (2018b)

6.4 Programming interfaces and access to configuration data

As mentioned in Section 3.6 and Section 5.3, a complete image of the configuration (CRAM) and data memory (BRAM) can be written or read from the FPGA in the form of a bitstream, using different communication interfaces such as the internal ICAP and PCAP ports and the external JTAG and SelectMAP ports (Fig. 6.8).

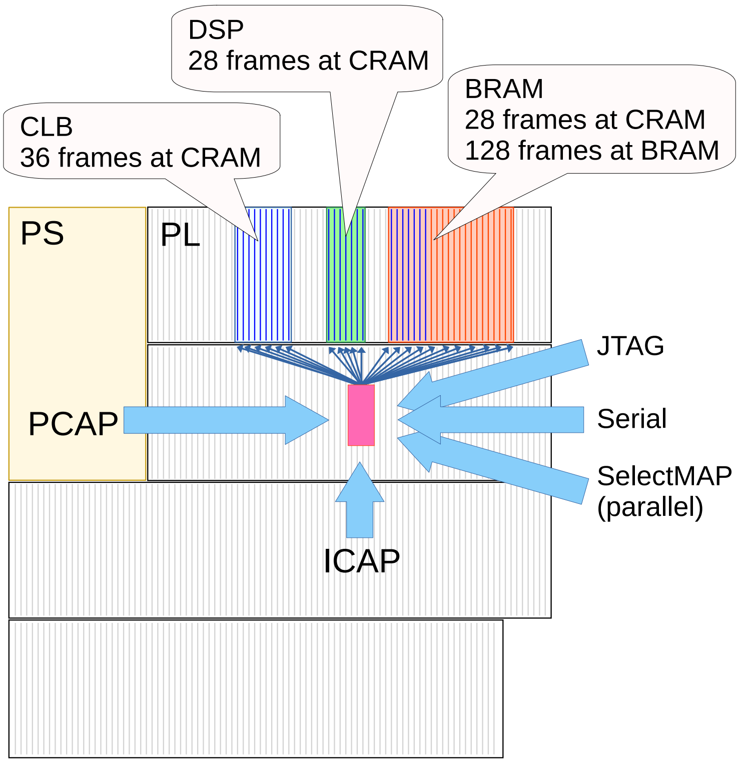
These different interfaces can operate in widths from 1 bit up to 32 bits in the case of SelectMAP. Some of these interfaces are designed to load the bitstream during the device power-up or during partial reconfiguration while others, like the JTAG, are designed primarily for testing purposes. The internal and external configuration port can also be used to correct SEUs by periodic memory scrubbing.

In Xilinx Zynq-7000 devices, the PCAP port allows software running in the Arm Cortex-A9 microprocessors to read and write configuration data. This is the mechanism used by the bootloader software to load the bitstream into the FPGA during power-up but can also be used for partial reconfiguration the to manipulated single configuration data frames.

Both Xilinx Zynq-7000 and 7 Series devices have an internal port, the ICAP, allowing circuits implemented in the FPGA to manipulate the memory of the FPGA itself.

The PCAP and ICAP have the same constraint that minimal data unit to read or write is a complete frame of 3,232 bits that are read or written as an array of 101 words

Figure 6.8 – Hardware access to FPGA configuration data



Source: The Author

of 32 bits.

Xilinx design tools can be used to read and write the bitstream using cable adapters to the programming interfaces discussed previously, for instance the JTAG port.

On some development board that have an external Flash to store the bitstream this Flash device can be programmed with the Xilinx `program_flash` tool and the bitstream can also be loaded directly from the computer to the FPGA using Xilinx `xsdb` and `xmd` tools.

The graphical user interface of Xilinx Vivado tool can also be used to program the board external Flash memory and load or readback the bitstream on the FPGA.

7 DETAILED DEVICE CARTOGRAPHY

7.1 Mapping floorplan to bitstream file

For generality, in this thesis the readback of the bitstream was done with the Xilinx Vivado tool in batch mode using the JTAG port of the development boards.

To verify the bitstream, comparing the original bitstream file with the bitstream read from the FPGA, was done with a tool named `readback_compare` developed by the same team that developed the UFRGS fault injector (Section 5.3). In this thesis this tool was extensively modified to allow different file formats for bitstream and mask files and to allow to enforce mask starting from arbitrary positions defined by the user.

The configuration and data frames for CRAM and BRAM are positioned sequentially inside the bitstream file. The Xilinx synthesis tool can generate a report of the logical location of some FPGA elements, such as flip-flops and BRAM bits, instantiated at the design. A sample of this logic location report is presented in (Fig. 7.1). Despite not informing detailed location for routing and all elements of CLB and DSP, the logic location file provides exact location of every data memory bit in memory blocks (BRAM), which are the BRAM frames for BRAM (Section 3.6).

Figure 7.1 – Fragment of logic location report (`.ll` file)

```
...
Bit 22358275 0x0042141f   2531 Block=SLICE_X58Y89 Latch=AQ
         Net=ps7/design_1_i/axi_smc/inst/m00_nodes/m00_aw_node/inst/
         inst_mi_handler/areset_r
...
Bit 22361839 0x00421420   2863 Block=SLICE_X58Y94 Ram=A:55
Bit 22361840 0x00421420   2864 Block=SLICE_X58Y94 Ram=B:1
Bit 22361841 0x00421420   2865 Block=SLICE_X58Y94 Ram=B:3
...
Bit 44925038 0x00c40002    238 Block=RAMB36_X0Y0 Ram=B:BIT695
Bit 44925039 0x00c40002    239 Block=RAMB36_X0Y0 Ram=B:BIT759
Bit 44925040 0x00c40002    240 Block=RAMB36_X0Y0 Ram=B:PARBIT65
Bit 44925041 0x00c40002    241 Block=RAMB36_X0Y0 Ram=B:PARBIT73
...
```

Source: The Author

Although the logical mapping for BRAM is clearly defined in the logic location file, other researchers, such as Gomez-Cornejo et al. (2017), also investigated the internal organization of the BRAM. The main motivation, in those case, is to determine the mapping between bits positions in BRAM and in data words in the user space since adjacent bits in user data word may not be adjacent in the logical addressing of the frames or adjacent in the physical location in the die.

In this study of the geometry of the configuration data in the physical device it is expected to determine the mapping of the bits in three different coordinate systems:

- The (x,y) position, in micrometer, relative to the die surface.
- The offset of that bits in the bitstream file.
- The bit and frame addresses (Table 6.1) to the configuration memory and logical floorplan.

The strategy adopted to map between the bitstream file and the bit and frame addresses consisted in three steps that were repeated several times in different regions of the device floorplan:

1. Change the content of the memory inside the FPGA in a delimited region.
2. Read back the modified bitstream from the FPGA.
3. Compare the readback bitstream with the original bitstream file.

The bitstream comparison allowed to determine the offset of the bit and frame address inside the bitstream file.

In the case of the configuration memory (CRAM), the fault injector was used to change the contents of the memory in every column. Since the logic location report from Xilinx Vivado tool already provide the offset and frame address for every BRAM bit in the design, the complete map was obtained by synthesizing a circuit where all the memory blocks in the device were instantiated. To verify this mapping the block memories were also modified in random positions to compare the offset on the readback with the offset informed in the logic location file.

Table 7.1 presents a fragment of the mapping between the start of the column in the bitstream file and the logical floorplan for the configuration memory (CRAM) and user data (BRAM) frames in the Zynq-7000 XC7Z030 device.

The organization of the bits inside the SRAM array of the memory blocks is relatively complex but the distribution of the columns and frames among the several memory block is rather simple. Each row in the floorplan have, typically, 10 memory blocks arranged vertically in each column. For each frame, the first 320 bits belong to the first memory block, the next 320 bits belong to the memory block immediately above, and so

Table 7.1 – Fragment of floorplan mapped to Zynq-7000 XC7Z030 bitstream

Frame type	*Column type*	*Region*	*Row*	*Column*	*Bitstream file byte offset*
...					
CRAM	CLB	Top	0	20	278,360
CRAM	CLB	Top	0	21	292,904
CRAM	BRAM	Top	0	22	307,448
CRAM	CLB	Top	0	23	318,760
CRAM	CLB	Top	0	24	333,304
CRAM	DSP	Top	0	25	347,848
CRAM	CLB	Top	0	26	359,160
...					
CRAM	CLB	Bottom	2	71	4,328,864
CRAM	CLB	Bottom	2	72	4,343,408
CRAM	I/O Blocks	Bottom	2	73	4,357,952
...					
BRAM	BRAM	Top	0	2	4,475,112
BRAM	BRAM	Top	0	3	4,526,824
BRAM	BRAM	Top	0	4	4,578,536
...					
BRAM	BRAM	Bottom	2	4	5,822,048
BRAM	BRAM	Bottom	2	5	5,873,760
BRAM	BRAM	Bottom	2	6	5,925,472

Source: The Author

on. Each block memory have 4,096 parity bits and 32,768 data bits distributed horizontally along the 128 frames of the BRAM data memory column.

A descriptive model was build, in the form of a Python script, using the information gathered from fault injection, logic location report and readback to reverse map bits in the readback file to the respective bit and frame address as defines in Table 6.1. This model is not complete in the sense that is was not extensively validated in some columns such as the clock buffer columns (BUFHCE and BUFG) and the I/O blocks for PCI-Express.

The descriptive model was verified with additional fault injection campaigns injecting faults in randomized positions of the floorplan. This procedure consisted in injecting a fault in some random row, column, frame and bit, reading back the bitstream, identify the bitstream file position of the fault, apply the model the reverse map that position to the bit and frame address and, finally, compare the randomized addresses with the address obtained from the model. This procedure was repeated for 10 thousand different positions to build a reference database for validation of the model.

This whole mapping process was repeated also for the Xilinx Zynq-7000 XC7Z020 to which was built another descriptive model in the Python script.

7.2 Laser cartography on Xilinx 7 Series FPGA

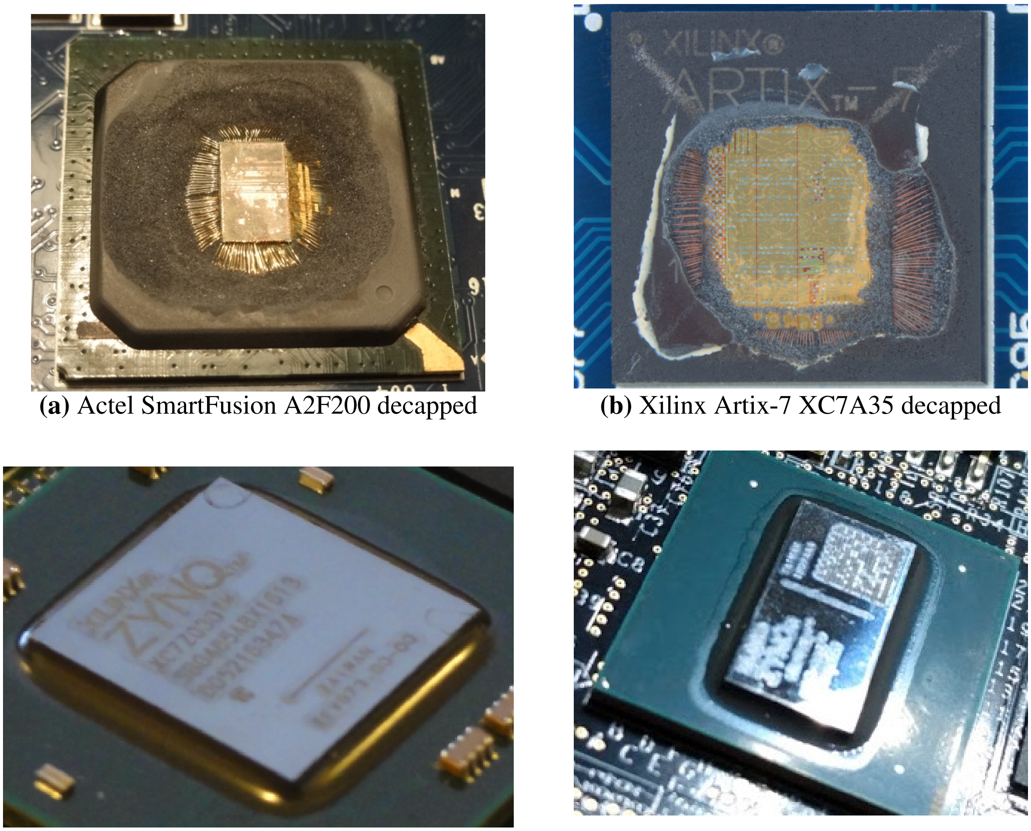
Although the process described previously could determine the mapping between the logical floorplan and the bitstream file, it is not enough to map the logical floorplan to the die surface. To this, static tests with laser were executed for study of the bit density and organization of the configuration memory and end user memory blocks.

A first step for physical fault injection using laser may be the sample preparation. When the die is covered by packaging it must be decapsulated (decapped), at least partially to open a window for laser test. Figure 7.2a shows the example of an Actel, now Microchip, SmartFusion A2F200M3F Flash-based FPGA decapped using fuming nitric acid at the Analytical Laboratory facilities of the Centro Nacional de Tecnologia Eletrônica Avançada (CEITEC, National Center for Advanced Electronic Technology), Porto Alegre. Figure 7.2b shows a Xilinx Artix-7 XC7A35T decapped at the Electronic Products Qualification and Analysis Center (NAPE) facilities of the Centro de Tecnologia da Informação Renato Archer (CTI, Center for Information Technology Renato Archer), Campinas.

Dies assembled by wire bonding typically have its backside (bulk) glued to the package frame leaving the frontside exposed. In the case of FPGAs, a first challenge will be the dense metallization on the frontside of the device that prevents the laser from passing through, as seen in Fig. 7.2a and 7.2b. Conversely, dies assembled in flip-chip will have the frontside attached to the package frame or interposer leaving the backside (bulk). Some products assembled in flip-chip are offered without exterior lid having the die exposed, for instance to allow improved heat transfer solutions, as seen in Fig. 7.2c and 7.2d.

The flip-chip die assembly is more convenient for laser fault injection from backside (die bulk) but manufacturer markings on the die and the die thickness introduce new challenges. The die on Xilinx Zynq-7000 and 7 Series devices have a thickness around 700 μm (POUGET et al., 2017) and the absorption coefficient becomes an import param-

Figure 7.2 – FPGA dies frontside and backside assembly



(a) Actel SmartFusion A2F200 decapped

(b) Xilinx Artix-7 XC7A35 decapped

(c) Zynq-7000 XC7Z030 delidded flip-chip

(d) Zynq UltraScale+ ZU3EG delidded flip-chip

Source: The Author

eter in selecting the wavelength for laser fault injection. Some researchers also reduce the thickness of the die to around 100 μm or below using combinations of mechanical die backgrinding, laser and chemical processes for thinning and polishing.

While laser in the region of near-infrared (NIR) have better penetration in silicon, the photons have lower energy and the success in the generation of electron-hole pairs depends on the combined absorption of two photons (GÖPPERT-MAYER, 2009) and on the laser focusing at the depth of the transistors active region (Section 2.6), increasing the complexity and cost of laser facilities. Conversely, high-energy photons have a higher absorption coefficient.

Data used in this thesis was collected in the facilities of the Institut d'Électronique et des Systèmes (IES) at the Université de Montpellier using laser at two wavelengths. A laser with 1.064 nm wavelength was used in a single-photon absorption (SPA) process with pulses of 250 pJ, 275 pJ and 300 pJ. Another laser with 1.550 nm wavelength was used in a process of two-photon absorption (TPA) with pulses of 220 pJ and 330 pJ.

The experimental dataset consists of just over 130 hours of experiment but the data collected specifically to this thesis, discussed in the following, only add up to about 4 hours.

To a preliminary analysis of the bit density and memory organization of the configuration (CRAM) and data (BRAM) memories, two circuits were implemented in the Xilinx Zynq-7000 XC7Z030 used in the laser experiments initializing bits in particular regions of the device with values zero (0) and one (1). This initialization was applied to the LUT contents at the CRAM, as in the example of Fig. 7.3, and to the contents of the block memory, as in the example of Fig. 7.4.

Figure 7.3 – Code fragment initializing LUT6 primitive with 1's

```
library UNISIM;
use UNISIM.VCOMPONENTS.ALL;

entity LUT6_ones is
end LUT6_ones;

architecture dummy of LUT6_ones is
    attribute DONT_TOUCH : string;
    attribute DONT_TOUCH of LUT6_ones_inst: label is "TRUE";
begin
       LUT6_ones_inst : LUT6
       generic map (
          -- Initial content of LUT equation/truth table
          INIT => X"FFFFFFFFFFFFFFFF")
...
end dummy;
```

Source: The Author

The whole BRAM capacity in the Xilinx Zynq-7000 XC7Z030 was instantiated while LUTs were instantiated to only two rows of the floorplan as presented in Fig. 7.5 and Fig. 7.6 e Fig. 7.7

The procedure for static test was implemented as two independent processes of laser control and bitstream readback, as summarized in Fig. 7.8.

All the tests were executed at the IES by the team of IES with support of researchers from UFRGS present at the IES. The FPGA experimental board and all the routines for the laser control station were provided by the IES. The experimental board is a PicoZed module featuring the Xilinx Zynq-7000 XC7Z030 device assembled on a FMC PicoZed Carrier Card (Fig. 7.9).

The routines for the readback control computer were developed at UFRGS in the scope of this thesis.

The pulse period of laser was adjusted to 4 seconds and each cycle of readback and comparison of the bitstream file took 16 seconds. There is no synchronization between the

Figure 7.4 – Code fragment initializing RAMB36E1 primitive with 1's

```
library UNISIM;
use UNISIM.VCOMPONENTS.ALL;

entity BRAM4K_ones is
end BRAM4K_ones;

architecture dummy of BRAM4K_ones is
    attribute DONT_TOUCH : string;
    attribute DONT_TOUCH of BRAM4K_ones_inst: label is "TRUE";
begin
       BRAM4K_ones_inst : RAMB36E1
       generic map (
          -- Initial content of memory parity data
          INITP_00 => X"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
                        FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
...
          INITP_0F => X"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
                        FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
          -- Initial content of memory user data
          INIT_00 => X"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
                       FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
...
          INIT_7F => X"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
                       FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
...
       )
...
end dummy;
```

Source: The Author

Figure 7.5 – Code fragment instantiating static elements

```
architecture arch of z030_top is
...
    LUT6_ones_many: for i in 0 to 959 generate
    begin
        LUT6_ones_inst_i : LUT6_ones;
    end generate LUT6_ones_many;

    BRAM4K_ones_many: for j in 0 to 264 generate
    begin
        BRAM4K_ones_inst_j : BRAM4K_ones;
    end generate BRAM4K_ones_many;

end arch;
```

Source: The Author

two processes of laser control and readback, therefore the number of pulses accumulated between each readback will not be necessarily the same. There is the chance of the laser pulse occur during the readback operation of during the comparison of the files.

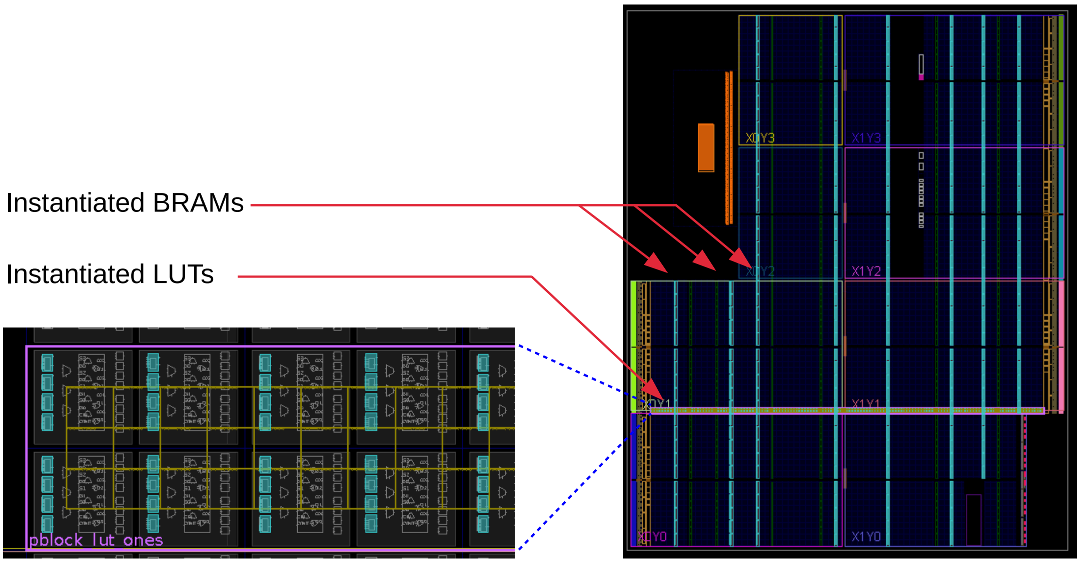
Four runs of test were executed, as summarized in Table 7.2. All tests used an nominal horizontal step of 1 µm between laser shots and vertical step of 5 µm between

Figure 7.6 – Placement constraints for static elements

```
create_pblock pblock_lut_ones
add_cells_to_pblock [get_pblocks pblock_lut_ones] \
             [get_cells -quiet -hierarchical \
                          -filter { REF_NAME =~ *LUT6_ones* }
             ]
resize_pblock [get_pblocks pblock_lut_ones] \
             -add {SLICE_X0Y50:SLICE_X119Y51}
```

Source: The Author

Figure 7.7 – Placement of static elements in the floorplan



Source: The Author

Figure 7.8 – Static test procedure with laser fault injection

At the laser control station	***At the readback control computer***
1. Define pulse period, energy, displacement speed, vertical and horizontal steps 2. Define region of interest for laser fault injection 3. Start line scan 4. Advance to next line 5. Repeat steps 3 and 4 until coverage of region of interest	1. Load bitstream with `xsdb` or`xmd` 2. Read back bitstream with Vivado batch mode 3. Compare bitstream files with `readback_compare` and report bit-flip locations 4. Repeat steps 2 and 3 continuously

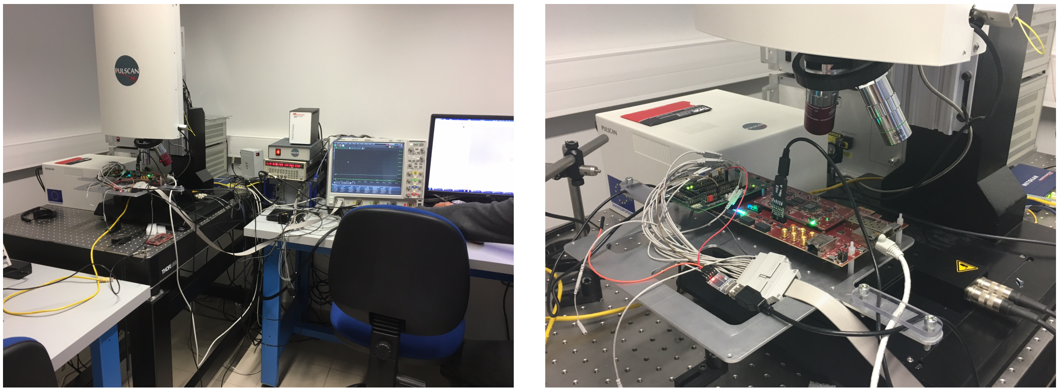
Source: The Author

horizontal lines.

In later analysis it was observed that the high number of changes in BRAM for the bitstream files at the second run, which targeted the CLB, were caused by hits of the laser beam in a BRAM configuration bit that reset a to zero value large portions of BRAM filled with ones.

Figure 7.10 presents some sample artifacts, seen in the readback file as clusters of bit-flips, produced by the laser fault injection at the Xilinx Zynq-7000 XC7Z030 device.

Figure 7.9 – Laser test facility



(a) TPA laser facility at IES

(b) Detail of the PicoZed module and FMC Carrier

Source: Images courtesy of Fernanda Lima Kastensmidt

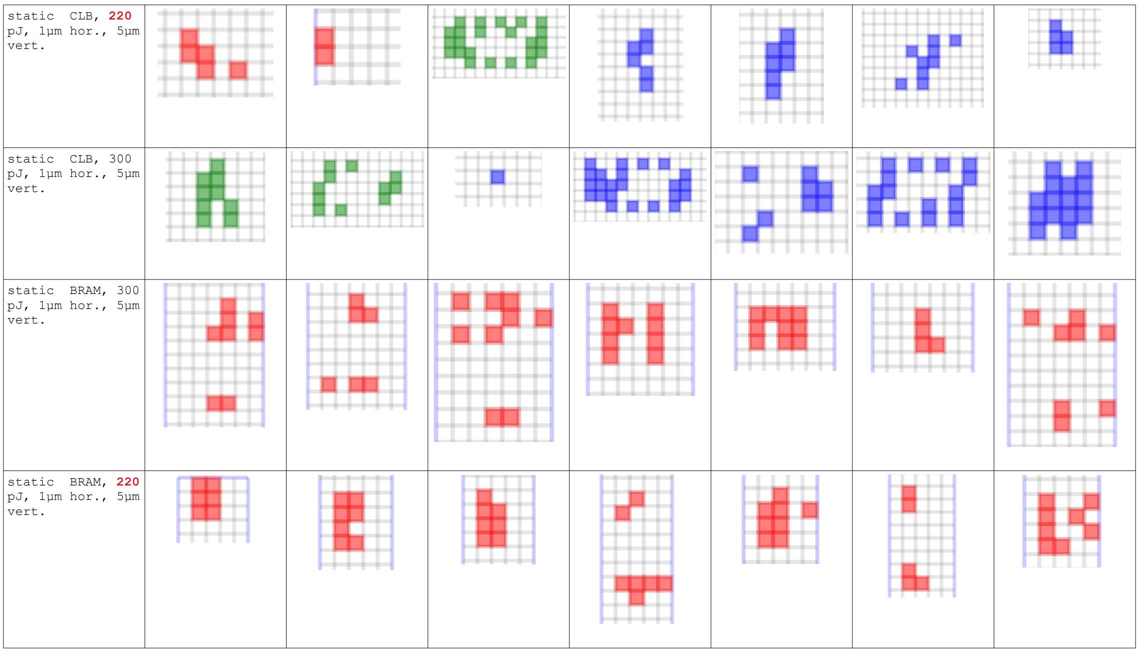
Table 7.2 – Laser cartography static test runs

Run	***Target region***	***Laser mode and pulse energy***	***# Readback files***	***Total test duration***	***Faults on CRAM***	***Faults on BRAM***
1	CLB	TPA 220 pJ	441	120 min	2138	–
2	CLB	TPA 300 pJ	385	108 min	2377	1408
3	BRAM	TPA 300 pJ	52	15 min	–	290
4	BRAM	TPA 220 pJ	62	16 min	–	252

Source: The Author

As there are multiple laser shots between readbacks, these bit-flips are cumulative from all the shots until the readback.

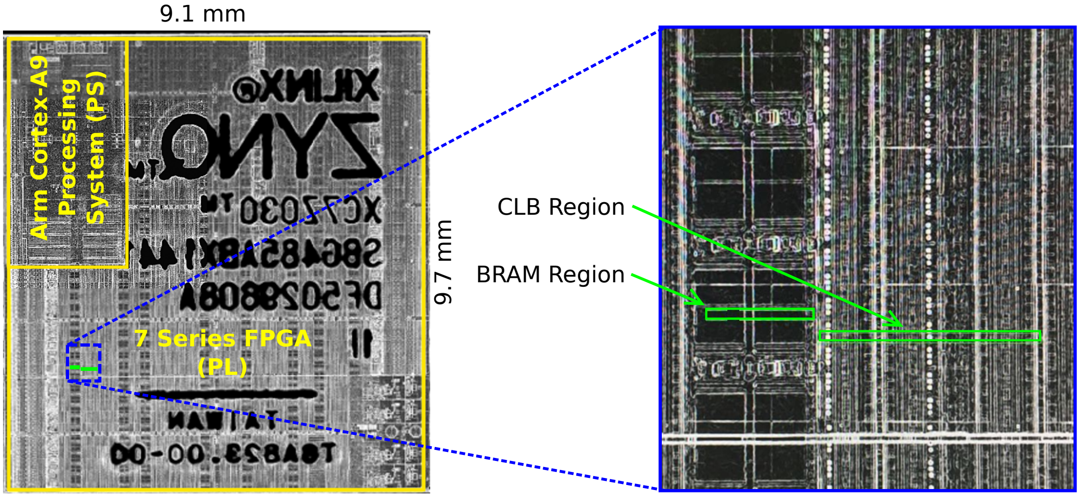
Figure 7.10 – Sample laser artifacts


static CLB, 220 pJ, 1μm hor., 5μm vert.
static CLB, 300 pJ, 1μm hor., 5μm vert.
static BRAM, 300 pJ, 1μm hor., 5μm vert.
static BRAM, 220 pJ, 1μm hor., 5μm vert.

Source: The Author

The region of interest in the device covered by the static tests using laser fault injection is shown in Fig. 7.11, corresponding to approximately 0.01% of the total die area of $9.1\,\mathrm{mm} \times 9.7\,\mathrm{mm}$ (XILINX, 2021g).

Figure 7.11 – Laser scan regions of interest for static tests



Source: Die infrared images courtesy IES

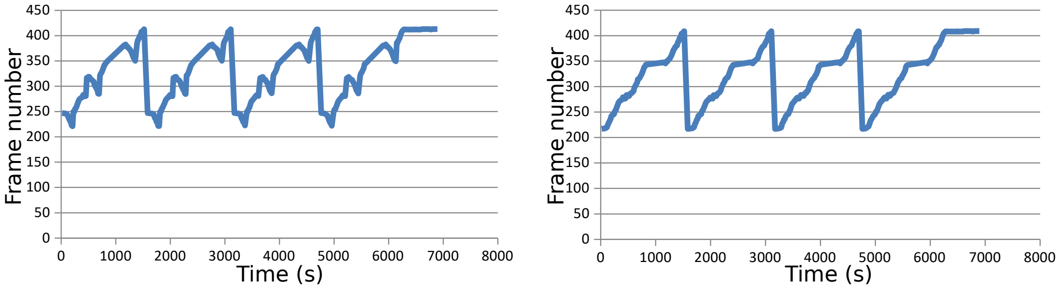
The bitstream files collected during the experiment were preserved for post-processing and detailed analysis.

7.3 Analysis of physical arrangement of configuration memory

Aligning at the time the bit-flips caused by laser fault injection, extracted from the readback files, it was identified that the physical memory columns have different directions. Different directions were already present at the logical floorplan provided by the Xilinx Vivado synthesis tool, as in the example of Fig. 6.5, and suggested by the back-to-back interconnect shown in Fig. 6.4, but the laser experiments demonstrated that the different directions occurs also in the physical layout.

Without this direction adjustments the laser scan appears as a discontinuous trajectory (Fig. 7.12a) over the die, which is incorrect. After adjusting the frame addressing considering the column direction we can align the frame address with the laser trajectory (Fig. 7.12b). In some columns the memory frames are arranged from left to right, as in columns CLBLM_L and BRAM_L in Fig. 6.5, while in other columns the frames are physically arranged from right to left, as in columns CLBLM_R and BRAM_R in Fig. 6.5. Conveniently this information is already available at the Xilinx Vivado synthesis tool not requiring further laser scanning in the whole die width to determine columns directions.

Figure 7.12 – Laser trajectory time and frame number (trajectory x axis)



(a) No adjustment in column direction

(b) Adjusted column direction

Source: The Author

It was also observed that the column of BRAM frames for BRAM (frames associated with the parity and user data in memory blocks) is physically beside the corresponding column of CRAM frames for BRAM (frames associated with routing and configuration of memory blocks). The direction of these frames also follows the aforementioned direction of the column. The main consequence for the fault injector is that the column direction must be taken into account, both in CRAM and BRAM, to determine bits neighborhood when analyzing the occurrence of multiple-bit memory upsets (MBU).

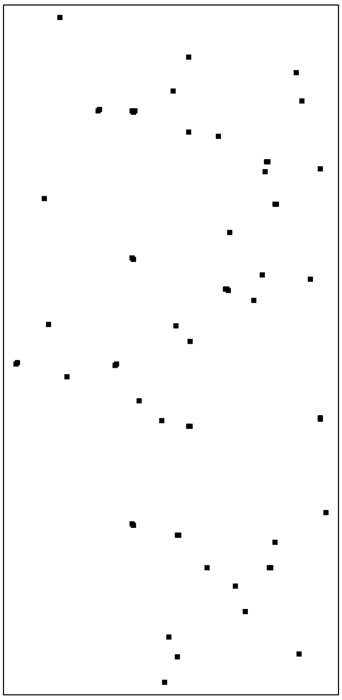
Using these findings from the laser tests, the descriptive model of the devices discussed in Section 7.1 was extended with the column directions to compute correctly the reverse map from bitstream file offset to die location.

Other relevant aspect, already documented by the manufacturer (XILINX, 2015a; CHAPMAN, 2015) and other researchers (WIRTHLIN, M. J. et al., 2014; HARDING et al., 2014; STODDARD, 2015), is that the configuration memory is interleaved such that bit-flips in physically adjacent memory cells, for instance due to MBU, are not adjacent in the logical map of memory frames.

This laser cartography experiment was not reproduced with the Xilinx Zynq-7000 XC7Z020 device but a similar experiment using bitstream readback was executed with a $43.2\,\mathrm{MeV}$ ^{16}O heavy ion beam at the Nuclear Physics Open Laboratory at University of São Paulo (LAFN) using a moving collimating target (pinhole) coupled to a motorized stepper. Although the beam spot diameter in this heavy ion setup is one order of magnitude larger than the spot diameter of the laser beam (compare Fig. 7.13 with Fig. 7.10), the trajectory of the heavy ions microbeam indicated that the geometric model built for XC7Z020 is consistent.

The microbeam experiment also give insights about the aspect ratio of the memory array but the die surface covered and the number of bitstream readback collected with

Figure 7.13 – Sample heavy ions microbeam artifacts



Source: The Author

microbeam is not enough to build a metric model of the configuration memory. Notwithstanding, these are promising results and the heavy ion microbeam may become as useful as laser for this kind of experiment.

8 STATIC TESTS AND MULTIPLE-BIT UPSETS INVENTORY

8.1 Static tests of Xilinx 7 Series under proton, alpha and heavy ions

Multiple-bit memory upset (MBU) in Xilinx SRAM-Based FPGAs are studied by Quinn et al. (2005) and Manuzzato et al. (2008) covering several device generations from Xilinx Virtex to Xilinx Virtex-4, from 220 nm to 90 nm technology.

M. Wirthlin et al. (2014) analyze of MBU in Xilinx 7 Series Kintex-7 devices under heavy ions and provides the frequency of occurrence of MBUs with different number of bit-flips for LET from $1.5\,\mathrm{MeV mg^{-1} cm^{-2}}$ to $126\,\mathrm{MeV mg^{-1} cm^{-2}}$, but without distinguishing the behavior of configuration (CRAM) and user data (BRAM) memories. An important contribution of M. Wirthlin et al. (2014) is on the statistical methodology to identify MBUs in the memory array.

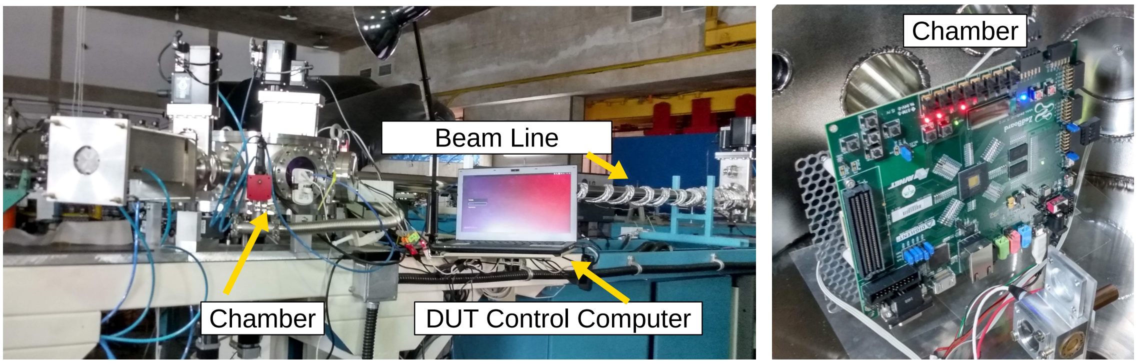
Du et al. (2019) also analyze MBUs in the same class of Xilinx Kintex-7 devices under heavy ions with LET of $3.7\,\mathrm{MeV mg^{-1} cm^{-2}}$. The clustering method adopted by Du et al. (2019) groups in the same MBUs bit-flips occurring in the same FPGA column and with distance up to $\sqrt{2}$.

Tonfat et al. (2017) analyze the occurrence of MBU in Xilinx 7 Series Artix-7 under heavy ions, varying the angle of incidence in an attempt to cover a larger range of LET, but do not provide detailed information about occurrence of different MBU artifact geometries of different sizes of bit-flip clusters. Moreover, the analysis focus in the occurrence of MBUs inside a single configuration memory (CRAM) frame, which hinders the efficacy of Xilinx native memory scrubber, or MBUs inside a single user data (BRAM) word, which hinders the efficacy of Xilinx native block memory error detection and correction codes.

The device of interest in this thesis is the Xilinx Zynq-7000 XC7Z020, which features an FPGA equivalent to the Xilinx Artix-7 (XILINX, 2021h), hence results from M. Wirthlin et al. (2014) and Du et al. (2019) may not be directly comparable.

This study of multiple bit upsets in Xilinx 7 Series SRAM-based FPGA under protons and heavy ions was based on a dataset of readback files shared by Aguiar et al. (2020) as part of the cooperation in the development of the SAFIIRA (Sistema de Feixes Iônicos para IRradiações e Aplicações, ion beam system for irradiation and applications) facility at the LAFN (AGUIAR, 2019). Figure 8.1 shows an example of experimental setup in SAFIIRA facility with a ZedBoard development board featuring a decapped Xilinx Zynq-7000 XC7Z020 device installed inside the experimental chamber.

Figure 8.1 – SAFIIRA facility at the LAFN 0° beamline



Source: The Author

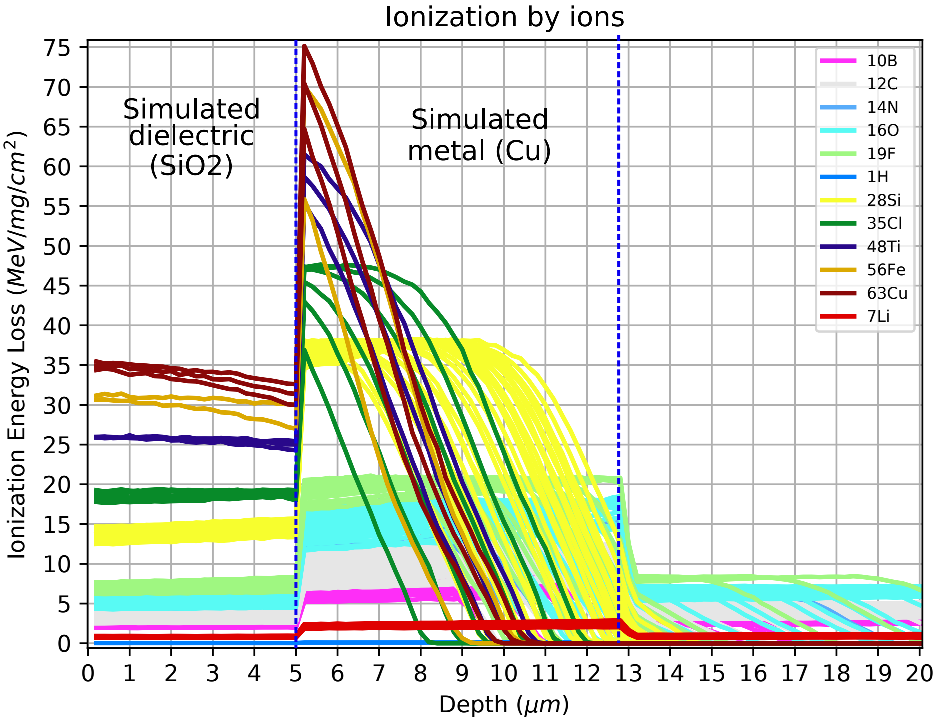
The dataset from Aguiar et al. (2020) was collected with a PYNQ-Z1 development board featuring a Xilinx Zynq-7000 XC7Z020 device decapped at the Electronic Products Qualification and Analysis Center (NAPE) facilities of the Centro de Tecnologia da Informação Renato Archer (CTI, Center for Information Technology Renato Archer), Campinas. The device was irradiated with protons at $12\,\mathrm{MeV}$ and heavier ions ranging from ^{7}Li to ^{63}Cu with energies from $13.1\,\mathrm{MeV}$ to $87.5\,\mathrm{MeV}$. Figure 8.2 shows LET estimated using SRIM/TRIM software tool (ZIEGLER et al., 2010) for these ions and energies.

The device manufacturer indicates the use of high-k dielectric at the passive layers, which is expected to be porous composite such as SiCO(H) (BAO et al., 2009; TSMC, 2011). S.H. Yang et al. (2011), for instance, suggests and all the TSMC 28 nm processes share a common back-end of the line (BEOL) and documents that the TSMC HPM process uses a extreme high-k (ELK) inter-metal dielectric with k=2.5, which is consistent with data from TSMC (2011) and TSMC (2015). Simulations with SRIM/TRIM suggested increased penetration range of the particles with a porous dielectric but with small variation of the LET at the depth of the transistor active regions. Hence, the model from the literature (TAMBARA et al., 2016; YANG, W. et al., 2019; YANG, W.-T. et al., 2019; AGUIAR, 2019) with SiO_2 (k=4) as dielectric and the worst case scenario of maximum Cu layers the was adopted in the estimates of Fig. 8.2.

The dataset was extended with additional experiments with a ^{241}Am radioactive source of α particles ($5.5\,\mathrm{MeV}$), ^{16}O ($38.4\,\mathrm{MeV}$ and $43.2\,\mathrm{MeV}$) and ^{28}Si ($73.5\,\mathrm{MeV}$) heavy ions also performed at the SAFIIRA facility at the LAFN by researchers of UFRGS with cooperation of researchers from IF-USP.

Data for $12\,\mathrm{MeV}$ protons extracted from the Aguiar et al. (2020) dataset must be analyzed with caution as energy in this range is scarcely approached in the literature and is

Figure 8.2 – Estimated LET for LAFN beam



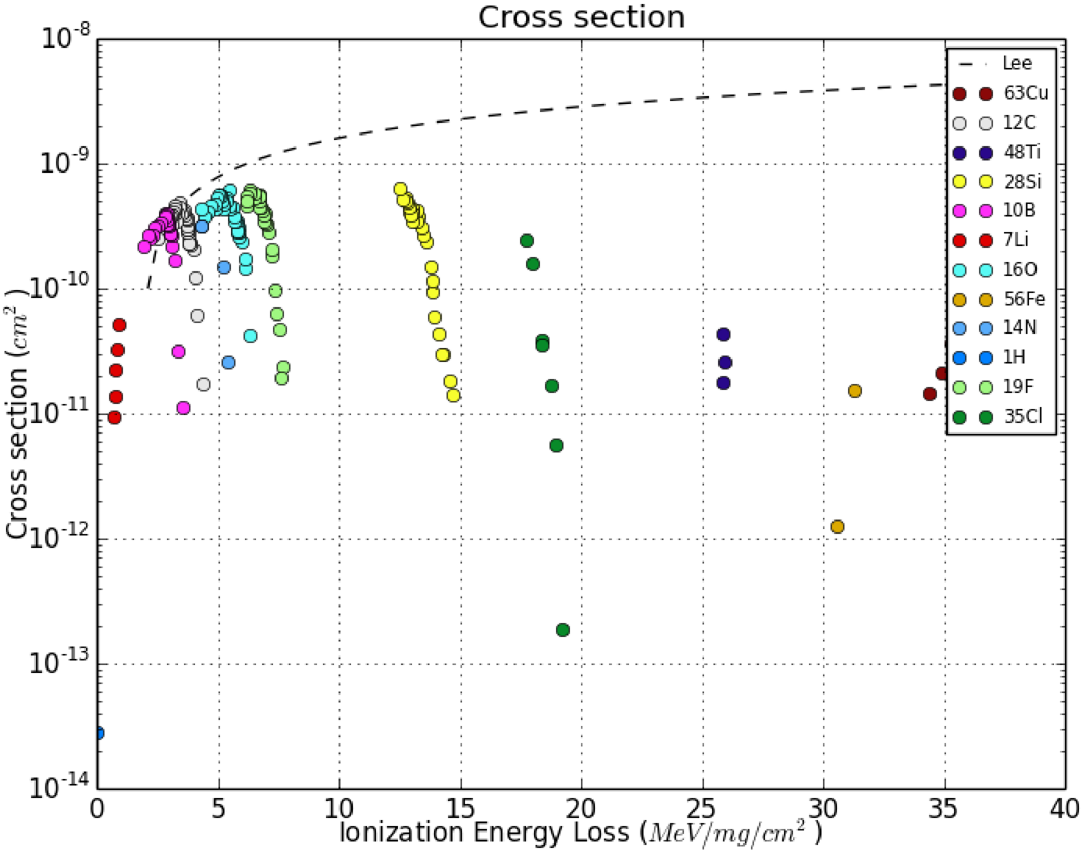
Source: The Author

not expected to have a significant contribution to the occurrence of SEUs (SIERAWSKI et al., 2009). P. Maillard et al. (2015), Tambara et al. (2016) and W.-T. Yang et al. (2019), for instance, use proton with energies higher than $60\,\mathrm{MeV}$. Notwithstanding, Y. Zhang et al. (2016) also use protons at lower energies of $3\,\mathrm{MeV}$, $5\,\mathrm{MeV}$ and $10\,\mathrm{MeV}$. More recently, Azimi et al. (2022) used protons in the range of $16\,\mathrm{MeV}$ to $150\,\mathrm{MeV}$ for comparative static tests of Xilinx Zynq-7000 XC7Z020 device against Xilinx UltraScale+ Zynq MPSoC XCZU7EV, with results discussed briefly in Section 4.3.

Figure 8.3 show a preliminary estimate of static cross-section over estimated LET at the surface of the device, which as estimated by SRIM/TRIM. For comparison, the static cross section curve for Xilinx Kintex-7 parameterized by Lee et al. (2014) is shown as a dashed line. The decay observed in the plot static cross-section for higher surface LET can be considered situations where the particle depth does not always reach the transistor active area with the SEUs observed being due to occasional lower number of metal layers. For comparison, Figure 8.4 shows cross-section over the estimated LET at the depth of the transistor with the worst case of maximum number of metal layers.

This indicate that some beam configurations at the SAFIIRA facility should no be used in irradiation test in this device because they do not reach uniformly the transistors

Figure 8.3 – Heavy ions cross section considering LET at surface

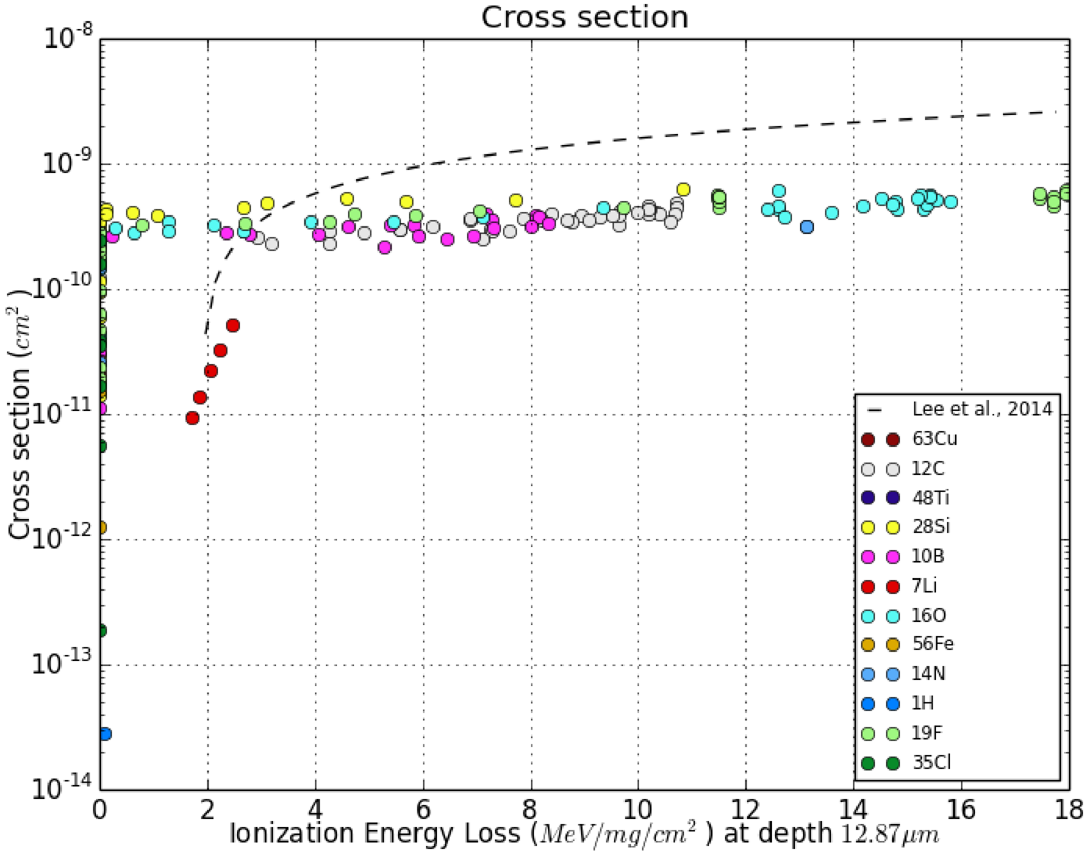


Source: The Author

active area. Figure 8.5 show the beams and energy range available at the SAFIIRA facility classified in configurations that should not be used in irradiation tests for Xilinx 7 Series because they do not reach uniformly the transistors, configurations that is expected to always reach the transistors but are below the saturation level of the cross-section curve, and configurations that are safer to use.

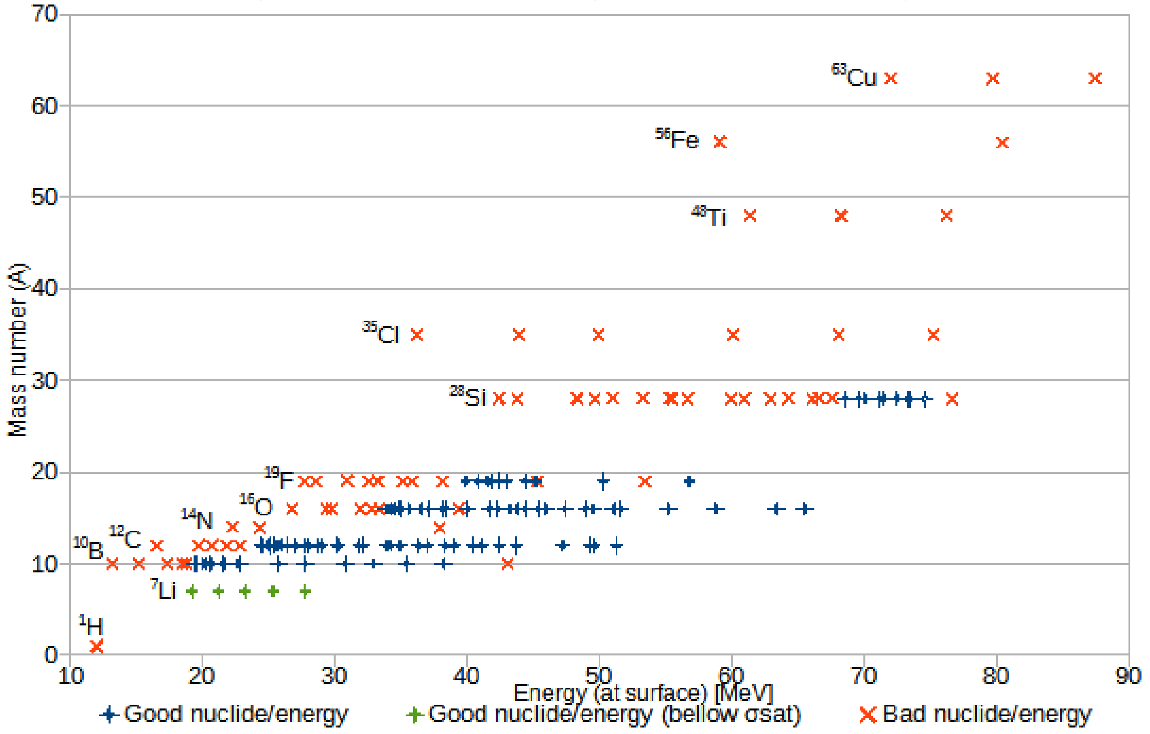
The most frequent SEU geometries and MBU cluster sizes observed in the read-back files, covering more than 95% of the cases, are presented in Fig. 8.6 for SEUs at the configuration memory (CRAM) and user data memory (BRAM), under heavy ions and α particles. Figure 8.7 shows examples of fitted cross-section curves for some of the most frequent SEU cluster sizes up to two bit-flips. Curve fitting was based on all SEUs (in green) but plots in Fig.8.7 also distinguishes bit-flips where the original bit value was one that flipped to zero (in red) and value zero that flipped to one (in blue) but there is no significant deviation in the static cross-section associated with the bit value.

Figure 8.4 – Heavy ions cross section considering LET at active region



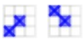
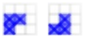
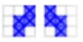
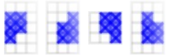
Source: The Author

Figure 8.5 – Nuclides and energies at the SAFIIRA facility



Source: The Author

Figure 8.6 – Most frequent heavy ions and alpha particles SEUs

Type of memory	*Type of SEU*	*Examples*	*Frequency*	
			Heavy ions	*α Particles*
BRAM	SBU 1-1-1		82.0%	100.0%
	MBU 2-1-2		16.2%	—
	MBU 1-2-2		—	—
	Others		1.8%	—
CRAM	SBU 1-1-1		38.1%	97.6%
	MBU 2-2-2		41.9%	2.4%
	MBU 2-1-2		4.4%	0.0%
	MBU 1-2-2		—	—
	MBU 2-2-3		3.0%	—
	MBU 2-2-4		0.2%	—
	MBU 2-3-4		8.3%	—
	MBU 2-3-5		0.6%	—
	Others		3.4%	—

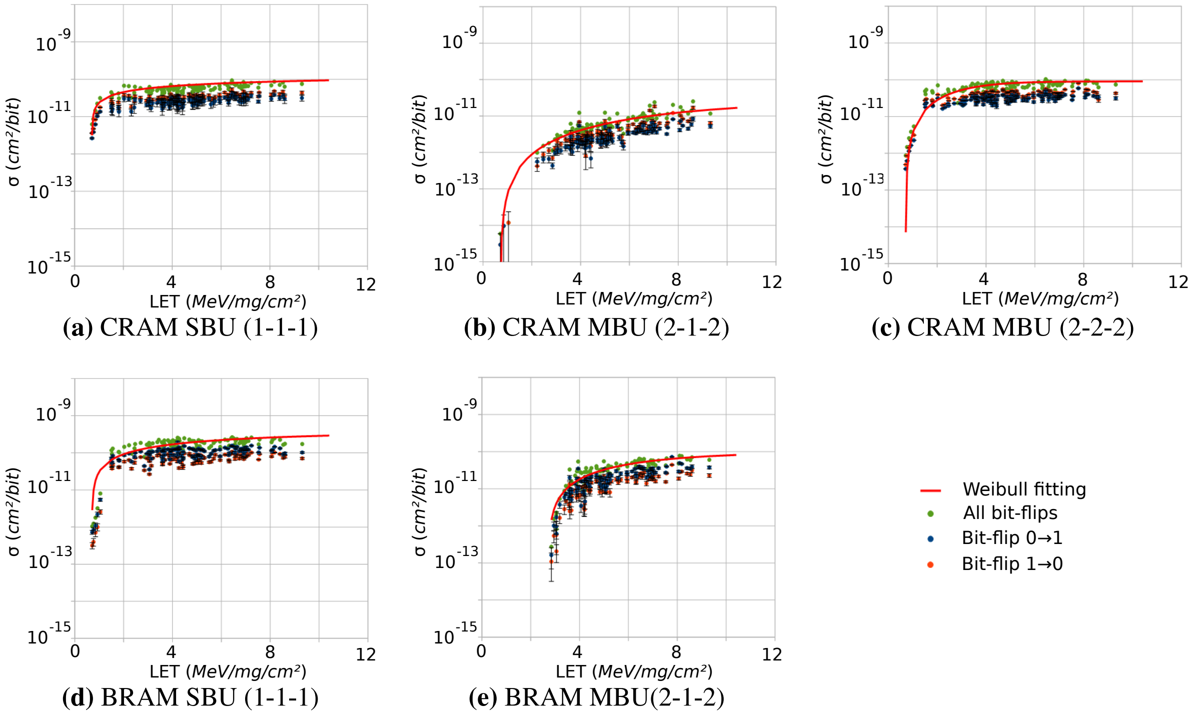
Source: The Author

8.2 Static tests of Xilinx 7 Series under fast neutrons

Static cross section of Xilinx 7 Series devices have already been studied by other researchers. Xilinx (2021b) reports static cross section for many Xilinx 7 Series at wide spectrum spallation neutron beam source from Los Alamos Neutron Science Center (LANSCE) but does not provide detailed information about multiple-bit upsets.

Tsiligiannis et al. (2018) report static cross for configuration (CRAM) and user data (BRAM) memory for a Xilinx 7 Series Artix-7 device at $14\,\mathrm{MeV}$ neutron but also do not provide detailed information about multiple-bit upsets. The authors use two technical approaches for data collection. The first one is using Xilinx Vivado tool and JTAG test interface to read back the bitstream file, which is the same procedure adopted in this thesis. The second approach is using the native memory scrubbing hardware present in Xilinx 7 Series devices that allows to report the position of bit-flips only in CRAM. The scrubber method was also used by Aguiar et al. (2019) to study static cross-section for thermal neutrons in a Xilinx Zynq-7000 XC7Z020 device in a facility with very low flow flux. Early

Figure 8.7 – Heavy ions cross-section for most frequent SEU geometries



(a) CRAM SBU (1-1-1)

(b) CRAM MBU (2-1-2)

(c) CRAM MBU (2-2-2)

(d) BRAM SBU (1-1-1)

(e) BRAM MBU(2-1-2)

Source: The Author

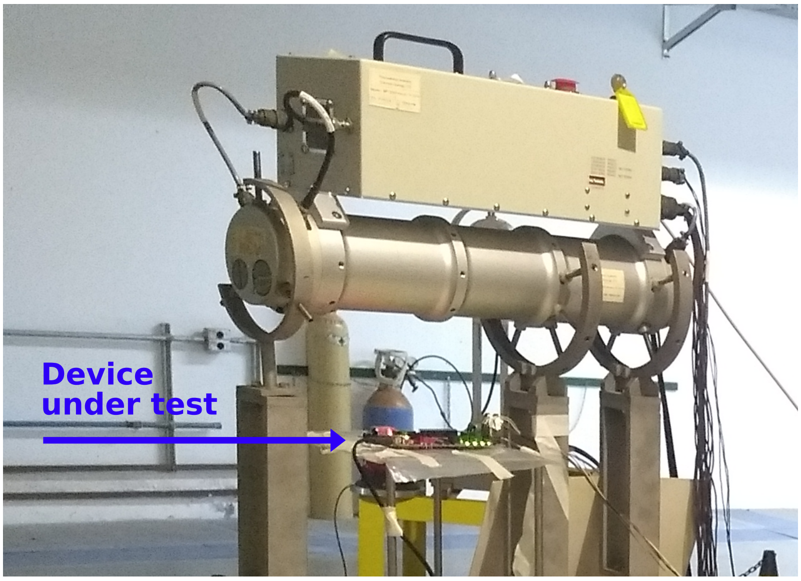
experiments with heavy ions in Aguiar et al. (2020) also used the scrubber method but this approach was abandoned because did not operate correctly at higher particle fluxes, a problem also reported by Tsiligiannis et al. (2018).

Fabero et al. (2020) uses $14.2\,\mathrm{MeV}$ neutrons to study MBUs in a Xilinx 7 Series Artix-7 device, which is the same technology of the Xilinx 7 Series Zynq-7000 XC7Z020 tested here. The clustering method adopted by Fabero et al. (2020) is based on the works of Clemente et al. (2016) and Franco et al. (2017), although the authors also compare results with Du et al. (2019) that used the Euclidean distance up to $\sqrt{2}$ rule. Fabero et al. (2020) reports statistics about the occurrence of SEUs with different number of bit-flips at the configuration memory (CRAM) and frequency of different geometric shapes in MBU clusters but did not report SEUs in user data memory (BRAM) and argued that static cross-section for BRAM should be similar to the static cross-section in CRAM. Instead of using Xilinx tools and test cables to interface with the experimental board collect the bitstream readback files, the authors used a microprocessor implementing JTAG interface to read back the FPGA memory and analyze and report bit-flips. This approach is similar to the readback procedure implemented at UFRGS for the Xilinx Artix-7 XC7A100T inside the NanosatC-BR2 small satellite (BENEVENUTI et al., 2019b).

The static cross-section and MBU analysis in this thesis is based on irradiation experiments performed at the Labotatório de Radiações Ionizantes (LRI, Ionizing Radiation Laboratory) at the Instituto de Estudos Avançados (IEAv, Institute for Advanced Studies), São José dos Campos, a research unit under the Aeronautics Science and Technology Department of the Brazilian Air Force. Tambara et al. (2014) studied SEUs in a Xilinx Spartan-6 device in this same laboratory but using different instrument with fast neutrons at a lower energy.

Readback files were collected from a Xilinx Zynq-7000 XC7Z020 device while irradiated using a deuterium-tritium (D-T) $14.1\,\mathrm{MeV}$ neutron generator (Fig. 8.8). During the experiments, the FPGA board was protected with $0.5\,\mathrm{mm}$ cadmium sheet to avoid the influence of thermal neutrons. Instrumentation was shielded with borated polyethylene to avoid interference.

Figure 8.8 – IEAv $14.1\,\mathrm{MeV}$ D-T neutron generator facility

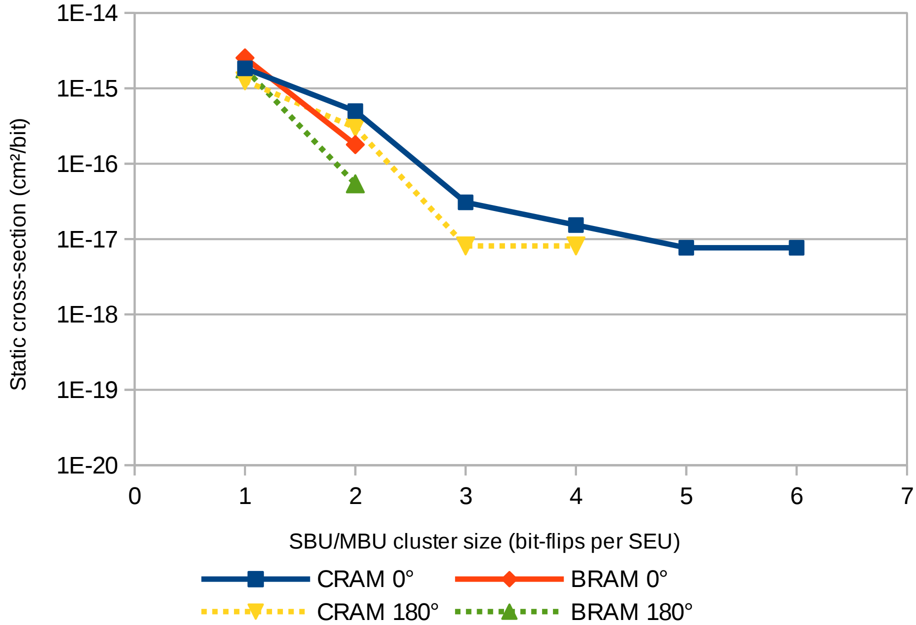


Source: Image courtesy IEAv

The static tests were performed with the experimental board in two orientations for frontal irradiation (0°) and backside irradiation (180°). Figure 8.9 summarizes static cross section at 0° and 180° incidence angles with different numbers of bit-flips per SEU at the configuration memory (CRAM) and user data memory (BRAM).

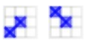
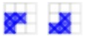
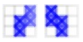
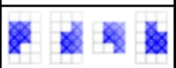
The most frequent SEU geometries and MBU cluster sizes observed in the readback files for $14.1\,\mathrm{MeV}$ neutron are presented in Fig. 8.10 for SEUs at the configuration memory (CRAM) and user data memory (BRAM), at 0° and 180° incidence angles.

Figure 8.9 – Static cross-section in different irradiation angles



Source: The Author

Figure 8.10 – Most frequent 14.1 MeV neutron SEUs

Type of memory	*Type of SEU*	*Examples*	*Frequency*	
			0°	*180°*
BRAM	SBU 1-1-1		93.4%	97.1%
	MBU 2-1-2		4.7%	2.9%
	MBU 1-2-2		—	—
	Others		1.9%	—
CRAM	SBU 1-1-1		76.7%	79.9%
	MBU 2 bits (2×2)		16.9%	15.5%
	MBU 2 bits (2×1)		3.5%	2.1%
	MBU 2 bits (1×2)		0.3%	1.5%
	MBU 3 bits (2×2)		1.3%	0.5%
	MBU 4 bits (2×2)		—	0.5%
	MBU 4 bits (2×3)		0.6%	—
	MBU 5 bits (2×3)		0.3%	—
	Others		0.3%	—

Source: The Author

8.3 Static tests of Xilinx 7 Series under thermal neutrons

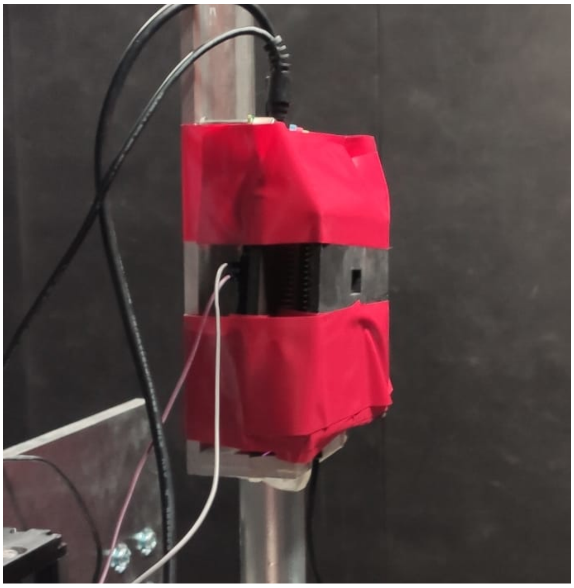
As in the case of fast neutrons, Xilinx (2021b) and Tsiligiannis et al. (2018) report static cross section for configuration (CRAM) and user data (BRAM) memory for a Xilinx 7 Series Artix-7 under thermal neutrons, but do not provide detailed information about multiple-bit upsets.

Aguiar et al. (2019) tested a Xilinx Zynq-7000 XC7Z020 device under thermal neutrons using the memory scrubbing hardware present in Xilinx 7 Series devices to report bit-flips in CRAM. The method was used to report and count bit-flips detected at in CRAM frames and no data was reported about the address of those bit-flips. Therefore, this experiment did not produce detailed information about multiple-bit upsets.

Experiments for analysis of multiple-bit upsets under thermal neutrons were performed at the Thermal and Epi-thermal Neutron irradiation Station (TENIS) of the Institute Laue–Langevin (ILL), Grenoble, in collaboration with the Université Grenoble Alpes (UGA). The beam presents a fission spectrum with maximum thermal neutron flux of $3 \times 10^9\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$ and thermal to epithermal ratio around 18.6.

To minimize interference of radiation on other electronic components, the board was covered with a boron carbide (B4C) shielding with a hole over the Xilinx Zynq-7000 SoC FPGA device (Fig. 8.11).

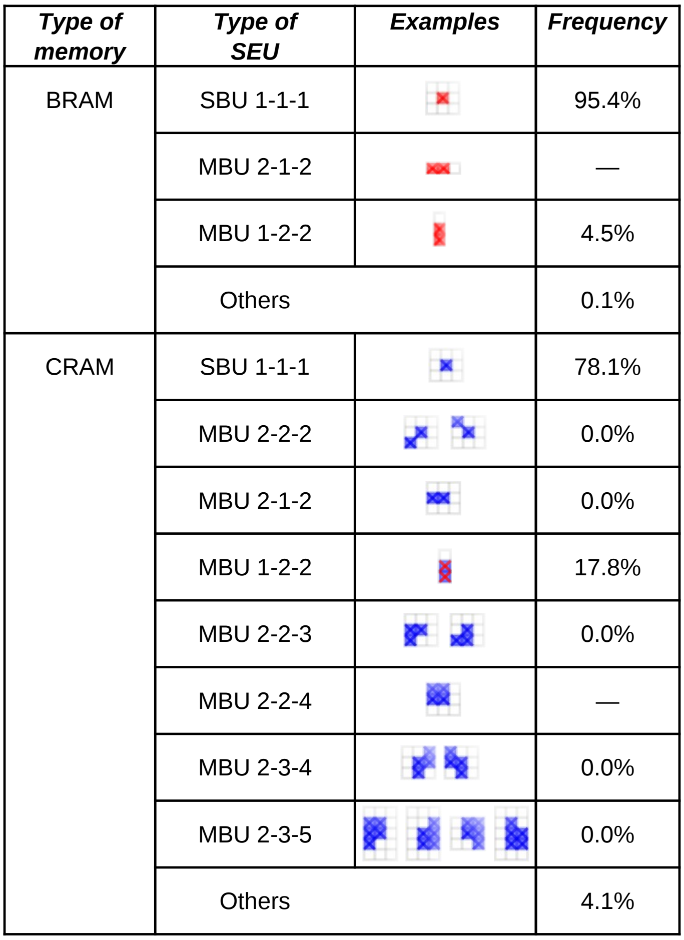
Figure 8.11 – Board shielding for thermal neutrons experiments.



Source: Image courtesy ILL

The most frequent SEU geometries and MBU cluster sizes observed in the readback files for thermal neutron are presented in Fig. 8.12 for SEUs at the configuration memory (CRAM) and user data memory (BRAM).

Figure 8.12 – Most frequent thermal and epithermal neutron SEUs



Type of memory	*Type of SEU*	*Examples*	*Frequency*
BRAM	SBU 1-1-1		95.4%
	MBU 2-1-2		—
	MBU 1-2-2		4.5%
	Others		0.1%
CRAM	SBU 1-1-1		78.1%
	MBU 2-2-2		0.0%
	MBU 2-1-2		0.0%
	MBU 1-2-2		17.8%
	MBU 2-2-3		0.0%
	MBU 2-2-4		—
	MBU 2-3-4		0.0%
	MBU 2-3-5		0.0%
	Others		4.1%

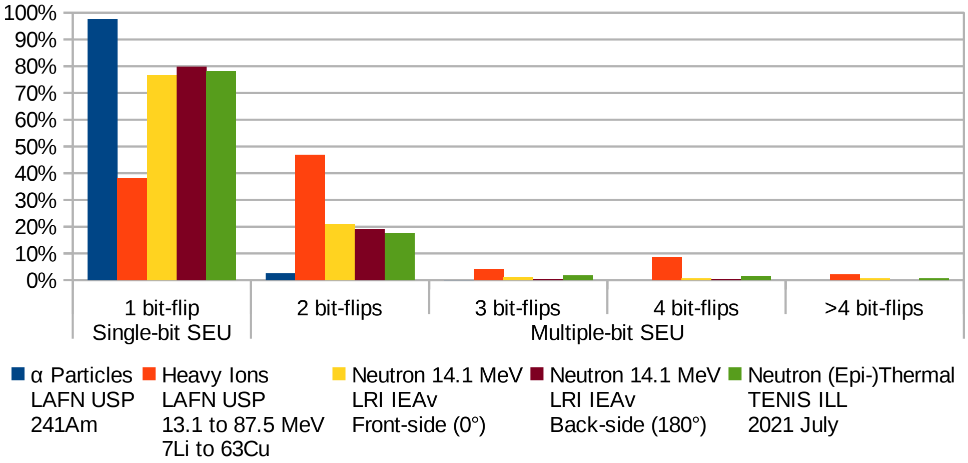
Source: The Author

8.4 Summary of multiple-bit upsets profile on Xilinx 7 Series FPGA

A summary of most frequent cluster sizes is presented in Fig. 8.13 and Fig. 8.14, for SEUs in configuration memory (CRAM) and user data memory (BRAM), respectively.

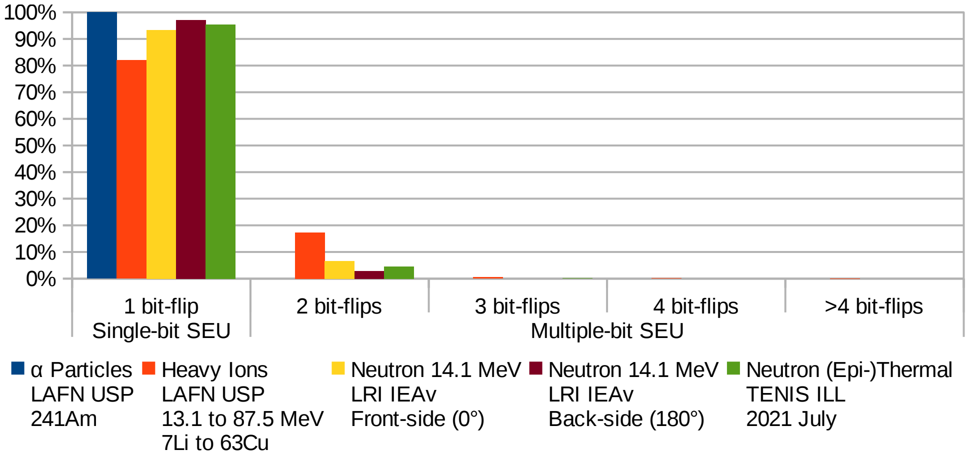
The support for emulation of multiple-bit upsets in the fault injector should take into account the proportion of SBU and the different sizes of MBU and the different MBU artifacts geometry that occurs with irradiation. Moreover, as different particles and energies have shown different profiles of multiple-bit upsets, the fault injection campaigns must also use the proper profile for the targeted environment of application of the design under test.

Figure 8.13 – Frequency of SEU cluster sizes in configuration memory (CRAM)



Source: The Author

Figure 8.14 – Frequency of SEU cluster sizes in user data memory (BRAM)



Source: The Author

Part III

Enhancements on Fault Injector and Methodology

9 MANY-BITS FAULT INJECTION

9.1 Motivation

A first improvement on the UFRGS fault injection tool for Xilinx 7 Series (TONFAT et al., 2016), discussed briefly in Section 5.3, was to emulate arbitrary shapes and sizes of bit-flip clusters in the configuration memory (CRAM) of the FPGA.

This feature was originally motivated by the potential use of the fault injector in the characterization of attack countermeasures for active laser fault attacks in security sensitive applications of SRAM-based FPGAs.

Laser fault attacks can target malicious access to embedded circuit design or software, private user data or cryptographic information such as shared keys installed by the device managing authority or user private keys.

The attack potential can be graded according to a threat model, which includes the attacker profile, such as a vandal kid, a trespasser, a thief or a nation supported spy, and the level of sophistication of the tools and technologies the attacker has access to.

In the case of laser, that level of sophistication can also be graded in terms of the diameter of the laser spot that can be focused on the circuit under attack. Vandals motivated by curiosity or fame may have access to low-end or homemade laser equipment, possibly scrapping near-infrared laser heads from old CD writers, achieving a focused laser spot diameter and XY stepper resolution in the scale millimeters. At the other end of the spectrum, highly skilled spies targeting a military communication equipment powered by software-defined radio (SDR) implemented on SRAM-based FPGA may have access to high-end state supported laser facilities achieving a focused laser spot diameter in the order of micrometers.

Fault artifacts induced by laser may have several shapes, as seen in Fig. 7.10. Other examples of artifact shapes can be found in Canivet et al. (2009), Darracq et al. (2009), Sterpone et al. (2011) e Castro et al. (2015).

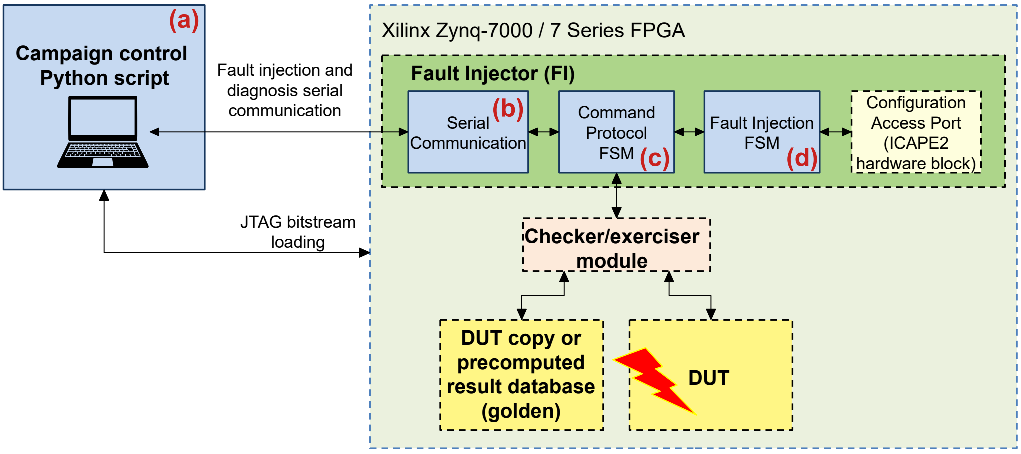
A laser fault attack detection circuit implemented in the FPGA may be easily triggered by a laser spot in the scale of millimeters but may overlook smaller laser spots. Hence, an interframe many-bits fault injection methodology was conceived to grow the fault artifact in the memory starting from a single bit-flip and adding more bits following an arbitrary shape model for the fault.

9.2 Implementation

The decisions made for the implementation of the many-bits fault injection was strongly influenced by the main mode of operation of the fault injector, which was the sequential-exhaustive scanning the region of interest where the DUT was placed.

The UFRGS fault injection for Xilinx 7 Series as implemented by Tonfat et al. (2016) can be organized in four layers. We have the campaign scripting (Fig. 9.1a) that coordinates the sequential-exhaustive fault injection, a serial communication module (Fig. 9.1b) interfacing with the campaign script, a command interpreter finite state machine (FSM) (Fig. 9.1c) which also coordinates the fault injection and DUT diagnosis and, finally, a fault injection finite state machine (FSM) (Fig. 9.1d) which transmits the required commands for the Xilinx ICAP hardware to read and write CRAM frames. In that setup the FPGA was usually programmed through JTAG interface.

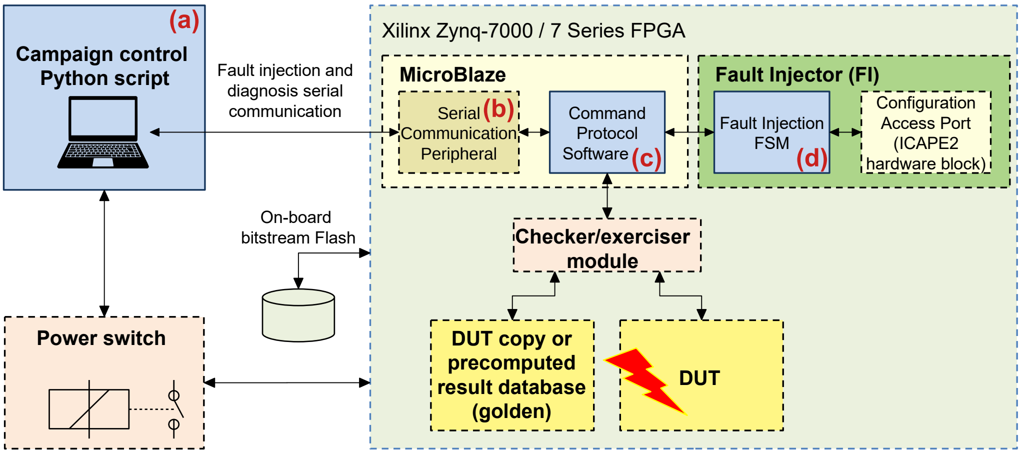
Figure 9.1 – Legacy fault injector setup


(a)
Campaign control Python script
Fault injection and diagnosis serial communication
JTAG bitstream loading
Xilinx Zynq-7000 / 7 Series FPGA
Fault Injector (FI)
Serial Communication
(b)
Command Protocol FSM
(c)
Fault Injection FSM
(d)
Configuration Access Port (ICAPE2 hardware block)
Checker/exerciser module
DUT copy or precomputed result database (golden)
DUT

Source: The Author

As the serial communication is time consuming, it was considered to forward to the fault injector modules inside the FPGA the range of frame and bit addresses where faults should be injected sequentially, instead of issuing each command to inject fault individually. This approach would have little effect to the laser attack emulation but could improve the speed in the sequential-exhaustive fault injection. To implement this modification, the command FSM (Fig. 9.1c) was replaced by a Xilinx MicroBlaze ™ allowing more flexible protocol and DUT diagnosis routines to be implemented in software (Fig. 9.2c). Another modification consisted in replacing the FPGA programming through JTAG interface by power-cycling and loading the bitstream from on-board non-volatile memory (Flash), further improving the campaign speed.

Figure 9.2 – Modified setup for many-bits fault injection


(a)
Campaign control Python script
Fault injection and diagnosis serial communication
Xilinx Zynq-7000 / 7 Series FPGA
MicroBlaze
Serial Communication Peripheral
(b)
Command Protocol Software
(c)
Fault Injector (FI)
Fault Injection FSM
(d)
Configuration Access Port (ICAPE2 hardware block)
On-board bitstream Flash
Checker/exerciser module
Power switch
DUT copy or precomputed result database (golden)
DUT

Source: The Author

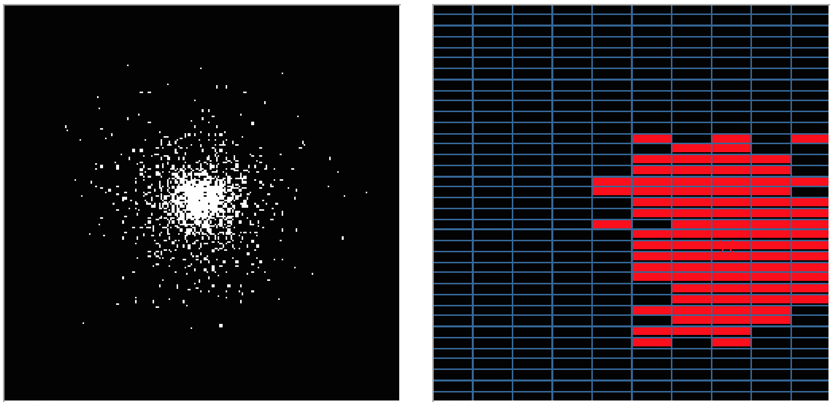
9.3 Methodology and scripting

At the experimental implementation of the methodology for laser attack emulation, the FPGA memory array was modeled as a rectangular structure and the fault was modeled by a bivariate normal distribution random pattern as in the example of Fig. 9.3a.

The process starts with a determined location for the fault in the die surface, in this case the location of the emulated laser spot. Then the position for a bit-flip is randomized around that fault location following the fault artifact geometric model, the bit-flip is injected at that position, the design under test (DUT) is stimulated for one or more processing cycles and the diagnosis is collected. The outcomes can be that the DUT is faulty or operates correctly, with or without triggering the fault attack detection mechanism implemented. Depending on the diagnosis, a new bit-flip is added to the same fault location at another randomized position, again following the fault artifact geometric model, growing the fault cluster of bit-flips and enlarging the spot diameter. This process is repeated until fault is detected, the DUT fails or the maximum spot diameter is reached. The bit-flips cluster is checked on each step to avoid repeating bit-flip positions so that each bit is hit only once and bit-flips are not reversed. Figure 9.3b shows an example of the realization of the fault artifact in the CRAM array.

If gradually increasing the emulated laser spot diameter is related to increasing the fault attack potential, then the size of the spot when the fault detection mechanism is triggered can be associated with the security assurance levels where that detection mechanism can be applied.

Figure 9.3 – An example of fault artifact geometric model



(a) Bivariate normal distribution **(b)** Randomized pattern on CRAM

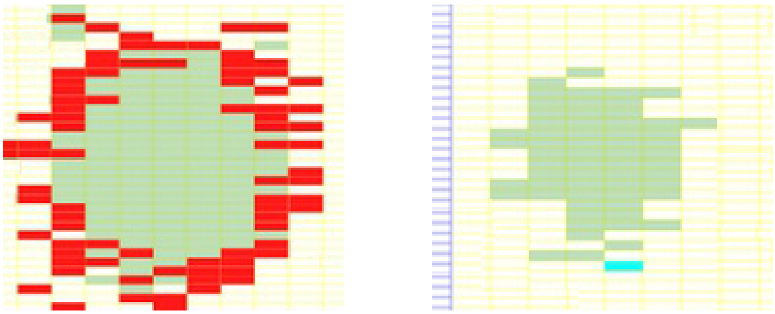
Source: The Author

9.4 Study case and results

This interframe many-bits fault injection methodology was applied to evaluate a fault attack detection mechanism permeating a cryptographic module implemented on a Xilinx 7 Series FPGA (BENEVENUTI et al., 2017). The design under test is the substitution box (S-box) from Rijndael symmetric cryptography and the Advanced Encryption Standard (AES). The attack detection mechanism is based on functional redundancy.

Example of Fig. 9.4a shows a situation where, at a given emulated laser spot size, the DUT starts to show functional failure (upon injecting bit-flips at the positions marked in red). In the example of Fig. 9.4b, at a certain emulated spot size, the detection mechanism is triggered by a bit-flip (at the position marked in blue) before the DUT starts to malfunction.

Figure 9.4 – Example of diagnosis of the design under test



(a) DUT functional failure with attack undetected **(b)** Fault detected before DUT functional failure

Source: The Author

10 ACCUMULATED FAULT INJECTION

10.1 Motivation

The result from the exhaustive-sequential fault injection methodology proposed by Tonfat et al. (2016) (Section 5.3) is a list of addresses for each critical bit in the CRAM. The number of critical bits can be used for direct comparison between alternative design implementations, with different levels of fault mitigation. By adequate design floorplanning or the reverse mapping of addresses to design modules, the list of critical bits can also be used to identify modules in the design where additional fault mitigation techniques can be introduced to improve the design reliability.

While the sequential-exhaustive method provides a detailed inventory with positions of each critical bit in the DUT, the reverse mapping from these positions to high-level design modules is cumbersome and the critical bits positions are prone to change every time the design is modified or synthesize with a new starting condition to the place and route step.

One limitation of this exhaustive-sequential methodology is that it is not comparable with the results from irradiation tests because it does not reflect the cumulative effect of the faults over time.

Based on the results of the many-bits fault injection presented previously, it was proposed to inject faults in CRAM in a randomized-accumulated process.

Different from the test methodology in Chapter 9, where bit-flips were injected randomly following a bivariate normal distribution growing a cluster around a focused location, in the random-accumulated methodology bit-flips are injected in positions randomized uniformly over the whole FPGA or over a constrained region of interest at the FPGA floorplan where the DUT was implemented.

The randomized-accumulated methodology does not provide an exhaustive list of critical bits, but it can still provide an estimate of the number of critical bits and, with adequate design floorplanning, it can indicate modules deserving additional fault mitigation techniques. The results from randomized-accumulated methodology can be used to build a reliability curve in the form of the empirical cumulative probability density function (CDF), using the same methods adopted for irradiation tests.

10.2 Implementation

The target of the fault injection was still only the configuration memory (CRAM) and required only minimal modifications in the fault injection module implemented by Tonfat et al. (2016), mostly to improve serial communication speed. On the other hand, the fault injection campaign scripting was intensively modified to implement the new methodology.

10.3 Methodology and scripting

Since faults in CRAM are mostly persistent in the absence of memory scrubbing, the fault injection operated synchronously with the DUT execution. The DUT is stimulated for one or more processing cycles to collect diagnosis after each fault is injected, as occurs in the method of Chapter 9 and in the original implementation of Tonfat et al. (2016).

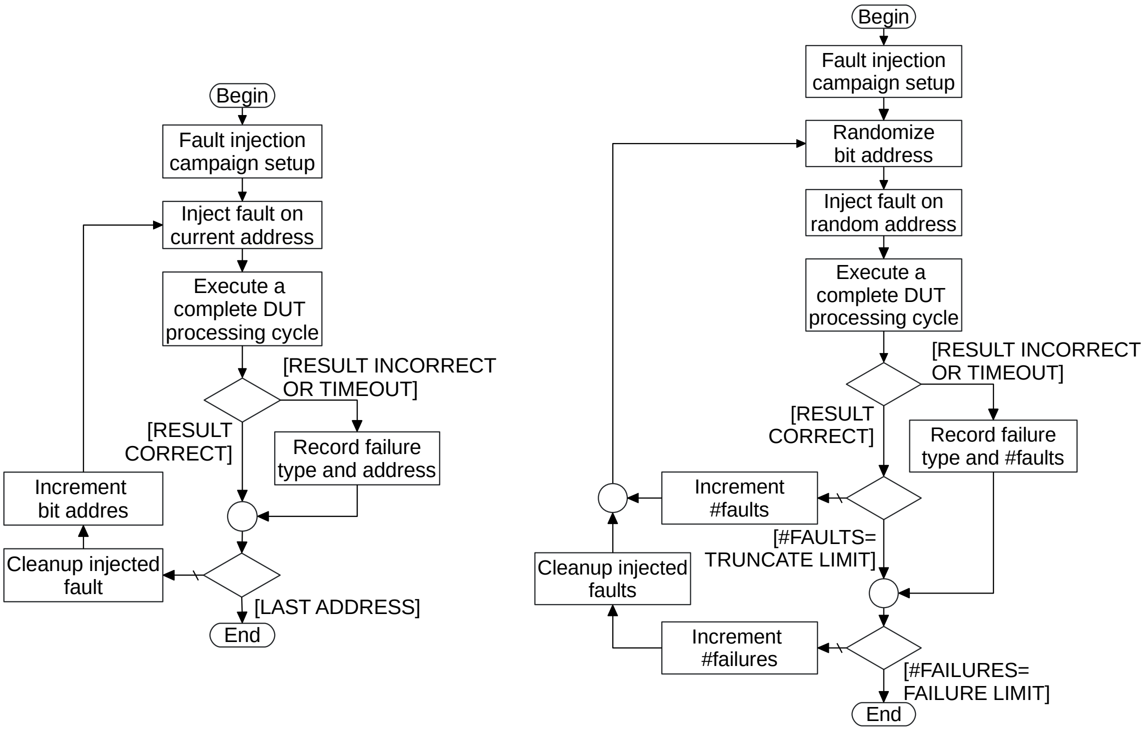
In the random-accumulated method, faults in CRAM are accumulated over time, as occurs in radiation-induced SEUs, representing a better approximation of the physical phenomena compared to the exhaustive-sequential test methodology of Tonfat et al. (2016) present in the original implementation of the fault injector. Figure 10.1 shows the main procedure of the two fault injection processes.

One important difference between the two methodologies is that in the sequential-exhaustive methodology every fault injected must be cleaned after the DUT diagnosis. It is done mainly by injecting fault again in the same address, thus reversing the bit-flip. After the fault is cleaned, the DUT diagnosis must be run again to confirm that it is working properly. However, in some situations, injecting fault again in the same place cannot revert the effect of the bit-flip. One example of such situation is when the fault is injected in the control bits for the shift register over LUT in the CLB. In those cases, the FPGA must be reprogrammed to fully revert the fault injected.

10.4 Study case and results

This random-accumulated methodology was applied to investigate fault tolerance and reliability in several types of designs, including computing accelerators (SANTOS

Figure 10.1 – Two fault injection test approaches


Begin
Fault injection campaign setup
Inject fault on current address
Execute a complete DUT processing cycle
[RESULT INCORRECT OR TIMEOUT]
[RESULT CORRECT]
Record failure type and address
Increment bit addres
Cleanup injected fault
[LAST ADDRESS]
End
Begin
Fault injection campaign setup
Randomize bit address
Inject fault on random address
Execute a complete DUT processing cycle
[RESULT INCORRECT OR TIMEOUT]
[RESULT CORRECT]
Record failure type and #faults
Increment #faults
[#FAULTS= TRUNCATE LIMIT]
Cleanup injected faults
Increment #failures
[#FAILURES= FAILURE LIMIT]
End

(a) Sequential-exhaustive fault injection

(b) Random-accumulated fault injection

Source: The Author

et al., 2017; BENEVENUTI et al., 2019b; RODRIGUES et al., 2019), communication infrastructures (BENEVENUTI et al., 2018b,a), softcore microprocessors (GONCALVES et al., 2020; BRAGA et al., 2021) and machine learning inference engines (BENEVENUTI et al., 2018c; LOPES et al., 2018; TRINDADE et al., 2021).

Two main advantages of the randomized-accumulated methodology are the better comparability with irradiation tests by the use of the reliability curves and the aggressive reduction of fault injection campaign time.

The random-accumulated method does not provide the positions of every critical bit, but it provides an estimate of the number of such bits that may be enough to steer the fault tolerance and reliability engineering process. Moreover, and of special interest in the scope of this thesis, random-accumulated fault injection is significantly faster than the sequential-exhaustive approach.

A study case design of neural network (NN) (BENEVENUTI et al., 2018c, 2019a) was used to compare results from irradiation, sequential-exhaustive methodology and the new randomized-accumulated methodology.

As faults are injected twice on each address, once to test and again to clean up the fault, and the DUT must be executed twice, once to collect diagnosis and again to verify the fault clean-up, the rate of fault injection is naturally lower in sequential-exhaustive methodology than in the randomized-accumulated methodology.

For the study case design, a rate of 12 faults injected per second was obtained for the sequential-exhaustive methodology while the randomized-accumulated methodology reached a rate of 36 faults injected per second.

However, what makes the test campaign time for sequential-exhaustive exceedingly longer compared to randomized-accumulated is its exhaustive nature where every bit in the region of interest must be tested. In the randomized-accumulated, conversely, a randomized sampling of the region of interest requires only a fraction of the time. Additionally, upon discretion of the engineer, more sampling can be added gradually in the randomized-accumulated methodology until the adequate level of uncertainty is obtained in the reliability curve.

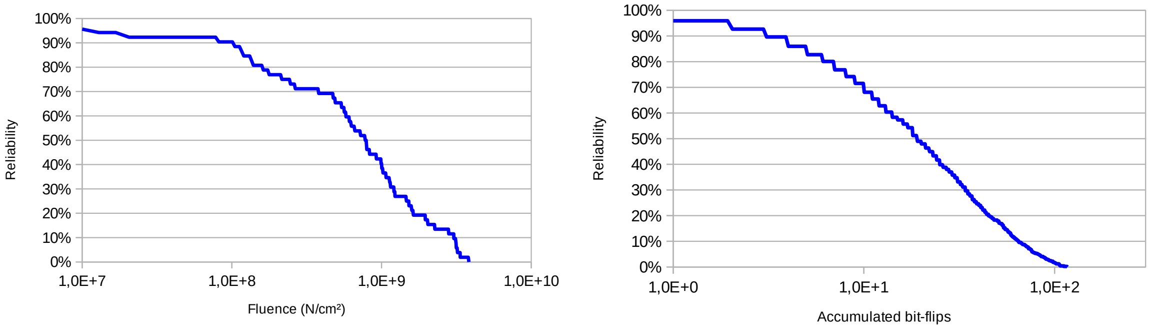
For the study case design, the investigation of reliability of the four main design modules in five different design floorplans took 142 hours of fault injection campaign while the randomized-accumulated methodology took less than 4 hours, being over 36 times faster than the previous methodology. Table 10.1 summarizes the information gathered with the fault injection campaigns for the study case design. The difference between the number of faults injected and the estimated number of bits on the design is related to different heights of the physical block and the configuration memory frame. While in exhaustive campaigns we injected a total of 1.5×10^7 faults, in the random campaigns we injected a total of 5×10^5 faults.

The study case design was also tested under a spallation neutron source at the Los Alamos Neutron Science Center (LANSCE) ICE House facility (BENEVENUTI et al., 2018c) under an average flux of 6.2$\times$10^9 N/cm^2/h in 13 hours of neutron irradiation campaign. After discarding occasional time periods where beam was off and other spurious data, we obtained N_{SEU}=52 failure events.

From the fluence for each individual failure event we can obtain the empirical reliability curve presented on Fig. 10.2a. In a similar process, from the number of accumulated faults for each individual failure event under fault injection we can obtain he reliability curve on Fig. 10.2b

For this specific design, with no significant presence of redundancy, the reliability can be modeled by an exponential distribution described by a constant failure rate λ that, when considering no relevant reposition time, can be obtained directly from mean time to failure (MTTF) as $\lambda = 1/MTTF$. In this case, the cumulative failure can be expressed as

Figure 10.2 – Reliability curves for study case design



(a) Neutron irradiation (LANSCE) **(b)** Fault injection (randomized)

Source: The Author

$F(t) = 1 - e^{-\lambda t}$. The reliability can be computed as its complement as $R(t) = 1 - F(t)$. Using the number of accumulated faults as a time dimension, we can compute MTTF as total number of accumulated injected bit-flips τ over the number of functional failure events N observed during each campaign.

No advanced curve fitting tool is required for an exponential distribution. To other designs, however, the exponential distribution may not be the best choice for modeling the system reliability. Alternatively, metrics can be extracted directly from the empirical reliability curve (CDF).

Using the obtained parameters for the statistical distribution, we can take the failure rate at exactly one fault accumulated $F(t = 1)$ to estimate the probability of critical bits. This can, in turn, be multiplied by the total number of bits on the target area of the random fault injection to estimate the number of critical bits, as seen in Table 10.1.

As presented in Table 10.1 and Fig. 10.3, the estimated number of critical bits, using results from random-accumulated fault injection, is consistent with the number of critical bits obtained from exhaustive-sequential fault injection.

As the estimation of the number of critical bits using results random fault injection are based on the statistical distribution parameters and the system behavior on the high reliability region only, for reliability engineering processes based on critical bits only, further acceleration can be obtained with experiment censoring (truncation), which does not prevent extraction of the relevant statistical parameters for the estimates.

We also compared the exhaustive and random fault injection in terms of the qualitative result of the study case design represented by the type of error observed. Fig. 10.4 show that the two methodologies present very similar profiles of error type in the study case design.

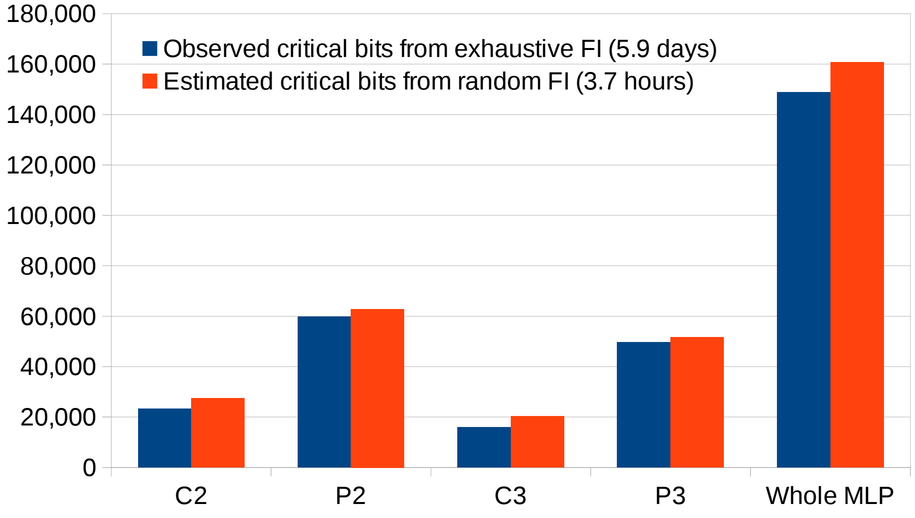
Table 10.1 – Design Area Assessment and Fault Injection Results

	Design Module				**Whole**
	C2	***P2***	***C3***	***P3***	**Design**
Resources Usage					
Essential bits	–	–	–	–	2,567,227[c]
Estim. phys. block bits[a]	889,446	1,791,821	667,085	2,785,984	5,014,771[c]
Occupancy[b]	52%	98%	56%	92%	98%[c]
SLICEs	331	1,233	267	1,831	3,543[c]
LUTs	1,192	3,735	918	5,428	11,274
FFs	–	64	–	96	160
DSPs	32	–	24	–	56
Exhaustive Fault Injection					
Faults injected	1,104,744	1,977,145	1,103,985	5,498,826	–
Campaign time (hours)	19	41	17	65	–
Critical bits	23.328	59.818	15.979	49.756	148.881
% of Critical bits[d]	2.6%	3.3%	2.4%	1.8%	3.0%
Random Fault Injection					
Faults injected (τ)	211,125	33,287	112,314	126,947	17,585
Failures observed (N)	1,011	367	400	1,150	500
MTTF (fault,τ/N)	208.8	90.7	280.8	110.4	35.2
Failure rate (λ)	0.0048	0.011	0.0036	0.0091	0.028
Failure rate $(F(t=1))$	0.5%	1.1%	0.4%	0.9%	2.8%
Estim. critical bits[e]	27.272	62.595	20.295	51.481	160.032
% of Critical bits[d]	3.1%	3.5%	3.0%	1.8%	3.2%

[a] Estimated CRAM bits on the floorplan physical block containing the layer. [b] Slices occupancy for the physical block. [c] Whole neural network implemented on a single physical block. [d] Relative to the estimated CRAM bits. [e] Computed from 5,708,732 bits on FI target area from where λ was extracted.

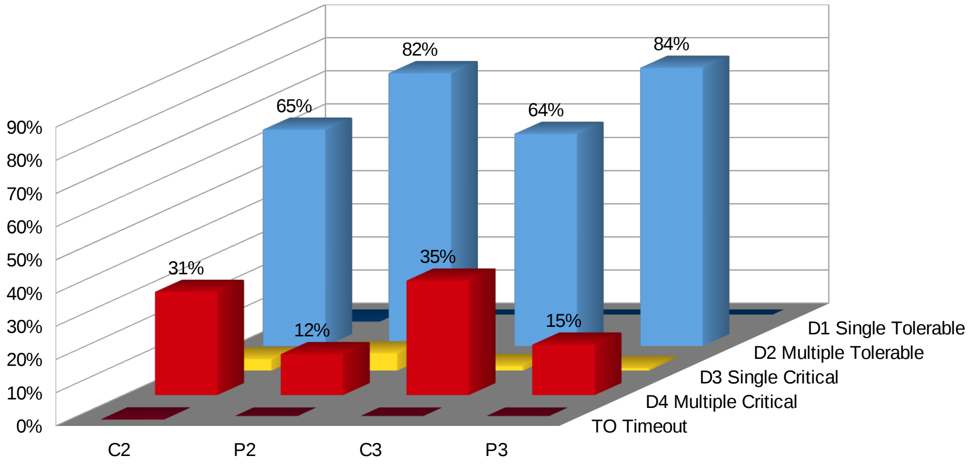
Source: The Author

Figure 10.3 – Critical bits comparison.

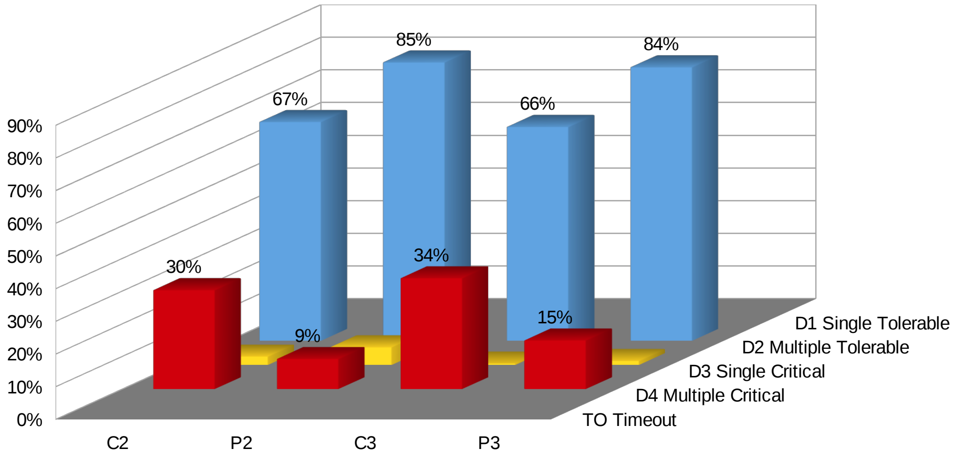


Source: The Author

Figure 10.4 – Comparative failure distribution by neural network layer and failure criticality.



(a) Failures from exhaustive-sequential FI



(b) Failures from random-accumulated FI

Source: The Author

11 ASYNCHRONOUS FAULT INJECTION

11.1 Motivation

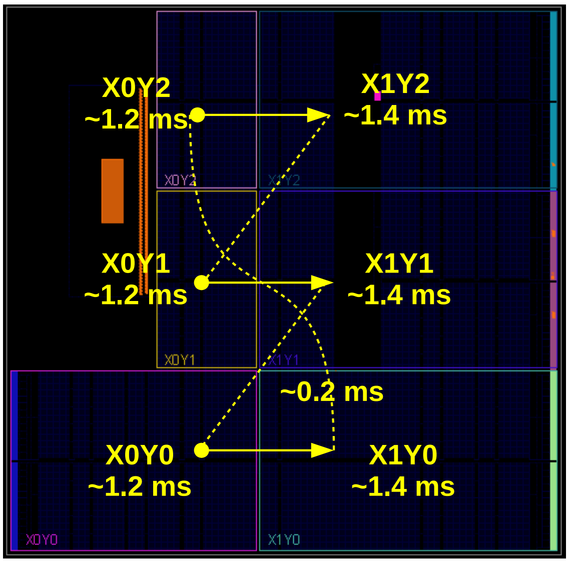
As mentioned previously, an assumption in the original implementation of the UFRGS fault injector for Xilinx 7 Series was that faults in CRAM are persistent and, consequently, the fault injector could operated synchronously starting the DUT processing cycles and diagnosis after each fault is injected.

What happens in the presence of the memory scrubber hardware native to the Xilinx 7 Series devices is that faults can be corrected at any moment in time because the memory scrubber operates autonomously and asynchronous with the DUT.

Even if a bit in CRAM is a critical bit, a fault in that bit will remain latent until the point in time inside the DUT processing cycle when that part of the circuit contributes to the DUT result. If the scrubber has the chance to correct the fault during that period in which it is latent, then the malfunction will be avoided.

The Xilinx memory scrubber operates over the memory frames in the CRAM, starting from the first frame address and scanning continuously until the end of the memory or a fault is found. Figure 11.1 shows an example of the operating cycle of the Xilinx scrubber in a Xilinx Zynq-7000 XC7Z020 device with approximate time it takes to scan each clock region with the scrubbing clock at 100 MHz, adding up to about 8 ms to scan the whole device in the absence of faults.

Figure 11.1 – Example of scrubber scan cycle



Source: The Author

If a fault is found and the scrubber succeed in correcting it, the scrubber does not continue to the next frame address but, instead, rewind to the first frame address and restart the scrubbing from the beginning.

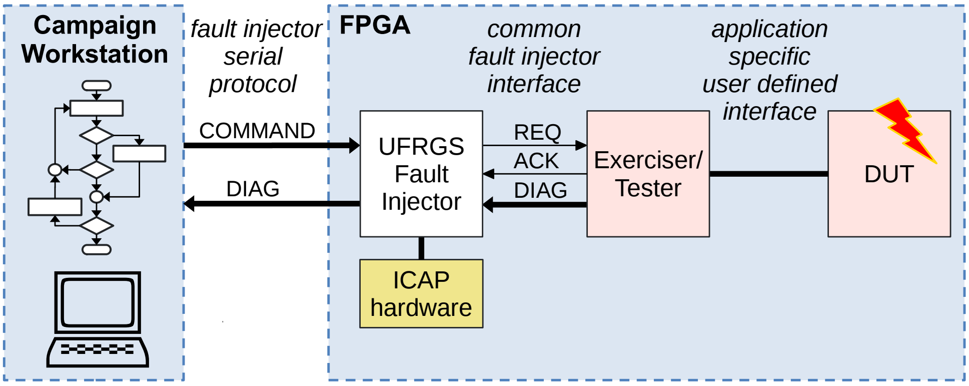
The Xilinx memory scrubber hardware automatically suspends its operations when the UFRGS fault injector activates the ICAP hardware to inject a fault. The scrubber automatically restarts its operation once the fault injection is completed and the ICAP hardware is deactivated. However, the scrubber always restarts its operation from the first frame address, independent of the frame address were it was working in the moment is was suspended.

With the synchronous operation in the original implementation of the UFRGS fault injector, the fault would always be corrected in the time window of the scan cycle of the scrubber starting from the point in time when the fault injector released the ICAP hardware and started the DUT processing cycle. Consequently, if the point in time when the latent fault would cause an error is after that time, the fault will always be corrected.

11.2 Implementation

The modular organization of the UFRGS fault injector is shown in Fig. 11.2 with a common handshake interface between the fault injector and the DUT exerciser, which is used to start the DUT processing and collect diagnostic data every time a fault is injected as a consequence of commands issued by the campaign workstation.

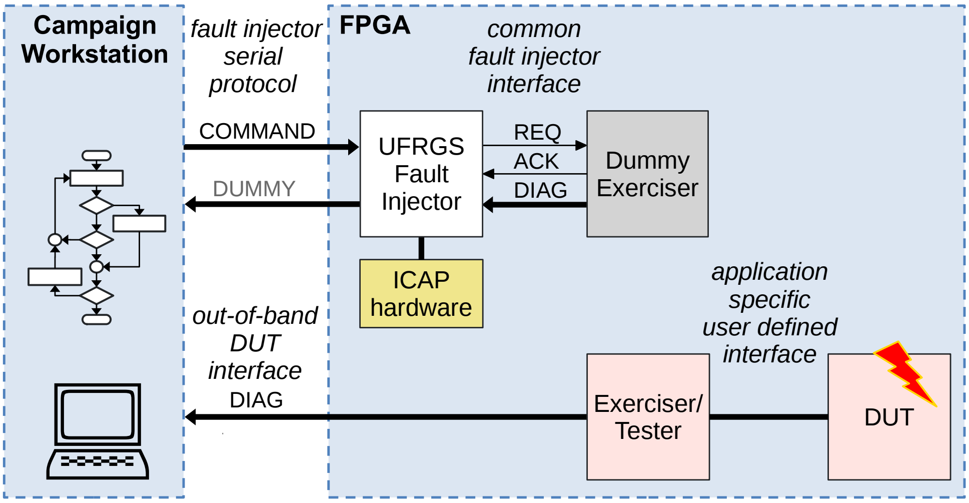
Figure 11.2 – Fault injector operating in synchronous mode


Campaign Workstation
fault injector serial protocol
FPGA
common fault injector interface
application specific user defined interface
COMMAND
DIAG
UFRGS Fault Injector
REQ
ACK
DIAG
Exerciser/ Tester
DUT
ICAP hardware

Source: The Author

As the synchronous operation of the fault injector is faster and still useful in scenarios where memory scrubber is not used, it was opted to do only minor changes in the fault injector maintaining the functionalities of the common handshake interface and serial protocol. It was accomplished by using a dummy exerciser interfacing with the fault injector that responds instantly to requests with a null diagnosis. A side channel is used to collect the diagnostic data from the real DUT exerciser that operates independently. Figure 11.3 shows a fault injection setup for asynchronous fault injection. This approach allows the reuse of a same fault injector source code in different modes of operation.

Figure 11.3 – Fault injector operating in asynchronous mode


Campaign Workstation
fault injector serial protocol
FPGA
common fault injector interface
COMMAND
DUMMY
UFRGS Fault Injector
REQ
ACK
DIAG
Dummy Exerciser
ICAP hardware
application specific user defined interface
out-of-band DUT interface
DIAG
Exerciser/ Tester
DUT

Source: The Author

11.3 Methodology and scripting

With the asynchronous fault injection, the campaign script injecting faults to emulate radiation and the script monitoring the DUT diagnosis can work as two completely independent processes, occasionally in different campaign workstations.

The fault injection script operates with a given targeted rate of faults that must take into account the scrubbing period and the DUT processing time. As faults are injected now asynchronously, the interval time between faults must be larger than the DUT processing cycle time to ensure that there is no more than one fault per processing cycle. The interval time between faults must also be larger than the scrubbing period.

It is important that the interval time between faults has a random component to ensure that faults will be inject, and corrected, at any point in time during the processing cycle. As a general rule one can use a fixed parcel of interval that is larger than both the DUT processing time and the scrubber period plus another parcel that is a random fraction of that time. More sophisticated fault rates can be implemented, for instance modeling the interval between faults as a Poisson distribution.

11.4 Study case and results

A random-accumulated and asynchronous fault injection methodology was applied in the study of fault tolerance and reliability of designs where memory scrubbing was used in conjunction of fault mitigation techniques than can mask temporarily faults in CRAM, such as triple modular redundancy (TMR). Designs under analysis included the Arm Cortex-M0 softcore microprocessor (BENITES et al., 2019) and convolutional neural networks for image classification (BENEVENUTI et al., 2019c, 2021).

Asynchronous fault injection, operating isolated as an irradiation facility simulator, was also used successfully to verify and validate design instrumentation, hardware setup and radiation test scripts before sending to the radiation facility.

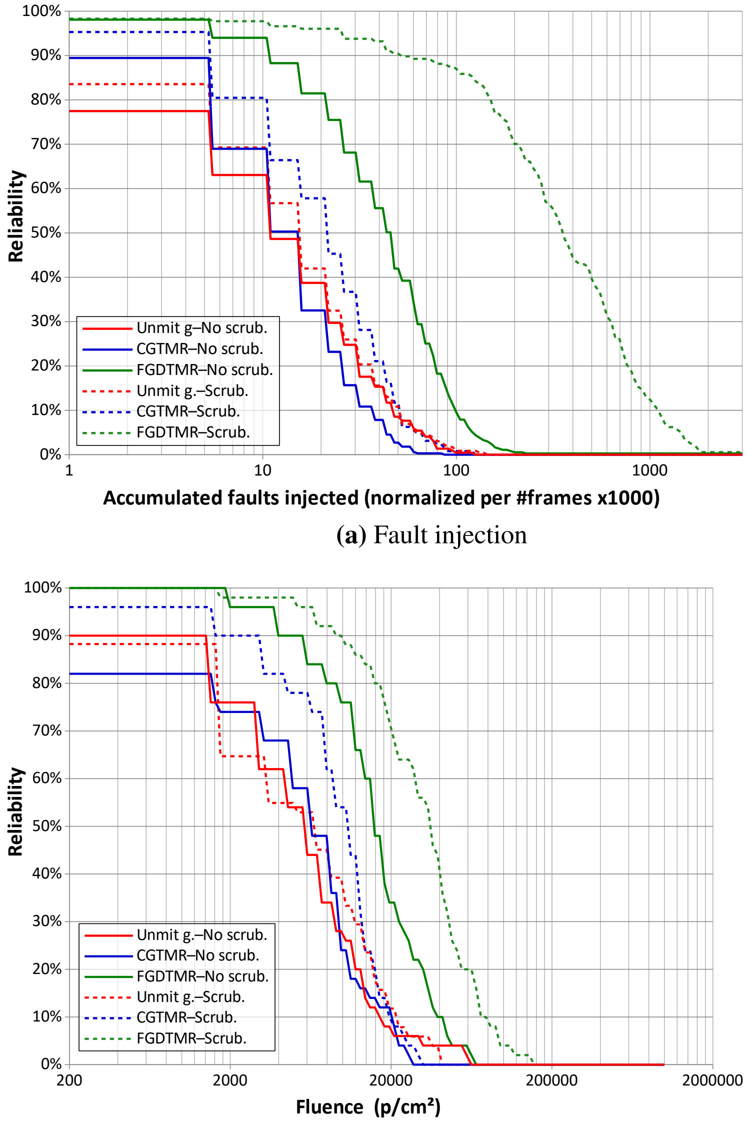
The asynchronous fault injection was essential at the analysis of the study case design of softcore microprocessors by Benites et al. (2019) because it required the comparison of the mitigation strategies based on memory scrubbing associated with different levels of modular redundancy, ranging from coarse-grain TMR (CGTMR) to fine-grain distributed TMR (FGDTMR).

The software tasks in the softcore microprocessor had an execution period of approximately $60\,\mathrm{ms}$. For the FPGA device present in the experimental board, at the given configuration clock, the memory scrubbing period was approximately $8\,\mathrm{ms}$ in the absence of bit-flips in CRAM. It was adopted an interval of $150\,\mathrm{ms}$ between faults injected, being a non-integer multiple of the DUT processing cycle time. Figure 11.4a presents the reliability curves obtained from fault injection for each DUT configuration.

The study case design was also tested with heavy ions at the SAFIIRA facility from the tandem electrostatic 8UD Pelletron accelerator at the LAFN-USP, Brazil (AGUIAR et al., 2020). The experimental board was irradiated in-vacuum using a $39\,\mathrm{MeV}$ ^{16}O beam at 0° normal incidence. By preliminary static tests, the beam flux was adjusted to produce a bit-flips rate around $5\,\mathrm{s}^{-1}$ in the entire FPGA. During the tests, the regular beam flux was from $2.3 \times 10^2\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$ to $3.2 \times 10^2\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$.

The study case design was tested under radiation with different levels of modular redundancy, with and without memory scrubbing. These different DUT configurations were irradiated accumulating a total of approximately 6 hours of beam time, during which it was acquired at least 50 failure events for each DUT configuration. For comparison, Fig. 11.4b presents the reliability curves obtained from heavy ions irradiation for each DUT configuration.

Figure 11.4 – Observed reliability under fault injection and radiation



(a) Fault injection

(b) Heavy ions

Source: The Author

From Fig. 11.4, it is noteworthy that, for some configurations, the effect of memory scrubbing was relatively less expressive under radiation when compared to emulated fault injection results. A reliability metric relevant for analysis in this context is the error rate from fault injection $\tau_{failure}$ (Eq. 4.10), which is presented in Fig. 11.5a, and its relation to the dynamic cross section $\sigma_{failure}$ (Eq. 4.1), which is presented in Fig. 11.5b.

The three quantities for static cross section σ_{SEU}, dynamic cross section $\sigma_{failure}$ and error rate from fault injection $\tau_{failure}$ all convey a sense of probability. The inverse of the static cross section gives how many particles are required to obtain an SEU, whereas the inverse of the dynamic cross section provides the number of particles to occur a functional failure. Finally, the inverse of the fault injection error rate gives how many injected faults (i.e., emulated SEUs) are required to obtain a functional failure.

Given that the same device and similar irradiation conditions were used on all heavy ions' experiments presented above, we can consider the static cross section σ_{SEU} as a constant. Also, for all fault injection campaigns, bit-flips were injected on the same fault density for all design variants. It would be expected, then, that $\sigma_{failure}$ and $\tau_{failure}$ could be related by a proportionality constant κ as seen in Eq. 4.11, which can be simplified as in Eq. 11.1 given that σ_{SEU} here is also a constant.

$$\kappa = \frac{\sigma_{failure}}{\tau_{failure}}. \tag{11.1}$$

Considering the existence of such scale factors between $\sigma_{failure}$ and $\tau_{failure}$, it is straightforward to identify and understand the similarities between charts in Fig. 11.5a and Fig. 11.5b. However, comparing these charts, one can also observe a greater discrepancy in the case of the configuration FGDTMR with scrubbing, as seen in Fig. 11.5c.

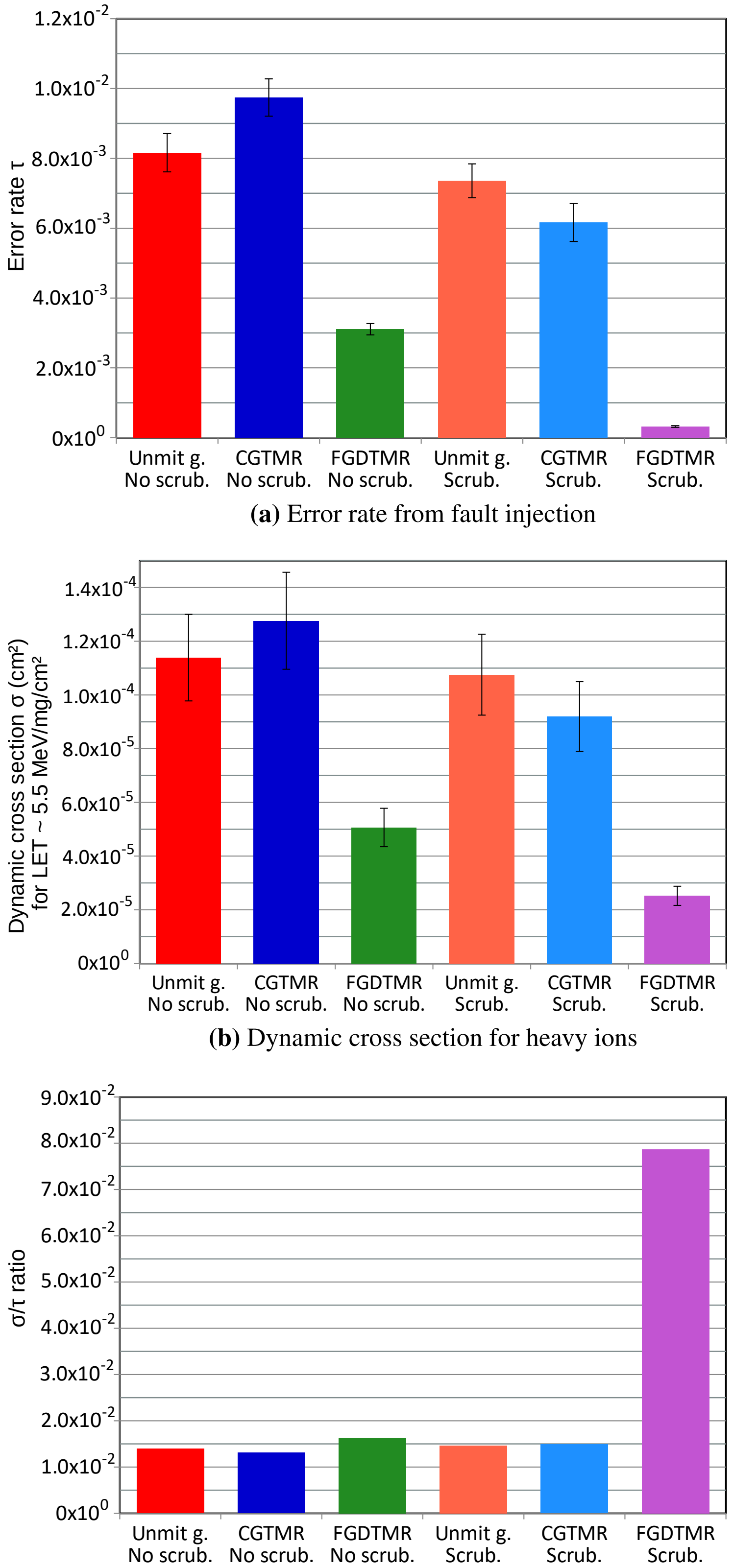
The observed dynamic cross section for FGDTMR with scrubbing is higher than the expected considering the results of fault injection. The causes of this discrepancy are yet to be confirmed, but it can be conjectured that the mitigation with FGDTMR and scrubbing raised the design reliability such that the FPGA configuration memory ceased to be the weakest link on the system.

In this case, the upper bound for reliability would now be defined by other block components inside the FPGA that was not targeted by this implementation of fault injection, such as clock tree, power-on-reset, configuration blocks, scrubbing engine, or even user data on block memory and flip-flops.

Is is also noteworthy that this implementation of the fault injector emulated only single-bit upsets (SBU) while the occurrence of multiple-bit upsets (MBU), specially when the bit-flips occur in the same CRAM frame, have a great impact over the memory scrubber. Multiple-bit upsets also have the potential to disrupt TMR in the cases where the redundant modules are not physically isolated in the FPGA floorplan, which is impracticable for FGDTMR.

These results suggests further improvements in the fault injector, first extending the reach of fault injection to other FPGA resources, notably memory blocks, and then emulating multiple-bit upsets.

Figure 11.5 – Observed reliability under fault injection and radiation



(a) Error rate from fault injection

(b) Dynamic cross section for heavy ions

(c) Ratio between dynamic cross section and fault injection error rate

Source: The Author

12 FAULT INJECTION ON MEMORY BLOCKS

12.1 Motivation

Different from the contents of the configuration memory (CRAM), which is typically persistent, the contents of block memory (BRAM) holding user data is produced and updated dynamically at runtime.

As faults in user data are subject to temporal masking, fault injection on BRAM frames should also be asynchronous so that it can occur at any point in time during the DUT processing cycle, as already seen in Chapter 11.

Fault injection on BRAM can occur concomitantly or not with fault injection on CRAM, depending on what the engineer wants to analyze in the DUT. To emulate the radiation process, the random position of faults should be distributed uniformly over the BRAM bits in the region of interest. The same must occurs with CRAM bits. However, if BRAM and CRAM have different static cross sections for the targeted radiation environment, then the cross-section must be taken into account in the randomization process. In static tests it was also observed that the geometry of MBU clusters in BRAM have a different distribution from CRAM, possibly due to interleaving and higher density in the memory array for BRAM blocks.

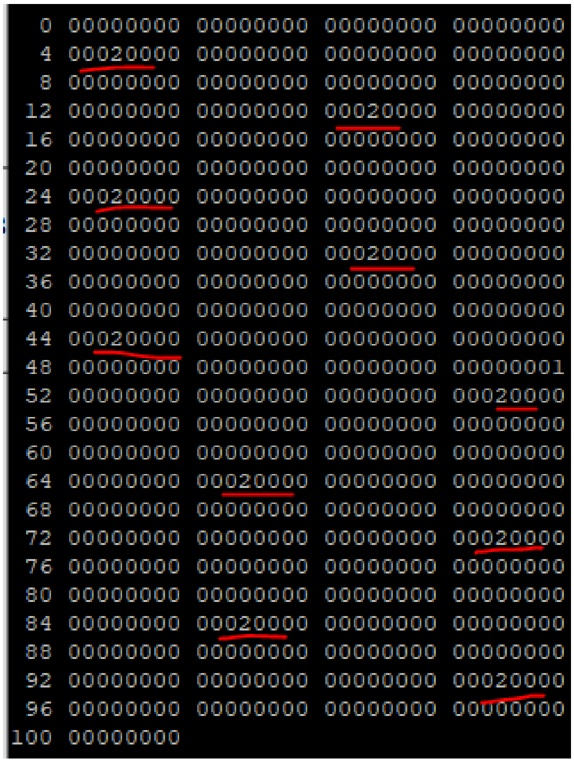
12.2 Implementation

In the implementation of the fault injection in BRAM frames, one first modification on the fault injector code is related to the BRAM flags that are set in the frame readback process (Fig. 12.1) and must be masked before writing the frame with bit-flip into the BRAM. This is a characteristic already documented in the literature, for instance by Gomez-Cornejo et al. (2017), and does not characterize reverse engineering violating the licensing and use terms of Xilinx products.

If these flags (Fig. 12.2) are not masked the frame is not accepted and the fault is not injected. The masking does not apply to CRAM frames, but BRAM and CRAM can be easily distinguished by the type of block field in the frame address as described in Table 6.1.

Another aspect that must be taken into account is the interference of the frame manipulation in the interface of the block memory hardware as seen by the design at the

Figure 12.1 – Examples of BRAM frame bits set during readback



```
  0 00000000 00000000 00000000 00000000
  4 00020000 00000000 00000000 00000000
  8 00000000 00000000 00000000 00000000
 12 00000000 00000000 00020000 00000000
 16 00000000 00000000 00000000 00000000
 20 00000000 00000000 00000000 00000000
 24 00020000 00000000 00000000 00000000
 28 00000000 00000000 00000000 00000000
 32 00000000 00000000 00020000 00000000
 36 00000000 00000000 00000000 00000000
 40 00000000 00000000 00000000 00000000
 44 00020000 00000000 00000000 00000000
 48 00000000 00000000 00000000 00000001
 52 00000000 00000000 00000000 00020000
 56 00000000 00000000 00000000 00000000
 60 00000000 00000000 00000000 00000000
 64 00000000 00020000 00000000 00000000
 68 00000000 00000000 00000000 00000000
 72 00000000 00000000 00000000 00020000
 76 00000000 00000000 00000000 00000000
 80 00000000 00000000 00000000 00000000
 84 00000000 00020000 00000000 00000000
 88 00000000 00000000 00000000 00000000
 92 00000000 00000000 00000000 00020000
 96 00000000 00000000 00000000 00000000
100 00000000
```

Source: The Author

logical level. To the design the memory block typically operates with an input address, input data and output data port (Fig. 3.3). Values seen at the output data port is the logical content of the memory at the given address. However, when the ICAP hardware accesses a frame inside the BRAM, this interferes with the normal operation of the memory block hardware changing the form how the address port is seen or the form how the output port is populated, causing an incorrect value to be seen by the application and, ultimately, a functional failure.

To cope with this behavior of the memory blocks, we considered the use of the mechanism of capturing and restoring registers in the region of interest must be used during the fault injection. In Xilinx 7 Series, the BRAM blocks have internal registers which can optionally be enabled. It was conjectured that the capture operation could copy the contents of such registers to the bitstream image, before the fault injection, to be copied back into the BRAM by a restore operation, after the fault injection.

To check the validity of this assumption, first we implemented several test designs with instances of BRAM with placement constrained to physical blocks marked with the attribute `RESET_AFTER_RECONFIG` (RAR). By moving BRAM instances along he FPGA floorplan it was possible to devise the values in the global reset mask associated with each BRAM column. In the following this reset mask was used before the restore

Figure 12.2 – Masking of BRAM bits set on frame readback for Xilinx 7 Series

```
...

constant MAGIC_BIT_HIGH_MASK : std_logic_vector (31 downto 0) := X"00020000";

...

frame_is_bram_s <=
  true when ((faddr_to_icap_reg and BLOCK_TYPE_BRAM_MASK) = BLOCK_TYPE_BRAM_MASK ) else
  false;

word_addr_is_magic_s <=
  true when ( (frame_mem_addr = X"04") and
        ( bit_pos >=    0 ) and ( bit_pos <=  319 ) ) else
  true when ( (frame_mem_addr = X"0E") and
        ( bit_pos >=  320 ) and ( bit_pos <=  639 ) ) else
  true when ( (frame_mem_addr = X"18") and
        ( bit_pos >=  640 ) and ( bit_pos <=  959 ) ) else
  true when ( (frame_mem_addr = X"22") and
        ( bit_pos >=  960 ) and ( bit_pos <= 1279 ) ) else
  true when ( (frame_mem_addr = X"2C") and
        ( bit_pos >= 1280 ) and ( bit_pos <= 1599 ) ) else
  true when ( (frame_mem_addr = X"37") and
        ( bit_pos >= 1632 ) and ( bit_pos <= 1951 ) ) else
  true when ( (frame_mem_addr = X"41") and
        ( bit_pos >= 1952 ) and ( bit_pos <= 2271 ) ) else
  true when ( (frame_mem_addr = X"4B") and
        ( bit_pos >= 2272 ) and ( bit_pos <= 2591 ) ) else
  true when ( (frame_mem_addr = X"55") and
        ( bit_pos >= 2592 ) and ( bit_pos <= 2911 ) ) else
  true when ( (frame_mem_addr = X"5F") and
        ( bit_pos >= 2912 ) and ( bit_pos <= 3231 ) ) else
  false;
...
```

Source: The Author

operation to avoid interference with the fault injection itself.

Unfortunately, experiments evidenced that the capture and restore mechanism had no effect on the registers inside the BRAM block. In fact, Xilinx technical documentation (XILINX, 2019) indicates that BRAM blocks have configuration for initial values and reset values for registers, but those reset values are loaded upon BRAM specific reset signal (RSTRAM/RSTREG) while BRAM block has no connection to the global restore signal. However, the manipulation of the BRAM specific reset signal also could not solve the problem as the configured values does not seem to be affected by the capture operation.

More recently, a similar strategy of BRAM manipulation was presented by Gomez-Cornejo et al. (2022) in an implementation of memory scrubbing for BRAM blocks, again suggesting the feasibility of reading and writing BRAM frames without disrupting the design in execution. The authors have indicated, however, that they have not analyzed the content of the BRAM output ports during the manipulation of the memory contents (BARRENA, 2022), suggesting that their BRAM scrubbing mechanism was tested only with persistent data stored inside BRAM, such as constant data tables or firmware object code.

Notwithstanding, out implementation of fault injection in BRAM block can still be used with such persistent data, although its usefulness is very limited and does not contribute to the emulation of radiation effects on BRAM for data subject to temporal masking.

12.3 Methodology and scripting

The fault injection emulating persistent faults on BRAM works similar to the fault injection methodology presented on Chapter 10. The main difference is that it must invoke an specific command of the fault injector because it uses a different frame type for BRAM (Table 6.1) and the BRAM control bits must be masked (Fig. 12.2).

Other BRAM fault injection methodologies can be tailored to evaluate error detection and correction codes as well as policies for data scrambling and data interleaving.

12.4 Study case and results

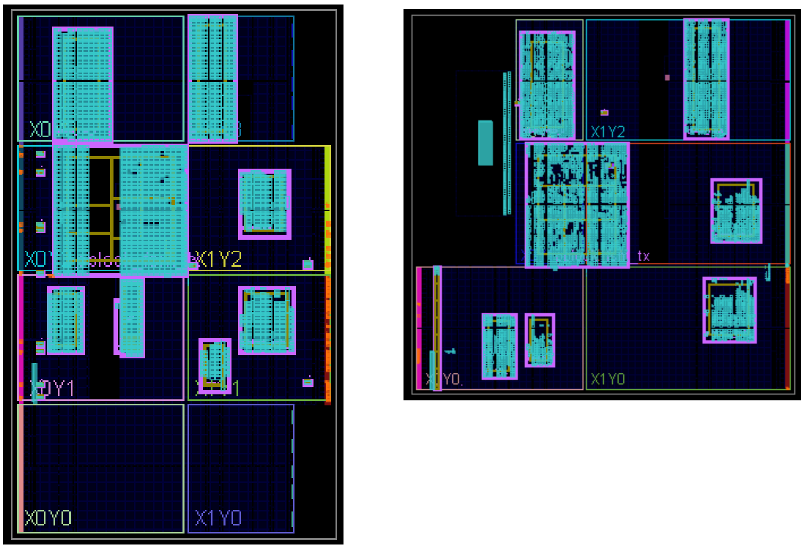
As study case design for fault injection on BRAM we adopted the investigation of the effects of software implemented hardware fault tolerance (SIHFT) in the benchmark application embedded on the INPE-SMDH/UFSM-UFRGS payload board for NanosatC-BR2 CubeSat mission described briefly in Appendix C.

The design implemented in the SRAM-based FPGA comprises seven main modules, being one supervisory and six versions of the benchmark application.

In Benevenuti et al. (2019b) we investigated the hardening of the supervisory module by the use of triple modular redundancy (TMR) with different granularities. For that, we used randomized-cumulative fault injection (Chapter 10) over each of the seven modules, individually, as well as over the whole FPGA design. The focus was then on the supervisory module but we also collected diagnosis data about the benchmark application. During these experiments it was collected more 250 failure events in the set of applications.

A similar experiment was performed by heavy ions irradiation at the SAFIIRA facility from the tandem electrostatic 8UD Pelletron accelerator at the LAFN-USP, Brazil (AGUIAR et al., 2020). We ported the NanosatC-BR2 design from Xilinx Artix-7 floorplan (Fig. 12.3a) to fit the Xilinx Zynq-7000 used in the radiation experiment (Fig. 12.3b). The input pins selecting the active applications (`APP_EN_*`) were configured for pull-up at the Xilinx device input/output block (IOB), hence all the applications were active. The clear to send (`CTS`) signal is active low, hence the input pin was configured for pull-down.

Figure 12.3 – Floorplan of NanosatC-BR2 benchmark application



(a) Xilinx Artix 7 A100T FPGA **(b)** Xilinx Zynq-7000 Z020

Source: The Author

The FPGA was irradiated in-vacuum using a $73.5\,\mathrm{MeV}$ ^{28}Si and $43.2\,\mathrm{MeV}$ ^{16}O beams, amounting to a total fluence of $7.8 \times 10^{6}\,\mathrm{cm}^{-2}$. During these experiments it was collected more 100 failure events in the set of applications.

As the soft-core microprocessors state is stored into flip-flops on the FPGA CLB while machine code is stored into BRAM, neither covered by the fault injector presented in (Chapter 10), it was not surprising that there was no occurrences of status message for `MIPSH_DUE` or the `MBH_DUE` (Table C.1) during fault injection.

The use of hardening techniques originally designed for hard-core microprocessors does not always produces relevant effects when applied to soft-core microprocessors because the higher volume of CRAM, and its susceptibility, masks the effects of the techniques or disrupt the circuit before any benefit from the hardening is accrued.

Notwithstanding, the use of soft-core microprocessors in FPGA allows the construction of a large library of microprocessors architectures that can be used to evaluate and steer the development of hardening techniques targeting the equivalent hard-core microprocessors. In this sense, BRAM fault injection in FPGA can still be used, for instance, in focused fault injection only for data elements equivalent to that existing hard-core microprocessors.

To illustrate this approach, we implemented the BRAM fault injection in the NanosatC-BR2 design with the exclusive purpose of evaluating the impact of the SIHFT techniques in regard to the persistent object code stored in BRAM.

Two fault injection campaigns were performed, each targeting one of the soft-core microprocessors executing hardened software. Faults were injected in randomized positions inside the BRAM memory matrices using mechanisms described in Section 12.2.

For these fault injection campaigns the NanosatC-BR2 supervisory, depicted inside Fig. C.6f, was replaced by a test driver for tighter control of soft-core and software execution life-cycle and to collect additional diagnosis information. The software executing in the microprocessors is exactly the same on-board NanosatC-BR2, only the supervisory being replaced. In both the microprocessors under test the software is the SIHFT-hardened version of the code on Fig. C.7. Under this set-up the microprocessors are most of the time in busy-wait idle state or busy-wait spin-lock after the end of a computation, `lbl_wait_start` and `lbl_wait_forever`, respectively, in Fig. C.7.

The fault injection sequence consists of injecting a fault in BRAM while the microprocessor is running but in one of the idle or lock states and then invoking the test driver to collect diagnosis. The test-driver starts by issuing a logical reset that causes the initialization and software boot-up of the microprocessor. The test-driver then coordinates the execution of microprocessor while monitoring the execution time.

The outcome may be (1) a timer run-out after the logical reset and boot-up, meaning that the software did not wrote `IDLE` in the memory, (2) a timer run-out during the computation, meaning that the software did not wrote `DONE` in the memory, (3) a detected improper execution flow, signaled on `SYNC_RESULT_DUE` memory address, (4) an undetected incorrect computation, detected by the test driver as an incorrect checksum value on `SYNC_RESULT_CHECKSUM` memory address, or, finally, (5) a correct computation matching the known golden value. During fault injection, if a correct computation is obtained then another fault is injected and the process is repeat until a failure is observed.

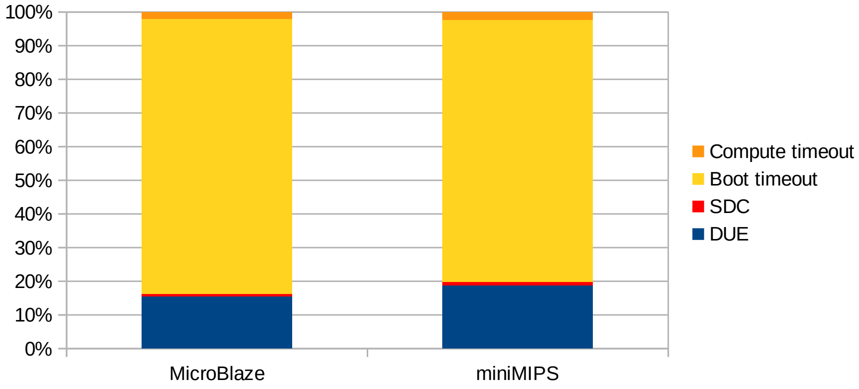
Table 12.1 summarizes the results of fault injection on BRAM. Results from MicroBlaze and miniMIPS are not directly comparable because the memory volume is slightly different in the two microprocessors and the software uses different runtime libraries and compilation flow. Notwithstanding, the classification of the type of failure observed follows a very similar distribution, with most of the failures seen as execution time exceeded. It is noteworthy that the timer run-out during the processor boot-up routine is the more frequent event. All the timer run-out events amounts up to 85% of the functional failures, as depicted on Fig. 12.4. However, when we take into account only the software executions that ended in an incorrect computed result, the SIHFT technique was capable of detecting 95% of the cases.

Table 12.1 – Results from fault injection on BRAM

Failure type	*MicroBlaze*	*miniMIPS*
Failure type		
Detected unrecoverable error (DUE)	99	97
Silent data corruption (SDC)	5	5
Compute timer run-out	13	12
Boot-up timer run-out	520	401
Reliability metrics		
Faults injected	1,528	4,283
BRAM volume (bits)	204,800	163,840
Mean faults to failure (MFTF)	2.4	8.3

Source: The Author

Figure 12.4 – Type of software failure under BRAM fault injection



Source: The Author

13 FAULT INJECTION ON FLIP-FLOPS

13.1 Motivation

The flip-flops (FF) inside logical blocks (CLB) represents a very small portion of the designs essential bits. Each CLB column has 36 frames of CRAM, amounting to 116,352, while the configuration of flip-flops in a CLB column takes only 800 bits, a ratio of 145 to 1. Using as example the design in Table 10.1, we have a ratio of 930 to 1 between critical bits and flip-flops, and a ratio 16045 to 1 between essential bits and flip-flops. Data stored on flip-flops in a CLB columns are also eminently volatile and, consequently, susceptible to temporal masking.

These aspects make fault injection on flip-flops in SRAM-based FPGAs a low priority task compared to the persistent configuration stored in CRAM in general.

Notwithstanding, as we discussed for the case of BRAM fault injection in the previous Chapter 12, emulation-based fault injection on flip-flops in FPGAs can also be used as test methodology in the development of fault tolerance techniques for ASICs, replacing the use of simulation-based methods that are slower.

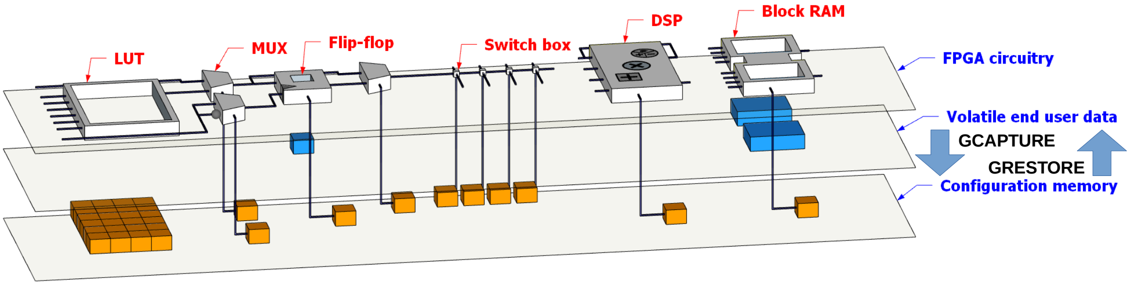
Some examples of fault tolerance development that can use emulation-based fault injection on flip-flops in test circuits implemented in FPGAs include development of software implemented hardware fault tolerance (SIHFT) (CHIELLE et al., 2015, 2016; CHIELLE, 2016) and modular redundancy by the use of fine-grain local TMR (FGLTMR) (BENITES et al., 2018, 2019).

13.2 Implementation

The implementation of fault injection on flip-flops follows a procedures similar to the `RESET_AFTER_RECONFIG` (RAR) discussed briefly in Section 12.2. The Xilinx 7 Series FPGA can be seen as three planes of configuration data (CRAM), volatile user data and FPGA circuitry, as depicted in Fig. 13.1.

In Xilinx 7 Series FPGAs there is a global capture (`GCAPTURE`) hardware operation that can copy data from the user plane to the configuration plane, for instance before a bitstream readback. Equivalently, there is a global restore (`GRESTORE`) operation than copy data from the configuration plane to the user plane. This restore operation occurs when a bitstream is loaded into the FPGA and the initial value of the flip-flops must be

Figure 13.1 – Capture and restore operations



Source: The Author

loaded.

However, the global restore operation does not need to be really global. In dynamic partial reconfiguration, for instance, only the partition of the FPGA that was reconfigured should be subject to the restore operation preventing interference with other partitions that were continuously in operation. The effect of the global restore signal can be constrained to a selected region of the FPGA, which is done by programming a global restore mask in the special CLB configuration frames (type of block 010b in Table 6.1). This mask inhibits the effect of global restore, except in the region of interest.

The programming of a global restore mask in inserted automatically in the bitstream by the Xilinx Vivado Synthesis tool when the physical block containing dynamic partial reconfiguration module is configured with the attribute `RESET_AFTER_RECONFIG` (RAR).

In our case, the configuration of the restore mask was implemented inside the fault injection module in the FPGA. New a fault injection commands were implemented to support fault injection with clock gating, global capture and restore. When the command for fault injection with capture and restore is issued, the fault injection module determines the CLB column from the frame address provided by the campaign script and configures the global restore mask for that single column only.

This fault injection operation would be typically associated with clock gating because the whole sequence of global capture, reading of CRAM frame, insertion of the bit-flip, writing back the CRAM frame, programming of the restore mask and, finally, masked global restore, takes several clock cycles. The clock gating of the design under test assure the consistency of the the state that will restored at the end of the sequence.

13.3 Methodology and scripting

The fault injection on flip-flops can be performed together with the general CRAM fault injection, with the main difference being the clock gating, capture and restore operations that would have no effect for CRAM bits that are not mirrored to the user data plane.

More specifically, for Xilinx 7 Series, in tests in real environment or under accelerated irradiation, SEUs can occur both in the CRAM plane and in the user data plane. If the SEU is on user data plane, the effect may be immediately seen as an error in the computed result of the design. Conversely, a SEU may also be masked if the value is not propagated after the SEU or a new value is stored by the design. However, if the SEU is on the CRAM plane, for instance in the flip-flop reset value it may stay latent until a logical reset in the circuit loads a corrupted value in the storage element in the user data plane. Conversely, a SEU in the CRAM plane changing the flip-flip operation mode from synchronous to asynchronous may also cause a functional failure immediately.

Flip-flop storage elements in a Xilinx 7 Series FPGA CLB column can operate either as latches or as flip-flop proper. The number of relevant configuration bits varies with the type of storage element used by the design, being latches or complete flip-flops, and also the mode of operation of the storage element, for instance if it has synchronous or asynchronous reset. Each storage element have also two configuration bits for its initial (INIT) value, loaded upon global restore, and its set-reset value (SRVAL), loaded upon the local logical set-reset signal.

As an experimental fault injection methodology for flip-flops we injected faults in selected relevant bits for the storage elements, amounting to 800 target bits in each CLB column. Faults are injected with the design clock on hold and the capture and restore operations are used to load any eventually changed data into the user plane. Faults are injected in random times during the design execution, as seem in Chapter 11.

Other flip-flop fault injection methodologies can be tailored to fault tolerant technique under analysis and the target of the development, if a SRAM-based FPGA or an ASIC. For instance, for a stricter emulation of SEUs in the user data plane, which would be of interest to the development of ASICs, fault should be injected only in the CRAM bits relative to the INIT value.

13.4 Study case and results

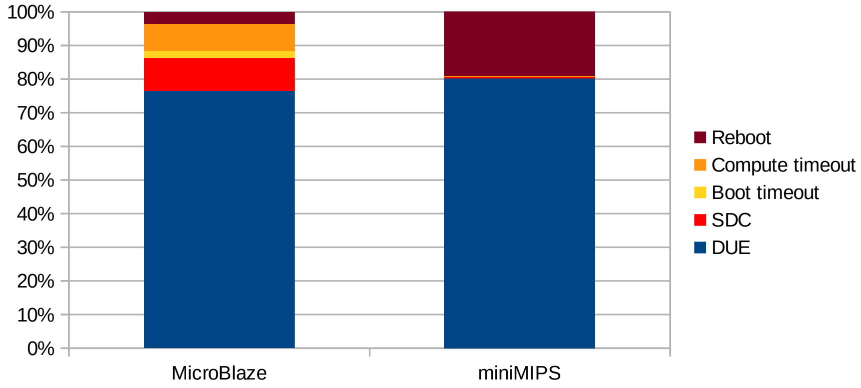
As study case design we adopted the same design from Section 12.4, consisting of the soft-core microprocessors execution benchmark applications with software implemented hardware fault tolerance (SIHFT) embedded on the INPE-SMDH/UFSM-UFRGS payload board for NanosatC-BR2 CubeSat mission.

The two softcore microprocessors under test, MicroBlaze and miniMIPS, have an very important different in the allocation of the register file. While the miniMIPS implements its entire register bank using flip-flops from CLB column, the MicroBlaze implements the register bank using LUTRAM, also from the CLB column as seem briefly in Section 3.3. This is relevant in this context because some of the SIHFT techniques present in the benchmark application includes redundancy of registers.

The diagnostic information was extended to present occurrences of machine reboot during the software execution, which may be caused by unhandled exceptions. This condition is detected by the presence of a `IDLE` value in the `SYNC` control memory address after the test driver have written `START` and exited with timer run-out without a `DONE` response from the application.

The results from fault-injection are summarized in Table 13.1. Overall, miniMIPS processor core used over three times more flip-flops than MicroBlaze and presented a mean-fault to failure (MFTF) also three times lower. The different distribution of failure type, also seen in Fig. 13.2, may be related both to the different number of pipeline stages and to the fact the MicroBlaze register file is not on CLB flip-flops. For instance, as the miniMIPS program counter register (PC) is implemented with flip-flips from CLB, it is plausible that fault injection in flip-flips increase the occurrence of machine reboot in miniMIPS due to exception caused by corrupted PC.

Figure 13.2 – Type of software failure under FF fault injection



Source: The Author

Table 13.1 – Results from fault injection on flip-flop

Failure type	***MicroBlaze***	***miniMIPS***
Failure type		
Detected unrecoverable error (DUE)	284	316
Silent data corruption (SDC)	36	2
Compute timer run-out	30	1
Machine reboot	13	75
Boot-up timer run-out	8	–
Reliability metrics		
Faults injected	1,785	617
CRAM volume (flip-flop bits)	6,400	6,400
Mean faults to failure (MFTF)	4.8	1.6
Flip-flops instantiated by the design	496	1,676

Source: The Author

14 FAULT INJECTION EMULATING MULTIPLE-BIT UPSETS

14.1 Motivation

As discussed in Section 11.4, the emulated fault injection tends to present a larger discrepancy from irradiation tests as the reliability of the design under test increases.

In the study case application of Arm Cortex-M0 softcore microprocessor, that discrepancy was larger for designs with memory scrubbing enabled, as seen in design with the FGDTMR and scrubbing on Fig. 11.4. A similar effect was observed in the design for image classification using convolutional neural network (CNN) presented in Section B.6 (BENEVENUTI et al., 2019c, 2021).

The cumulative effect of SEUs in CRAM can disrupt mitigation techniques such as modular redundancy. The effect of multiple-bit upsets (MBU) can disrupt both modular redundancy and memory scrubbing. The impact on modular redundancy is higher when the redundant modules are collocated (STERPONE et al., 2017). In the case of native memory scrubbing available on Xilinx 7 Series hardware, the error detection and correction codes tolerates only a single bit-flip on each CRAM frame.

It is expected that the emulation of multiple-bit upsets in fault injection may decrease the discrepancy between emulated fault injection and radiation experiments, specially in designs using modular redundancy and memory scrubbing.

14.2 Implementation

The statistical profiles of single-bit (SBU) and multiple-bit upsets collected from static tests in Chapter 8 play a central role in the emulation of multiple-bit upsets.

When the target of the fault injection campaign is persistent configuration data stored in CRAM, in design without the use of memory scrubbing, the multiple-bit upsets can be emulated by the simple injection of many bit-flips is neighboring locations following the profiles as presented in Fig. 8.6, Fig. 8.10 or Fig. 8.12. To some degree, it would be similar to the emulation of arbitrary artifact shapes seen in Chapter 9.

This approach of building the MBU artifact by injection of multiple SBU was applied in preliminary fault injection campaigns over a convolutional neural network for aerial image classification described in Section B.7 (BENEVENUTI et al., 2022).

The challenge is different when memory scrubbing is involved. In this case, the fault injection method presented in Chapter 11 is not enough to inject multiple-bit upsets because the scrubber would be activated during the injection of the MBU. Also, as the fault injection takes several clock cycles, the bit-flips may not be seen by the design in CRAM as a single MBU.

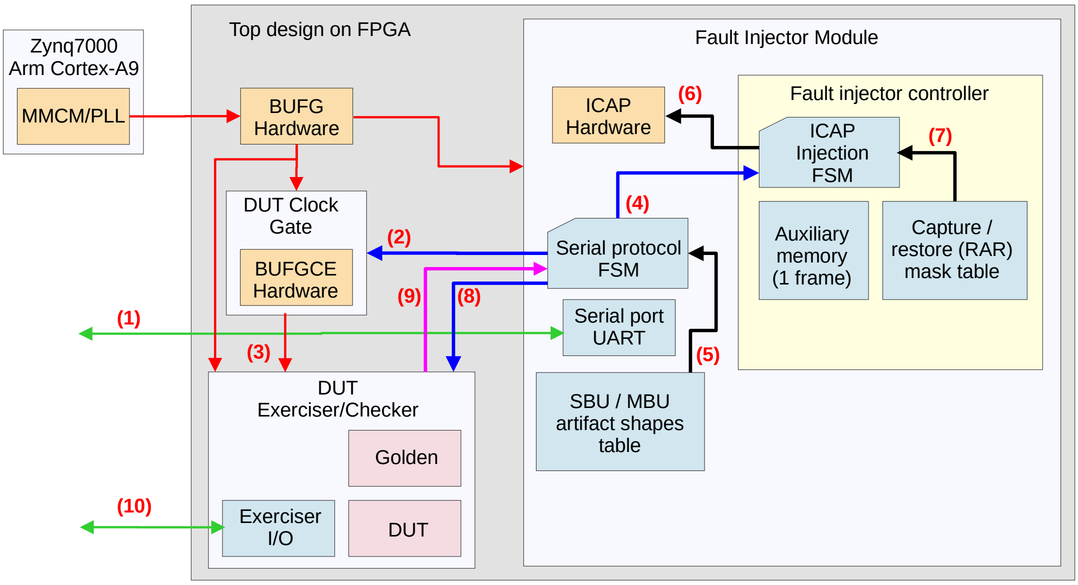
To cope with these challenges, two main modifications were performed in the fault injector. First, the memory scrubber is disabled before the injection of the bit-flips composing the MBU and enabled again at the end of the injection. Second, clock gating must be applied to the design under test, as seen in the case of flip-flop fault injection in Chapter 13.

To sustain the productivity of the fault injection, considering the relatively low speed of serial communication, all the operations over memory scrubber, clock gating and injection of multiple bit-flips were embedded in a single command.

The basic operations of the fault injection presented in previous chapters, and the main fault injector modules seem in Fig. 9.1, can be seen as building blocks for the different fault injection commands. Each fault injection command indicates if memory scrubber should be put on hold or not, if the clock gate should be put on hold or not, if the capture and restore should be applied of not, and, finally, the shape of the SBU or MBU artifact to the injected.

An example of embodiment of the fault injector module is presented in Fig. 14.1. The fault injector receives commands (1) from the campaign script, which are handled by the protocol FSM. The protocol FSM was dismembered from the fault injection control logic from the original fault injection tool for Xilinx 7 Series (TONFAT et al., 2016) for easier maintenance of the protocol and injection logic independently. Whenever the command involves clock gating, the fault injector first commands the clock gate (2) to suspend the DUT clock (3). The implementation of the clock gate module is external to the fault injection enabling the user to define how many clock signals are relevant and which design modules should be coordinated with the DUT gating. The interpreted command is issued to the fault injector (4). However, a single command from the campaign script may result in many commands to the fault injector. For instance, one command to the fault injector may be to use the ICAP interface (6) to write a new value to the FPGA control register to temporarily shutdown or re-enable the Xilinx native memory scrubber. Several commands may also be send to the fault injector to inject the many bit-flips according to the given fault shape (5) selected by the script. If the fault injection also involves capture and restore of data on the user plane, the fault injector should first capture data and program the restore mask (7) except in the column where the fault is injected. To inject a bit-flip, the fault injector module reads the whole frame using the ICAP interface (6), storing temporarily in an auxiliary memory, flips a single bit in the frame by an XOR operation, and then write back the frame using the ICAP interface (6). This is repeated until all the bit-flips for the SBU or MBU were injected (7). Again, if the fault injection also involves capture and restore, the fault injector should now restore data to the user data plane. If the scrubber was shutdown, it should then be enabled again in the control register (6) while the DUT clock is also enabled (2)(3). The fault injector then can command the execution (8) and collection of diagnostic information (9). This information is send back the to campaign script (1). The DUT exerciser/checker can also collect diagnostic information independently before, during or after the fault injection, except during the time interval when the DUT clock is suspended (3). While a brief diagnostic word is sent through the interface with the fault injector (9), more comprehensive diagnose data can be transmitted by the exerciser/checker through a secondary channel (10).

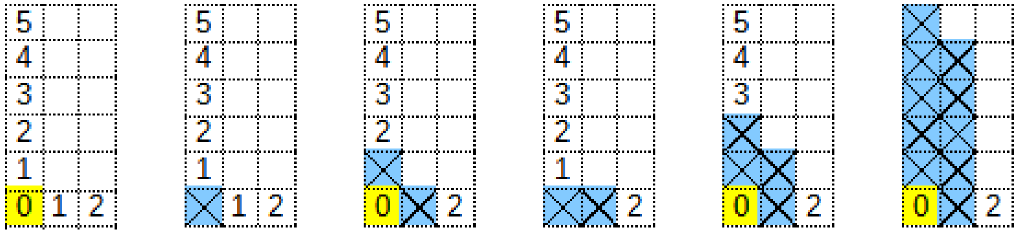
Figure 14.1 – Fault injector modules



Source: The Author

Each SBU or MBU artifact is represented in relation to rectangular matrix. Given the most frequent MBU shapes observed in static tests from Chapter 8, it was observed that a matrix of six rows and three column would suffice to represent all the relevant shapes. The command from the campaign script to the fault injector always inform the position relative to the lower-left corner of the reference frame. Each shape has a well-known identifier shared shared with the campaign script and the fault injection module. The reference frame and some examples of SBU and MBU shapes are presented in Fig.

Figure 14.2 – Reference frame and examples of SBU and MBU shapes



(a) Reference **(b)** #1 **(c)** #2 **(d)** #3 **(e)** #5 **(f)** #40

Source: The Author

14.2. The shape for the MBU has the identifier #1 (Fig. 14.2b). Hence, using the new command for injecting the fault artifact with identifier #1 is equivalent to the legacy command that injects a single bit-flip at the given CRAM position.

Inside the fault injection module, as well as in the campaign script, the fault artifact shape can the represented in a compact form of a bitmap with 18 bits, as in the example of Table 14.1.

Table 14.1 – Examples of SBU and MBU shapes encoding bitmaps

Identifier	***Bitmap***
#1	100 000 000 000 000 000 000
#2	010 100 000 000 000 000 000
#3	110 000 000 000 000 000 000
#5	010 110 100 000 000 000 000
#40	010 110 110 110 110 110 100

Source: The Author

Given the higher complexity of this fault injection module implemented inside the FPGA, the the higher complexity of the coordination of the different building blocks of the fault injector along its finite-state machines, additional simulation testbenchs were implemented to validate its correct operation in different fault injection scenarios.

14.3 Methodology and scripting

Despite the complexity of the fault injector, the fault injection methodology is not very different from the presented in Section 10.3 or in Section 11.3. Overall, the fault injection procedure will always follow a process similar to the presented in Fig. 10.1b, the main difference being that delays may are introduced between faults injected, to emulate a desired flux and adjust to the memory scrubbing period, and the shape of the fault injected should follow a desired statistical profile of SBU and MBU.

14.4 Study case and results

Two different designs were used as study case during the implementation and validation of fault injection supporting multiple-bit upsets. The first design is a simple matrix multiplication benchmark generated by high level synthesis (MXM HLS). The smaller area and simpler interface and operation of this design allow for easier execution of regression tests in the fault injector. Another design used as study case is the Arm Cortex-M0 softcore microprocessor with different levels of mitigation from Section 11.4.

The benchmark application was implemented without mitigations, with coarse-grain triple modular redundancy (CGTMR) coded manually, and with fine-grain distributed triple modular redundancy (FGDTMR) generated automatically by the TMRi tool (BENITES et al., 2018, 2019). The three mitigation levels were tested with and without memory scrubbing. Consequently, we have different levels of fault masking, which may sustain the reliability for a short period of time, and the healing by memory scrubbing, which clean up faults periodically and prevents its accumulation.

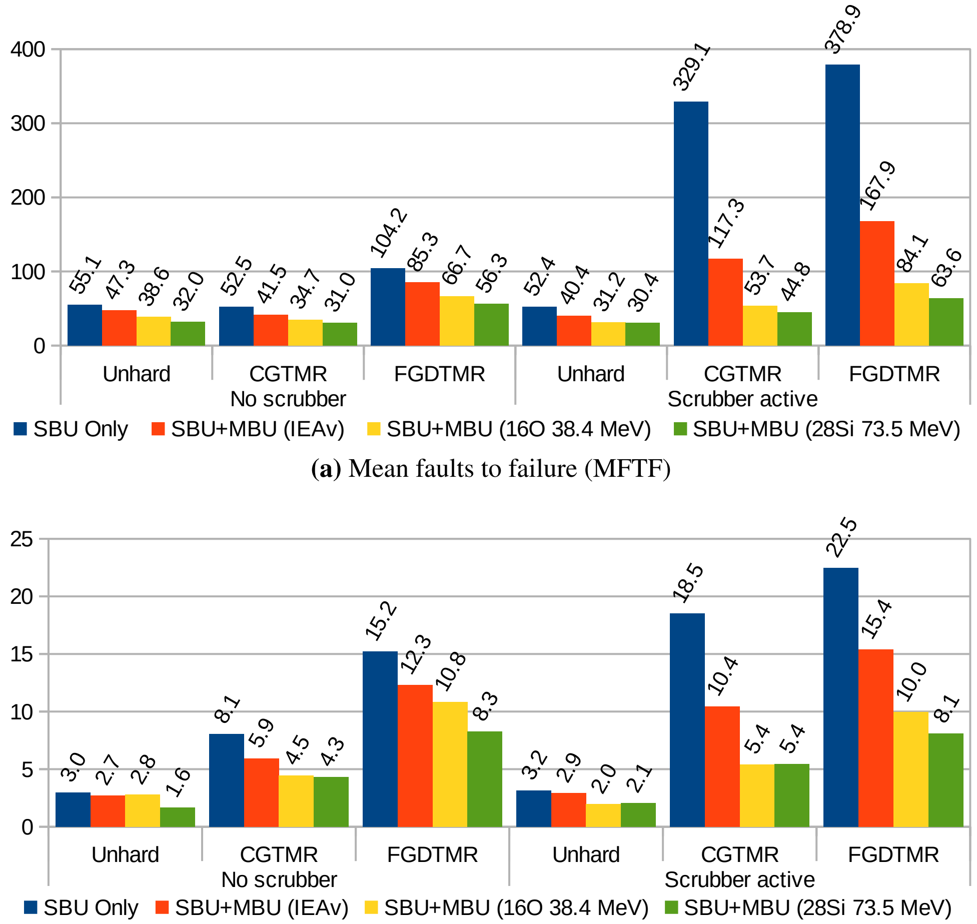
The fault injection campaigns were performed emulating single-bit upsets only, which is the legacy method presented in previous chapters, and emulating both single-bit and multiple-bit upsets following the statistical profile from fast neutrons at IEAv LRI facilities and heavy ions from USP LAFN SAFIIRA with beams of ^{16}O and ^{28}Si.

Figure 14.3 presents reliability metrics for the matrix multiplication benchmark used for regression testes along the several stages of development of the multiple-bit upsets emulation. From the charts we can observe how excessively optimistic the fault injection can be in relation to the memory scrubbing when we inject only single-bit upsets.

In reality, occurrences of multiple-bit upsets in the same frame (intraframe MBU) prevents the memory scrubbing at that frame. The memory scrubbing is not interrupted, but the scrubber will skip frames with multiple bit-flips and continue the periodic CRAM scan correcting only frames with one bit-flip. For instance, the faults from Fig. 14.2b, Fig. 14.2c and Fig. 14.2d can be correct by the scrubber, while the examples from Fig. 14.2e and Fig. 14.2f can not be corrected and would accumulate over time, potentially causing a functional failure once the masking capability of the modular redundancy is exceeded.

As discussed in Chapter 8, the proportion of single-bit upsets and multiple-bit

Figure 14.3 – Reliability metrics for MXM HLS benchmark application



(a) Mean faults to failure (MFTF)

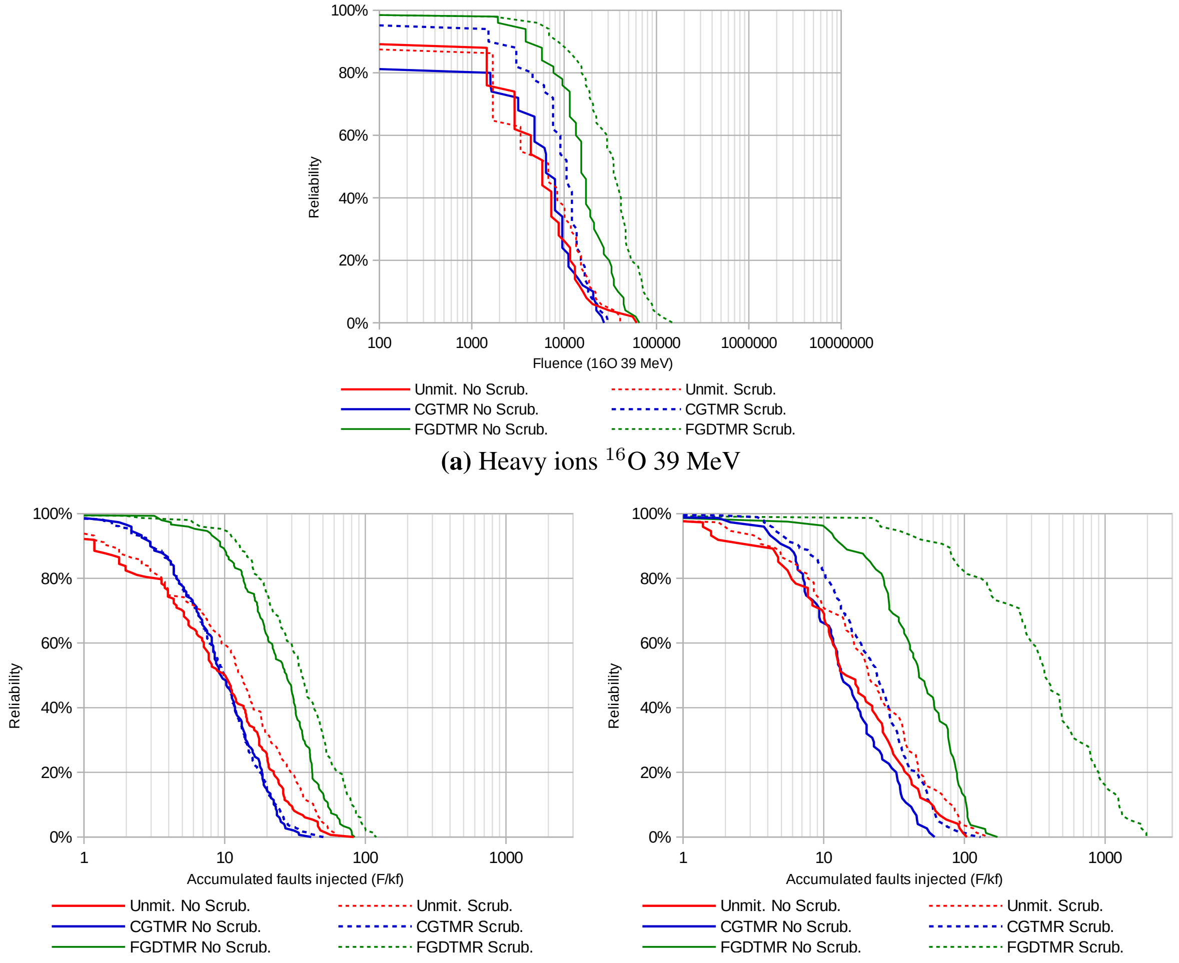
(b) Estimated mission time for 95% reliability (MT, in accumulated faults, R$\geq$95%)

Source: The Author

upsets varies according to the particle and energy. However, the proportion between upsets with a single bit-flip and with multiple bit-flips per frame also varies greatly. For instance, the heavy ions beams of ^{16}O at $38.4\,\mathrm{MeV}$ and ^{28}Si at $73.5\,\mathrm{MeV}$ produce, respectively, 4.7× and 6.8× more intraframe MBUs than fast neutrons. With this in mind, we can see from Fig. 14.3 how the contribution of the memory scrubber in improving reliability is less and less significant as we increase the occurrence of intraframe MBU.

The Arm Cortex-M0 softcore microprocessor was tested under heavy ions with a ^{16}O beam at $39\,\mathrm{MeV}$ (BENITES et al., 2019). For comparison, Fig. 14.4 shows the reliability curves obtained from heavy ions irradiation with ^{16}O beam at $39\,\mathrm{MeV}$, from fault injection emulating only SBU and from fault injection emulating both SBU and MBU using the statistical profile from ^{16}O beam at $38.4\,\mathrm{MeV}$. Figure 14.5 shows the ratio between dynamic cross section (σ) and fault injection error rate (τ), as already discussed in Section 11.4, to illustrate the improvement in the consistency between irradiation tests and emulated fault injection with MBU. The ratio obtained with MBU is not yet a constant,

Figure 14.4 – Reliability curves from radiation and fault injection



(a) Heavy ions ^{16}O 39 MeV

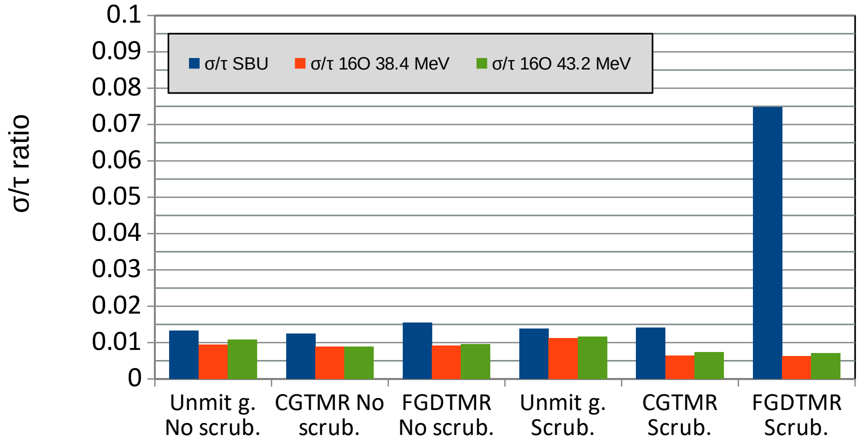
(b) SBU and MBU (^{16}O 38.4 MeV)

(c) SBU Only

Source: The Author

suggesting that there are still elements not covered by fault injection, but the discrepancy is significantly smaller.

Figure 14.5 – Ratio between dynamic cross section and fault injection error rate



Source: The Author

15 PORT OF THE UFRGS FAULT INJECTOR TO ULTRASCALE+

15.1 Motivation

The UltraScale device family using 20 nm planar technology was launched by Xilinx by the end of 2013 following its 7 Series family from mid 2010 that used planar 28 nm technology. UltraScale introduced architectural improvements in the FPGA logic and in the configuration memory array, for instance with a higher bit interleaving to further reduce the occurrence of intraframe MBU. The same UltraScale FPGA architecture is present in the UltraScale+ device family, the main difference being the use of 16 nm FinFET technology. The Versal device family was launched commercially in 2019 and uses 7 nm FinFET technology.

With the launch of low cost UltraScale+ devices, this family becomes a candidate to replace the 7 Series in general purpose applications, considering the lower maturity of the Versal family in the market, the higher cost of the new technology and the positioning of the Versal brand in niches of high performance such as 5G mobile networks and AI/ML in datacenters.

15.2 Implementation

The main characteristics of Xilinx UltraScale+, compared to Xilinx 7 Series, affecting the UFRGS fault injector are:

- The number of bits in the frame is different (93 words of 32 bits).
- The ICAP hardware block has new ports in its interface (Table 15.1).
- The interconnect frames are now addressed independently from the other columns.
- The numbering of rows changed and there is no more the concept of TOP and BOTTOM rows.
- The number of bits and organization of the fields in the frame address was adapted to the new memory organization.
- There are changes in the FPGA configuration registers and commands.
- The BRAM flag bits are in new positions in UltraScale+ (Fig. 15.1).

Table 15.1 – Differences in ICAP hardware interface

		Xilinx Virtex-5	Xilinx 7 Series	Xilinx Ultrascale+
Primitive		ICAP_VIRTEX5	ICAPE2	ICAPE3
Generics				
ICAP_WIDTH => "X32"	Specifies the input and output data width.	Yes	Yes	No
DEVICE_ID => X"03628093"	Specifies the pre-programmed Device ID value to be used for simulation purposes.	No	Yes	Yes
SIM_CFG_FILE_NAME => "NONE"	Specifies the Raw Bitstream (RBT) file to be parsed by the simulation model.	No	Yes	Yes
ICAP_AUTO_SWITCH => "DISABLE"	Enable switch ICAP using sync word	No	No	Yes
Ports				
O : out STD_LOGIC_VECTOR (31 downto 0)	32-bit output: Configuration data output bus	Yes	Yes	Yes
I : in STD_LOGIC_VECTOR (31 downto 0)	32-bit input: Configuration data input bus	Yes	🟩	🟩
CLK : in STD_LOGIC	1-bit input: Clock Input	Yes	Yes	Yes
WRITE : in STD_LOGIC	0 = write, 1 = read (RDWR_B for SelectMAP)	Yes	No	No
RDWRB	1-bit input: Read/Write Select input	No	Yes	Yes
CE : in STD_LOGIC	clock enable (active low) (CS_B for SelectMAP)	Yes	No	No
CSIB	1-bit input: Active-Low ICAP Enable	No	Yes	Yes
BUSY : out STD_LOGIC	active high, only used during read	Yes	No	No
AVAIL	1-bit output: Availability status of ICAP	No	No	Yes
PRDONE	1-bit output: Indicates completion of Partial Reconfiguration	No	No	Yes
PRERROR	1-bit output: Indicates Error during Partial Reconfiguration	No	No	Yes

Source: The Author

Figure 15.1 – Masking of BRAM bits set on frame readback for Xilinx UltraScale+

```
python 08_check_fi_toggle_bram.py
[INFO] do_inject_bram at 01040000 bit 0 bit mask 108 response: 000000
[INFO] do_inject_bram at 01040000 bit 240 bit mask 348 response: 000000
[INFO] do_inject_bram at 01040000 bit 480 bit mask 588 response: 000000
[INFO] do_inject_bram at 01040000 bit 720 bit mask 828 response: 000000
[INFO] do_inject_bram at 01040000 bit 960 bit mask 1068 response: 000000
[INFO] do_inject_bram at 01040000 bit 1200 bit mask 1308 response: 000000
[INFO] do_inject_bram at 01040000 bit 1536 bit mask 1644 response: 000000
[INFO] do_inject_bram at 01040000 bit 1776 bit mask 1884 response: 000000
[INFO] do_inject_bram at 01040000 bit 2016 bit mask 2124 response: 000000
[INFO] do_inject_bram at 01040000 bit 2256 bit mask 2364 response: 000000
[INFO] do_inject_bram at 01040000 bit 2496 bit mask 2604 response: 000000
[INFO] do_inject_bram at 01040000 bit 2736 bit mask 2844 response: 000000
```

Source: The Author

Due to the number of changes in the code, it was opted for maintaining two independent sets of the VHDL source code for the fault injector to Xilinx 7 Series and to Xilinx UltraScale+.

15.3 Methodology and scripting

As little modifications were introduced in the command and control protocol at the serial communication interface between the campaign scripting and the fault injector module inside the FPGA, many of the campaign scripts development of methodologies applied to the 7 Series FPGA can be adapted with little effort to the UltraScale+ devices.

The main aspect that must be changed in the campaign scripting is the inventory of rows, columns and frames on each column, for each device model. However, this is a modification that is necessary even when using different devices from 7 Series family such as the Artix-7 A100T onboard NanosatC-BR2 and the Zynq-7000 Z020 on ZedBoard development board.

Overall, there is no new test methodology specific to the UltraScale+ devices, the test approaches being the same applicable to 7 Series. However, not all test approaches implemented to the 7 Series family are fully functional in UltraScale+ and even those that are functional were not yet cross validated with data from radiation tests.

15.4 Study case and results

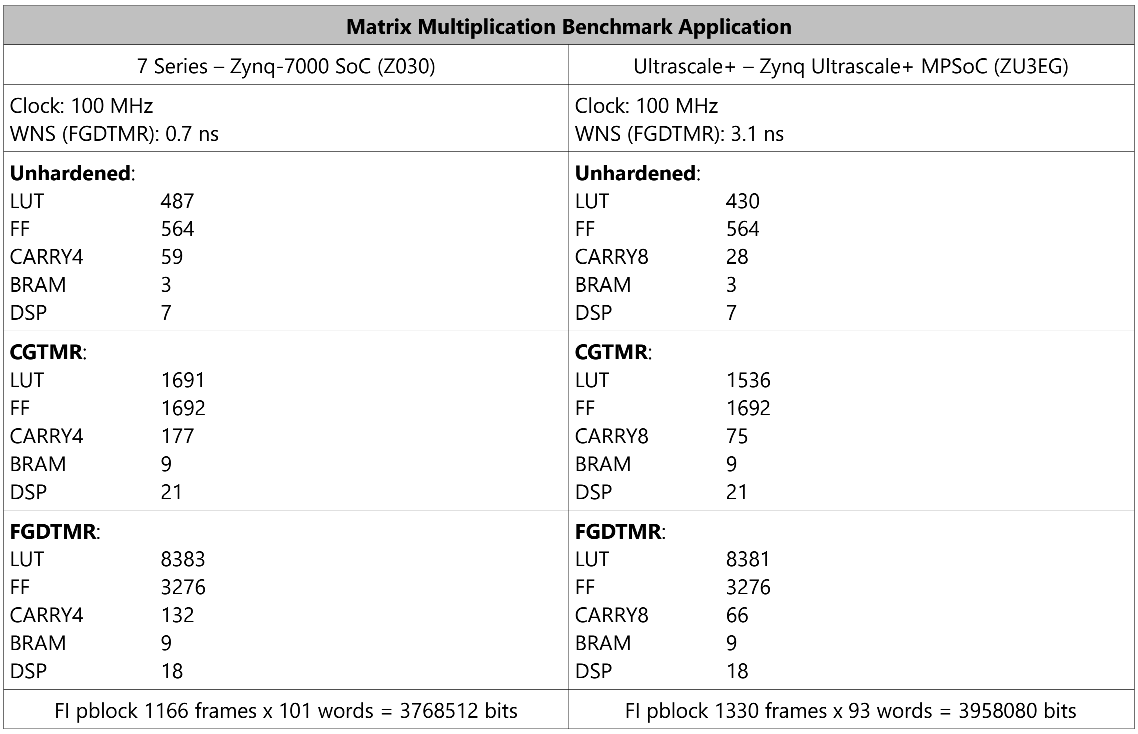
The port of the fault injection in configuration memory (CRAM) for UltraScale+ was implemented and tested with a small benchmark design.

Xilinx High-Level Synthesis (HLS) was used to generate VHDL implementation of a matrix multiplication benchmark and respective exerciser/tester. The TMRi tool described in Benites et al. (2018, 2019) was ported to Xilinx Vivado Tcl and used in hardening the VHDL code of the benchmark application, generating two new variants for coarse-grain TMR (CGTMR) and fine-grain distributed TMR (FGDTMR).

Each of the three variants of the benchmark application was implemented in a Xilinx Zynq-7000 XC7Z030 device and in a Xilinx UltraScale+ Zynq ZU3EG device. Table 15.2 shows resources usage on both FPGA architectures. The number of flip-flops, DSP and BRAM blocks is exactly the same in both devices. The carry hardware in Xilinx

UltraScale+ has twice the size of Xilinx 7 Series. Considering this proportion the carry usage on both devices is similar. The number of LUTs is also similar. The variation on the usage of these resources may be explained by the new architecture of Xilinx UltraScale+ where the two slices of Xilinx 7 Series was fused into a single one opening new opportunities for circuit optimizations inside the slice.

Table 15.2 – Resources utilization by the benchmark application



Matrix Multiplication Benchmark Application	
7 Series – Zynq-7000 SoC (Z030)	Ultrascale+ – Zynq Ultrascale+ MPSoC (ZU3EG)
Clock: 100 MHz WNS (FGDTMR): 0.7 ns	Clock: 100 MHz WNS (FGDTMR): 3.1 ns
Unhardened: LUT 487 FF 564 CARRY4 59 BRAM 3 DSP 7	**Unhardened**: LUT 430 FF 564 CARRY8 28 BRAM 3 DSP 7
CGTMR: LUT 1691 FF 1692 CARRY4 177 BRAM 9 DSP 21	**CGTMR**: LUT 1536 FF 1692 CARRY8 75 BRAM 9 DSP 21
FGDTMR: LUT 8383 FF 3276 CARRY4 132 BRAM 9 DSP 18	**FGDTMR**: LUT 8381 FF 3276 CARRY8 66 BRAM 9 DSP 18
FI pblock 1166 frames x 101 words = 3768512 bits	FI pblock 1330 frames x 93 words = 3958080 bits

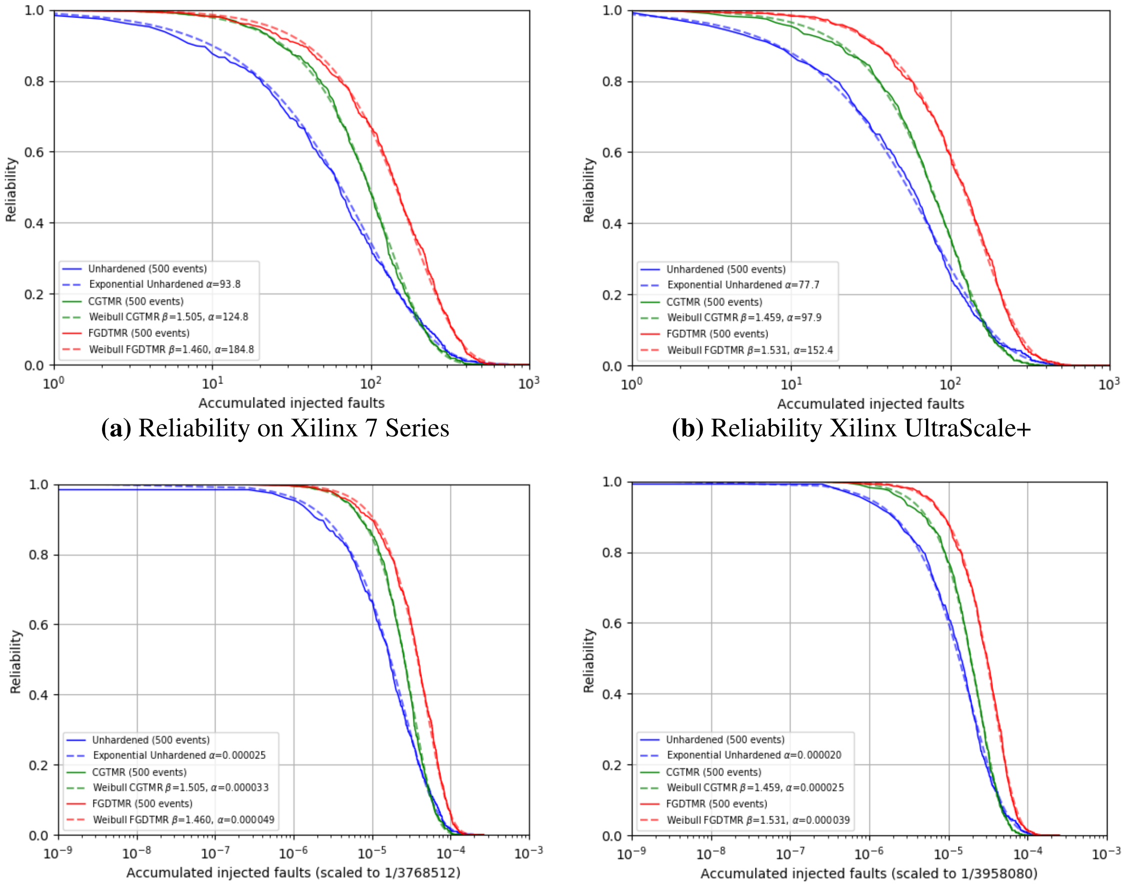
Source: The Author

Random-accumulated fault injection on configuration memory (CRAM) was performed for each variant of the benchmark on each device. Reliability curves obtained from fault injection on the two target devices are shown in Fig. 15.2a and Fig.15.2b. Due to different device architecture and different geometries of the configuration memory array, the number of bits in the region of interest for fault injection is different on each device, as indicated in Table 15.2. For better comparability, Fig. 15.2c and Fig.15.2d shows the reliability curves normalized to a common fault density scale of injected fault per bit.

Despite having different reliability on different devices, notably in the case of CGTMR, we can observe in Fig. 15.2 that:

- The reliability curves (continuous lines) present good fitting to the expected behavior of exponential and Weibull probability distribution (dashed lines), for unhardened and TMR variants respectively.

Figure 15.2 – Benchmark application reliability curves



(a) Reliability on Xilinx 7 Series

(b) Reliability Xilinx UltraScale+

(c) Scaled reliability on Xilinx 7 Series

(d) Scaled reliability on Xilinx UltraScale+

Source: The Author

- The reliability curves show great similarities in both devices.
- The unhardened, CGTMR and FGDTMR variants follows the same ranking in terms of reliability.

This demonstrates that the fault injection on Xilinx UltraScale+ is working properly and the port of the fault injection to this platform was successful.

Some support for fault injection on user data memory (BRAM) was ported provisionally. The basic fault injection CRAM and BRAM, emulating single-bit upset, is currently working in Xilinx UltraScale+ at the same extension of the fault injection for Xilinx 7 Series.

16 CONCLUSIONS AND FURTHER DEVELOPMENTS

Improvements in the fault injector for Xilinx 7 Series devices and the implementation of the random-accumulated test methodology reduced significantly the fault injection campaign time obtaining results similar to the exhaustive-accumulated fault injection test methodology used by Tonfat et al. (2016) for this fault injector.

Additional improvements in the fault injector for 7 Series enabling asynchronous fault injection allowed the use of the fault injection with the memory scrubbing native to the Xilinx 7 Series devices.

Radiation experiments of a CNN using different levels of fault mitigation demonstrated that the original implementation of the fault injector lacked fidelity in scenarios of higher reliability. A higher discrepancy was observed in the design where mitigation techniques such as modular redundancy were associated with healing by memory scrubber. Improvements in the fault injector, notably by the emulation of multiple-bit upsets, reduced this discrepancy increasing the fidelity against radiation tests.

Fault injector implemented for Xilinx 7 Series supporting fault injection only on the configuration memory (CRAM) was modified to inject faults in the user data memory (BRAM) but preliminary results demonstrated that BRAM fault injection through ICAP interface has limited use. The fault injector was also extended to support fault injection on flip-flops in the user data plane on Xilinx 7 Series CLB columns.

The construction of a cartography model of the configuration memory, using readback, laser and heavy-ions microbeam, provided critical information about memory cells neighborhood and important insights for improvement of the fault injector.

Laser cartography experiments were not originally planned for Xilinx UltraScale+ devices. Heavy ions microbeam cartography, heavy ions characterization of static cross section and multiple-bits upsets and irradiation tests of benchmark applications in Xilinx UltraScale+ devices also were not originally planned to this thesis because those tests are not feasible in the SAFIIRA facility at LAFN. However, neutron irradiation tests on Xilinx UltraScale+ that were originally planned, either in LRI at IEAv or at TENIS at ILL, could not be performed due to laboratories being inoperative, lack of personnel or failure to schedule the experiments with the collaborating institutions.

Meanwhile, the fault injector implemented for Xilinx 7 Series was modified to work with Xilinx UltraScale+ devices and the reliability curves obtained by random-accumulated fault injection in the configuration memory for a benchmark design are consistent with reliability curves obtained for Xilinx 7 Series devices.

Static tests on Xilinx 7 Series with protons, alpha particles, heavy ions, thermal neutrons and fast neutrons provided statistical information about the occurrence of multiple-bit upsets in the different types of memory, with different particles and energies. This information was used with random-accumulated test methodology to emulate multiple-bit upsets using the fault injector.

Extensive databases of multiple-bit upsets on Xilinx 7 Series under fast neutrons, from IEAv LRI facility, thermal and epithermal neutrons, from ILL TENIS facility, and heavy ions, from USP LAFN SAFIIRA facility, were used to build statistical profiles of multiple-bit upsets, which were integrated in the fault injection methodology.

The fault injection module supporting multiple-bit upset emulation can be promptly ported to Xilinx UltraScale+, but the use of the methodology still requires a statistical profile of multiple-bit upsets for the technology in a targeted environment, which could be obtained from static tests. Results from static tests on the Xilinx UltraScale+ published by Lee et al. (2018) e Zhaoqun Chen et al. (2021) allow the comparative analysis of static cross section with the previous Xilinx 7 Series family, but does not provide enough information regarding the relative occurrence of single-bit upsets and multiple-bit upsets with different cluster sizes.

Table 16.1 summarizes the maturity level of the new implementations of the fault injector module and the fault injection methodology scripting.

The next development steps in this line of research should aim in increasing the maturity level of the fault injection on Xilinx UltraScale+, primarily by profiling multiple-bit upsets with static tests and cross-validation with benchmark applications tested under accelerated irradiation. The development should also aim in porting the fault injector from Xilinx UltraScale+ family of 16 nm FinFET devices, to the newer Xilinx Versal family of 7 nm FinFET devices. Preliminary assessment of the CRAM on Xilinx Versal family under radiation are already available at the literature (CHEN, Y. P. et al., 2022), including the analysis of occurrences of intraframe MBU.

Table 16.1 – Development maturity level of fault injector implementations

Technique and methodology	***Maturity level***	
	7 Series	***UltraScale+***
Exhaustive-sequential (TONFAT et al., 2016)	**High**. Validated in laboratory and qualified against accelerated irradiation.	**Medium**. Proof-of-concept. Concept and application formulated.
Many-bits fault injection (Chap. 9)	**Medium**. Demonstrated implementation. Concept and application implemented.	**Low**. Untested concept.
Randomized-accumulated fault injection (Chap. 10)	**High**. Validated in laboratory and qualified against accelerated irradiation.	**Medium**. Demonstrated implementation. Concept and application implemented.
Scrubbing compliant (asynchronous) fault injection (Chap. 11)	**High**. Validated in laboratory and qualified against accelerated irradiation.	**Low**. Untested concept. Some development effort required to adapt to new softcore scrubber.
Memory blocks (BRAM) fault injection (Chap. 12)	**Medium**. Demonstrated implementation. Feasibility or boundary conditions identified. Concept and application implemented. Potentially limited usefulness.	**Medium**. Proof-of-concept. Concept and application implemented. Potentially limited usefulness.
Flip-flop (CLB) fault injection (Chap. 13)	**Medium**. Demonstrated implementation. Feasibility or boundary conditions identified. Concept and application implemented. Potentially limited usefulness.	**Low**. Untested concept. Some development effort required to adapt to new reconfiguration strategy.
Multiple-bit upsets (MBU) emulation (Chap. 14)	**Medium**. Demonstrated implementation.	**Low**. Untested concept. Minor development effort required. Static tests database required.

Source: The Author

REFERÊNCIAS

ABDELOUAHAB, K. et al. Tactics to Directly Map CNN Graphs on Embedded FPGAs. **IEEE Embedded Systems Letters**, IEEE, v. 9, n. 4, p. 113–116, dez. 2017. Disponível em: <https://doi.org/10.1109/LES.2017.2743247>.

ACTEL CORPORATION. **ProASIC 500K Family: Data Sheet**. Sunnyvale, USA, fev. 2002. v. 3.0. Disponível em: <https://www.microsemi.com/document-portal/doc_download/130709-proasic-datasheet>. Acesso em: 27 out. 2021.

ACTEL CORPORATION. **SX-A Family FPGAs: Data Sheet**. Mountain View, USA, fev. 2007. v. 5.3. Disponível em: <https://www.microsemi.com/document-portal/doc_download/130722-sx-a-family-fpgas-datasheet>. Acesso em: 27 out. 2021.

AGUIAR, V. A. P. **Desenvolvimento de um sistema de medidas para estudos de efeitos de radiação em dispositivos eletrônicos: metodologias e estudos de casos**. 6 jun. 2019. Doctor of Science Thesis – Instituto de Física, Universidade de São Paulo, São Paulo, Brazil. Disponível em: <https://doi.org/10.11606/T.43.2019.tde-18072019-151550>.

AGUIAR, V. A. P. et al. SAFIIRA: A heavy-ion multi-purpose irradiation facility in Brazil. **Review of Scientific Instruments**, v. 91, n. 5, p. 053301, 2020. Disponível em: <https://doi.org/10.1063/1.5138644>.

AGUIAR, V. A. P. et al. Thermal neutron induced upsets in 28nm SRAM. **Journal of Physics: Conference Series**, IOP, v. 1291, p. 012025, jul. 2019. Disponível em: <https://doi.org/10.1088/1742-6596/1291/1/012025>.

AKKERMAN, A.; BARAK, J.; YITZHAK, N. M. Role of elastic scattering of protons, muons, and electrons in inducing single-event upsets. **IEEE Transactions on Nuclear Science**, v. 64, n. 10, p. 2648–2660, 2017. Disponível em: <https://doi.org/10.1109/TNS.2017.2747658>.

AL KADI, M. et al. General-Purpose Computing with Soft GPUs on FPGAs. **ACM Transactions on Reconfigurable Technology and Systems**, ACM, v. 11, n. 1, p. 1–22, mar. 2018. Disponível em: <https://doi.org/10.1145/3173548>.

ALANIS, A. Y.; ARANA-DANIEL, N.; LÓPEZ-FRANCO, C. **Bio-inspired algorithms for engineering**. [S.l.]: Elsevier, 2018. Disponível em: <https://doi.org/10.1016/C2017-0-00350-6>.

ALDERIGHI, M. et al. Evaluation of Single Event Upset Mitigation Schemes for SRAM based FPGAs using the FLIPPER Fault Injection Platform. In: 22ND IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007). Rome, Italy: IEEE, set. 2007. Disponível em: <https://doi.org/10.1109/DFT.2007.45>.

ALTERA CORPORATION. **FLEX 8000 Programmable Logic Device Family: Data Sheet**. San Jose, USA, jan. 2003. DS-F8000, v. 11.1. Disponível em: <https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ds/archives/dsf8k.pdf>. Acesso em: 27 out. 2021.

ANDREOU, A. G. et al. Bio-inspired system architecture for energy efficient, BIGDATA computing with application to wide area motion imagery. In: 2016 IEEE 7th Latin American Symposium on Circuits Systems (LASCAS). Florianopolis, Brazil: IEEE, 2016. Disponível em: <https://doi.org/10.1109/LASCAS.2016.7450995>.

ANTONI, L.; LEVEUGLE, R.; FEHER, B. Using run-time reconfiguration for fault injection in hardware prototypes. In: PROCEEDINGS IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems. Yamanashi, Japan: IEEE Computer Society, out. 2000. Disponível em: <https://doi.org/10.1109/DFTVS.2000.887181>.

EL-ARABY, E. et al. Reconfigurable processing for satellite on-board automatic cloud cover assessment. **Journal of Real-Time Image Processing**, Springer, v. 4, n. 3, p. 245–259, jan. 2009. Disponível em: <https://doi.org/10.1007/s11554-008-0107-8>.

ARECHIGA, A. P.; MICHAELS, A. J.; BLACK, J. T. Onboard Image Processing for Small Satellites. In: NAECON 2018 - IEEE National Aerospace and Electronics Conference. Dayton, USA: IEEE, jul. 2018. Disponível em: <https://doi.org/10.1109/NAECON.2018.8556744>.

AZIMI, S. et al. A comparative radiation analysis of reconfigurable memory technologies: FinFET versus bulk CMOS. **Microelectronics Reliability**, Elsevier BV, v. 138, p. 114733, nov. 2022. Disponível em: <https://doi.org/10.1016/j.microrel.2022.114733>.

BAO, T.I. et al. Challenges of Low Effective-K approaches for future Cu interconnect. In: 2009 IEEE International Interconnect Technology Conference. Sapporo, Japan: IEEE, jun. 2009. Disponível em: <https://doi.org/10.1109/IITC.2009.5090329>.

BARNABY, H. J. et al. Modeling ionizing radiation effects in solid state materials and CMOS devices. In: 2008 IEEE Custom Integrated Circuits Conference. San Jose, USA: IEEE, 2008. P. 273–280. Disponível em: <https://doi.org/10.1109/CICC.2008.4672075>.

BARRENA, Julen Gomez-Cornejo. Access to Xilinx BRAM columns through PCAP: To: Fabio Benevenuti. E-mail, Message-ID: <20220809141255.Horde.-FVZSBpgZsP2Eqa9sXTkCvp@webposta.ehu.eus>. Bilbao, Spain, 9 ago. 2022.

BASU, S. et al. DeepSat: a learning framework for satellite imagery. In: PROCEEDINGS of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems. Seattle, USA: ACM, nov. 2015. Disponível em: <http://dx.doi.org/10.1145/2820783.2820816>.

BAUMANN, R. et al. Boron compounds as a dominant source of alpha particles in semiconductor devices. In: PROCEEDINGS of 1995 IEEE International Reliability Physics Symposium. Las Vegas, USA: IEEE, 1995. P. 297–302. Disponível em: <https://doi.org/10.1109/RELPHY.1995.513695>.

BEHRENDT, K.; NOVAK, L.; BOTROS, R. A deep learning approach to traffic lights: Detection, tracking, and classification. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). Singapore: IEEE, mai. 2017. Disponível em: <https://doi.org/10.1109/ICRA.2017.7989163>.

BENEVENUTI, F.; KASTENSMIDT, F. L. Analyzing AXI Streaming interface for hardware acceleration in AP-SoC under soft errors. In: VOROS, N. et al. (Ed.). **Applied Reconfigurable Computing. Architectures, Tools, and Applications**. Cham: Springer, 2018. P. 243–254. ISBN 978-3-319-78890-6. Disponível em: <https://doi.org/10.1007/978-3-319-78890-6_20>.

BENEVENUTI, F.; KASTENSMIDT, F. L. Comparing exhaustive and random fault injection methods for configuration memory on SRAM-based FPGAs. In: 2019 IEEE 20th Latin-American Test Symposium (LATS). Santiago, Chile: IEEE, mar. 2019. P. 87–92. Disponível em: <https://doi.org/10.1109/LATW.2019.8704647>.

BENEVENUTI, F.; KASTENSMIDT, F. L. Evaluation of fault attack detection on SRAM-based FPGAs. In: 2017 18th IEEE Latin American Test Symposium (LATS). Bogota, Colombia: IEEE, 2017. Disponível em: <https://doi.org/10.1109/LATW.2017.7906747>.

BENEVENUTI, F.; KASTENSMIDT, F. L. Reliability evaluation on interfacing with AXI and AXI-S on Xilinx Zynq-7000 AP-SoC. In: 2018 IEEE 19th Latin-American Test Symposium (LATS). São Paulo, Brazil: IEEE, 2018. Disponível em: <https://doi.org/10.1109/LATW.2018.8347233>.

BENEVENUTI, F. et al. Comparative analysis of inference errors in a neural network implemented in SRAM-based FPGA induced by neutron irradiation and fault injection methods. In: 2018 31st Symposium on Integrated Circuits and Systems Design (SBCCI). Bento Gonçalves, Brazil: IEEE, 2018. Disponível em: <https://doi.org/10.1109/SBCCI.2018.8533235>.

BENEVENUTI, F. et al. Experimental applications on SRAM-based FPGA for the NanosatC-BR2 scientific mission. In: 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). Rio de Janeiro, Brazil: IEEE, 2019. P. 140–146. Disponível em: <https://doi.org/10.1109/IPDPSW.2019.00032>.

BENEVENUTI, F. et al. Heavy ions testing of an all-convolutional neural network for image classification evolved by genetic algorithms and implemented on SRAM-based FPGA. In: 2019 European Conference on Radiation and Its Effects on Components and Systems (RADECS). Montpellier, France: IEEE, 2019. Disponível em: <https://doi.org/10.1109/RADECS47380.2019.9745650>.

BENEVENUTI, F. et al. Investigating the Reliability Impacts of Neutron-induced Soft Errors in Aerial Image Classification CNNs Implemented in a Softcore SRAM-based FPGA GPU. **Microelectronics Reliability**, Elsevier, Berlin, Germany, v. 138, p. 114738, 2022. 33rd European Symposium on Reliability of Electron Devices, Failure Physics and Analysis. ISSN 0026-2714. Disponível em: <https://doi.org/10.1016/j.microrel.2022.114738>.

BENEVENUTI, F. et al. Robust Convolutional Neural Networks in SRAM-based FPGAs: a Case Study in Image Classification. **Journal of Integrated Circuits and Systems**, v. 16, n. 2, p. 504, 19 ago. 2021. Disponível em: <https://doi.org/10.29292/jics.v16i2.504>.

BENGIO, Y. Learning deep architectures for AI. **Foundations and Trends in Machine Learning**, Now Publishers, v. 2, n. 1, p. 1–127, 2009. Disponível em: <http://dx.doi.org/10.1561/2200000006>.

BENGIO, Y.; LECUN, Y.; HINTON, G. Deep learning for AI. **Communications of the ACM**, ACM, v. 64, n. 7, p. 58–65, jul. 2021. Disponível em: <https://doi.org/10.1145/3448250>.

BENITES, L. A. C.; KASTENSMIDT, F. L. Automated design flow for applying Triple Modular Redundancy (TMR) in complex digital circuits. In: 2018 IEEE 19th Latin-American Test Symposium (LATS). Sao Paulo, Brazil: IEEE, mar. 2018. Disponível em: <https://doi.org/10.1109/LATW.2018.8349668>.

BENITES, L. A. C. et al. Reliability calculation with respect to functional failures induced by radiation in TMR Arm Cortex-M0 soft-core embedded into SRAM-based FPGA. **IEEE Transactions on Nuclear Science**, v. 66, n. 7, p. 1433–1440, 2019. Disponível em: <https://doi.org/10.1109/TNS.2019.2921796>.

BENSANA, E.; LEMAITRE, M.; VERFAILLIE, G. Earth Observation Satellite Management. **Constraints**, Springer, v. 4, n. 3, p. 293–299, 1999. Disponível em: <https://doi.org/10.1023/A:1026488509554>.

BETZ, V.; ROSE, J.; MARQUARDT, A. **Architecture and CAD for Deep-Submicron FPGAS**. [S.l.]: Springer, 1999. Disponível em: <https://doi.org/10.1007/978-1-4615-5145-4>.

BHUVA, B. Soft Error Trends in Advanced Silicon Technology Nodes. In: 2018 IEEE International Electron Devices Meeting (IEDM). [S.l.]: IEEE, dez. 2018. Disponível em: <https://doi.org/10.1109/IEDM.2018.8614526>.

BLACKETT, P. M. S.; OCCHIALINI, G. P. S. Some photographs of the tracks of penetrating radiation. **Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences**, Royal Society, v. 139, n. 839, p. 699–726, mar. 1933. Disponível em: <https://doi.org/10.1098/rspa.1933.0048>.

BLANCO, A. et al. **Forecast Overview: Semiconductors,Worldwide, 2017 Update**. Stamford, USA, 28 jul. 2017.

BLUM, E. K.; LI, L. K. Approximation theory and feedforward networks. **Neural Networks**, Elsevier, v. 4, n. 4, p. 511–515, jan. 1991. Disponível em: <https://doi.org/10.1016/0893-6080(91)90047-9>.

BOCQUILLON, A. et al. Highlights of laser testing capabilities regarding the understanding of SEE in SRAM based FPGAs. In: 2007 9th European Conference on Radiation and Its Effects on Components and Systems. Deauville, France: IEEE, 2007. Disponível em: <https://doi.org/10.1109/RADECS.2007.5205500>.

BORGEAUD, M. et al. SwissCube: The First Entirely-Built Swiss Student Satellite with an Earth Observation Payload. In: SMALL Satellite Missions for Earth Observation. [S.l.]: Springer, 2010. P. 207–213. Disponível em: <https://doi.org/10.1007/978-3-642-03501-2_19>.

BOUDENOT, J.-C. Radiation space environment. In: VELAZCO, R.; FOUILLAT, P.; REIS, R. (Ed.). **Radiation Effects on Embedded Systems**. Dordrecht: Springer, 2007. P. 1–9. ISBN 978-1-4020-5646-8. Disponível em: <https://doi.org/10.1007/978-1-4020-5646-8_1>.

BOZZOLI, L.; STERPONE, L. Self rerouting of dynamically reconfigurable SRAM-based FPGAs. In. Disponível em: <https://doi.org/10.1109/AHS.2017.8046362>.

BRAGA, G. et al. Evaluating softcore GPU in SRAM-based FPGA under radiation-induced effects. In: 2021 European Symposium on Reliability of Electron Devices, Failure Physics and Analysis (ESREF). Bordeaux, France: Elsevier, nov. 2021. v. 126. Disponível em: <https://doi.org/10.1016/j.microrel.2021.114348>.

BRÄUNIG, D.; WULF, F. Atomic displacement and total ionizing dose damage in semiconductors. **Radiation Physics and Chemistry**, v. 43, n. 1, p. 105–127, 1994. ISSN 0969-806X. Disponível em: <https://doi.org/10.1016/0969-806X(94)90205-4>.

BRZOZOWSKI, W. et al. CubeSat Camera: A Low Cost Imaging System for CubeSat Platforms. In: INTERPLANETARY CubeSat Workshop. Paris, France: iCubeSat, mai. 2018. (iCubeSat 2018). Disponível em: <https://icubesat.org/archive/2018-2/icubesat-program-2018/>.

BUCHNER, S. P. et al. Laser simulation of single event upsets. **IEEE Transactions on Nuclear Science**, v. 34, n. 6, p. 1227–1233, 1987. Disponível em: <https://doi.org/10.1109/TNS.1987.4337457>.

CAI, Chang et al. SEE Sensitivity Evaluation for Commercial 16 nm SRAM-FPGA. **Electronics**, MDPI AG, v. 8, n. 12, p. 1531, dez. 2019. Disponível em: <https://doi.org/10.3390/electronics8121531>.

CAI, X. et al. Low-Power SDR Design on an FPGA for Intersatellite Communications. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, IEEE, v. 26, n. 11, p. 2419–2430, nov. 2018. Disponível em: <https://doi.org/10.1109/TVLSI.2018.2850746>.

CALIFORNIA POLYTECHNIC STATE UNIVERSITY. **CubeSat Design Specification (CDS)**. San Luis Obispo, USA: [s.n.], abr. 2015. Revision 13, Updated 6 April 2015. Disponível em: <http://www.cubesat.org/s/cds_rev13_final2.pdf>. Acesso em: 30 jul. 2021.

CANIVET, G. et al. Characterization of Effective Laser Spots during Attacks in the Configuration of a Virtex-II FPGA. In: 2009 27th IEEE VLSI Test Symposium. Santa Cruz, USA: IEEE, mai. 2009. Disponível em: <https://doi.org/10.1109/VTS.2009.19>.

CANIVET, G. et al. Detailed analyses of single laser shot effects in the configuration of a Virtex-II FPGA. In: 2008 14th IEEE International On-Line Testing Symposium. Rhodes, Greece: IEEE, 2008. P. 289–294. Disponível em: <https://doi.org/10.1109/IOLTS.2008.41>.

CASTRO, S. de et al. Figure of Merits of 28nm Si Technologies for Implementing Laser Attack Resistant Security Dedicated Circuits. In: 2015 IEEE Computer Society Annual Symposium on VLSI. Montpellier, France: IEEE, jul. 2015. Disponível em: <https://doi.org/10.1109/ISVLSI.2015.76>.

CHAPMAN, K. SEU Mitigation Techniques for SRAM based FPGAs. In: TWEPP 2015 - Topical Workshop on Electronics for Particle Physics. [S.l.: s.n.], 2015. Invited Plenary, Contribution ID 240. Disponível em: <https://indico.cern.ch/event/357738/contribution/848841/attachments/1161500/1675459/Chapman_ID240.pptx>. Acesso em: 30 jul. 2021.

CHARETTE, R. de; NASHASHIBI, F. Traffic light recognition using image processing compared to learning processes. In: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems. St. Louis, USA: IEEE, out. 2009. Disponível em: <https://doi.org/10.1109/IROS.2009.5353941>.

CHEN, S.-J.; LI, Z.; HU, M. Research Progress on Requirements Integrated Preprocessing and Mission Planning for Earth Observation Satellites. **IOP Conference Series: Materials Science and Engineering**, IOP, v. 608, p. 012030, ago. 2019. Disponível em: <https://doi.org/10.1088/1757-899X/608/1/012030>.

CHEN, Yanran Paula et al. 64MeV Proton single-event evaluation of Xilinx Single Event Mitigation (XilSEM) firmware on 7nm Versal™ ACAP devices. In: 2022 IEEE Radiation Effects Data Workshop (REDW) (in conjunction with 2022 NSREC). Provo, USA: IEEE, jul. 2022. Disponível em: <https://doi.org/10.1109/REDW56037.2022.9921508>.

CHEN, Z.; HUANG, X. Pedestrian Detection for Autonomous Vehicle Using Multi-Spectral Cameras. **IEEE Transactions on Intelligent Vehicles**, IEEE, v. 4, n. 2, p. 211–219, jun. 2019. Disponível em: <https://doi.org/10.1109/TIV.2019.2904389>.

CHEN, Zhaoqun et al. New method for predicting heavy ion-induced SEE cross-section based on proton experimental data. **Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms**, Elsevier BV, v. 509, p. 27–33, dez. 2021. Disponível em: <https://doi.org/10.1016/j.nimb.2021.08.014>.

CHIELLE, Eduardo. **Selective software-implemented hardware fault tolerance tecnhiques to detect soft errors in processors with reduced overhead**. 2016. Doctor Thesis – Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Microeletrônica. Disponível em: <http://hdl.handle.net/10183/142568>.

CHIELLE, Eduardo et al. Reliability on ARM Processors Against Soft Errors Through SIHFT Techniques. **IEEE Transactions on Nuclear Science**, Institute of Electrical e Electronics Engineers (IEEE), p. 1–9, 2016. Disponível em: <https://doi.org/10.1109/TNS.2016.2525735>.

CHIELLE, Eduardo et al. S-SETA: Selective Software-Only Error-Detection Technique Using Assertions. **IEEE Transactions on Nuclear Science**, Institute of Electrical e Electronics Engineers (IEEE), v. 62, n. 6, p. 3088–3095, dez. 2015. Disponível em: <https://doi.org/10.1109/TNS.2015.2484842>.

CHIEN, S. et al. Onboard Autonomy on the Intelligent Payload EXperiment (IPEX) Cubesat Mission – A pathfinder for the proposed HyspIRI Mission Intelligent Payload Module. In: PROC. Intl Symposium on Artificial Intelligence, Robotics, and Automation for Space. Montreal, Canada: European Space Agency, 2014. Disponível em: <http://robotics.estec.esa.int/i-SAIRAS/isairas2014/Data/Session%207c/ISAIRAS_FinalPaper_0013.pdf>.

CHO, D.-. et al. Optimization-Based Scheduling Method for Agile Earth-Observing Satellite Constellation. **Journal of Aerospace Information Systems**, American Institute of Aeronautics e Astronautics (AIAA), v. 15, n. 11, p. 611–626, nov. 2018. Disponível em: <https://doi.org/10.2514/1.I010620>.

CHO, D.-H. et al. High-Resolution Image and Video CubeSat (HiREV): Development of Space Technology Test Platform Using a Low-Cost CubeSat Platform. **International Journal of Aerospace Engineering**, Hindawi Limited, v. 2019, p. 1–17, mai. 2019. Disponível em: <https://doi.org/10.1155/2019/8916416>.

CHRISTIE, G. et al. Functional Map of the World. In: CVPR. Salt Lake City, USA: [s.n.], jun. 2018. Disponível em: <https://doi.org/10.1109/CVPR.2018.00646>.

CLAEYS, C.; SIMOEN, E. Radiation environments and component selection strategy. In: RADIATION Effects in Advanced Semiconductor Materials and Devices. [S.l.]: Springer, 2002. Disponível em: <https://doi.org/10.1007/978-3-662-04974-7_1>.

CLEMENTE, J. A. et al. Statistical Anomalies of Bitflips in SRAMs to Discriminate SBUs From MCUs. **IEEE Transactions on Nuclear Science**, IEEE, v. 63, n. 4, p. 2087–2094, ago. 2016. Disponível em: <https://doi.org/10.1109/TNS.2016.2551263>.

CRYSTALSPACE OÜ. **Crystalspace CAM1U CubeSat Camera**. Tartu, Estonia, 29 mai. 2016. Disponível em: <https://crystalspace.eu/products/crystalspace-c1u-cubesat-camera>.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. **Mathematics of Control, Signals, and Systems**, Springer, v. 2, n. 4, p. 303–314, dez. 1989. Disponível em: <https://doi.org/10.1007/BF02551274>.

DARRACQ, F. et al. Investigation of single event burnout sensitive depth in power MOSFETS. In: 2009 European Conference on Radiation and Its Effects on Components and Systems. Brugge, Belgium: IEEE, set. 2009. Disponível em: <https://doi.org/10.1109/RADECS.2009.5994563>.

DAVIS, Philip et al. Single-Event Characterization of the 16 nm FinFET Xilinx UltraScale RFSoC Field-Programmable Gate Array under Proton Irradiation. In: 2019 IEEE Radiation Effects Data Workshop. [S.l.]: IEEE, jul. 2019. Disponível em: <https://doi.org/10.1109/REDW.2019.8906566>.

DAWOOD, A. S.; VISSER, S. J.; WILLIAMS, J. A. Reconfigurable FPGAS for real time image processing in space. In: 2002 14th International Conference on Digital Signal Processing Proceedings. DSP 2002 (Cat. No.02TH8628). Santorini, Greece: IEEE, 2002. Disponível em: <https://doi.org/10.1109/ICDSP.2002.1028222>.

DIXON-WARREN, S. **A review of TSMC 28 nm process technology**. Ottawa, Canada: Tech Insights, dez. 2012. Disponível em: <https://www.techinsights.com/blog/review-tsmc-28-nm-process-technology>. Acesso em: 28 jan. 2018.

DODD, P. E. et al. Production and propagation of single-event transients in high-speed digital logic ICs. **IEEE Transactions on Nuclear Science**, v. 51, n. 6, p. 3278–3284, 2004. Disponível em: <https://doi.org/10.1109/TNS.2004.839172>.

DU, B. et al. Ultrahigh Energy Heavy Ion Test Beam on Xilinx Kintex-7 SRAM-Based FPGA. **IEEE Transactions on Nuclear Science**, IEEE, v. 66, n. 7, p. 1813–1819, jul. 2019. Disponível em: <https://doi.org/10.1109/TNS.2019.2915207>.

DUNAI, T. J. **Cosmogenic nuclides: principles, concepts and applications in the earth surface sciences**. Cambridge: Cambridge University Press, 2010. ISBN 978-0-511-67752-6.

ECOFFET, R. In-flight anomalies on electronic devices. In: VELAZCO, R.; FOUILLAT, P.; REIS, R. (Ed.). **Radiation Effects on Embedded Systems**. Dordrecht: Springer, 2007. P. 31–68. ISBN 978-1-4020-5646-8. Disponível em: <https://doi.org/10.1007/978-1-4020-5646-8_3>.

FABERO, J. C. et al. Single Event Upsets Under 14-MeV Neutrons in a 28-nm SRAM-Based FPGA in Static Mode. **IEEE Transactions on Nuclear Science**, IEEE, v. 67, n. 7, p. 1461–1469, jul. 2020. Disponível em: <https://doi.org/10.1109/TNS.2020.2977874>.

FANG, Yi-Pin; OATES, Anthony S. Characterization of Single Bit and Multiple Cell Soft Error Events in Planar and FinFET SRAMs. **IEEE Transactions on Device and Materials Reliability**, Institute of Electrical e Electronics Engineers (IEEE), v. 16, n. 2, p. 132–137, jun. 2016. Disponível em: <http://dx.doi.org/10.1109/TDMR.2016.2535663>.

FANG, Yi-Pin; OATES, Anthony S. Neutron-Induced Charge Collection Simulation of Bulk FinFET SRAMs Compared With Conventional Planar SRAMs. **IEEE Transactions on Device and Materials Reliability**, Institute of Electrical e Electronics Engineers (IEEE), v. 11, n. 4, p. 551–554, dez. 2011. Disponível em: <https://doi.org/10.1109/TDMR.2011.2168959>.

FARRAG, A. et al. Satellite swarm survey and new conceptual design for Earth observation applications. **The Egyptian Journal of Remote Sensing and Space Science**, Elsevier, v. 24, n. 1, p. 47–54, fev. 2021. Disponível em: <https://doi.org/10.1016/j.ejrs.2019.12.003>.

FEDERAL AVIATION ADMINISTRATION'S OFFICE OF COMMERCIAL SPACE TRANSPORTATION. **The Annual Compendium of Commercial Space Transportation: 2018**. [S.l.: s.n.], jan. 2018. Disponível em: <https://www.faa.gov/about/office_org/headquarters_offices/ast/media/2018_ast_compendium.pdf>.

FRANCO, F. J. et al. Statistical Deviations from the Theoretical only-SBU Model to Estimate MCU rates in SRAMs. **IEEE Transactions on Nuclear Science**, IEEE, p. 1–1, 2017. Disponível em: <https://doi.org/10.1109/TNS.2017.2726938>.

GAO, Y. et al. Remote sensing scene classification based on high-order graph convolutional network. **European Journal of Remote Sensing**, Informa UK Limited, v. 54, sup1, p. 141–155, jan. 2021. Disponível em: <https://doi.org/10.1080/22797254.2020.1868273>.

GARRANZO, D. et al. APIS: the miniaturized Earth observation camera on-board OPTOS CubeSat. **Journal of Applied Remote Sensing**, SPIE-Intl Soc Optical Eng, v. 13, n. 03, p. 1, mai. 2019. Disponível em: <https://doi.org/10.1117/1.JRS.13.032502>.

GEIGER, A. et al. 3D Traffic Scene Understanding From Movable Platforms. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 36, n. 5,

p. 1012–1025, mai. 2014. Disponível em: <https://doi.org/10.1109/TPAMI.2013.185>.

GEORGE, V.; RABAEY, J. M. **Low-Energy FPGAs — Architecture and Design**. [S.l.]: Springer, 2001. Disponível em: <https://doi.org/10.1007/978-1-4615-1421-3>.

GIM, Y. et al. Three-dimensional particle tracking velocimetry using shallow neural network for real-time analysis. **Experiments in Fluids**, Springer, v. 61, n. 2, jan. 2020. Disponível em: <https://doi.org/10.1007/s00348-019-2861-8>.

GLORIEUX, Maximilien et al. Single-Event Characterization of Xilinx UltraScale+ MPSOC under Standard and Ultra-High Energy Heavy-Ion Irradiation. In: 2018 IEEE Nuclear and Space Radiation Effects Conference (NSREC 2018). [S.l.]: IEEE, jul. 2018. Disponível em: <https://doi.org/10.1109/NSREC.2018.8584296>.

GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. In: GORDON, G.; DUNSON, D.; DUDÍK, M. (Ed.). **Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics**. Fort Lauderdale, USA: PMLR, abr. 2011. v. 15. (Proceedings of Machine Learning Research), p. 315–323. Disponível em: <http://proceedings.mlr.press/v15/glorot11a.html>.

GOMEZ-CORNEJO, J. et al. A novel BRAM content accessing and processing method based on FPGA configuration bitstream. **Microprocessors and Microsystems**, Elsevier, v. 49, p. 64–76, mar. 2017. Disponível em: <http://dx.doi.org/10.1016/j.micpro.2017.01.009>.

GOMEZ-CORNEJO, J. et al. Data content scrubbing approach for SRAM based FPGA designs. In: 2022 IEEE 31st International Symposium on Industrial Electronics (ISIE). [S.l.]: IEEE, jun. 2022. Disponível em: <https://doi.org/10.1109/ISIE51582.2022.9831467>.

GONCALVES, M. M. et al. Investigating floating-point implementations in a softcore GPU under radiation-induced faults. In: 2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS). Glasgow, UK: IEEE, 2020. Disponível em: <https://doi.org/10.1109/ICECS49266.2020.9294939>.

GÖPPERT-MAYER, M. Elementary processes with two quantum transitions. **Annalen der Physik**, v. 18, n. 7-8, p. 466–479, 2009. Reprint of "Über Elementarakte mit zwei Quantensprüngen", Translation by Daniel C. Koepke, Montana State University. Disponível em: <https://doi.org/10.1002/andp.200910358>.

GRAZIANI, A.; MELEGA, N.; TORTORA, P. A Low-Cost Microsatellite Platform for Multispectral Earth Observation. In: SMALL Satellites for Earth Observation. [S.l.]: Springer, 2008. P. 309–318. Disponível em: <https://doi.org/10.1007/978-1-4020-6943-7_29>.

GSCHWEND, D. **Zynq Net: An FPGA-accelerated embedded convolutional neural network**. Ago. 2016. Master Thesis – ETHZürich, Department of Information Technology e Electrical Engineering, Zürich, Switzerland. Disponível em: <https://github.com/dgschwend/zynqnet>. Acesso em: 27 out. 2021.

GUARESCHI, W. et al. Configurable test bed design for nanosats to qualify commercial and customized integrated circuits. In: 2013 IEEE Aerospace Conference. [S.l.: s.n.], mar. 2013. Disponível em: <https://doi.org/10.1109/AERO.2013.6497170>.

HABINC, S. (Ed.). **Suitability of reprogrammable FPGAs in space applications**. Paris, France, set. 2002. version 0.4. Disponível em: <http://microelectronics.esa.int/techno/fpga_002_01-0-4.pdf>. Acesso em: 1 mai. 2021.

HABING, D. H. The use of lasers to simulate radiation-induced transients in semiconductor devices and circuits. **IEEE Transactions on Nuclear Science**, v. 12, n. 5, p. 91–100, 1965. Disponível em: <https://doi.org/10.1109/TNS.1965.4323904>.

HAN, J.; GUO, G. Characteristics of energy deposition from 1-1000 MeV proton and neutron induced nuclear reactions in silicon. **AIP Advances**, v. 7, n. 11, p. 115220, nov. 2017. Disponível em: <https://doi.org/10.1063/1.4995529>.

HANDS, A. et al. New data and modelling for single event effects in the stratospheric radiation environment. **IEEE Transactions on Nuclear Science**, v. 64, n. 1, p. 587–595, 2017. Disponível em: <https://doi.org/10.1109/TNS.2016.2612000>.

HAO, J.-B.; LIU, Y.; WANG, Z.-J. Research of transient radiation effects on FinFET SRAMs compared with planar SRAMs. In: 2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT). Hangzhou, China: IEEE, 2016. P. 1005–1007. Disponível em: <https://doi.org/10.1109/ICSICT.2016.7998633>.

HARADA, R. et al. Angular dependency of neutron-induced multiple cell upsets in 65-nm 10T subthreshold SRAM. **IEEE Transactions on Nuclear Science**, v. 59, n. 6, p. 2791–2795, 2012. Disponível em: <https://doi.org/10.1109/TNS.2012.2224373>.

HARDING, A.; WIRTHLIN, M. Improving the Reliability of Xilinx 7 Series FPGAs through Configuration Scrubbing, jun. 2014. Utah Space Grant Consortium Conference. Disponível em: <http://digitalcommons.usu.edu/spacegrant/2014/Session4/1>.

HARWARD, N. A. et al. Estimating Soft Processor Soft Error Sensitivity through Fault Injection. In: 2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines. Vancouver, Canada: IEEE, mai. 2015. Disponível em: <https://doi.org/10.1109/FCCM.2015.61>.

HEBB, D. O. **The organization of behavior: A neuropsychological theory**. New York, USA: John Wiley & Sons, 1949.

HEIDEL, D. F. et al. Low energy proton single-event-upset test results on 65 nm SOI SRAM. **IEEE Transactions on Nuclear Science**, v. 55, n. 6, p. 3394–3400, 2008. Disponível em: <https://doi.org/10.1109/TNS.2008.2005499>.

HEIDT, H. et al. CubeSat: A new Generation of Picosatellite for Education and Industry Low-Cost Space Experimentation. In: PROCEEDINGS of the 14^{th} AIAA/USU Conference on Small Satellites. [S.l.: s.n.], 2000. Technical Session V: Lessons Learned - In Success and Failure, Paper 32. Disponível em: <https://digitalcommons.usu.edu/smallsat/2000/All2000/32>.

HERNANDEZ, H. G. M. et al. A Modular Software Library for Effective High Level Synthesis of Convolutional Neural Networks. In: APPLIED Reconfigurable Computing. Architectures, Tools, and Applications. [S.l.]: Springer, 2020. P. 211–220. Disponível em: <https://doi.org/10.1007/978-3-030-44534-8_16>.

HERNANDEZ, H. G. M. et al. Accelerating Convolutional Neural Networks in FPGA-based SoCs using a Soft-Core GPU. In: APPLIED Reconfigurable Computing. Architectures, Tools, and Applications. [S.l.]: Springer, 2021. P. 275–284. Disponível em: <https://doi.org/10.1007/978-3-030-79025-7_20>.

HOELSCHER, I. G. **Detecção e classificação de sinalização vertical de trânsito em cenários complexos**. 2017. Master Dissertation – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil. Disponível em: <http://hdl.handle.net/10183/163777>.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. **Neural Networks**, Elsevier, v. 2, n. 5, p. 359–366, jan. 1989. Disponível em: <https://doi.org/10.1016/0893-6080(89)90020-8>.

HOUBEN, S. et al. Detection of traffic signs in real-world images: The German traffic sign detection benchmark. In: THE 2013 International Joint Conference on Neural Networks (IJCNN). Dallas, USA: IEEE, ago. 2013. Disponível em: <https://doi.org/10.1109/IJCNN.2013.6706807>.

HSIEH, H.-C. et al. Third-generation architecture boosts speed and density of field-programmable gate arrays. In: IEEE Proceedings of the Custom Integrated Circuits Conference. Boston, USA: IEEE, 16 mai. 1990. Disponível em: <https://doi.org/10.1109/CICC.1990.124841>.

HUGHES, H. et al. Total ionizing dose radiation effects on 14 nm FinFET and SOI UTBB technologies. In: 2015 IEEE Radiation Effects Data Workshop (REDW). Boston, USA: IEEE, 2015. Disponível em: <https://doi.org/10.1109/REDW.2015.7336740>.

IANDOLA, F. N. et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and $<$1MB model size. **arXiv:1602.07360**, 2016. arXiv: 1602.07360.

IBE, E. H. **Terrestrial radiation effects in ULSI devices and electronic systems**. Singapore: Wiley, 2015. (Wiley-IEEE). ISBN 978-1-118-47929-2. Disponível em: <https://www.wiley.com/en-us/9781118479322>.

IEREMEIEV, O. et al. Full-Reference Quality Metric Based on Neural Network to Assess the Visual Quality of Remote Sensing Images. **Remote Sensing**, MDPI, v. 12, n. 15, p. 2349, jul. 2020. Disponível em: <https://doi.org/10.3390/rs12152349>.

INTERNATIONAL ELECTROTECHNICAL COMMISSION. **IEC 61508**: Functional safety of electrical/electronic/programmable electronic safety related systems. Geneva, Switzerland, 30 abr. 2010.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO 26262-11**: Road Vehicles – Functional Safety – Part 11 – Guidelines on application of ISO 26262 to semiconductors. Geneva, Switzerland, 2018.

IRIE, B.; MIYAKE, S. Capabilities of three-layered perceptrons. In: IEEE International Conference on Neural Networks. San Diego, USA: IEEE, 1988. Disponível em: <https://doi.org/10.1109/ICNN.1988.23901>.

ISLAM, M. Md. et al. Towards Benchmarking of Functional Safety in the Automotive Industry. In: LECTURE Notes in Computer Science. [S.l.]: Springer, 2013. P. 111–125. Disponível em: <https://doi.org/10.1007/978-3-642-38789-0_10>.

JAMES, D. High-k/metal gates in leading edge silicon devices. In: 2012 SEMI Advanced Semiconductor Manufacturing Conference. Saratoga Springs, USA: IEEE, 2012. P. 346–353. Disponível em: <https://doi.org/10.1109/ASMC.2012.6212925>.

JEDEC SOLID STATE TECHNOLOGY ASSOCIATION. **JESD89A**: Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices. Arlington, USA, out. 2006. Disponível em: <https://www.jedec.org/sites/default/files/docs/jesd89a.pdf>.

JOHNSON, E.; WIRTHLIN, M. J.; CAFFREY, M. **Single-Event Upset Simulation on an FPGA**. Los Alamos, USA, 2002. Disponível em: <http://www.ericjohnsonweb.net/papers/BYU_ERSA_2002_1017er.pdf>. Acesso em: 1 mai. 2021.

JOHNSON, E. et al. Accelerator validation of an FPGA SEU simulator. **IEEE Transactions on Nuclear Science**, IEEE, v. 50, n. 6, p. 2147–2157, dez. 2003. Disponível em: <https://doi.org/10.1109/TNS.2003.821791>.

KAK, A. et al. Performance Evaluation of SDN-Based Internet of Space Things. In: 2018 IEEE Globecom Workshops (GC Wkshps). Abu Dhabi, United Arab Emirates: IEEE, dez. 2018. Disponível em: <https://doi.org/10.1109/GLOCOMW.2018.8644237>.

KAMPERT, K.-H.; WATSON, A. A. Extensive air showers and ultra high-energy cosmic rays: a historical review. **The European Physical Journal H**, Royal Society, v. 37, n. 3, p. 359–412, ago. 2012. Disponível em: <https://doi.org/10.1140/epjh/e2012-30013-x>.

KANO, K. **Semiconductor devices**. Upper Saddle River, NJ, USA: Prentice-Hall, 1998. ISBN 0-02-361938-4.

KASTENSMIDT, F. L. et al. Designing and Testing Fault-tolerant Techniques for SRAM-based FPGAs. In: PROCEEDINGS of the 1st Conference on Computing Frontiers. Ischia, Italy: ACM, 2004. (CF ’04), p. 419–432. ISBN 1-58113-741-9. Disponível em: <http://doi.acm.org/10.1145/977091.977150>.

KASTENSMIDT, F. L. et al. Laser testing methodology for diagnosing diverse soft errors in a nanoscale SRAM-based FPGA. **IEEE Transactions on Nuclear Science**, v. 61, n. 6, p. 3130–3137, 2014. Disponível em: <https://doi.org/10.1109/TNS.2014.2369008>.

KATO, T. et al. Neutron-induced multiple-cell upsets in 20-nm bulk SRAM: Angular sensitivity and impact of multiwell potential perturbation. **IEEE Transactions on Nuclear Science**, v. 66, n. 7, p. 1381–1389, 2019. Disponível em: <https://doi.org/10.1109/TNS.2019.2900629>.

KORKIAN, G. et al. Experimental and analytical study of the responses of nanoscale devices to neutrons impinging at various incident angles. **IEEE Transactions on Nuclear Science**, v. 67, n. 11, p. 2345–2352, 2020. Disponível em: <https://doi.org/10.1109/TNS.2020.3025104>.

KRIZHEVSKY, A. **Learning Multiple Layers of Features from Tiny Images**. 18 abr. 2009. Master Thesis – University of Toronto, Department of Computer Science, Toronto, Canada. Disponível em: <http://www.cs.toronto.edu/~kriz/cifar.html>.

LAUFER, R.; PELTON, J. N. The Smallest Classes of Small Satellites Including Femtosats, Picosats, Nanosats, and CubeSats. In: HANDBOOK of Small Satellites. [S.l.]: Springer, 2020. P. 87–101. Disponível em: <https://doi.org/10.1007/978-3-030-36308-6_5>.

LE CUN, Y. et al. Handwritten digit recognition: applications of neural network chips and automatic learning. **IEEE Communications Magazine**, IEEE, v. 27, n. 11, p. 41–46, nov. 1989. Disponível em: <https://doi.org/10.1109/35.41400>.

LEE, D. S. et al. Single-Event Characterization of 16 nm FinFET Xilinx UltraScale+ Devices with Heavy Ion and Neutron Irradiation. In: 2018 IEEE Nuclearand Space Radiation Effects Conference (NSREC 2018). Waikoloa, USA: IEEE, jul. 2018. Disponível em: <https://doi.org/10.1109/NSREC.2018.8584313>.

LEE, D. S. et al. Single-Event Characterization of the 28 nm Xilinx Kintex-7 Field-Programmable Gate Array under Heavy Ion Irradiation. In: 2014 IEEE Radiation Effects Data Workshop (REDW). Paris, France: IEEE, jul. 2014. Disponível em: <https://doi.org/10.1109/REDW.2014.7004595>.

LEIPNITZ, M. T.; GEFERSON, L. H.; NAZAR, G. L. A fault injection platform for FPGA-based communication systems. In: 2016 IEEE 7th Latin American Symposium on Circuits Systems (LASCAS). Florianópolis, Brazil: IEEE, 2016. P. 59–62. Disponível em: <https://doi.org/10.1109/LASCAS.2016.7451009>.

LEMIEUX, G. et al. Directional and single-driver wires in FPGA interconnect. In: PROCEEDINGS of 2004 IEEE International Conference on Field- Programmable Technology (FPT2004). Brisbane, Australia: IEEE, 2004. Disponível em: <https://doi.org/10.1109/FPT.2004.1393249>.

LEWIS, D. et al. The Stratix routing and logic architecture. In. Disponível em: <https://doi.org/10.1145/611817.611821>.

LIBANO, F. et al. On the Reliability of Linear Regression and Pattern Recognition Feedforward Artificial Neural Networks in FPGAs. **IEEE Transactions on Nuclear Science**, IEEE, v. 65, n. 1, p. 288–295, jan. 2018. Disponível em: <https://doi.org/10.1109/TNS.2017.2784367>.

LIMA, F. et al. A fault injection analysis of Virtex FPGA TMR design methodology. In: RADECS 2001. 2001 6th European Conference on Radiation and Its Effects on Components and Systems (Cat. No.01TH8605). Grenoble, France: IEEE, 2001. Disponível em: <https://doi.org/10.1109/RADECS.2001.1159293>.

LIN, M.; CHEN, Q.; YAN, S. Network In Network. In: BENGIO, Y.; LE CUN, Y. (Ed.). **2nd International Conference on Learning Representations (ICLR 2014)**. Banff, Canada: ICLR, abr. 2014. (ICLR 2014). Disponível em: <https://iclr.cc>.

LIU, Q. et al. DeepSat V2: feature augmented convolutional neural nets for satellite image classification. **Remote Sensing Letters**, Informa UK Limited, v. 11, n. 2, p. 156–165, nov. 2019. Disponível em: <https://doi.org/10.1080/2150704X.2019.1693071>.

LOPES, I. C. et al. Reliability analysis on case-study traffic sign convolutional neural network on APSoC. In: 2018 IEEE 19th Latin-American Test Symposium (LATS). São Paulo, Brazil: IEEE, 2018. Disponível em: <https://doi.org/10.1109/LATW.2018.8347234>.

LOPES, I. da C. **Convolutional neural network reliability on an APSoC platform a traffic-sign recognition case study**. 2017. Master Dissertation – Programa de Pós-Graduação em Microeletrônica, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil. Disponível em: <http://hdl.handle.net/10183/171094>.

LOPEZ-MARTIN, M. et al. Shallow neural network with kernel approximation for prediction problems in highly demanding data networks. **Expert Systems with Applications**, Elsevier, v. 124, p. 196–208, jun. 2019. Disponível em: <https://doi.org/10.1016/j.eswa.2019.01.063>.

LUTZ, G. **Semiconductor radiation detectors: Device physics**. Berlin, Heidelberg: Springer, 2007. ISBN 978-3-540-71679-2.

MA, Y. et al. End-to-end scalable FPGA accelerator for deep residual networks. In: 2017 IEEE International Symposium on Circuits and Systems (ISCAS). Baltimore, USA: IEEE, mai. 2017. Disponível em: <https://doi.org/10.1109/ISCAS.2017.8050344>.

MAASS, W. Networks of spiking neurons: The third generation of neural network models. **Neural Networks**, Elsevier, v. 10, n. 9, p. 1659–1671, dez. 1997. Disponível em: <https://doi.org/10.1016/S0893-6080(97)00011-7>.

MADRY, S.; PELTON, J. N. Historical Perspectives on the Evolution of Small Satellites. In: HANDBOOK of Small Satellites. [S.l.]: Springer, 2020. P. 33–48. Disponível em: <https://doi.org/10.1007/978-3-030-36308-6_2>.

MAILLARD, P. et al. Neutron, 64 MeV Proton, Thermal Neutron and Alpha Single-Event Upset Characterization of Xilinx 20nm UltraScale Kintex FPGA. In: 2015 IEEE Radiation Effects Data Workshop (REDW). Boston, USA: IEEE, jul. 2015. Disponível em: <https://doi.org/10.1109/REDW.2015.7336723>.

MAILLARD, Pierre et al. Neutron, 64 MeV Proton and Alpha Single-event Characterization of Xilinx 16nm FinFET Zynq UltraScale+ MPSoC. In: 2017 IEEE Radiation Effects Data Workshop (REDW). [S.l.]: IEEE, jul. 2017. Disponível em: <https://doi.org/10.1109/NSREC.2017.8115449>.

MAILLARD, Pierre et al. Radiation Tolerant Deep Learning Processor Unit (DPU) based platform using Xilinx 20nm Kintex UltraScale FPGA. **IEEE Transactions on Nuclear Science**, Institute of Electrical e Electronics Engineers (IEEE), p. 1–1, 2022. Disponível em: <https://doi.org/10.1109/TNS.2022.3216360>.

MÂNICA, Thales Ramos. **Missão NanoSatC-BR1 - análise de telemetria e resultados em órbita**. 18 dez. 2018. Trabalho de Conclusão de Curso (Graduação) – Universidade Federal de Santa Maria, Centro de Tecnologia. Disponível em: <http://repositorio.ufsm.br/handle/1/15423>.

MANNING, J. et al. Machine-Learning Space Applications on SmallSat Platforms with TensorFlow. In: PROCEEDINGS of the 32^{nd} AIAA/USU Conference on Small Satellites. [S.l.: s.n.], 2018. Technical Session 7: Advanced Concepts II. Disponível em: <https://digitalcommons.usu.edu/smallsat/2018/all2018/458/>.

MANUZZATO, A. et al. On the Static Cross Section of SRAM-Based FPGAs. In: 2008 IEEE Radiation Effects Data Workshop. Tucson, USA: IEEE, jul. 2008. Disponível em: <https://doi.org/10.1109/REDW.2008.24>.

MARAIS, I. van Z.; STEYN, W. H.; PREEZ, J. A. du. Construction of an Image Quality Assessment Model for Use on Board an LEO Satellite. In: IGARSS 2008 - 2008 IEEE International Geoscience and Remote Sensing Symposium. Boston, USA: IEEE, 2008. Disponível em: <https://doi.org/10.1109/IGARSS.2008.4779183>.

MASUD, M. I.; WILTON, S. J. E. A New Switch Block for Segmented FPGAs. In: FIELD Programmable Logic and Applications. [S.l.]: Springer, 1999. P. 274–281. Disponível em: <https://doi.org/10.1007/978-3-540-48302-1_28>.

MAY, T. C.; WOODS, M. H. Alpha-particle-induced soft errors in dynamic memories. **IEEE Transactions on Electron Devices**, v. 26, n. 1, p. 2–9, 1979. Disponível em: <https://doi.org/10.1109/T-ED.1979.19370>.

MCCABE, M. F. et al. The future of Earth observation in hydrology. **Hydrology and Earth System Sciences**, v. 21, n. 7, p. 3879–3914, 2017. Disponível em: <https://doi.org/10.5194/hess-21-3879-2017>.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, v. 5, n. 4, p. 115–133, dez. 1943. ISSN 1522-9602. Disponível em: <https://doi.org/10.1007/BF02478259>.

MCDONNELL, M. D. et al. Fast, simple and accurate handwritten digit classification by training shallow neural network classifiers with the ‘Extreme Learning Machine’ algorithm. Edição: F. Schwenker. **PLOS ONE**, Public Library of Science (PLoS), v. 10, n. 8, e0134254, ago. 2015. Disponível em: <https://doi.org/10.1371/journal.pone.0134254>.

MCMORROW, D. et al. Single-event upsets in substrate-etched CMOS SOI SRAMs using ultraviolet optical pulses with sub-micrometer spot size. **IEEE Transactions on Nuclear Science**, v. 60, n. 6, p. 4184–4191, 2013. Disponível em: <https://doi.org/10.1109/TNS.2013.2290307>.

MCMORROW, D. et al. Subbandgap laser-induced single event effects: carrier generation via two-photon absorption. **IEEE Transactions on Nuclear Science**, v. 49, n. 6, p. 3002–3008, 2002. Disponível em: <https://doi.org/10.1109/TNS.2002.805337>.

MENCER, O. et al. The history, status, and future of FPGAs. **Communications of the ACM**, ACM, v. 63, n. 10, p. 36–39, set. 2020. Disponível em: <https://doi.org/10.1145/3410669>.

MICROSEMI CORPORATION. **ProASIC3 FPGA Fabric: User’s Guide**. Aliso Viejo, USA, set. 2012. rev. 4. Disponível em: <http://www.actel.com/documents/PA3_UG.pdf>. Acesso em: 27 out. 2021.

MILLER, F. et al. Investigation of 14 MeV neutron capabilities for SEU hardness evaluation. **IEEE Transactions on Nuclear Science**, v. 60, n. 4, p. 2789–2796, 2013. Disponível em: <https://doi.org/10.1109/TNS.2013.2241078>.

MINETTO, R.; PAMPLONA SEGUNDO, M.; SARKAR, S. Hydra: An Ensemble of Convolutional Neural Networks for Geospatial Land Classification. **IEEE Transactions on Geoscience and Remote Sensing**, IEEE, v. 57, n. 9, p. 6530–6541, set. 2019. Disponível em: <https://doi.org/10.1109/TGRS.2019.2906883>.

MOGOLLON, J. M. et al. FTUNSHADES2: A novel platform for early evaluation of robustness against SEE. In: 2011 12th European Conference on Radiation and Its Effects on Components and Systems. Seville, Spain: IEEE, set. 2011. Disponível em: <https://doi.org/10.1109/RADECS.2011.6131392>.

MORCEL, R. et al. FeatherNet. **ACM Transactions on Reconfigurable Technology and Systems**, ACM, v. 12, n. 2, p. 1–27, jun. 2019. Disponível em: <https://doi.org/10.1145/3306202>.

NAGEL, G. W.; M. NOVO, E. M. L. de; KAMPEL, M. Nanosatellites applied to optical Earth observation: a review. **Ambiente e Agua - An Interdisciplinary Journal of Applied Science**, Instituto de Pesquisas Ambientais em Bacias Hidrograficas (IPABHi), v. 15, n. 3, p. 1, jun. 2020. Disponível em: <http://doi.org/10.4136/ambi-agua.2513>.

NANNIPIERI, P. et al. Introduction to Satellite on-Board Data-Handling. In: NEXT-GENERATION High-Speed Satellite Interconnect. [S.l.]: Springer, 2021. P. 1–21. Disponível em: <https://doi.org/10.1007/978-3-030-77044-0_1>.

NAPOLES, J. et al. A Complete Emulation System for Single Event Effects Analysis. In: 2008 4th Southern Conference on Programmable Logic. Bariloche, Argentina: IEEE, mar. 2008. Disponível em: <https://doi.org/10.1109/SPL.2008.4547760>.

NAPOLES, J. et al. Radiation Environment Emulation for VLSI Designs: A Low Cost Platform based on Xilinx FPGA's. In: 2007 IEEE International Symposium on Industrial Electronics. Vigo, Spain: IEEE, jun. 2007. Disponível em: <https://doi.org/10.1109/ISIE.2007.4375150>.

NARASIMHAM, Balaji et al. Scaling trends and bias dependence of the soft error rate of 16 nm and 7 nm FinFET SRAMs. In: 2018 IEEE International Reliability Physics Symposium (IRPS). [S.l.]: IEEE, mar. 2018. Disponível em: <https://doi.org/10.1109/IRPS.2018.8353583>.

NAZAR, G. L.; CARRO, L. Fast single-FPGA fault injection platform. In: 2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT). Austin, USA: IEEE, 2012. P. 152–157. Disponível em: <https://doi.org/10.1109/DFT.2012.6378216>.

NEWHAUSER, W. D.; ZHANG, R. The physics of proton therapy. **Physics in Medicine and Biology**, IOP, v. 60, n. 8, r155–r209, mar. 2015. Disponível em: <https://doi.org/10.1088/0031-9155/60/8/r155>.

NISHIDA, K.; KURITA, T. Boosting Soft-Margin SVM with Feature Selection for Pedestrian Detection. In: MULTIPLE Classifier Systems. [S.l.]: Springer, 2005. P. 22–31. Disponível em: <https://doi.org/10.1007/11494683_3>.

NOH, Jinhyun et al. Study of Neutron Soft Error Rate (SER) Sensitivity: Investigation of Upset Mechanisms by Comparative Simulation of FinFET and Planar MOSFET SRAMs. **IEEE Transactions on Nuclear Science**, Institute of Electrical e Electronics Engineers (IEEE), v. 62, n. 4, p. 1642–1649, ago. 2015. Disponível em: <https://doi.org/10.1109/TNS.2015.2450997>.

NORMAND, E. Single event upset at ground level. **IEEE Transactions on Nuclear Science**, v. 43, n. 6, p. 2742–2750, 1996. Disponível em: <https://doi.org/10.1109/23.556861>.

NSENGIYUMVA, Patrick et al. A Comparison of the SEU Response of Planar and FinFET D Flip-Flops at Advanced Technology Nodes. **IEEE Transactions on Nuclear Science**, Institute of Electrical e Electronics Engineers (IEEE), v. 63, n. 1, p. 266–272, fev. 2016. Disponível em: <https://doi.org/10.1109/TNS.2015.2508981>.

O'GORMAN, T. J. The effect of cosmic rays on the soft error rate of a DRAM at ground level. **IEEE Transactions on Electron Devices**, v. 41, n. 4, p. 553–557, 1994. Disponível em: <https://doi.org/10.1109/16.278509>.

OGAWA, T.; HINES, J. F. **Market Insight – Prepare for Surging Semiconductor Business Opportunities Driven by Autonomous Vehicles**. Stamford, USA, 12 mai. 2017.

OLIVEIRA, D. et al. Thermal neutrons: a possible threat for supercomputers and safety critical applications. In: 2020 IEEE European Test Symposium (ETS). Tallinn, Estonia: IEEE, 2020. Disponível em: <https://doi.org/10.1109/ETS48528.2020.9131597>.

PACINI, A. A. Cosmic rays: bringing messages from the sky to the Earth's surface. **Revista Brasileira de Ensino de Física**, SciELO, São Paulo, v. 39, 2017. ISSN 1806-1117. Disponível em: <https://doi.org/10.1590/1806-9126-rbef-2016-0168>.

PAIKOWSKY, D. What Is New Space? The Changing Ecosystem of Global Space Activity. **New Space**, Mary Ann Liebert Inc, v. 5, n. 2, p. 84–88, jun. 2017. Disponível em: <https://doi.org/10.1089/space.2016.0027>.

PAN, X. et al. Spatial as Deep: Spatial CNN for Traffic Scene Understanding. In: PROCEEDINGS of the Thirty-Second AAAI Conference on Artificial Intelligence. New Orleans, USA: AAAI Press, fev. 2018. (AAAI'18), p. 7276–7283. Disponível em: <https://aaai.org/Conferences/AAAI-18/>.

PARK, Y.; BAEK, S.; PAIK, S.-B. A brain-inspired network architecture for cost-efficient object recognition in shallow hierarchical neural networks. **Neural Networks**, Elsevier, v. 134, p. 76–85, fev. 2021. Disponível em: <https://doi.org/10.1016/j.neunet.2020.11.013>.

PARK, Y.; CHOI, W.; PAIK, S.-B. Symmetry of learning rate in synaptic plasticity modulates formation of flexible and stable memories. **Scientific Reports**, Springer, v. 7, n. 1, mai. 2017. Disponível em: <https://doi.org/10.1038/s41598-017-05929-2>.

PETERSEN, E. Soft errors due to protons in the radiation belt. **IEEE Transactions on Nuclear Science**, v. 28, n. 6, p. 3981–3986, 1981. Disponível em: <https://doi.org/10.1109/TNS.1981.4335659>.

PLANET LABS INC. **Planet imagery: product specification**. [S.l.], jan. 2018. Disponível em: <https://www.planet.com/products/satellite-imagery/files/Planet_Combined_Imagery_Product_Specs_December2017.pdf>. Acesso em: 27 out. 2021.

POIVEY, C. **TNID total non ionizing dose or DD displacement damage**. Geneva: [s.n.], 10 mai. 2017. ESA-CERN-SCC Workshop, Session TNID effect, mechanisms and testing. Disponível em: <https://indico.cern.ch/event/635099/>. Acesso em: 1 jun. 2021.

POPOWICZ, A. Image processing in the BRITE nano-satellite mission. In: MACEWEN, H. A. et al. (Ed.). **Space Telescopes and Instrumentation 2016: Optical, Infrared, and Millimeter Wave**. Edinburgh, UK: SPIE, jul. 2016. Disponível em: <https://doi.org/10.1117/12.2229141>.

POUGET, V. et al. Structural pattern extraction from asynchronous two-photon laser fault injection using spectral analysis. **Microelectronics Reliability**, v. 76-77, p. 650–654, 2017. ISSN 0026-2714. Disponível em: <https://doi.org/10.1016/j.microrel.2017.07.028>.

POUGET, V. et al. Tools and methodology development for pulsed laser fault injection in SRAM-based FPGAs. In: 8TH LATIN-AMERICAN TEST WORKSHOP (LATW'07). Cuzco, Peru: IEEE Computer Society, 2007. Session 8. Disponível em: <https://hal.archives-ouvertes.fr/hal-00156318>.

QUINN, H.; WIRTHLIN, M. Validation Techniques for Fault Emulation of SRAM-based FPGAs. **IEEE Transactions on Nuclear Science**, IEEE, v. 62, n. 4, p. 1487–1500, ago. 2015. Disponível em: <https://doi.org/10.1109/TNS.2015.2456101>.

QUINN, H. et al. Radiation-induced multi-bit upsets in SRAM-based FPGAs. **IEEE Transactions on Nuclear Science**, IEEE, v. 52, n. 6, p. 2455–2461, dez. 2005. Disponível em: <https://doi.org/10.1109/TNS.2005.860742>.

RASHEVSKY, N. Outline of a physico-mathematical theory of excitation and inhibition. **Protoplasma**, Springer, v. 20, n. 1, p. 42–56, out. 1933. Disponível em: <https://doi.org/10.1007/BF02674811>.

RECH, P. et al. Impact of GPUs Parallelism Management on Safety-Critical and HPC Applications Reliability. In: 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. [S.l.]: IEEE, jun. 2014. Disponível em: <https://doi.org/10.1109/DSN.2014.49>.

REED, R. A. et al. Evidence for angular effects in proton-induced single-event upsets. **IEEE Transactions on Nuclear Science**, v. 49, n. 6, p. 3038–3044, 2002. Disponível em: <https://doi.org/10.1109/TNS.2002.805446>.

RODBELL, K. P. et al. Low-energy proton-induced single-event-upsets in 65 nm node, silicon-on-insulator, latches and memory cells. **IEEE Transactions on Nuclear Science**, v. 54, n. 6, p. 2474–2479, 2007. Disponível em: <https://doi.org/10.1109/TNS.2007.909845>.

RODRIGUES, G. S. et al. Exploiting approximate computing for low-cost fault tolerant architectures. In: PROCEEDINGS of the 32nd Symposium on Integrated Circuits and Systems Design. São Paulo, Brazil: ACM, 2019. (SBCCI '19). ISBN 9781450368445. Disponível em: <https://doi.org/10.1145/3338852.3339875>.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, American Psychological Association (APA), v. 65, n. 6, p. 386–408, 1958. Disponível em: <https://doi.org/10.1037/h0042519>.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, Springer, v. 323, n. 6088, p. 533–536, out. 1986. Disponível em: <https://doi.org/10.1038/323533a0>.

RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. **International Journal of Computer Vision (IJCV)**, v. 115, n. 3, p. 211–252, 2015. Disponível em: <https://doi.org/10.1007/s11263-015-0816-y>.

S. CURIEL, A. da; CAWTHORNE, A.; SWEETING, M. Progress in small satellite technology for Earth obsevation missions. In: SMALL Satellites for Earth Observation. [S.l.]: De Gruyter, dez. 2005. P. 50–63. Disponível em: <https://doi.org/10.1515/9783110919806.50>.

SADEH, Y. et al. Fusion of Sentinel-2 and PlanetScope time-series data into daily 3 m surface reflectance and wheat LAI monitoring. **International Journal of Applied Earth Observation and Geoinformation**, Elsevier, v. 96, p. 102260, abr. 2021. Disponível em: <https://doi.org/10.1016/j.jag.2020.102260>.

SAE INTERNATIONAL. **SAE J3016 – Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles**. Warrendale, USA, abr. 2021. Also published as ISO/SAE PAS 22736. Disponível em: <https://www.sae.org/standards/content/j3016_202104/>.

SAMU, D.; SETH, A. K.; NOWOTNY, T. Influence of wiring cost on the large-scale architecture of human cortical connectivity. Edição: O. Sporns. **PLoS Computational Biology**, PLoS, v. 10, n. 4, e1003557, abr. 2014. Disponível em: <https://doi.org/10.1371/journal.pcbi.1003557>.

SANTOS, A. F. dos et al. Applying TMR in hardware accelerators generated by High-Level Synthesis design flow for mitigating multiple bit upsets in SRAM-based FPGAs. In: WONG, S. et al. (Ed.). **Applied Reconfigurable Computing: 13th International Symposium, ARC 2017, Delft, The Netherlands, April 3-7, 2017, Proceedings**. Cham: Springer, 2017. P. 202–213. ISBN 978-3-319-56258-2. Disponível em: <https://doi.org/10.1007/978-3-319-56258-2_18>.

SCHRIMPF, R. D. Radiation effects in microelectronics. In: VELAZCO, R.; FOUILLAT, P.; REIS, R. (Ed.). **Radiation Effects on Embedded Systems**. Dordrecht: Springer, 2007. P. 11–29. ISBN 978-1-4020-5646-8. Disponível em: <https://doi.org/10.1007/978-1-4020-5646-8_2>.

SCHUCH, Nelson J. et al. The NanosatC-BR, CubeSat Development Program - A joing CubeSat program developed by UFSM and INPE/MCTIC - Space geophysics mission payloads and first results. **Brazilian Journal of Geophysics**, Sociedade Brasileira de Geofisica, v. 37, n. 1, p. 95, mar. 2019. Disponível em: <http://dx.doi.org/10.22564/rbgf.v37i1.1992>.

SCHUCH, Nelson Jorge; DURÃO, Otávio Cupertino. The Brazilian INPE-UFSM NANOSATC-BR CubeSat Program. In: PROCEEDINGS of the 2^{nd} IAA Conference on University Satellites Missions and the CubeSat Winter Workshop. [S.l.]: International Academy of Astronautics, fev. 2013. Technical Session VII: CubeSat Missions, Paper IAA-CU-13-09-02.

SELVA, D.; KREJCI, D. A survey and assessment of the capabilities of Cubesats for Earth observation. **Acta Astronautica**, Elsevier, v. 74, p. 50–68, mai. 2012. Disponível em: <https://doi.org/10.1016/j.actaastro.2011.12.014>.

SERNA, C. G.; RUICHEK, Y. Classification of Traffic Signs: The European Dataset. **IEEE Access**, IEEE, v. 6, p. 78136–78148, 2018. Disponível em: <https://doi.org/10.1109/ACCESS.2018.2884826>.

SIERAWSKI, B. D. et al. Impact of low-energy proton induced upsets on test methods and rate predictions. **IEEE Transactions on Nuclear Science**, v. 56, n. 6, p. 3085–3092, 2009. Disponível em: <https://doi.org/10.1109/TNS.2009.2032545>.

SKOROBOGATOV, S. P.; ANDERSON, R. J. Optical fault induction attacks. In: KALISKI, B. S.; KOÇ, Ç. K.; PAAR, C. (Ed.). **Cryptographic Hardware and Embedded Systems - CHES 2002**. Berlin, Heidelberg: Springer, 2003. P. 2–12. ISBN 978-3-540-36400-9. Disponível em: <https://doi.org/10.1007/3-540-36400-5_2>.

SPRINGENBERG, J. T. et al. Striving for Simplicity: The All Convolutional Net. In: BENGIO, Y.; LE CUN, Y. (Ed.). **3rd International Conference on Learning Representations (ICLR 2015)**. San Diego, USA: ICLR, mai. 2015. (ICLR 2015). Disponível em: <https://iclr.cc>.

STALLKAMP, J. et al. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. **Neural Networks**, Elsevier, v. 32, p. 323–332, ago. 2012. Disponível em: <https://doi.org/10.1016/j.neunet.2012.02.016>.

STASSINOPOULOS, E. G.; RAYMOND, J. P. The space radiation environment for electronics. **Proceedings of the IEEE**, v. 76, n. 11, p. 1423–1442, 1988. Disponível em: <https://doi.org/10.1109/5.90113>.

STERPONE, L.; BORAGNO, L. Analysis of radiation-induced cross domain errors in TMR architectures on SRAM-based FPGAs. In: 2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS). Thessaloniki, Greece: IEEE, jul. 2017. Disponível em: <https://doi.org/10.1109/IOLTS.2017.8046214>.

STERPONE, L.; VIOLANTE, M. A New Partial Reconfiguration-Based Fault-Injection System to Evaluate SEU Effects in SRAM-Based FPGAs. **IEEE Transactions on Nuclear Science**, IEEE, v. 54, n. 4, p. 965–970, ago. 2007. Disponível em: <https://doi.org/10.1109/TNS.2007.904080>.

STERPONE, L. et al. Layout-Aware Multi-Cell Upsets Effects Analysis on TMR Circuits Implemented on SRAM-Based FPGAs. **IEEE Transactions on Nuclear Science**, IEEE, v. 58, n. 5, p. 2325–2332, out. 2011. Disponível em: <https://doi.org/10.1109/TNS.2011.2161887>.

STODDARD, A. G. **Configuration Scrubbing Architectures for High Reliability FPGA Systems**. Dez. 2015. Master of Science Thesis – Brigham Young University, Provo, USA. Disponível em: <https://scholarsarchive.byu.edu/etd/5704>.

SWARTWOUT, M. University-Class Satellites: From Marginal Utility to 'Disruptive' Research Platforms. In: PROCEEDINGS of the 18^{th} AIAA/USU Conference on Small Satellites. [S.l.: s.n.], 2004. Technical Session II: Measuring Small Satellite Utility, Paper 12. Disponível em: <https://digitalcommons.usu.edu/smallsat/2004/All2004/12/>.

SZE, S. M.; NG, K. K. **Physics of semiconductor devices**. 3. ed. New Delhi, India: Wiley, 2007. ISBN 978-81-265-1702-2.

TAIWAN SEMICONDUCTOR MANUFACTURING CO.,LTD. **28nm Technology**. Hsinchu, Taiwan: TSMC, 2018. Disponível em: <http://www.tsmc.com/english/dedicatedFoundry/technology/28nm.htm>. Acesso em: 28 jan. 2018.

TAIWAN SEMICONDUCTOR MANUFACTURING CO.,LTD. Fang-Wen Tsai et al. **Extreme low-K dielectric film scheme for advanced interconnects**. 5 jul. 2011. US n. RE42,514 E, Jan. 2, 2008, Jul. 5, 2011. Disponível em: <https://patentscope.wipo.int/search/en/detail.jsf?docId=US73761086>.

TAIWAN SEMICONDUCTOR MANUFACTURING CO.,LTD. Shiauhan Wu; Joung-Wei Liou; Han-Ti Hsiaw. **Method of back-end-of-line (BEOL) fabrication, and devices formed by the method**. 8 set. 2015. US 9,130,022 B2, May 9, 2013, Sep. 8, 2015. Disponível em: <https://patentscope.wipo.int/search/en/detail.jsf?docId=US107408853>.

TAMBARA, L. A. et al. Analyzing the Impact of Radiation-Induced Failures in Programmable SoCs. **IEEE Transactions on Nuclear Science**, IEEE, v. 63, n. 4, p. 2217–2224, ago. 2016. Disponível em: <https://doi.org/10.1109/TNS.2016.2522508>.

TAMBARA, L. A. et al. Heavy Ions Induced Single Event Upsets Testing of the 28 nm Xilinx Zynq-7000 All Programmable SoC. In: 2015 IEEE Radiation Effects Data Workshop (REDW). [S.l.]: IEEE, jul. 2015. Disponível em: <https://doi.org/10.1109/REDW.2015.7336716>.

TAMBARA, L. A. et al. On the Characterization of Embedded Memories of Zynq-7000 All Programmable SoC under Single Event Upsets Induced by Heavy Ions and Protons. In: 2015 15th European Conference on Radiation and Its Effects on Components and Systems (RADECS). Moscow, Russia: IEEE, set. 2015. Disponível em: <https://doi.org/10.1109/RADECS.2015.7365643>.

TAMBARA, L. A. et al. Soft error rate in SRAM-based FPGAs under neutron-induced and TID effects. In: 2014 15th Latin American Test Workshop - LATW. Fortaleza, Brazil: IEEE, mar. 2014. Disponível em: <https://doi.org/10.1109/LATW.2014.6841920>.

TARRILLO, J. et al. Multiple fault injection platform for SRAM-based FPGA based on ground-level radiation experiments. In: 2015 16th Latin-American Test Symposium (LATS). Puerto Vallarta: IEEE, 2015. Disponível em: <https://doi.org/10.1109/LATW.2015.7102494>.

TARRILLO, J. et al. Neutron Cross-Section of N-Modular Redundancy Technique in SRAM-Based FPGAs. **IEEE Transactions on Nuclear Science**, IEEE, v. 61, n. 4,

p. 1558–1566, ago. 2014. Disponível em: <https://doi.org/10.1109/TNS.2014.2343259>.

THURLOW, C.; ROWBERRY, H.; WIRTHLIN, M. TURTLE: A Low-Cost Fault Injection Platform for SRAM-based FPGAs. In: 2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig). Cancun, Mexico: IEEE, dez. 2019. Disponível em: <https://doi.org/10.1109/ReConFig48160.2019.8994782>.

TONFAT, J. et al. Analyzing the Influence of the Angles of Incidence and Rotation on MBU Events Induced by Low LET Heavy Ions in a 28-nm SRAM-Based FPGA. **IEEE Transactions on Nuclear Science**, IEEE, v. 64, n. 8, p. 2161–2168, ago. 2017. Disponível em: <https://doi.org/10.1109/TNS.2017.2727479>.

TONFAT, J. et al. Method to analyze the susceptibility of HLS designs in SRAM-based FPGAs under soft errors. In: BONATO, V.; BOUGANIS, C.; GORGON, M. (Ed.). **Applied Reconfigurable Computing**. Mangaratiba, Brazil: Springer, 2016. P. 132–143. ISBN 978-3-319-30481-6. Disponível em: <https://doi.org/10.1007/978-3-319-30481-6_11>.

TRINDADE, M. G. et al. Effects of thermal neutron radiation on a hardware-implemented machine learning algorithm. **Microelectronics Reliability**, v. 116, p. 114022, jan. 2021. ISSN 0026-2714. Disponível em: <https://doi.org/10.1016/j.microrel.2020.114022>.

TRIPPE, J. M. et al. Electron-induced single event upsets in 28 nm and 45 nm bulk SRAMs. **IEEE Transactions on Nuclear Science**, v. 62, n. 6, p. 2709–2716, 2015. Disponível em: <https://doi.org/10.1109/TNS.2015.2496967>.

TSAO, C. H.; SILBERBERG, R.; LETAW, J. R. Cosmic-ray heavy ions at and above 40,000 feet. **IEEE Transactions on Nuclear Science**, v. 31, n. 6, p. 1066–1068, 1984. Disponível em: <https://doi.org/10.1109/TNS.1984.4333456>.

TSILIGIANNIS, G. et al. Radiation Effects on Deep Submicrometer SRAM-Based FPGAs Under the CERN Mixed-Field Radiation Environment. **IEEE Transactions on Nuclear Science**, IEEE, v. 65, n. 8, p. 1511–1518, ago. 2018.

UNION OF CONCERNED SCIENTISTS. **UCS Satellite Database**. Cambridge, USA: [s.n.], 1 mai. 2021. Microsoft Office Excel Spreadsheet, 2405376 octets. Disponível em: <https://www.ucsusa.org/resources/satellite-database>. Acesso em: 30 jul. 2021.

VECTORBLOX COMPUTING INC. **ORCA FPGA optimized RISC-V**. Vancouver, Canada: Github, 2019. Disponível em: <https://github.com/VectorBlox/risc-v>. Acesso em: 4 jul. 2019.

VEDDER, B. **Testing Safety-Critical Systems using Fault Injection and Property-Based Testing**. 26 mai. 2015. Licenciate Thesis – Halmstad University, Centre for Research on Embedded Systems (CERES). ISBN 978-91-87045-28-8. Disponível em: <http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A808260>.

VELAZCO, R.; FOUCARD, G.; PERONNARD, P. Combining Results of Accelerated Radiation Tests and Fault Injections to Predict the Error Rate of an Application Implemented in SRAM-Based FPGAs. **IEEE Transactions on Nuclear Science**, v. 57, n. 6, p. 3500–3505, dez. 2010. ISSN 0018-9499. Disponível em: <http://dx.doi.org/10.1109/TNS.2010.2087355>.

VELAZCO, R. et al. Evidences of SEU tolerance for digital implementations of artificial neural networks: one year MPTB flight results. In: 1999 Fifth European Conference on Radiation and Its Effects on Components and Systems. RADECS 99 (Cat. No.99TH8471). Fontevraud, France: IEEE, 1999. Disponível em: <https://doi.org/10.1109/RADECS.1999.858648>.

VELHO, L.; FRERY, A.; GOMES, J. **Image Processing for Computer Graphics and Vision**. [S.l.]: Springer, 2009. Disponível em: <https://doi.org/10.1007/978-1-84800-193-0>.

VIAL, C. et al. A new approach for the prediction of the neutron-induced SEU rate. **IEEE Transactions on Nuclear Science**, v. 45, n. 6, p. 2915–2920, 1998. Disponível em: <https://doi.org/10.1109/23.736547>.

VILLALTA, I. et al. Fault injection system for SEU emulation in Zynq SoCs. In: DESIGN of Circuits and Integrated Systems. Madrid, Spain: IEEE, nov. 2014. Disponível em: <https://doi.org/10.1109/DCIS.2014.7035579>.

VON NEUMANN, J. First draft of a report on the EDVAC. **IEEE Annals of the History of Computing**, IEEE, v. 15, n. 4, p. 27–75, 1993. Disponível em: <https://doi.org/10.1109/85.238389>.

WESTE, N. H. E.; HARRIS, D. M. **CMOS VLSI design: A circuits and systems perspective**. 4. ed. Boston, USA: Addison-Wesley, 2010. ISBN 978-0-321-54774-3.

WIELGOSZ, M.; KARWATOWSKI, M. Mapping Neural Networks to FPGA-Based IoT Devices for Ultra-Low Latency Processing. **Sensors**, MDPI, v. 19, n. 13, p. 2981, jul. 2019. Disponível em: <https://doi.org/10.3390/s19132981>.

WINOKUR, P. S. et al. Use of COTS microelectronics in radiation environments. **IEEE Transactions on Nuclear Science**, v. 46, n. 6, p. 1494–1503, dez. 1999. ISSN 0018-9499. Disponível em: <https://doi.org/10.1109/23.819113>.

WIRTHLIN, M. et al. A Method and Case Study on Identifying Physically Adjacent Multiple-Cell Upsets Using 28-nm, Interleaved and SECDED-Protected Arrays. **IEEE Transactions on Nuclear Science**, IEEE, v. 61, n. 6, p. 3080–3087, dez. 2014. Disponível em: <https://doi.org/10.1109/TNS.2014.2366913>.

WIRTHLIN, M. J.; TAKAI, H.; HARDING, A. Soft error rate estimations of the Kintex-7 FPGA within the ATLAS Liquid Argon (LAr) Calorimeter. **Journal of Instrumentation**, v. 9, n. 01, p. c01025, 2014. Disponível em: <http://stacks.iop.org/1748-0221/9/i=01/a=C01025>.

WONG, S.; AS, T. van; BROWN, G. ρ-VEX: A reconfigurable and extensible softcore VLIW processor. In: 2008 International Conference on Field-Programmable Technology. Taipei, Taiwan: IEEE, dez. 2008. Disponível em: <https://doi.org/10.1109/FPT.2008.4762420>.

WROBEL, F. et al. Contribution of SiO2 in neutron-induced SEU in SRAMs. **IEEE Transactions on Nuclear Science**, v. 50, n. 6, p. 2055–2059, 2003. Disponível em: <https://doi.org/10.1109/TNS.2003.821596>.

XILINX, INC. **7 Series FPGA DSP48E1 Slice: User Guide**. San Jose, USA, 27 mar. 2018. UG479 v. 1.10. Disponível em: <https://www.xilinx.com/support/documentation/user_guides/ug479_7Series_DSP48E1.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **7 Series FPGAs Configurable Logic Block: User Guide**. San Jose, USA, 27 set. 2016. UG474 v. 1.8. Disponível em: <https://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **7 Series FPGAs Configuration: User Guide**. San Jose, USA, 20 ago. 2018. UG470 v. 1.13.1. Disponível em: <https://www.xilinx.com/support/documentation/user_guides/ug470_7Series_Config.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **7 Series FPGAs Memory Resources: User Guide**. San Jose, USA, 3 jul. 2019. UG473 v. 1.14. Disponível em: <https://www.xilinx.com/support/documentation/user_guides/ug473_7Series_Memory_Resources.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **7 Series FPGAs Mitigating Single-Event Upsets: White Paper**. San Jose, USA, 19 mai. 2015. WP395 v. 1.1. Disponível em: <https://www.xilinx.com/support/documentation/white_papers/wp395-Mitigating-SEUs.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **7 Series FPGAs Packaging and Pinout: Product Specification**. San Jose, USA, 7 abr. 2021. UG475 v. 1.19. Disponível em: <https://www.xilinx.com/support/documentation/user_guides/ug475_7Series_Pkg_Pinout.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **Device Reliability Report: First Half 2021**. San Jose, USA, 10 nov. 2021. UG116 v. 10.15. Disponível em: <https://www.xilinx.com/support/documentation/user_guides/ug116.pdf>. Acesso em: 15 nov. 2021.

XILINX, INC. **Isolation Design Flow for Xilinx 7 Series FPGAs or Zynq-7000 AP SoCs (Vivado Tools): Application Note**. San Jose, USA, 17 dez. 2020. XAPP1222 v. 1.4. Disponível em: <https://www.xilinx.com/support/documentation/application_notes/xapp1222-idf-for-7s-or-zynq-vivado.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **Soft Error Mitigation Using Prioritized Essential Bits: Application Note**. San Jose, USA, 4 abr. 2012. XAPP538 v.1.0. Disponível em: <https://www.xilinx.com/support/documentation/application_notes/xapp538-soft-error-mitigation-essential-bits.pdf>. Acesso em: 1 fev. 2018.

XILINX, INC. **UltraScale Architechture Configuration: User Guide**. San Jose, USA, 9 set. 2021. UG570 v. 1.15. Disponível em: <https://www.xilinx.com/support/documentation/user_guides/ug570-ultrascale-configuration.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **UltraScale Architecture Configurable Logic Block: User Guide**. San Jose, USA, 28 fev. 2017. UG574 v. 1.5. Disponível em: <https:

//www.xilinx.com/support/documentation/user_guides/ug574-ultrascale-clb.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **UltraScale Architecture DSP Slice User Guide**. San Jose, USA, 30 ago. 2021. UG579 v. 1.11. Disponível em: <https://www.xilinx.com/support/documentation/user_guides/ug479_7Series_DSP48E1.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **UltraScale Architecture Memory Resources User Guide**. San Jose, USA, 24 set. 2021. UG573 v. 1.13. Disponível em: <https://www.xilinx.com/support/documentation/user_guides/ug573-ultrascale-memory-resources.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **Versal ACAP DSP Engine Architecture Manual**. San Jose, USA, 15 jul. 2021. AM004 v. 1.1.2. Disponível em: <https://www.xilinx.com/support/documentation/architecture-manuals/am004-versal-dsp-engine.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **Versal: The First Adaptive Compute Acceleration Platform (ACAP): White Paper**. San Jose, USA, 29 set. 2020. WP505 v 1.1.1. Disponível em: <https://www.xilinx.com/support/documentation/white_papers/wp505-versal-acap.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **Virtex-5 FPGA XtremeDSP Design Considerations: User Guide**. San Jose, USA, 27 jul. 2017. UG193 v. 3.6. Disponível em: <https://www.xilinx.com/support/documentation/user_guides/ug193.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **Virtex-5 FPGA: User Guide**. San Jose, USA, 16 mar. 2012. UG190 v. 5.4. Disponível em: <https://www.xilinx.com/support/documentation/user_guides/ug190.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **Virtex-6 FPGA DSP48E1 Slice: User Guide**. San Jose, USA, 14 fev. 2011. UG369 v. 1.3. Disponível em: <https://www.xilinx.com/support/documentation/user_guides/ug369.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **Virtex-II Platform FPGA: User Guide**. San Jose, USA, 5 nov. 2007. UG002 v. 2.2. Disponível em: <https://www.xilinx.com/support/documentation/user_guides/ug002.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **Vivado Design Suite Getting Started: User Guide**. San Jose, USA, 18 nov. 2015. UG910 v. 2015.4. Disponível em: <http://www.xilinx.com/support/documentation/sw_manuals/xilinx2015_4/ug910-vivado-getting-started.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **Vivado Design Suite Properties Reference Guide: User Guide**. San Jose, USA, 6 jun. 2018. UG912 v. 2018.2. Disponível em: <https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_2/ug912-vivado-properties.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **Vixtex Field Programmable Gate Arrays: Product Specification**. San Jose, USA, 1 mar. 2013. DS003 v. 4.0. Disponível em: <https://www.xilinx.com/support/documentation/data_sheets/ds003.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **XC2064/XC2018 Logic Cell Array: Product Specification**. San Jose, USA, 1988.

XILINX, INC. **XC4000E and XC4000X Series Field Programmable Gate Arrays: Product Specification**. San Jose, USA, 14 mai. 1999. v. 1.6. Disponível em: <https://www.xilinx.com/support/documentation/data_sheets/4000.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **Xilinx Multi-node Technology Leadership Continues with UltraScale+ Portfolio "3D on 3D" Solutions: White Paper**. San Jose, USA, 15 dez. 2015. WP472 v. 1.0. Disponível em: <https://www.xilinx.com/support/documentation/white_papers/wp472-3D-on-3D.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **Xilinx Stacked Silicon Interconnect Technology Delivers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency: White Paper**. San Jose, USA, 11 dez. 2012. WP380 v. 1.2. Disponível em: <https://www.xilinx.com/support/documentation/white_papers/wp380_Stacked_Silicon_Interconnect_Technology.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. Ross H. Freeman. **Configurable electrical circuit having configurable logic elements and configurable interconnects**. 19 fev. 1988. US n. 4,870,302, Feb. 19

1988, Sep. 26 1989. Disponível em: <https://patentscope.wipo.int/search/en/detail.jsf?docId=US37788936>.

XILINX, INC. **XtremeDSP for Virtex-4 FPGAs: User Guide**. San Jose, USA, 15 mai. 2008. UG073 v. 2.7. Disponível em: <https://www.xilinx.com/support/documentation/user_guides/ug073.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **Zynq-7000 All Programmable SoC**. San Jose, USA, 28 jan. 2018. Disponível em: <https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html>. Acesso em: 28 jan. 2018.

XILINX, INC. **Zynq-7000 All Programmable SoC Packaging and Pinout: User Guide**. San Jose, USA, 28 jul. 2021. UG865 v. 1.9. Disponível em: <https://www.xilinx.com/support/documentation/user_guides/ug865-Zynq-7000-Pkg-Pinout.pdf>. Acesso em: 27 out. 2021.

XILINX, INC. **Zynq-7000 All Programmable SoC Technical Reference Manual: User Guide**. San Jose, USA, 2 abr. 2021. UG585 v1.13. Disponível em: <https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf>. Acesso em: 27 out. 2021.

YANG, S.H. et al. 28nm metal-gate high-K CMOS SoC technology for high-performance mobile applications. In: 2011 IEEE Custom Integrated Circuits Conference (CICC). San Jose, USA: IEEE, set. 2011. Disponível em: <https://doi.org/10.1109/CICC.2011.6055355>.

YANG, W. et al. Atmospheric neutron single event effect test on Xilinx 28 nm system on chip at CSNS-BL09. **Microelectronics Reliability**, Elsevier, v. 99, p. 119–124, ago. 2019. Disponível em: <https://doi.org/10.1016/j.microrel.2019.05.004>.

YANG, W.-T. et al. Single-event effects induced by medium-energy protons in 28 nm system-on-chip. **Nuclear Science and Techniques**, Springer, v. 30, n. 10, set. 2019. Disponível em: <https://doi.org/10.1007/s41365-019-0672-5>.

YANG, Y.; NEWSAM, S. Bag-of-visual-words and spatial extensions for land-use classification. In: PROCEEDINGS of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '10. San Jose, USA: ACM, 2010. Disponível em: <https://doi.org/10.1145/1869790.1869829>.

YUHANIZ, S.; VLADIMIROVA, T.; SWEETING, M. Embedded Intelligent Imaging On-Board Small Satellites. In: ADVANCES in Computer Systems Architecture. Berlin, Heidelberg: Springer, 2005. P. 90–103. Disponível em: <https://doi.org/10.1007/11572961_9>.

ZHANG, C. et al. Caffeine: Towards Uniformed Representation and Acceleration for Deep Convolutional Neural Networks. In: PROCEEDINGS of the 35th International Conference on Computer-Aided Design. Austin, USA: ACM, nov. 2016. Disponível em: <https://doi.org/10.1145/2966986.2967011>.

ZHANG, Y. et al. Primary single event effect studies on Xilinx 28-nm System-on-Chip (SoC). **Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment**, Elsevier, v. 831, p. 339–343, set. 2016. Disponível em: <https://doi.org/10.1016/j.nima.2016.05.120>.

ZHU, Z. et al. Traffic-Sign Detection and Classification in the Wild. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, USA: IEEE, jun. 2016. Disponível em: <https://doi.org/10.1109/CVPR.2016.232>.

ZIEGLER, J. F.; ZIEGLER, M. D.; BIERSACK, J. P. SRIM – The stopping and range of ions in matter (2010). **Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms**, v. 268, n. 11, p. 1818–1823, 2010. 19th International Conference on Ion Beam Analysis. ISSN 0168-583X. Disponível em: <https://doi.org/10.1016/j.nimb.2010.02.091>.

APPENDIX A — PUBLICATIONS

A.1 On experimental physics

Contributions on observations of radiation phenomena and characterization of electronic devices.

AGUIAR, V. A. P.; MEDINA, N. H.; ADDED, N.; MACCHIONE, E. L. A.; ALBERTON, S. G.; RODRIGUES, C. L.; SILVA, T. F.; ZAHN, G. S.; GENEZINI, F. A.; MORALLES, M.; BENEVENUTI, F.; GUAZZELLI, M. A. Thermal neutron induced upsets in 28nm SRAM. **Journal of Physics: Conference Series**, IOP, v. 1291, p. 012025, jul. 2019. Disponível em: <https://doi.org/10.1088/1742-6596/1291/1/012025>.

AGUIAR, V. A. P.; MEDINA, N. H.; ADDED, N.; MACCHIONE, E. L. A.; ALBERTON, S. G.; LEITE, A. R.; AGUIRRE, F. R.; RIBAS, R. V.; PEREGO, C. C.; FAGUNDES, L. M.; TERASSI, J. C.; BRAGE, J. A. P.; SIMÕES, R. F.; MORAIS, O. B.; ALMEIDA, E. A.; JOAQUIM, P. M.; SOUZA, M. S.; CECOTTE, A. F. M.; MARTINS, R.; DUARTE, J. G.; SCARDUELLI, V. B.; ALLEGRO, P. R. P.; ESCUDEIRO, R.; LEISTENSCHNEIDER, E.; OLIVEIRA, R. A. N.; SERVELO, W. A.; SILVA, M. T.; SARMENTO, V. E.; CARREIRA, C. A.; ABREU, J. C.; SILVA, S. C.; SANTOS, H. C.; RODRIGUES, C. L.; ASSIS, R. F.; SILVA, T. F.; TABACNIKS, M. H.; JOAQUIM, A. S.; MINAS, J. H. P.; KASHINSKY, D.; GUAZZELLI, M. A.; SEIXAS, L. E.; FINCO, S.; BENEVENUTTI, F. SAFIIRA: A heavy-ion multi-purpose irradiation facility in Brazil. **Review of Scientific Instruments**, v. 91, n. 5, p. 053301, 2020. Disponível em: <https://doi.org/10.1063/1.5138644>.

A.2 On fault injection methodology

Contributions on emulated fault injection methodologies for laser fault attack and radiation induced faults.

BENEVENUTI, F.; KASTENSMIDT, F. L. Evaluation of fault attack detection on SRAM-based FPGAs. In: 2017 18th IEEE Latin American Test Symposium (LATS).

Bogota, Colombia: IEEE, 2017. Disponível em:
<https://doi.org/10.1109/LATW.2017.7906747>.

BENEVENUTI, F.; KASTENSMIDT, F. L. **Reliability Evaluation on Xilinx 7 Series 28nm SRAM-based FPGAs Using Fault Injection Enhanced by Data Collected from Laser and Heavy Ions Irradiation**. São Paulo, Brazil: SBF, 2018. XX Escola de Ver ao Jorge André Swieca de Física Nuclear Experimental (XX EVJASFNE). Disponível em: <http://www.sbfisica.org.br/~evjasfne/xx/>.

BENEVENUTI, F.; KASTENSMIDT, F. L. Comparing exhaustive and random fault injection methods for configuration memory on SRAM-based FPGAs. In: 2019 IEEE 20th Latin-American Test Symposium (LATS). Santiago, Chile: IEEE, mar. 2019. P. 87–92. Disponível em:
<https://doi.org/10.1109/LATW.2019.8704647>.

BENEVENUTI, F.; LIBANO, F.; POUGET, V.; KASTENSMIDT, F. L.; RECH, P. Comparative analysis of inference errors in a neural network implemented in SRAM-based FPGA induced by neutron irradiation and fault injection methods. In: 2018 31st Symposium on Integrated Circuits and Systems Design (SBCCI). Bento Gonçalves, Brazil: IEEE, 2018. Disponível em:
<https://doi.org/10.1109/SBCCI.2018.8533235>.

SANTOS, F. F. dos; BENEVENUTI, F.; RODRIGUES, G.; KASTENSMIDT, F.; RECH, P. Physical stress. In: DI NATALE, G.; GIZOPOULOS, D.; DI CARLO, S.; BOSIO, A.; CANAL, R. (Ed.). **Cross-Layer Reliability of Computing Systems**. London: IET, 2020. P. 157–174. ISBN 978-178561-797-3.

A.3 On softcore microprocessors, high-level synthesis and communication buses

Fault injection, radiation experiments and studies on soft core microprocessors and high-level synthesis as migration path for legacy and vanguard applications into FPGA; use of ARM AMBA AXI as core interconnection technology as seem on modern SRAM-based MPSoC and FPGAs from major COTS suppliers.

BENEVENUTI, F.; KASTENSMIDT, F. L. Analyzing AXI Streaming interface for hardware acceleration in AP-SoC under soft errors. In: VOROS, N.; HUEBNER, M.; KERAMIDAS, G.; GOEHRINGER, D.; ANTONOPOULOS, C.; DINIZ, P. C. (Ed.).

Applied Reconfigurable Computing. Architectures, Tools, and Applications. Cham: Springer, 2018. P. 243–254. ISBN 978-3-319-78890-6. Disponível em: <https://doi.org/10.1007/978-3-319-78890-6_20>.

BENEVENUTI, F.; KASTENSMIDT, F. L. Reliability evaluation on interfacing with AXI and AXI-S on Xilinx Zynq-7000 AP-SoC. In: 2018 IEEE 19th Latin-American Test Symposium (LATS). São Paulo, Brazil: IEEE, 2018. Disponível em: <https://doi.org/10.1109/LATW.2018.8347233>.

BENEVENUTI, F.; CHIELLE, E.; TONFAT, J.; TAMBARA, L.; KASTENSMIDT, F. L.; ZAFFARI, C. A.; MARTINS, J. B. dos S.; DURÃO, O. S. C. Experimental applications on SRAM-based FPGA for the NanosatC-BR2 scientific mission. In: 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). Rio de Janeiro, Brazil: IEEE, 2019. P. 140–146. Disponível em: <https://doi.org/10.1109/IPDPSW.2019.00032>.

BENITES, L. A. C.; BENEVENUTI, F.; OLIVEIRA, Á. B. de; KASTENSMIDT, F. L.; ADDED, N.; AGUIAR, V. A. P.; MEDINA, N. H.; GUAZZELLI, M. A. Reliability calculation with respect to functional failures induced by radiation in TMR Arm Cortex-M0 soft-core embedded into SRAM-based FPGA. **IEEE Transactions on Nuclear Science**, v. 66, n. 7, p. 1433–1440, 2019. Disponível em: <https://doi.org/10.1109/TNS.2019.2921796>.

BRAGA, G.; BENEVENUTI, F.; GONCALVES, M. M.; HERNANDEZ, H. G. M.; HUBNER, M.; BRANDALERO, M.; KASTENSMIDT, F.; AZAMBUJA, J. R. Evaluating softcore GPU in SRAM-based FPGA under radiation-induced effects. In: 2021 European Symposium on Reliability of Electron Devices, Failure Physics and Analysis (ESREF). Bordeaux, France: Elsevier, nov. 2021. v. 126. Disponível em: <https://doi.org/10.1016/j.microrel.2021.114348>.

GONCALVES, M. M.; BENEVENUTI, F.; MUNOZ, H.; BRANDALERO, M.; HUBNER, M.; KASTENSMIDT, F.; AZAMBUJA, J. R. Investigating floating-point implementations in a softcore GPU under radiation-induced faults. In: 2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS). Glasgow, UK: IEEE, 2020. Disponível em: <https://doi.org/10.1109/ICECS49266.2020.9294939>.

OLIVEIRA, A.; BENEVENUTI, F.; BENITES, L.; RODRIGUES, G.; KASTENSMIDT, F.; ADDED, N.; AGUIAR, V.; MEDINA, N.; GUAZZELLI, M.;

TAMBARA, L. Dynamic heavy ions SEE testing of NanoXplore radiation hardened SRAM-based FPGA: Reliability-performance analysis. **Microelectronics Reliability**, v. 100-101, p. 113437, 2019. 30th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis. ISSN 0026-2714. Disponível em: <https://doi.org/10.1016/j.microrel.2019.113437>.

OLIVEIRA, Á. B. de; BENEVENUTI, F.; BENITES, L. A. C.; RODRIGUES, G. S.; KASTENSMIDT, F. L.; ADDED, N.; AGUIAR, V. A. P.; MEDINA, N. H.; SILVEIRA, M. A. G.; DEBARGE, C. Analyzing the influence of using reconfiguration memory scrubber and hardware redundancy in a radiation hardened FPGA under heavy ions. In: 2018 18th European Conference on Radiation and Its Effects on Components and Systems (RADECS). Goteborg, Sweden: IEEE, 2018. Disponível em: <https://doi.org/10.1109/RADECS45761.2018.9328683>.

OLIVEIRA, Á. B. de; TAMBARA, L. A.; BENEVENUTI, F.; BENITES, L. A. C.; ADDED, N.; AGUIAR, V. A. P.; MEDINA, N. H.; SILVEIRA, M. A. G.; KASTENSMIDT, F. L. Evaluating soft core RISC-V processor in SRAM-based FPGA under radiation effects. **IEEE Transactions on Nuclear Science**, v. 67, n. 7, p. 1503–1510, 2020. Disponível em: <https://doi.org/10.1109/TNS.2020.2995729>.

RODRIGUES, G. S.; FONSECA, J.; BENEVENUTI, F.; KASTENSMIDT, F.; BOSIO, A. Exploiting approximate computing for low-cost fault tolerant architectures. In: PROCEEDINGS of the 32nd Symposium on Integrated Circuits and Systems Design. São Paulo, Brazil: ACM, 2019. (SBCCI '19). ISBN 9781450368445. Disponível em: <https://doi.org/10.1145/3338852.3339875>.

SANTOS, A. F. dos; TAMBARA, L. A.; BENEVENUTI, F.; TONFAT, J.; KASTENSMIDT, F. L. Applying TMR in hardware accelerators generated by High-Level Synthesis design flow for mitigating multiple bit upsets in SRAM-based FPGAs. In: WONG, S.; BECK, A. C.; BERTELS, K.; CARRO, L. (Ed.). **Applied Reconfigurable Computing: 13th International Symposium, ARC 2017, Delft, The Netherlands, April 3-7, 2017, Proceedings**. Cham: Springer, 2017. P. 202–213. ISBN 978-3-319-56258-2. Disponível em: <https://doi.org/10.1007/978-3-319-56258-2_18>.

A.4 On machine learning and embedded inference engines

Basic skills, implementations and radiation experiments on the case study domain.

BENEVENUTI, F. **Redes Neurais Convolucionais Embarcadas em FPGA**. Porto Alegre, Brazil: UFRGS, 2021. 23 Escola Sul de Microeletrônica (EMICRO 2021). Disponível em: <https://www.ufrgs.br/emicro/>.

BENEVENUTI, F.; LOPES, I.; KASTENSMIDT, F. L.; ADDED, N.; AGUIAR, V. P.; MEDINA, N. H.; GUAZZELLI, M.; POUGET, V.; ROED, K. Heavy ions testing of an all-convolutional neural network for image classification evolved by genetic algorithms and implemented on SRAM-based FPGA. In: 2019 European Conference on Radiation and Its Effects on Components and Systems (RADECS). Montpellier, France: IEEE, 2019. Disponível em: <https://doi.org/10.1109/RADECS47380.2019.9745650>.

BENEVENUTI, F.; GONCALVES, M. M.; PEREIRA JR, E. C. F.; VAZ, R. G.; GONÇALEZ, O. L.; AZAMBUJA, J. R.; KASTENSMIDT, F. L. Neutron-induced faults on CNN for aerial image classification on SRAM-based FPGA using softcore GPU and HLS. In: 2021 European Conference on Radiation and Its Effects on Components and Systems (RADECS). Vienna, Austria: IEEE, set. 2021. Disponível em: <https://doi.org/10.1109/RADECS53308.2021.9954517>.

BENEVENUTI, F.; KASTENSMIDT, F. L.; OLIVEIRA, Á. B. de; ADDED, N.; AGUIAR, V. Â. P. de; MEDINA, N. H.; GUAZZELLI, M. A. Robust Convolutional Neural Networks in SRAM-based FPGAs: a Case Study in Image Classification. **Journal of Integrated Circuits and Systems**, v. 16, n. 2, p. 504, 19 ago. 2021. Disponível em: <https://doi.org/10.29292/jics.v16i2.504>.

BENEVENUTI, F.; GONÇALVES, Marcio M.; JR., Evaldo Carlos F. Pereira; VAZ, Rafael G.; GONÇALEZ, Odair L.; BASTOS, Rodrigo Possamai; LETICHE, Manon; KASTENSMIDT, Fernanda L.; AZAMBUJA, José Rodrigo. Investigating the Reliability Impacts of Neutron-induced Soft Errors in Aerial Image Classification CNNs Implemented in a Softcore SRAM-based FPGA GPU. **Microelectronics Reliability**, Elsevier, Berlin, Germany, v. 138, p. 114738, 2022. 33rd European Symposium on Reliability of Electron Devices, Failure Physics and Analysis. ISSN 0026-2714. Disponível em: <https://doi.org/10.1016/j.microrel.2022.114738>.

LOPES, I. C.; BENEVENUTI, F.; KASTENSMIDT, F. L.; SUSIN, A. A.; RECH, P. Reliability analysis on case-study traffic sign convolutional neural network on APSoC. In: 2018 IEEE 19th Latin-American Test Symposium (LATS). São Paulo, Brazil: IEEE, 2018. Disponível em: <https://doi.org/10.1109/LATW.2018.8347234>.

TRINDADE, M. G.; BENEVENUTI, F.; LETICHE, M.; BEAUCOUR, J.; KASTENSMIDT, F.; BASTOS, R. P. Effects of thermal neutron radiation on a hardware-implemented machine learning algorithm. **Microelectronics Reliability**, v. 116, p. 114022, jan. 2021. ISSN 0026-2714. Disponível em: <https://doi.org/10.1016/j.microrel.2020.114022>.

APPENDIX B — STUDY-CASE APPLICATIONS IN IMAGE CLASSIFICATION

There are many safety critical or mission critical tasks of machine learning deserving attention regarding reliability and fault tolerance, ranging from implantable medical devices to smart industrial plants, from autonomous cars to unmanned spacecrafts. To use convolutional neural networks for image classification as a study case, the first question to answer is about the data that will be fed into the machine learning process to train the convolutional neural network.

As study-case for this thesis we consider image classification in the context of two application fields that are driving automation for on-road vehicles and earth observation.

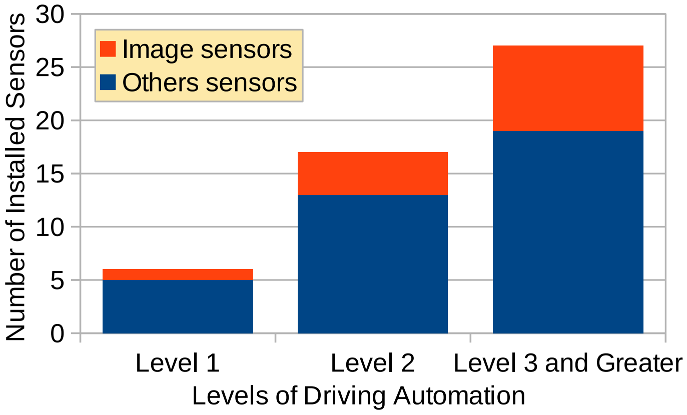
B.1 Driving automation and traffic sign classification

The SAE (2021) defines six levels, or categories, of driving automation ranging from no automation (Level 0) to full automation (Level 5). Level 1 (driver assistance) and Level 2 (partial automation) operates as driver support and requires the driver to perform some of the vehicle driving tasks while in Level 3 (conditional driving), Level 4 (high automation) and Level 5 (full automation) an automated driving system can perform all the vehicle driving tasks in a sustained basis while the level is engaged. In Level 3 the vehicle operates with a fallback-ready driver, in-vehicle or remote, that is expected to intervene and resume the driving while in the Level 4 and Level 5 the vehicle can execute fallback automatically, for instance by flashing hazard lights, pulling onto the road shoulder and calling roadside assistance.

Market analysis and forecast from Ogawa et al. (2017) suggests an increased number of image sensors following the implementation of higher automation levels, as presented in the chart of Figure B.1. These image sensors would be dedicated to the perception of the vehicle environment with functions like monitoring of the driver's condition and recognition of road lanes, recognition of vehicle surrounding for parking assistance, and recognition of road traffic including pedestrians and others vehicles. Ogawa et al. (2017) also consider future use of additional information such as weather and road surface conditions.

As part of the study case, we adopted the traffic sign recognition as a representative of image classification applied to automated driving systems. Traffic sign recognition could be used, for instance, in warning systems and automatic cruise control at the Level

Figure B.1 – Expected trend in electronic sensors count per car



Source: Adapted from Ogawa et al. (2017)

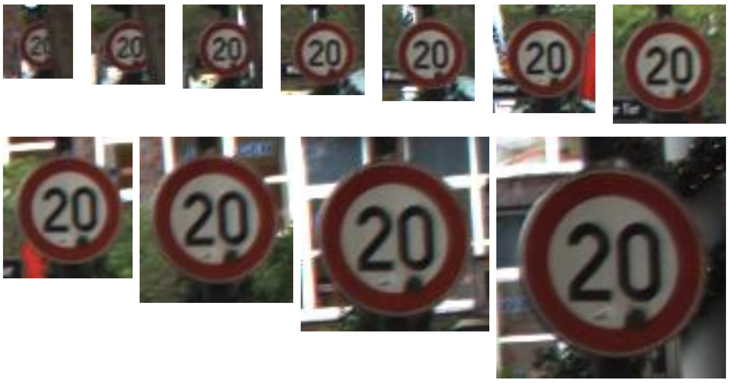
1 of driver assistance. Other example of image classification in automated driving is traffic light classification (CHARETTE et al., 2009; BEHRENDT et al., 2017). Many computer vision tasks in automated driving involves also detection of objects, not only traffic sign and traffic light, but pedestrians (NISHIDA et al., 2005; CHEN, Z. et al., 2019) and other vehicles, as well as an overall perception of the traffic environment (GEIGER et al., 2014; PAN et al., 2018).

Some publicly available datasets for traffic sign recognition of interest to this thesis include the aforementioned German Traffic Sign Recognition Benchmark (GTSRB) (STALLKAMP et al., 2012), as well as the Tsinghua-Tencent 100K dataset (TT100K) (ZHU et al., 2016) and the Brazilian Traffic Sign Database (BRTSD) (HOELSCHER, 2017). Each of these datasets have a different construction. Another dataset for traffic sign recognition is the European Traffic Sign Dataset (SERNA et al., 2018), composed of newly collected images merged with other datasets, including the GTSRB.

The GTSRB targets the task of traffic sign classification and consists of sequences of typically 30 images captured at different distances, consequently at different resolutions. GTSRB contains more than 10,000 images at the unlabeled test set and near 39,000 images at the labeled train set. However, as these 39,000 images consist of sequence of images with high similarity (Figure B.2), the variability of the dataset is limited with the number of sequences by traffic sign class ranging from 7 to 75 sequences in a total of only near 1300 different sequences. The GTSRB also has a companion dataset, the German Traffic Sign Detection Benchmark (GTSDB) (HOUBEN et al., 2013), dedicated to the task of detection of traffic sign in the scene.

The TT100K includes images from 100,000 different scenes and can be used both to traffic sign detection and classification. The accompanying metadata of TT100K have

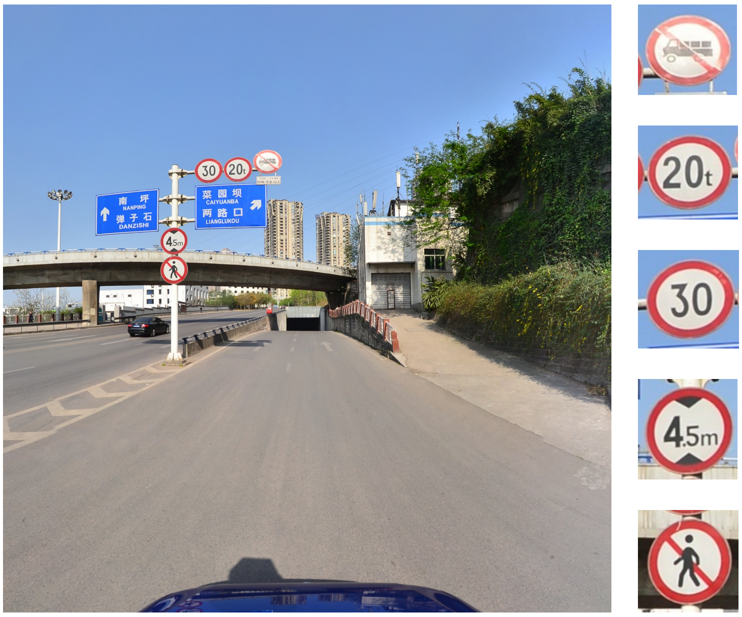
Figure B.2 – Fragment of a sequence from GTSRB



Source: Extracted from GTSRB dataset (STALLKAMP et al., 2012)

detailed position of traffic signs inside the scenes allowing the automated extraction of traffic signs region of interest when focusing specifically at the classification task (Figure B.3).

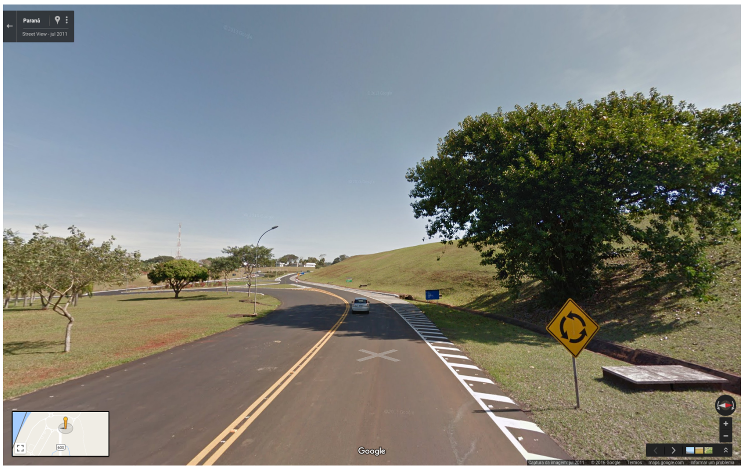
Figure B.3 – Example of a scene from TT100K and extracted traffic signs



Source: Extracted from TT100K dataset (ZHU et al., 2016)

Finally, the BRTSD is similar to TT100K in the sense that it presents traffic sign in the scene allowing both detection and classification (Figure B.4). However, it is limited to little more than 2,000 scene images. Some scene images from BRTSD are from sequence of images with the vehicle in different positions but these occurrences are rather uncommon. Both TT100K and BRTSD datasets were built reusing images already available for other purposes. BRTSD, specifically, consists of images captured from the Google Street View service that may not be unencumbered for generation of derivative works. Another disadvantage of BRTSD is the lack of metadata describing the position and class of each traffic sign at the scene.

Figure B.4 – Example of a scene from BRTSD



Source: Extracted from BRTSD dataset (HOELSCHER, 2017)

B.2 Earth observation and land use classification

Earth observation consists in monitoring and understanding our living planet using remote sensing and surveying technology, combining tools and methods for collection, storage and processing of geospatial data. It allows the assessment of the conditions of the planet, including its physical, chemical and biological systems, and the study of dynamic changes in its geophysical and ecological features.

Practical uses of Earth observation data include cartography, urban planning, traffic analysis, biomass estimation, definition of agricultural policies, support to sustainable development, monitoring of landslide hazard, hydrology studies, monitoring of flood propagation, monitoring of volcanic eruption, water and air quality monitoring, weather and climate monitoring, definition of policies and mitigations for climate change, natural disaster forecast, preparedness and recovery, wildfire detection, among many other uses.

Vehicles for Earth observation include satellites and airplanes, as well as drones, aerostats and, closer to the surface, tethered balloons. According to (MCCABE et al., 2017), start-up companies, with less than a decade of existence, operate more satellites in orbit than any space agency, and at costs that are a mere fraction of traditional satellite missions. 23% of more of more than 4,000 satellites on orbit reported by the nonprofit organization UCS (2021) are related to Earth observation. USA is the country of origin for more than half of the satellites, with Brazil counting with 13 satellites, among them the NanosatC-BR2 (BENEVENUTI et al., 2019b).

Miniaturization of artificial satellites decreased its complexity and cost, both by the use of commercial launch services and by the use of standard grade components with lower cost and unencumbered by restrictions seem on military grade components. However, the use of standard grade, commercially available off-the-shelf (COTS), components on those low cost satellites requires both qualification of the devices to grade its inherent susceptibility to radiation effects (WINOKUR et al., 1999) and implementation of mitigations for fault tolerance where required to satisfy the mission requirements.

There is no standardized nomenclature but small artificial satellites can range from femtosats to minisats. An example of classification given by Laufer et al. (2020) is summarized in Table B.1. The FAA AST (2018) uses a slightly different classification.

Table B.1 – Example of small satellite classes

Class	***Mass***
Femtosatellite (femtosat)	up to $100\,\mathrm{g}$
Picosatellite (picosat)	$100\,\mathrm{g}$ to $1\,\mathrm{kg}$
Nanosatellite (nanosat)	$1\,\mathrm{kg}$ to $10\,\mathrm{kg}$
Microsatellite (microsat)	$10\,\mathrm{kg}$ to $100\,\mathrm{kg}$
Minisatellite (minisat)	$100\,\mathrm{kg}$ to $500\,\mathrm{kg}$

Source: Laufer et al. (2020)

Among the small satellites, the CubeSat class enjoys great popularity. The CubeSat class is more than a weight classification as it include also form factor, constructive properties and interoperability aspects (CAL POLY, 2015; HEIDT et al., 2000). For instance, a one unit (1U) CubeSat can be at the picosat class while a larger CubeSat can be at the nanosat class. In fleets of small satellites, for example in a family formation (FARRAG et al., 2021), the use of FPGA in the implementation of integrated image processing, on-board data-handling (NANNIPIERI et al., 2021), software defined radio (SDR) (CAI, X. et al., 2018), and software defined network (SDN) (KAK et al., 2018), allows the mobility of roles among the fleet members increasing opportunities for fault tolerance and soft degradation during the mission lifetime.

In recent years the interest for small satellites increased both in academic research (SWARTWOUT, 2004) and professional use (MADRY et al., 2020). The interest includes also imaging applications (S. CURIEL et al., 2005; GRAZIANI et al., 2008; BORGEAUD et al., 2010; SELVA et al., 2012; CRYSTALSPACE, 2016; BRZOZOWSKI et al., 2018; CHO, D.-H. et al., 2019; GARRANZO et al., 2019; NAGEL et al., 2020), with the notable example of the commercial PlanetScope constellation with over 100

CubeSat 3U imaging satellites (PLANET LABS, 2018; SADEH et al., 2021).

Imaging sensors for Earth observation include panchromatic sensors, red-green-blue (RGB), near-infrared (NIR), multi- and hyperspectral sensors, multi-band thermal, multi-channel microwave, as as well as radar (radio detection and ranging) and lidar (light detection and ranging) sensors (MCCABE et al., 2017).

With the increasing volume of images, captured with low cost platforms such as small satellites or stored in publicly available databases, Earth observation also entered the field of big data. Of interest to this thesis are the earth observation activities executed at the aerospace segment, more specifically computer vision and image processing executed onboard aircrafts and satellites that may be subject to a higher flux of atmospheric and cosmic radiation rising reliability issues that can compromise its mission.

Scene classification plays an important role in earth observation by labeling a remote sensing image according to the semantic classes of natural features or land use, such as water body or parking lots. Other uses of scene classification includes pervasive surveillance with civilian and defense applications (ANDREOU et al., 2016).

In the context of aircrafts and spacecrafts with constrained resources, either in terms of storage capacity, energy budget or downlink communication bandwidth, integrated preprocessing, target tracking and scene classification, for instance to assess image quality of cloud coverage, can support management functions in decisions regarding image preservation and the scheduling of attitude, image acquisition and data transmission (BENSANA et al., 1999; CHIEN et al., 2014; CHO, D.-. et al., 2018; CHEN, S.-J. et al., 2019).

Yuhaniz et al. (2005) describes image processing onboard small satellites using traditional computer vision and summarize image processing functionality present on different satellites. El-Araby et al. (2009) e Chien et al. (2014) consider the use of image processing onboard satellites also using traditional algorithms of computer vision, El-Araby et al. (2009), more specifically, using FPGAs. Dawood et al. (2002) also describes low level image processing onboard satellites using FPGAs. Marais et al. (2008) describe image processing algorithms for image quality evaluation onboard small satellites. Popowicz (2016) discusses image classification of stellar objects onboard nanosatellites including defect detection and rejection of malformed images due to interference of proton radiation on the image sensor.

Ieremeiev et al. (2020) also deals with image quality evaluation, describes different metrics, and then proposes the use of a fully-connected neural network to emit a quality score. In that work, however, the goal is to process images on the ground.

Velazco et al. (1999) analyzes a space-borne fully-connected neural network trained to classify 24×24 pixels single channel grayscale satellite images in four texture classes of ‘industrial area’, ‘residential area’, ‘scrubland’ and ‘sea’. Using CNNs, Arechiga et al. (2018) describes image processing and object detection onboard satellite using CNN implemented on a NVIDIA TX2 SoC, an integrated circuit designed for AI computing on the edge, combining GPU and multicore CPU. Manning et al. (2018) proposes the use of CNN implemented on a SRAM-based SoC/FPGA for image classification onboard satellite, in five classes of ‘black’, ‘white’, ‘distorted’, ‘cloud or water’ and ‘land’.

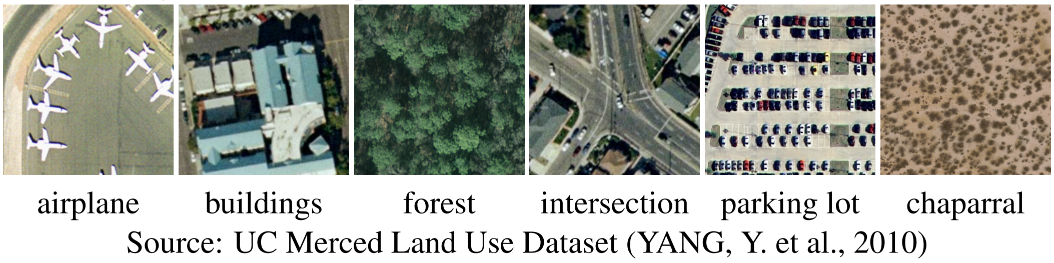
Also dealing with satellite image classification, but at the ground, Basu et al. (2015) e Liu et al. (2019) used CNN for image classification and compares with other methods of image classification. Minetto et al. (2019) e Gao et al. (2021) also used CNN in discriminative labeling of regions inside complex scenes in remote sensing images.

In many of those works we observe the use of commercial or publicly available satellite imagery datasets to train or validate the image processing.

Here we will adopt the task of land use classification as representative of image classification in mission critical systems for Earth observation and aerial surveillance. For this task, some candidate publicly available labeled datasets include UC Merced Land Use Dataset (YANG, Y. et al., 2010), Functional Map of the World (fMoW) Challenge (CHRISTIE et al., 2018), SAT-4 and SAT-6 (BASU et al., 2015).

A dataset well known in the literature, the UC Merced Land Use Dataset consists of a small number of images, only 2100, classified in 21 classes with 100 images per class. The images were extracted manually from the U.S. Geological Survey’s (USGS) National Map Urban Area Imagery collection and are available in 256×256 pixels with visible RGB color and a resolution of approximately 30 cm. A few samples of the dataset is presented in Figure B.5. The small number of images in this dataset is a disadvantage for its use in data intensive machine learning strategies such as CNNs.

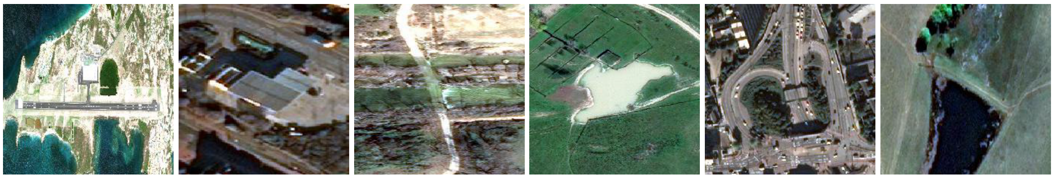
Figure B.5 – Examples from UC Merced Land Use Dataset in RGB



Source: UC Merced Land Use Dataset (YANG, Y. et al., 2010)

The dataset for the Functional Map of the World (fMoW) Challenge (CHRISTIE et al., 2018) consists of geospatial image scenes with multiple labeled regions of interest, using 63 different classes. The fMoW dataset counts with over 470,000 images available in multi-spectral 4-band and 8-band in the visible to near-infrared. Illustrative samples of the dataset are presented in Figure B.6 in visible RGB.

Figure B.6 – Examples from fMoW in RGB

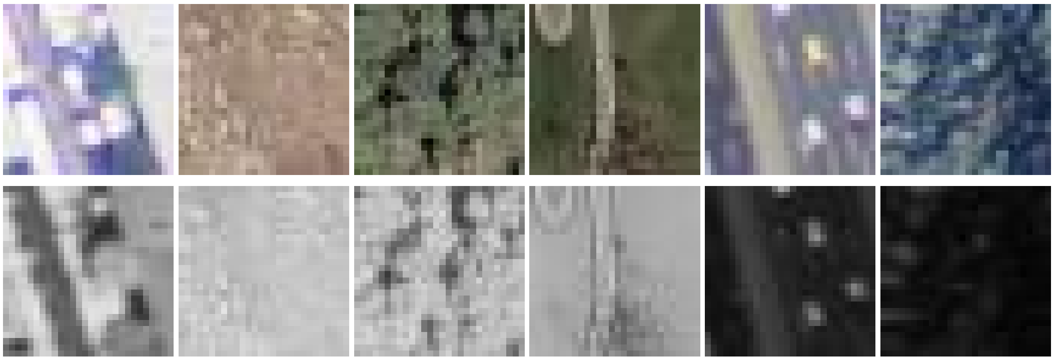


airport gas station flooded road lake or pond interchange dam

Source: Extracted from Christie et al. (2018)

The SAT-4 and SAT-6 datasets also counts with a large number of images, 500,000 and 405,000, respectively. Both datasets were extracted from aerial images of the U.S. Department of Agriculture (USDA) National Agriculture Imagery Program (NAIP) acquired with a ground resolution of $1\,\mathrm{m}$. The dataset images are available in 28×28 pixels with four channel, visible RGB color and near-infrared (NIR). One main difference between these two datasets is that SAT-4 images are labeled in four classes of ‘barren land’, ‘trees’, ‘grassland’ and ‘others’, while SAT-6 images are labeled in six classes of ‘barren land’, ‘trees’, ‘grassland’, ‘roads’, ‘buildings’ and ‘water bodies’. Figure B.7 presents samples of these classes.

Figure B.7 – Examples from SAT-6 dataset in RGB (top) and NIR (bottom)



building barren land trees grassland road water

Source: SAT-6 dataset (BASU et al., 2015)

B.3 Neural networks

Bio-inspired algorithms intent to emulate nature in order to solve real-life complex problems of classification, approximation, pattern recognition, identification and control, in different fields such as engineering, economics and social sciences (ALANIS et al., 2018).

The origins of neural networks mingle with the origins of digital computers going from Rashevsky (1933) to McCulloch et al. (1943) and then to von Neumann (1993).

Being originally inspired by biology, along the decades the most popular implementations of neural networks have been under a continuous process of approximating and then loosing biological plausibility. For instance, on the first model from Rashevsky (1933) the neurons had both excitatory and inhibitory aspects represented explicitly and on the model of Rosenblatt (1958) the multi-layer neuronal tissue was conceived with layers of sensory, association the recognition cells invoking synaptic plasticity (HEBB, 1949) in the determination of the neuron weights. However, the perceptron (ROSENBLATT, 1958), the sigmoidal neuron (RUMELHART et al., 1986) and the convolution with learned shared weights (LE CUN et al., 1989) are at the same time oversimplifications and overextrapolations of the biological neuron and the neuronal tissue for it does not capture the spiking nature of biological neurons and the connections on human brain cannot grow as deep as some popular deep neural networks (MAASS, 1997; SAMU et al., 2014; PARK et al., 2017).

It was with Rumelhart et al. (1986) that neural networks attained a huge momentum with the introduction of backpropagation algorithm and the sigmoidal activation function. Little after, Irie et al. (1988), Cybenko (1989), Hornik et al. (1989) e Blum et al. (1991) have shown that the feed-forward neural network is a universal approximator, meaning that it can uniformly approximate any real continuous nonlinear function to arbitrary degree of accuracy. Another significant improvement on neural networks came with Le Cun et al. (1989) with the use of learned shared weights that paved the road to the convolutional neural networks in image classification. Later, the introduction of the rectified linear unit (ReLU) activation function (GLOROT et al., 2011) reduced significantly the computational effort compared to the use of exponential operator at the sigmoidal activation function.

For some time, the concept of deep learning blended with the concept of deep neural networks since they were introduced in around the 2000's with applications in different fields and unprecedented performance on big data. This led to the impression that the neural network alone could solve the machine learning problem and that deep neural networks were always required to solve complex problems (BENGIO, 2009; BENGIO et al., 2021).

Engineering for execution environments with constrained resources and the need for optimizations conducted back to solutions of complex problems using shallow neural networks or multi-stage systems, even if, in some sense, suboptimal, but compliant with the requirements of the task at hand (MCDONNELL et al., 2015; LOPEZ-MARTIN et al., 2019; GIM et al., 2020; PARK et al., 2021). Also, there is no definitive rule to determine when a neural network is deep or shallow.

B.4 Image classification with neural networks

Computer vision is the process of extracting and interpreting information from images of the three-dimensional world, including physical, geometric, or topological properties of the objects that appear in the images (VELHO et al., 2009).

This traditional approach to computer vision is quickly challenged as we increase the number of objects of interest because every object class may require specific filters, algorithms or property descriptors to make it distinguishable from other objects.

To cope with this increasing complexity, we can take advantage of deep learning and convolutional neural networks to reduce of human work of devising the medium- and low-level computer vision processing by allowing the neural network to learn those steps and blend it seamlessly in the computer vision task.

This was possible due to advances in computing power available at low cost and also due to the big data phenomena that provided large data sets of labeled images with the objects of interest observed from different points of view and in different lighting conditions that can be used to learn the semantic information directly from images without the need of intermediary representations of the object geometry or topology.

Convolutional neural networks can solve more complex tasks of image classification, for instance the challenges and competitions such as GTSRB (STALLKAMP et al., 2012) with images labeled in 43 classes, CIFAR-100 (KRIZHEVSKY, 2009) where images are labeled in 100 classes and superclasses, and ILSVRC (RUSSAKOVSKY et al., 2015) with images labeled in 1000 classes.

The main characteristic of the convolutional neural networks is that the convolutional layers behave similar to the kernel filters used in image processing such as noise removal and morphological operations. This is accomplished by using the same weights on some neurons. Also, these neurons are not connected to all inputs or neurons from previous layers and all neurons on the next layer, as occurred with with the fully connected perceptron neural network. Different from the coefficients of convolution kernel filters that are determined by the designer, the shared weights of neurons at convolutional layers are learned in the same way as the weights on any other neural layer. In some sense, this opens the convolutional layers to also learn the best convolution kernel filters to the task at hand.

In deep learning, the convolutional layers are usually associated with feature extraction, equivalent to the preprocessing, segmentation and description processes in traditional computer vision, and its output is multidimensional data usually named as feature map. The feature maps can use a matrix structure similar to the input image, with a number of rows, columns and channels, except that in images the channels are usually associated with colors while on feature maps they carry arbitrary data.

Layers of fully connected neurons can be placed after the convolutional layers to do the final classification step using the data available on the feature map, what can be roughly associated with the recognition process.

The dense, fully-connected layers of neurons, bring many scalability challenges to the implementation of neural networks at embedded devices, notably FPGAs. As each neuron on fully-connected layers is unique, if each neuron is instantiated explicitly on hardware, parameterized by its weights and bias, they may occupy a large area, possibly making the whole neural network unfeasible on the targeted device. Alternatively, if a small number of generic neurons are instantiated a large memory with high-throughput may be required to store weights and bias that will be loaded into the reusable neuron module at run-time.

To cope with these challenges while implementing convolutional neural networks in FPGAs, some researchers, such as Lopes (2017), opted to divide the problem implementing the convolutional layers as multicycle hardware modules, exploiting the fact that common weights are shared at the convolutional layer, and implementing the fully-connected layers as software executing in general purpose microprocessors.

While working with CNNs for image classification, instead of adding fully-connected layers as classifiers, Lin et al. (2014) proposed to average the feature map obtained from convolutional layer to extract a classification result. With the use of global average pooling each channel in the output feature map could be directly associated with an output category, being more natural to the structure of the CNN and allowing the reinterpretation of the feature map at the output layer as a categories confidence map. Other advantage indicated by Lin et al. (2014) would be a higher robustness to spatial translations of the object in the input image.

Following the works from Lin et al. (2014), Springenberg et al. (2015) demonstrated the use of CNNs for image classification with other optimizations such as the use of small convolutional kernels to reduce the number of network weights and bias, and the combination of intermediate pooling layers between convolutional layers, originally designed for dimension reduction, into convolutional layers with stride greater than one.

A consequence, of special interest to this thesis, is a more regular or uniform structure in the CNN as all layers can now be of the same convolutional nature with shared weights, making it an *all-convolutional* neural network.

Despite the engineering advantages of lower complexity and smaller data footprint while achieving competitive or state of the art accuracy performance, this approach, unfortunately, takes the CNN a step farther from biological plausibility. Yet, all-convolutional neural networks have been adopted by researchers targeting embedded implementations, such as Iandola et al. (2016) and Gschwend (2016).

B.5 Convolutional neural networks on FPGA

There are many possible strategies for implementation of CNN in FPGAs. The first one would be coding the functional modules of the CNN inference engine, such as neurons and multiplexers, in a hardware description language (HDL), such as VHDL or Verilog. This approach has the advantage of allowing meticulous optimizations on the CNN design and can achieve faster processing times or lower resources usage. Libano et al. (2018) adopted this approach in the optimization of the sigmoidal activation function for fully-connected MLP where the neural network was assembled by manually coding the parameterized instantiation of the NN building blocks. In the case of complex CNNs, however, the manual instantiation of the NN modules becomes repetitive and error prone.

Another strategy for implementation of CNNs takes the basic NN parameterizable building blocks coded in HDL by a human specialist but automates the repetitive task of instantiation of these blocks following a machine readable description of the CNN structure and coefficients obtained at the NN training stage. This approach was adopted by Abdelouahab et al. (2017), Ma et al. (2017), Lopes (2017) e Morcel et al. (2019).

Up to this point the design on the FPGA was seen as a complete realization of the CNN or at least of its convolutional layers. A different approach is to implement on the FPGA only a general purpose and reusable convolution accelerator that can be invoked from a host microprocessor. Is this case, the CNN exists only conceptually and is realized at run time while the host microprocessor coordinates the configuration and dispatch of the reusable convolution accelerator for processing of each layer of the CNN.

In another dimension we can consider the level of specification of the CNN components. While specification in hardware description language is an alternative for FPGA, there are other options. At a higher level of specification one can consider to implement the CNN processing simply as software and then use the FPGA resources to implement a microprocessor, with a general purpose or a specialized instruction set architecture (ISA), that can run the CNN software at the targeted performance level. Examples in this case include and Hernandez et al. (2021) that uses a softcore general-purpose GPU as CNN accelerator. This approach has the advantage of direct port of newer CNN techniques and algorithms to the FPGA at a lower cost simply by compiling the well-proven software source code to execute and the microprocessor on the FPGA. Although an easier path for migration of the CNN into an FPGA, the software approach hardly achieves higher levels of performance.

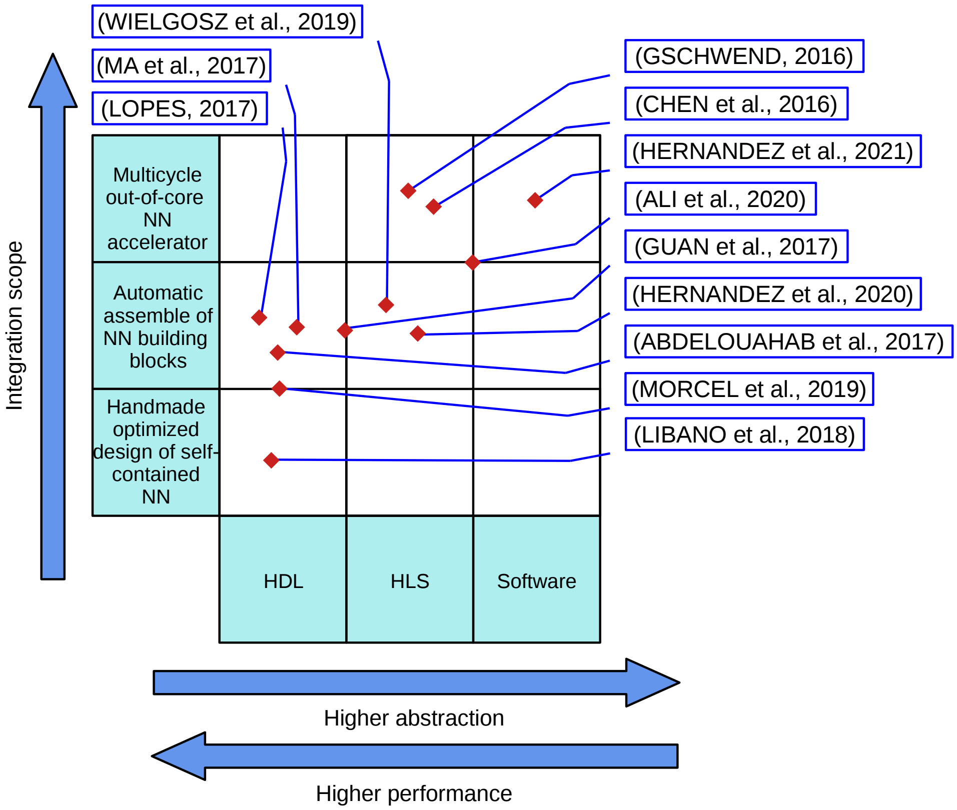
As a balance between high performance and easier port of CNN algorithms we can exploit high-level synthesis (HLS). In this case, the software source code is used as a specification of the CNN processing but, instead of compiled into machine code to be run in a microprocessor, the software is synthesized into circuit logic and becomes a dedicated hardware block.

HLS can be used by a human specialist to code manually an entire self-contained CNN, can be used to create base NN building blocks that will be assembled into a CNN by an automated tool that instantiates those blocks following the description of a trained CNN, as in Wielgosz et al. (2019) e Hernandez et al. (2020), or can be used to build a reusable convolution accelerator that will be invoked by a host processor, such as in C. Zhang et al. (2016), where all CNN processing is transformed into a canonical matrix multiplication form that is then implemented with optimization into the FPGA, and Gschwend (2016), where a dedicated convolution accelerator is optimized for efficiency of data movement between the accelerator local storage and the main memory shared with the host processor.

These different levels of abstraction and integration scope granularity are depicted in Figure B.8 summarizing some NN implementations reported in the literature or available as open source.

For comparison, an all-convolutional CNN designed for the image classification

Figure B.8 – Strategies for implementation of NN on FPGA

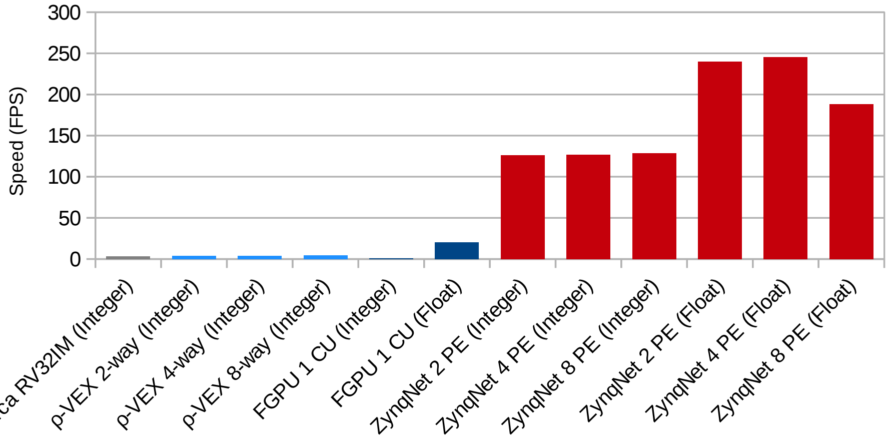


Source: The Author

task in the SAT-6 dataset, presented in Section B.2, was implemented in FPGA using software executing on softcore microprocessors and using hardware generated automatically by HLS. The CNN consists of only three convolutional layers, parameterized by 4,072 coefficients for weights and bias, and requires 368,640 multiply-accumulate (MACC) operations. The CNN software coded in C language using only fixed-point arithmetics was executed in two different microprocessors, the Orca (VECTORBLOX, 2019) 32 bits RISC-V processor, the ρ-VEX VLIW processor (WONG et al., 2008) configured for 2-way, 4-way and 8-way issues. The same CNN coded in OpenCL language was executed in the FGPU (AL KADI et al., 2018) general purpose GPU in two configurations for using floating-point arithmetics and fixed-point arithmetics, both using one compute unit (CU) with a compute vector (CV) of eight processing elements (PE). Finally, the CNN was executed in the ZynqNet CNN accelerator (GSCHWEND, 2016), also in two configurations for floating-point arithmetics and fixed-point arithmetics but with different numbers of two, four and eight processing elements (PE).

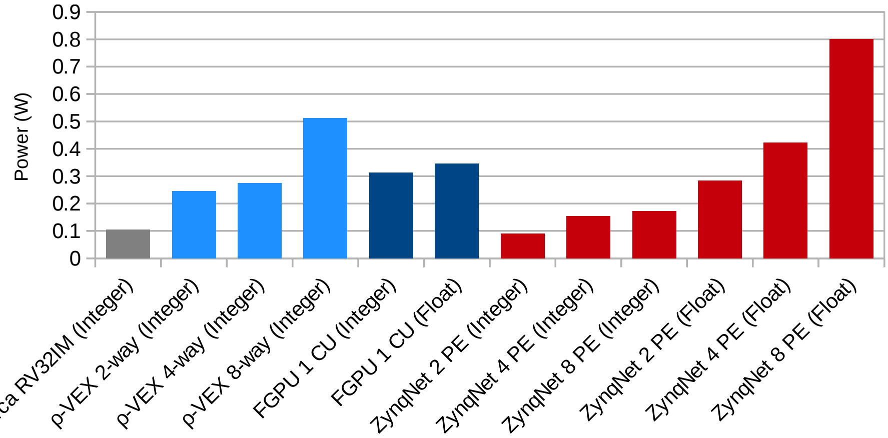
The chart at Figure B.9 presents the processing speed for each of these implementations, given in number of image frames per second where we can observe that the ZynqNet HLS approach is notably faster. The chart at Figure B.10 shows the power estimated by the Xilinx Vivado synthesis tool. Although ZynqNet HLS has a higher power requirement, when we consider its significantly smaller processing time ZynqNet has the lowest total energy consumption per image among all these implementations as seem in the chart of Figure B.11.

Figure B.9 – Processing speed for a SAT-6 three layers CNN



Source: The Author

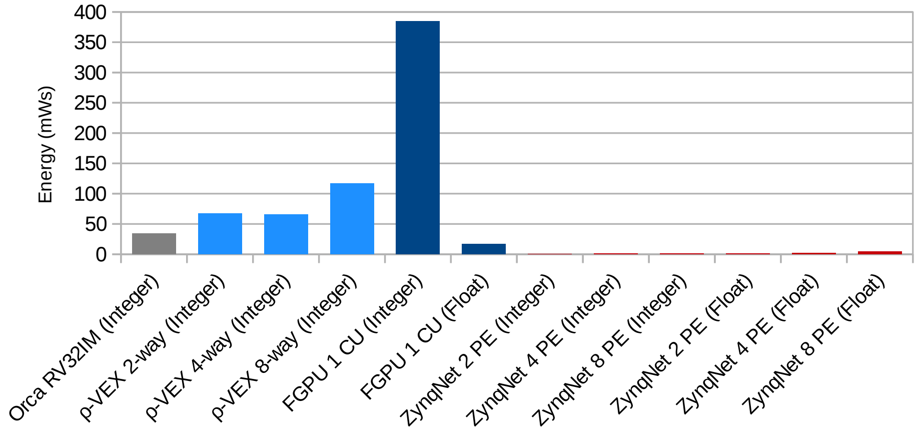
Figure B.10 – Required power for a SAT-6 three layers CNN



Source: The Author

Following these results, the ZynqNet CNN accelerator is adopted as the study case CNN inference engine for this thesis. The choice for ZynqNet resides in the fact that it was designed as an out-of-core accelerator, it scales well with shallow and deep CNNs even in small FPGA devices because is does not map the entire CNN topology directly on the FPGA, and it has fast processing time, low energy consumption and small footprint on FPGA resources. ZynqNet uses intensively the interface to the DDR memory shared with the host processor as all the CNN coefficients for weights and bias are read from DDR memory. In ZynqNet, the DDR memory is also the source for the input image to be classified and the input and output feature maps for each CNN layer.

Figure B.11 – Energy consumption by image for a SAT-6 three layers CNN



Source: The Author

Several technical characteristics of the ZynqNet inference engine must be taken into account to build a well-formed CNN compatible with that engine. These characteristics are not necessarily limitations of the inference engine but, on the contrary, optimizations for FPGA implementation devised by Iandola et al. (2016) and Gschwend (2016). The characteristics include:

- The convolution kernel size must be 1×1 or 3×3, with stride 1 or 2.
- The activation function is ReLU.
- The number of output channels in the convolutional layer must be multiple of the number of processing elements (PE).
- The maximum number of output channels in a feature map is 512 and up to 1024 input channels can be obtained by concatenation layers.
- The feature map is square, width equals height, with width and height being a power of 2.
- Pooling is allowed only in the last layer, and only global pooling is supported.

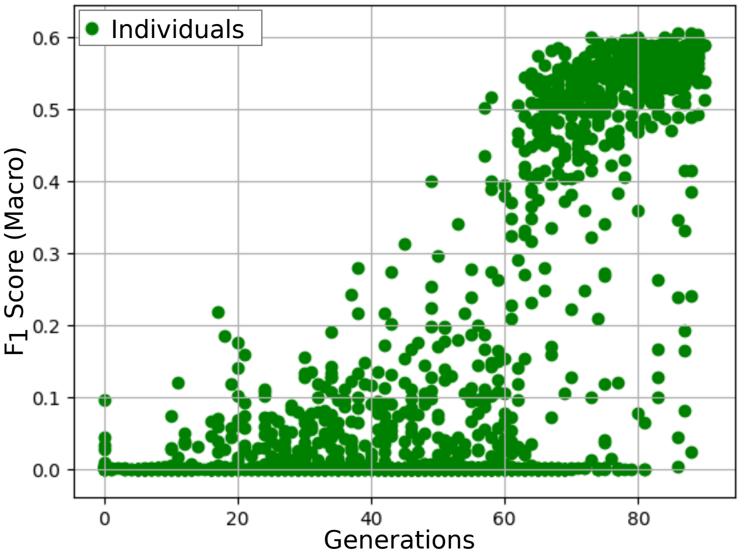
B.6 Prototype CNN for GTSRB traffic sign classification task

This CNN was designed to work with the GTSRB dataset presented on Section B.1. The traffic sign classification was the fist image classification task approached due to the familiarity with the dataset and task requirements acquired while working with another inference engine (LOPES et al., 2018).

This task served and testbed during the implementation of a prototype evolutionary flow and four main versions of the CNN were obtained along many runs of the evolutionary process.

The evolutionary process initially targeted the ZynqNet engine operating only with floating-point numbers. Figure B.12 presents the evolution of the accuracy in the CNN population. In early generations most individuals performed poorly by the best individuals were selected the new individuals created showed better performance. Again is must be noted that the evolutionary process is not about improving the individual but improving the population as a whole. Notwithstanding, an the end of the process the best of the best individuals can still be selected to be implemented in the FPGA.

Figure B.12 – CNN accuracy improvement over generations



Source: The Author

Table B.2 summarizes comparative information about the 5 best individuals on the population of CNNs for GTSRB after the first successful execution of the evolutionary flow producing a CNN synthesizable in the FPGA. The best individual, specimen `ace8288cd3a6eda2509a160d9de71a28` (Fig. B.13), have a Top_1 Accuracy of 81% and a Top_5 Accuracy of 93%. The input to this CNN is a grayscale image scaled to 32×32 pixels and preprocessed with adaptive histogram equalization (CLAHE). This CNN has 7 convolutional layers requiring 1,618,432 multiply-accumulate (MACC) operations in the inference stage.

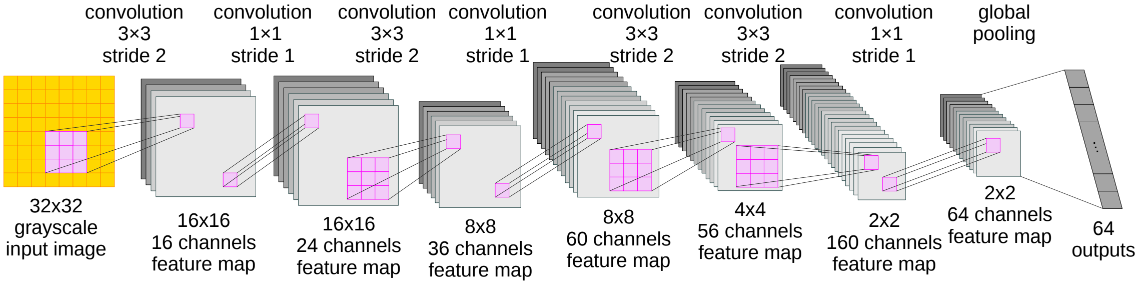
This evolutionary flow for GTSRB was driven exclusively by the accuracy metrics. More specifically, the fitness of each individual k in the population is the individual macro F_1 Score as in Eq. B.2. The F_1 Score metric was found the be highly correlated to the Top_1 Accuracy but, by fusing both precision and recall, it is expect to be safer in preventing

Table B.2 – Best fitted individuals in the GTSRB floating-point CNN population

Individual Identity Hash	*Layers*	*Input Image Format*	*Input Image Processing*	*CNN Training Iterations*	*Inference MACC Operations*	*F_1 Score*
`...1a28`	7	32×32 grayscale	CLAHE	2,000	1,618,432	0.536
`...049c`	8	32×32 RGB	CLAHE Y (YUV)	2,000	17,743,360	0.525
`...75ae`	8	64×64 grayscale	CLAHE	2,000	5,553,920	0.522
`...7090`	7	64×64 grayscale	CLAHE	2,000	4,878,336	0.516
`...0a36`	7	64×64 grayscale	CLAHE	2,000	6,332,672	0.516

Source: The Author

Figure B.13 – Structure of a CNN evolved for the GTSRB task


convolution 3×3 stride 2
convolution 1×1 stride 1
convolution 3×3 stride 2
convolution 1×1 stride 1
convolution 3×3 stride 2
convolution 3×3 stride 2
convolution 1×1 stride 1
global pooling
32x32 grayscale input image
16x16 16 channels feature map
16x16 24 channels feature map
8x8 36 channels feature map
8x8 60 channels feature map
4x4 56 channels feature map
2x2 160 channels feature map
2x2 64 channels feature map
64 outputs

Source: The Author

overfitting.

$$
\begin{aligned}
Fitness(k) &= F_1(k) && \text{(B.1)}\\
F_1(k) &= \frac{2}{Precision(k)^{-1} + Recall(k)^{-1}} && \text{(B.2)}
\end{aligned}
$$

Exploring the Ristretto feature, the CNN specimen `...1a28` was also fine-tuned for different arithmetic representations for fixed-point and floating-point. Based on the findings at the fine-tune process, the evolutionary flow was also modified to use Ristretto and different arithmetic representations were introduced as part of the solution space to be explored.

Other experiments in this task of traffic sign classification included improved balancing the the GTSRB dataset and the augmentation of this dataset using traffic signs from TT100K (Section B.1).

B.7 Prototype CNN for SAT-6 land cover classification task

This CNN was designed to work with the SAT-6 dataset presented on Section B.2. The selection of this image classification was motivated by the work of Velazco et al. (1999) and by its potential use embedded in small satellites in the NanosatC-BR program (Appendix C).

In this execution of the evolutionary process the metrics for classification accuracy and computational effort contributed to ranking each individual at the population. The fitness of each individual k in the population is computed according to Eq. B.3 where $PE(k)$ is the number of ZynqNet processing elements of the individual, $MACC(k)$ is the number of multiply-accumulate operations for the CNN inference, κ_{ac} is a smoothing factor for accuracy and κ_{ef} is a smoothing factor computational effort.

$$\begin{aligned}
Fitness(k) &= \exp\left[-\left(\frac{F_1(k)}{F_{1,max}}\right)^{\kappa_{ac}}\right] \times \exp\left[-\left(\frac{MACC_{PE}(k)}{MACC_{PE,max}}\right)^{\kappa_{ef}}\right] && \text{(B.3)}\\
F_{1,max} &= \max_{i \in population} F_1(i) && \text{(B.4)}\\
MACC_{PE,max} &= \max_{i \in population} MACC_{PE}(i) && \text{(B.5)}\\
MACC_{PE}(k) &= \frac{MACC(k)}{PE(k)} && \text{(B.6)}
\end{aligned}$$

The evolutionary process for this task was repeated twice from scratch, with CNN processing using floating-point arithmetic native to the Caffe framework and using fixed-point arithmetic with the help of the Ristretto feature.

In the first execution of the process, aiming at CNN processing with floating-point arithmetic, the SAT-6 dataset was used with minor modifications to convert the original 28×28 pixels image to 32×32 pixels to satisfy ZynqNet engine requirements.

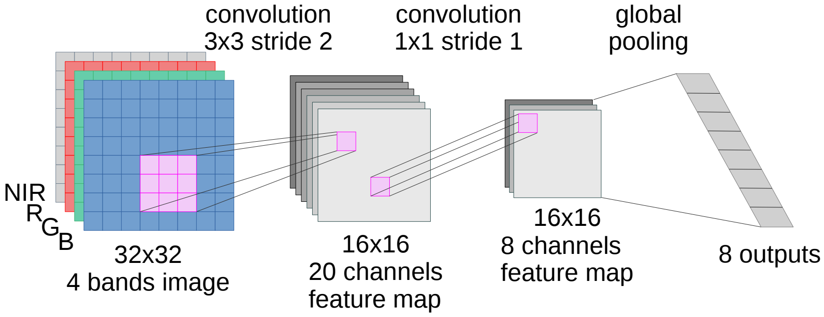
After a few generations of the first execution of the evolutionary process, a CNN candidate reached 97.6% accuracy requiring 7.7×10^6 MACC operations for inference. However, another individual of the population was chosen as the top-best, having a slightly lower accuracy of 97.3% but requiring only 2.3×10^5 MACC operations. Table B.3 summarizes the 5 best individuals on the population of CNNs for SAT-6 task operating in floating-point arithmetics. The best CNN specimen `a7a04d42e59a7942972714fd-61c8090b` has only two convolutional layers, as depicted in Fig B.14, described by a set of 908 weights and bias coefficients. The last CNN layer has eight output elements, satisfying ZynqNet engine requirements, of which six are used to indicate the image classes of the SAT-6 dataset.

Table B.3 – Best fitted individuals in the SAT-6 floating-point CNN population

Individual Identity Hash	*Layers*	*Input Image Format*	*Input Image Processing*	*CNN Training Iterations*	*Inference MACC Operations*	*F_1 Score*
`...090b`	2	32×32 RGB+NIR	None	74,223	225,280	0.954
`...6163`	2	32×32 RGB+NIR	None	7,596	7,659,520	0.967
`...6e11`	3	32×32 RGB+NIR	None	7,596	22,560,768	0.950
`...126d`	2	32×32 RGB+NIR	None	7,596	7,929,856	0.793
`...9940`	2	32×32 RGB+NIR	None	7,596	8,876,032	0.791

Source: The Author

Figure B.14 – Structure of a CNN evolved for the SAT-6 task in floating-point



Source: The Author

For the second execution of the process, two additional modifications were applied to the images of the SAT-6 dataset: color space conversion from red, green, and blue (RGB) channels to hue, saturation, and value (HSV) channels and contrast enhancement with adaptive histogram equalization (CLAHE).

Also, a fractional fixed-point numeric representation (Q$m.n$) was adopted for input images, feature maps, weights, and bias coefficients. To simplify the fixed-point multiplication, the input images and feature maps were chosen only to have the signal and integer parts (Q7.0), while weights and bias coefficients were chosen only to have the signal and fractional parts (Q0.7). However, the result of the multiplication, as well as accumulators inside the convolution, were implemented with both the integer and fractional parts (Q7.8), discarding the fractional part by rounding after the activation function (ReLU). Consequently, inputs and outputs of each convolutional layer fit into an eight-bit signed integer (INT8).

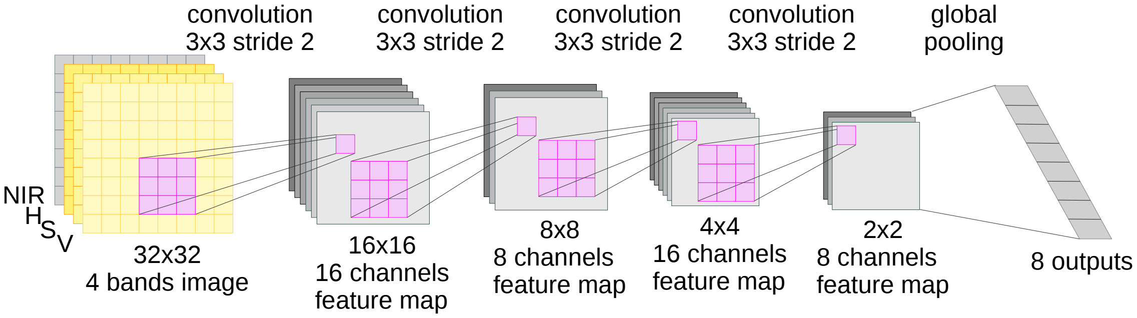
At this second execution, the evolutionary process reached 98.5% accuracy requiring 2.4×10^5 MACC operations for inference with 4,080 weights and bias coefficients. Table B.4 summarizes the 5 best individuals on the population of CNNs for SAT-6 task operating in fixed-point arithmetics. The best CNN specimen `80550ebda901f577df-ea7a1147dd28be` featuring four convolutional layers is depicted in Fig B.15, again with eight elements at the output CNN layer, of which six are used to indicate the image classes of the SAT-6 dataset.

Table B.4 – Best fitted individuals in the SAT-6 fixed-point CNN population

Individual Identity Hash	*Layers*	*Input Image Format*	*Input Image Processing*	*CNN Training Iterations*	*Inference MACC Operations*	*F_1 Score*
`...28be`	4	32×32 HSV+NIR	CLAHE V (HSV),NIR	75,960	244,224	0.976
`...9c8d`	5	32×32 HSV+NIR	CLAHE V (HSV),NIR	69,630	304,128	0.976
`...eaf9`	3	32×32 HSV+NIR	CLAHE V (HSV),NIR	82,290	230,400	0.975
`...8805`	5	32×32 HSV+NIR	CLAHE V (HSV),NIR	75,960	253,440	0.975
`...8753`	5	32×32 HSV+NIR	CLAHE V (HSV),NIR	164,580	193,536	0.974

Source: The Author

Figure B.15 – Structure of a CNN evolved for the SAT-6 task in fixed-point



Source: The Author

APPENDIX C — STUDY-CASE APPLICATION ONBOARD NANOSATC-BR2

C.1 NanosatC-BR2 mission

The NanosatC-BR2 is the second mission in the NanosatC-BR CubeSats Development Program led by the Brazilian Institute for Space Research (INPE), a research unit of the Brazilian Ministry of Science, Technology and Innovations (MCTI).

The INPE's CubeSats Development Program is in progress since 2006 (SCHUCH, Nelson Jorge et al., 2013; SCHUCH, Nelson J. et al., 2019) and has among its goals technology validation, personnel development and capacity building, representing a coordinated supraorganizational effort including stakeholders from governmental agencies, industry technology developers and academic research.

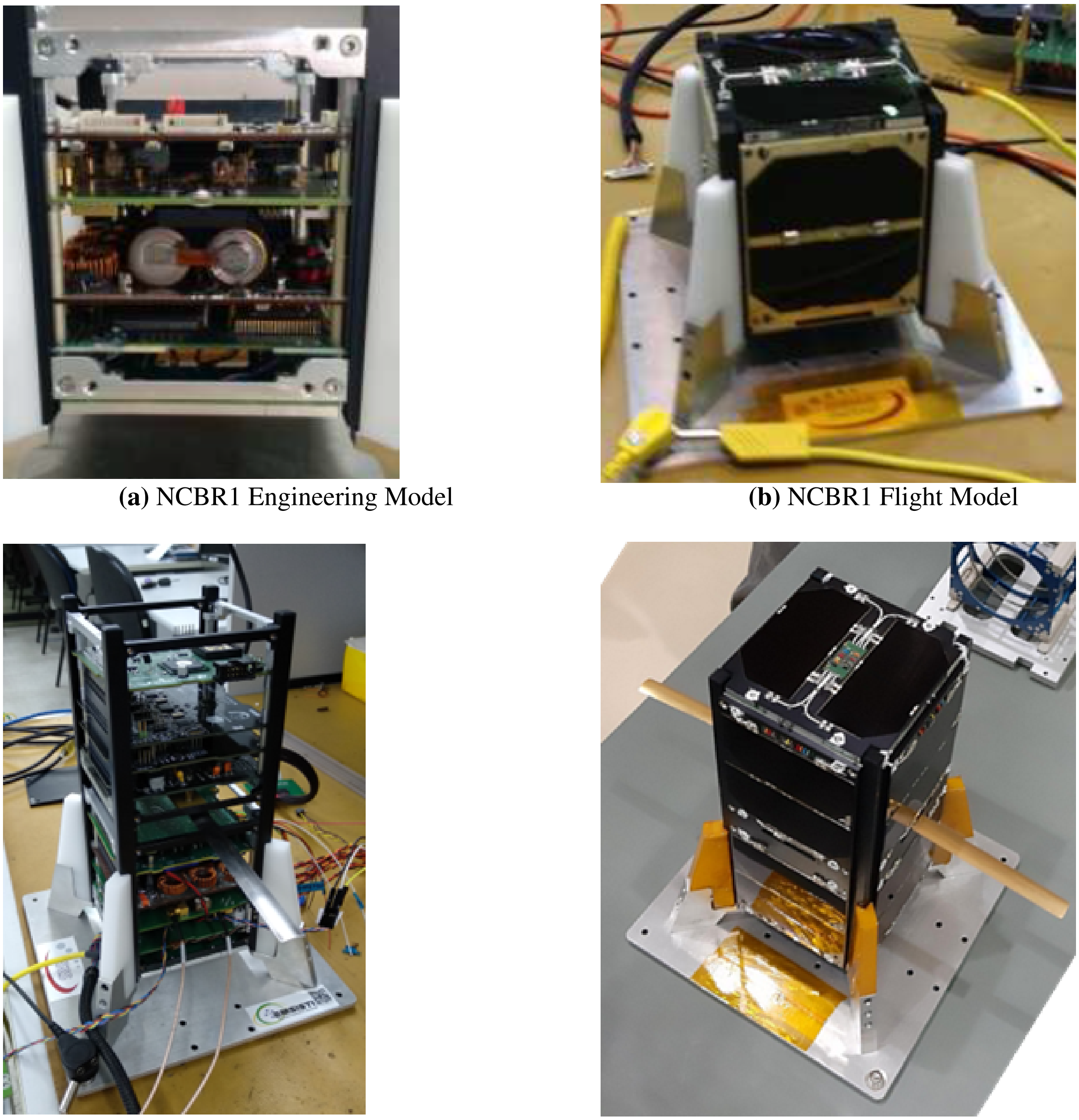
The NanosatC-BR1 (NCBR1, Fig. C.1a, C.1b and C.2) is a one unit (1U) CubeSat weighting $0.965\,\mathrm{kg}$ launched in June 2014 carrying a single payload board (MÂNICA, 2018).

The payload board for NanosatC-BR1 (Fig. C.3) was developed in partnership with Santa Maria Design House (SMDH) and Universidade Federal de Santa Maria (UFSM, Federal University of Santa Maria). The in-orbit experiments include measurements of Earth magnetic field, test of power switch and devices present in an ASIC test-chip developed by SMDH and test of a soft-core MIPS microprocessor and benchmark application implemented by UFRGS in a ProASIC3 Flash-based FPGA from Actel (later Microsemi and now Microchip).

Built upon the experience on NanosatC-BR1, the NanosatC-BR2 (NCBR2, Fig. C.1c, C.1d and C.4) is a two unit (2U) CubeSat weighting $1.720\,\mathrm{kg}$ launched in March 2021 carrying three payload boards aside additional experiments implemented as software at the on-board computer (OBC) board.

The INPE-SMDH/UFSM-UFRGS payload board for NanosatC-BR2 (Fig. C.5) extends the experimental platform (GUARESCHI et al., 2013) of NanosatC-BR1. One important modification is that it includes two FPGA devices, instead of one. One of these FPGAs is a Xilinx 7 Series SRAM-based FPGA while the other is a Actel (later Microsemi and now Microchip) SmartFusion Flash-based FPGA featuring also a hard-core Arm Cortex-M3 microprocessor in a system-on-chip (SoC). Both are standard commercial-grade devices.

Figure C.1 – NanosatC-BR CubeSats



(a) NCBR1 Engineering Model

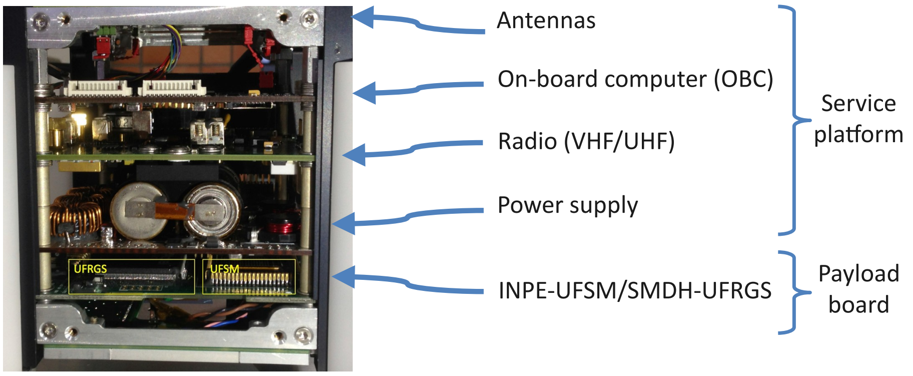
(b) NCBR1 Flight Model

(c) NCBR2 Engineering Model

(d) NCBR2 Flight Model

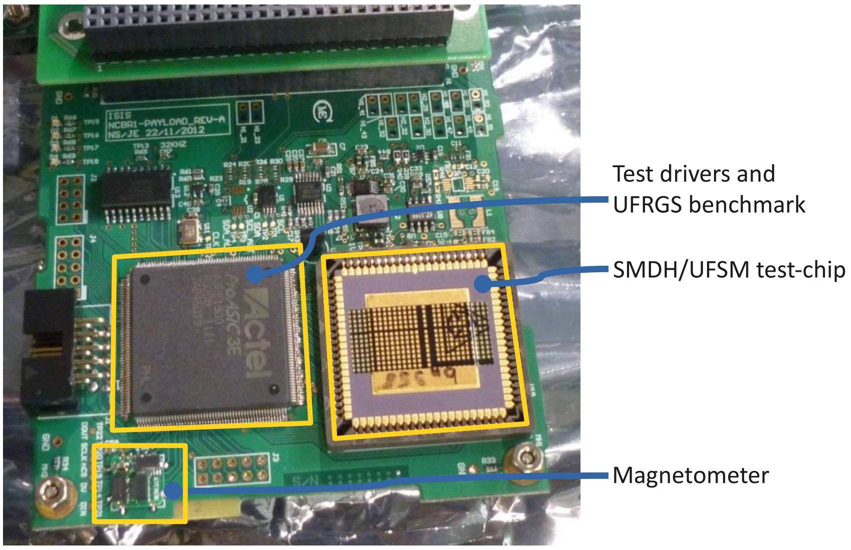
Source: INPE

Figure C.2 – NanosatC-BR1 Assembly



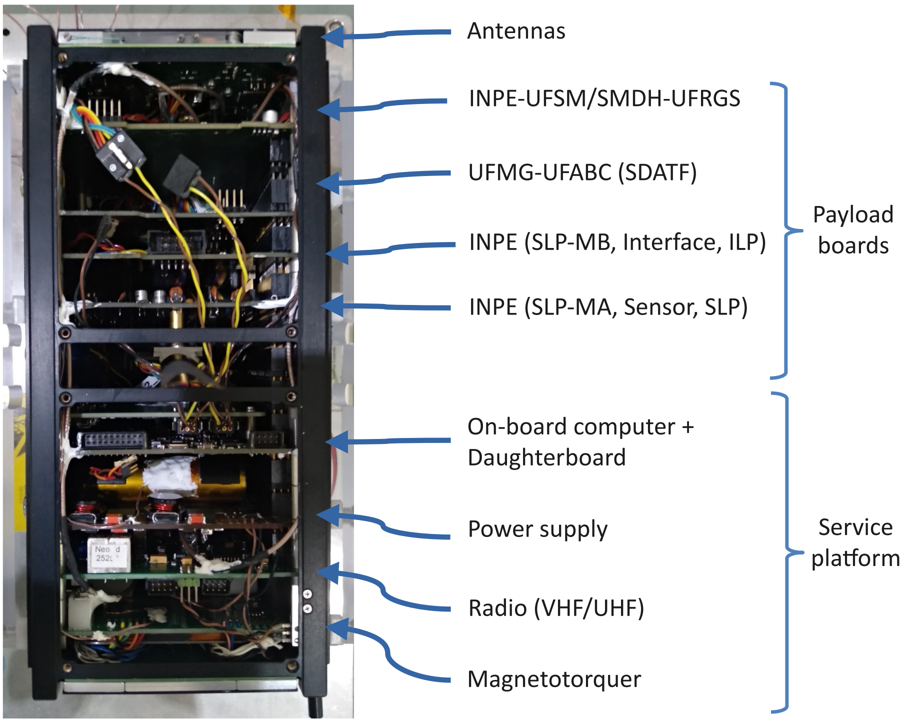
Source: INPE

Figure C.3 – NanosatC-BR1 INPE-SMDH/UFSM-UFRGS payload board



Source: INPE

Figure C.4 – NanosatC-BR2 Assembly

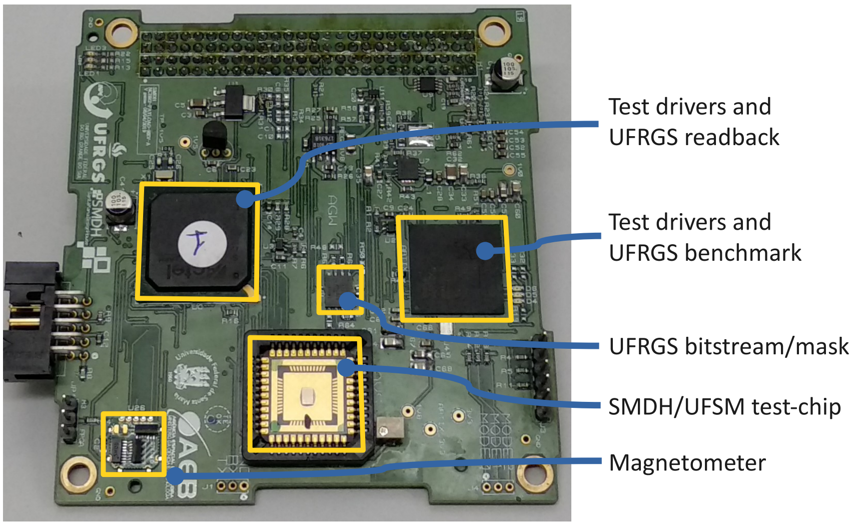


Source: INPE

C.2 NanosatC-BR2 payload overview

The operation mode of the INPE-SMDH/UFSM-UFRGS payload on NanosatC-BR2 is determined by configuration data stored inside the spacecraft configuration vector (SCCV) on the OBC. There are a few variables in the SCCV indicating wether the payload should be powered up and which tests should be run. It includes, for instance, flags indicating if the single event upsets (SEU) on SRAM-based FPGA should be tracked and

Figure C.5 – NanosatC-BR2 INPE-SMDH/UFSM-UFRGS payload board



Source: INPE

which versions of the UFRGS benchmark application should be tested.

An initial value was stored inside the SCCV on ground, before the satellite launch, which can be modified in-flight by sending telecommands to the OBC flight software whenever the satellite passes over a ground station.

A software application running at the Arm Cortex-M3 microprocessor (Fig. C.6a) communicates with OBC answering to telemetry and telecommand messages. Sequences of payload specific commands were stored inside the OBC flight software to initialize the payload informing its configuration as stored in SCCV, collect recorded data periodically and transfer to the OBC data handling (OBDH) storage for later transmission to ground, and shutdown the payload when the allotted test period is completed.

The INPE-SMDH/UFSM-UFRGS payload on NanosatC-BR2 implements five main functions:

- Housekeeping telemetry and telecommand, configuring the payload real time clock (RTC) and collecting information from temperature, current and voltage sensors (Fig. C.6b);

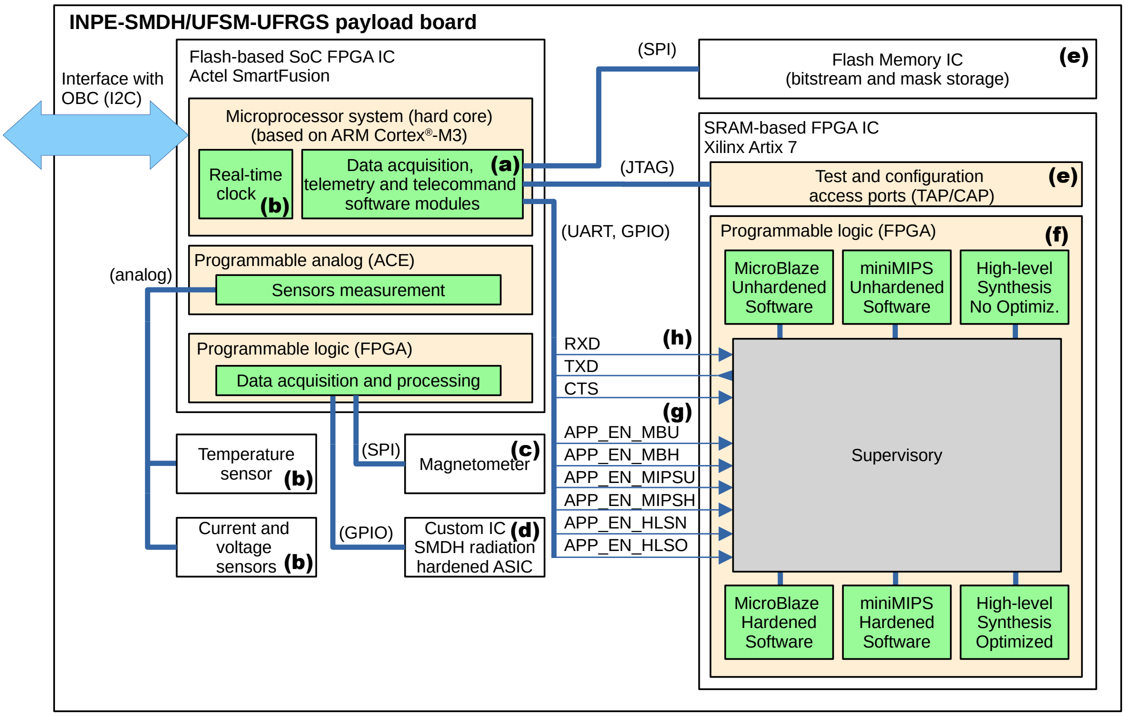
- Measurement of magnetic field intensity from magnetometer (Fig. C.6c) and, optionally, smoothing data with averaging filter ;

- Testing the circuit blocks inside SMDH custom IC (Fig. C.6d);

- Recording occurrences of single event upsets (SEU) in Xilinx Artix 7 FPGA (Fig. C.6e); and

- Testing benchmark application implemented in Xilinx Artix 7 FPGA using different processing and mitigation strategies (Fig. C.6f).

Figure C.6 – NanosatC-BR2 INPE-SMDH/UFSM-UFRGS payload overview


INPE-SMDH/UFSM-UFRGS payload board
Interface with OBC (I2C)
Flash-based SoC FPGA IC
Actel SmartFusion
Microprocessor system (hard core) (based on ARM Cortex®-M3)
Real-time clock (b)
Data acquisition, telemetry and telecommand software modules (a)
Programmable analog (ACE)
(analog)
Sensors measurement
Programmable logic (FPGA)
Data acquisition and processing
Temperature sensor (b)
(SPI)
Magnetometer (c)
Current and voltage sensors (b)
(GPIO)
Custom IC SMDH radiation hardened ASIC (d)
(SPI)
(JTAG)
(UART, GPIO)
Flash Memory IC (bitstream and mask storage) (e)
SRAM-based FPGA IC
Xilinx Artix 7
Test and configuration access ports (TAP/CAP) (e)
Programmable logic (FPGA) (f)
MicroBlaze Unhardened Software
miniMIPS Unhardened Software
High-level Synthesis No Optimiz.
RXD
TXD
CTS
(h)
(g)
APP_EN_MBU
APP_EN_MBH
APP_EN_MIPSU
APP_EN_MIPSH
APP_EN_HLSN
APP_EN_HLSO
Supervisory
MicroBlaze Hardened Software
miniMIPS Hardened Software
High-level Synthesis Optimized

Source: The Author

C.3 NanosatC-BR2 benchmark applications on Xilinx Artix-7

Two hardened software versions of the application seen on Fig. C.6f, for MicroBlaze and miniMIPS soft-core microprocessors, consist of software implemented hardware fault tolerance (SIHFT) (CHIELLE et al., 2015, 2016; CHIELLE, 2016), which are not intrinsically capable of masking faults or recovering from failures, but only to *detect* abnormal operation decreasing the confidence on the computed results. It is up to the client module to apply selection or voting mechanisms to extract a correct computation or computation with higher confidence, or to invoke recomputation or the proper system healing to restore its normal operation. This detection strategy solely, however, already allows for informed decision making and graceful exit or failsafe operation in safety- or mission-critical cyber-physical systems.

All the six versions of the benchmark application inside the SRAM-based FPGA are the same task of multiplication of a 32×32 matrix of unsigned integer 8 bit values stored in 32 bit words (Fig. C.7). All the six versions also follow a similar interface and control protocol with the test supervisory, which consists of a dual port memory block (BRAM) shared between supervisory and application, each connected to a different port and possibly at a different clock, and a set memory addresses and magic values settled between supervisory and application.

The data acquisition software running on Arm Cortex-M3 (Fig. C.6a) sets the values for GPIO pins (Fig. C.6g) signaling which applications should be run, according to the configuration received from OBC, which was, ultimately, stored on SCCV.

The supervisory enables the clock for the enabled application, according to its designated GPIO pin, and waits until the magic value for `IDLE` is written at the `SYNC` control memory address. After initialization, the application writes `IDLE` at the `SYNC` control memory address and waits until the magic value for `START` is written at the same `SYNC` address.

When the supervisory sees `IDLE` at the `SYNC` address, it writes `START` and waits until `DONE` magic value is written at the same `SYNC` address. When the application sees `START` it executes the matrix multiplication task, write the result values at the respective memory addresses and write `DONE` at the `SYNC` control memory address.

A maximum allowed execution time is configured for each application version, which varies depending on the processor and hardening. The supervisory have a timer associated with each application and signals failure in the case of timer run-out (timeout).

Upon completion of the application execution, signaled with `DONE`, the supervisory collects the result values from memory, compare with golden reference values configured for the application version, and reports failure in case of divergence.

If there is no divergence in the result, the supervisory issues a logical reset to the application and initiates the processing cycle again waiting until the magic value for `IDLE` is seen at the `SYNC` control memory address.

The supervisory reports periodically `ALIVE` messages to the data acquisition software through the serial communication port (Fig. C.6h). When one of the enabled applications fails the supervisory reports the failure of the application using its specific message. When all the enable applications have already failed the supervisory reports end of tests.

The known supervisory messages to the data acquisition software are described in Table C.1.

The legacy code of the matrix multiplication in Fig. C.7 should be improved at least by cleaning the result matrix before the computation since it works with constant

Figure C.7 – Simplified C source code for benchmark application

```c
#define SYNC_IDLE  0x00001D1E
#define SYNC_START 0x0000A5A5
#define SYNC_DONE  0x0000D9D9

#define SYNC_CTRL_ADDR        0
#define SYNC_RESULT_CHECKSUM  1
#define SYNC_RESULT_DUE       2

#define MATRIX_SQ_DIM        32
#define MATRIX_SQ_SIZE       (MATRIX_SQ_DIM*MATRIX_SQ_DIM)

#define MATRIX_A_START       1024
#define MATRIX_B_START       2048
#define MATRIX_C_START       3072

void matrix_mult(int ram[4096])
{
    unsigned int i = 0;
    unsigned int j = 0;
    unsigned int k = 0;
    unsigned int checksum = 0;

    // matrices A, B and C are contiguous in memory
    once_init_a_b: for(i=MATRIX_A_START; i<MATRIX_C_START; i++) {
        ram[i] = 0x55; // constant filler value
    }

    ram[SYNC_CTRL_ADDR] = SYNC_IDLE;

    lbl_wait_start: while(ram[SYNC_CTRL_ADDR] != SYNC_START);

    ram[SYNC_RESULT_CHECKSUM] = 0x0;
    ram[SYNC_RESULT_DUE] = 0x0;

    loop_a_rows: for(i=0; i<MATRIX_SQ_DIM; i++) {
        loop_b_cols: for(j=0; j<MATRIX_SQ_DIM; j++) {
            unsigned int sum = 0;

            loop_dot_product: for(k=0; k<MATRIX_SQ_DIM; k++) {
                unsigned int a = ram[ (i+(MATRIX_SQ_DIM*k))+MATRIX_A_START ];
                unsigned int b = ram[ (k+(MATRIX_SQ_DIM*j))+MATRIX_B_START ];

                sum = sum + a * b;
            }

            ram[ (i+(MATRIX_SQ_DIM*j))+MATRIX_C_START ] = sum;
        }
    }

    loop_check_sum: for(i=0; i<MATRIX_SQ_SIZE; i++) {
        unsigned int c = ram[i+MATRIX_C_START];
        checksum = checksum + c;
    }

    ram[SYNC_RESULT_CHECKSUM] = checksum;
    ram[SYNC_RESULT_DUE] = 0x0; // unhardened, will never trigger DUE
    ram[SYNC_CTRL_ADDR] = SYNC_DONE;

    lbl_wait_forever: while(1);
}
```

Source: The Author

input matrices and abnormal termination of the multiplication loops could leave the result matrix filled with data from a previous execution. In this situation the checksum could match the golden value, even in the case of incomplete multiplication loop, leading to an

Table C.1 – Benchmark supervisory serial communication messages

Message	***Value***	***Description***
TX_ALIVE	0x3e	Supervisory serial communication module is alive, reported nearly every 30 seconds.
FSM_ALIVE:	0x3d	Supervisory test driver module is alive, reported nearly every 1 second.
SYS_FAIL	0x3c	End of tests or system failure, communication end.
NONE	0x00	Filler, no relevant event regarding application status.
MIPSU_SEFI_SDC	0x01	Application running as miniMIPS unhardened software is dead (either timeout or a computed result mismatch).
MBU_SEFI_SDC	0x02	Application running as MicroBlaze unhardened software is dead (either timeout or a computed result mismatch).
MIPSH_SEFI_SDC	0x04	Application running as miniMIPS hardened software is dead (either timeout or an undetected computed result mismatch).
MIPSH_DUE	0x0c	Application running as miniMIPS hardened software is dead (a detected computed result mismatch).
MBH_SEFI_SDC	0x10	Application running as MicroBlaze hardened software is dead (either timeout or an undetected computed result mismatch).
MBH_DUE	0x30	Application running as MicroBlaze hardened software is dead (a detected computed result mismatch).
HLSO_SEFI_SDC	0x80	Application running as High-level synthesis with pipeline optimization is dead (either timeout or a computed result mismatch).

Source: The Author

undetectable failure. It would also be convenient if the input matrices elements were not filled with the same constant value because it will not exercise properly the circuits in the arithmetic logic unit, leading to latent failures. Filling the input matrices with values generated from a simple pseudo-random number generator, even with a constant seed, would increase the coverage in the arithmetic logic unit. Alternatively, two or more different seeds, with well-known checksum values, could be used by the test driver passing the seed through the memory interface. This would avoid the artificial effort of cleaning up the result matrix. Finally, the simple sum may be too simple to be used as checksum and should be replaced by an position-sensitive algorithm.

APPENDIX D — RESUMO EXPANDIDO EM PORTUGUÊS

Aperfeiçoamentos em Injeção de Falhas para FPGAs SRAM Xilinx 7 Series e UltraScale+

D.1 Introdução

Aplicações críticas quanto a segurança e aplicações de missão crítica exploram a crescente capacidade computacional dos sistemas digitais, incluindo aplicações a bordo de satélites, aeronaves, ou mesmo aplicações em solo como instalações de computação de alto desempenho (HPC), assistência ao motorista (ADAS) e veículos autônomos.

Dentre os sistemas digitais que suportam estas aplicações críticas e de missão crítica, os FPGAs baseados em SRAM são de especial interesse devido a alta densidade computacional e ao custo e tempo de engenharia relativamente menores, quando comparado com o desenvolvimento de ASICs.

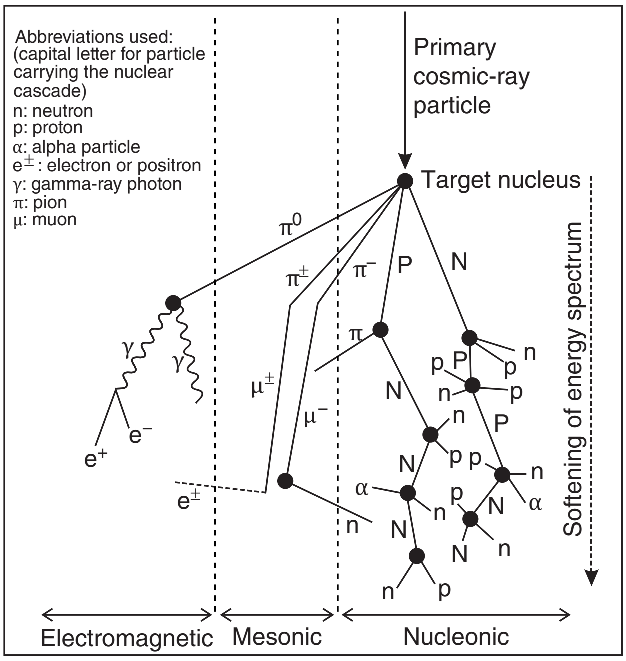
O uso de FPGAs de prateleira (COTS) de classe comercial tem as vantagens de baixo custo e alta disponibilidade por não estarem sujeitos a restrições como International Traffic in Arms Regulations (ITAR) como os produtos de classe militar ou aeroespacial.

Além disso, o uso da tecnologia SRAM tem as vantagens de permitir uma fácil reconfiguração do dispositivo sendo de interesse para aplicações onde há chaveamento de tarefas, ou seja, o mesmo hardware reprogramado para diferentes funções ao longo da missão, ou onde há necessidade de atualizações para correções ou adaptação a novas técnicas de processamento ou novas condições do ambiente, favorecendo, por exemplo, a sobrevida tecnológica em aplicações de aprendizado de máquina (IA/ML).

Uma desvantagem é que as células de memória SRAM em dispositivos COTS de classe comercial são especialmente suscetíveis a efeitos causados por radiação. Em órbita, estes efeitos são causados por prótons, ions pesados e elétrons enquanto em altas altitudes e em solo o efeito predominante tem origem em nêutrons (Fig. D.1). Estes efeitos de radiação podem afetar eletrônica a bordo de satélites artificiais, veículos espaciais, aeronaves, veículos autônomos, dispositivos médicos implantáveis e instalações de computação de alto desempenho (HPC).

Os efeitos causados pela incidência de uma única partícula isoladamente são conhecidos como efeitos de eventos singulares (SEE, *single-event effect*).

Figura D.1 – *Air shower*: radiação secundária de raios cósmicos na atmosfera



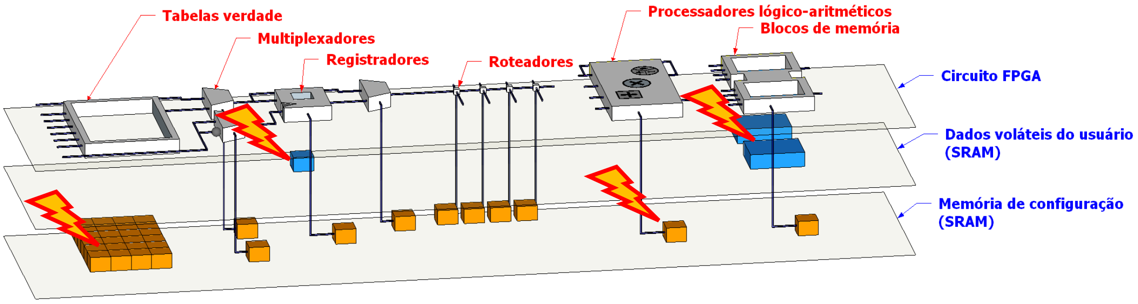
Fonte: Dunai (2010)

O tipo de evento singular relevante para esta tese ocorre quando a interação de radiação com as células de memória SRAM no FPGA causa a inversão dos valores armazenados na memória, por exemplo invertendo o valor lógico de 0 para 1 ou vice-versa, o que é conhecido como *bit-flip* e também como *upset* de memória (SEU, *single-event upset*). Uma mesma partícula pode ter efeito sobre múltiplas células de memória. Quando o SEU afeta apenas uma célula de memória ele também é chamado de *single-bit upset* (SBU) e quando afeta mais de uma célula de memória ele é chamado de *multiple-bit upset* (MBU).

Em FPGA SRAM, a susceptibilidade a radiação, em especial aos upsets de memória (SEUs), afeta não apenas os dados dos usuários, mas também a própria definição do circuito que é armazenada na memória de configuração (CRAM) (Fig. D.2). O volume de CRAM é muito maior que os outros tipos de memória e torna-se o foco principal do problema.

Uso de FPGAs SRAM em aplicações críticas envolve, portanto, a caracterização dos dispositivos e aplicações, o uso de técnicas de mitigação de falhas para atender requisitos de confiabilidade, e a qualificação das aplicações com tolerância a falha.

Figura D.2 – Efeitos de radiação em FPGA SRAM


Tabelas verdade
Multiplexadores
Registradores
Roteadores
Processadores lógico-aritméticos
Blocos de memória
Circuito FPGA
Dados voláteis do usuário (SRAM)
Memória de configuração (SRAM)

Fonte: O autor

D.1.1 Motivação

Idealmente, os sistemas digitais e aplicações seriam testados e qualificados em seu ambiente operacional, onde estariam sujeitos aos efeitos da radiação. A avaliação de confiabilidade em condições operacionais possui diversas desvantagens, incluindo alto custo, longo tempo de teste, e execução em estágios mais avançados do desenvolvimento, o que também aumenta o custo de eventuais correções necessárias. Como alternativa, a injeção de falhas, em laboratório, permite a avaliação desde os estágios iniciais de prova de conceito até cenários de uso mais próximos do ambiente real.

Por outro lado, a injeção de falhas é também complexa porque exige um conhecimento aprofundado do dispositivo alvo e do seu comportamento sob radiação, aumentando o esforço em aspectos como:

- Conhecer a arquitetura do dispositivo, incluindo detalhamento dos módulos ou elementos de interesse dentro do dispositivo;
- Identificar um mecanismo de acesso para o dispositivo e para os elementos internos de interesse;
- Conhecer os tipos de falhas que podem ocorrer nos diferentes cenários de operação;
- Prover uma cobertura de teste adequada dos pontos de interesse e das funcionalidades do sistema;
- Realizar uma validação rigorosa e análise comparativa.

No processo de engenharia, a injeção de falhas pode ser usada não apenas como método de teste para analisar confiabilidade e verificar a resiliência a erros, mas também para direcionar a seleção das técnicas de mitigação aplicadas. Porém, maior consistência entre injeção de falhas e radiação é necessária para garantir os testes foram realizados em condições representativas do ambiente operacional real e também garantir que o processo de engenharia avance na direção correta. Além disso, a injeção de falhas também precisa fornecer uma avaliação adequada dentro de um intervalo de tempo razoável para não comprometer a produtividade do processo de engenharia.

D.1.2 Objetivos

Implementar aperfeiçoamentos no injetor de falhas para FPGAs Xilinx baseados em SRAM desenvolvido na UFRGS para ampliar a consistência entre injeção de falhas e testes em radiação, acelerar o processo de injeção de falhas para reduzir o tempo de campanha, e estender o suporte do injetor para uma nova família de produtos Xilinx.

D.2 Conceitos preliminares

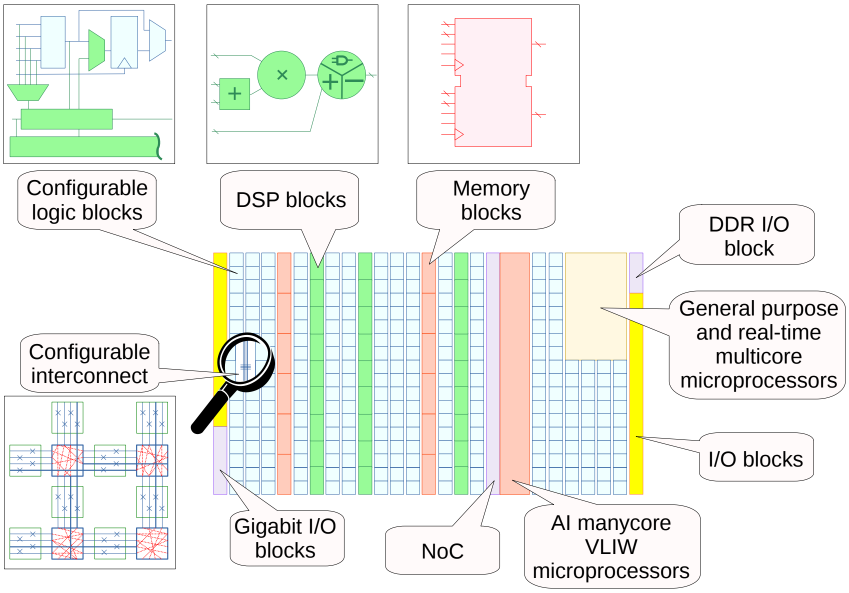
Os FPGAs são circuitos integrados predominantemente digitais de propósito geral, constituidos de elementos básicos configuráveis, como visto na Fig. D.2, incluindo lógica combinacional, registradores, blocos lógico-aritméticos (DSP), blocos de memória (BRAM) e matriz de roteamento de sinais (PIP/INT). Alguns FPGAs incluem também recursos mais especializados, incluindo conversores analógico-digital (ADC), interfaces de comunicação de alta velocidade (LVDS de uso geral, DDR, SATA, PCI, USB), microprocessadores hardcore (RISC, GPU, VLIW), e redes on-chip (NoC).

Os diversos recursos dos FPGA Xilinx são organizados em colunas (Fig. D.3) e dentro destas colunas existem subcolunas de vetores de memória (*frames*) que formam a memória de configuração (CRAM) ou memória de dados (BRAM) do FPGA.

Os vetores de memória CRAM e BRAM podem ser acessados através de diversos mecanismos, internos ao FPGA, como a Porta Interna de Acesso a Configuração (ICAP) e a Porta de Processor de Acesso a Configuração (PCAP), ou externos, como a interface padronizada JTAG ou interface proprietária SelectMAP (Fig. D.4).

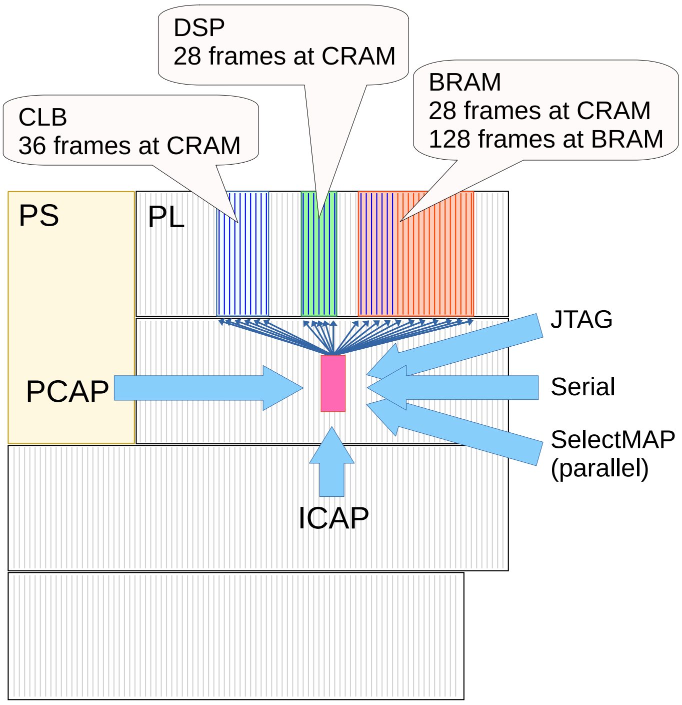
Estas diferentes interfaces de configuração e teste que dão acesso as memórias do FPGA podem ser utilizadas na injeção de falhas emulando os efeitos de radiação. O injetor de falhas desenvolvido na UFRGS, mais especificamente, utiliza a interface ICAP para esse fim.

Figura D.3 – Visão simplificada dos elementos no FPGA Xilinx


Configurable logic blocks
DSP blocks
Memory blocks
DDR I/O block
General purpose and real-time multicore microprocessors
Configurable interconnect
I/O blocks
Gigabit I/O blocks
NoC
AI manycore VLIW microprocessors

Fonte: O autor

Figura D.4 – Acesso de hardware à memoria no FPGA Xilinx



Fonte: O autor

D.3 Estado-da-arte e trabalho proposto

Diferentes injetores de falha foram desenvolvidos para os produtos Xilinx ao longo do tempo, utilizando diferentes estratégias para emular os efeitos de radiação (Tabela D.1).

Uma implementação do injetor de falhas UFRGS para Xilinx Virtex-5 é apresen-

Tabela D.1 – Injetores de falha para FPGA Xilinx

Injetor de falhas	***Manipulação da memória***	***Dispositivo alvo***	***Interface de acesso***
Antoni et al. (2000)	Modifica o arquivo de bitstream antes de carregar no FPGA. O bitstream pode ser de reconfiguração parcial, por exemplo apenas 1 frame.	Xilinx Virtex	JTAG/ MultiLINX
Johnson et al. (2003)		Xilinx Virtex	JTAG/ MultiLINX
Aldeghiri et al. (2007) ESA FLIPPER	Usa um FPGA para manipular bitstream em um segundo FPGA.	Xilinx Virtex-II	n.d.
Napoles et al. (2007) FT-UNSHADES		Xilinx Virtex-II	SelectMAP
Mogollon et al. (2011) FT-UNSHADES2		Xilinx Virtex-5	SelectMAP
Hardward et al. (2015) BYU XRTC-V5FI		Xilinx Virtex-5	SelectMAP
Thurlow et al. (2019) BYU TURTLE		Xilinx 7 Series	JTAG
Sterpone et al. (2007)	Modifica o bitstream localmente dentro do FPGA.	Xilinx Virtex-II	ICAP
Villalta et al. (2014)	Modifica o bitstream usando processador Arm Cortex-A dentro do mesmo dispositivo.	Xilinx Zynq-7000	PCAP
Gomes-Cornejo et al. (2017)	Modifica conteúdo dos blocos de memória (BRAM) usando processador Arm Cortex-A dentro do mesmo dispositivo.	Xilinx Zynq-7000	PCAP

Fonte: O autor

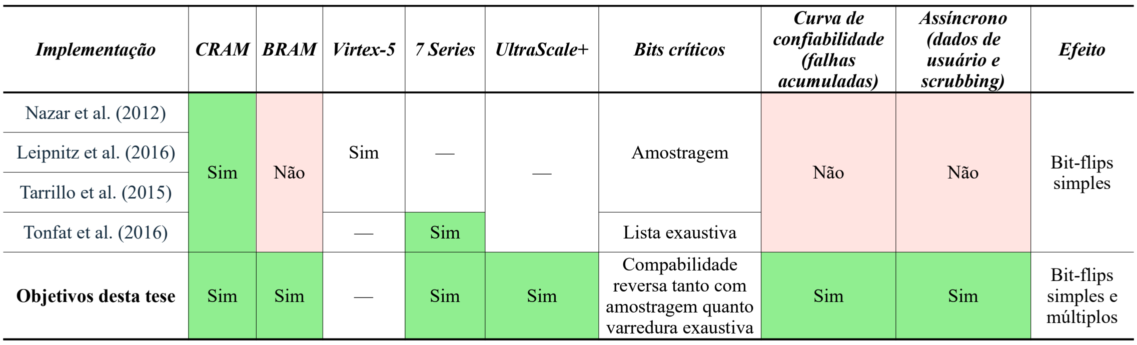
tada por Nazar et al. (2012), e tem como características todos os componentes do injetor de falhas embarcado no FPGA, incluindo os vetores de teste, o módulo de injeção de falhas, o relatório de diagnóstico e, naturalmente, o circuito a set testado. Outras implementações de Leipnitz et al. (2016) e de Tarrillo et al. (2015) para o mesmo dispositivo Xilinx Virtex-5 introduzem pequenas variações nos módulos que são embarcados no FPGA ou na forma de interconexão do FPGA com o veículo de teste.

Tonfat et al. (2016) implementaram o injetor de falhas UFRGS para a família Xilinx Artix-7 tendo como principais características o uso de um pequeno módulo de injeção de falhas que se comunica com o computador de coordenação da campanha através de uma interface de comunicação serial. O modelo de uso desse injetor se baseava pricipalmente na implementação de duas cópias do circuito sob teste dentro do mesmo FPGA, uma copia alvo da injeção e outra utilizada como resultado verdadeiro, as quais era comparadas a cada falha injetadas.

O quadro da Fig. D.5 resume as principais características das implementações anteriores do injetor de falhas UFRGS para FPGA Xilinx, assim como as funcionalidades propostas como aperfeiçoamentos nesta tese.

Para atingir estes objetivos, foram coletadas informações detalhadas sobre a organização da matriz de memória, utilizando tanto pesquisa bibliográfica quanto experimentos de cartografia com laser e microfeixe de íons pesados, sobre a ocorrência de eventos causados por radiação na matriz de memória do FPGA, utilizando testes estáticos com

Figura D.5 – Características do injetor de falhas UFRGS para FPGA Xilinx



Implementação	*CRAM*	*BRAM*	*Virtex-5*	*7 Series*	*UltraScale+*	*Bits críticos*	*Curva de confiabilidade (falhas acumuladas)*	*Assíncrono (dados de usuário e scrubbing)*	*Efeito*
Nazar et al. (2012)	Sim	Não	Sim	—	—	Amostragem	Não	Não	Bit-flips simples
Leipnitz et al. (2016)									
Tarrillo et al. (2015)									
Tonfat et al. (2016)			—	Sim		Lista exaustiva			
Objetivos desta tese	Sim	Sim	—	Sim	Sim	Compabilidade reversa tanto com amostragem quanto varredura exaustiva	Sim	Sim	Bit-flips simples e múltiplos

Fonte: O autor

prótons, partículas alfa, íons pesados, neutrons térmicos e neutrons rápidos, e sobre a organização do arquivo de programação do FPGA, utilizando operações de *readback* da memória do FPGA. Com estas informações foram implementadas modificações na lógica do módulo injetor de falhas, no protocolo de comunicação serial, e nos scripts de execução das campanhas de injeção de falhas. O injeção de falhas também foi adaptado para funcionar com os dispositivos da família Xilinx UltraScale+.

D.4 Resultados experimentais

Os principais resultados experimentais desta tese são resumidos nas próximas seções.

D.4.1 Testes estáticos

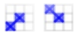
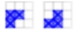
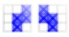
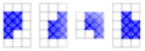
A análise de dados coletados em testes estáticos permitiu identificar a ocorrência de diferentes geometrias de MBU e a proporção entre elas. Alguns exemplos são apresentados no quadro da Fig. D.6, onde se pode observar que a ocorrência de MBU é bastante sensível ao tipo de partícula.

D.4.2 Injeção de falhas acumuladas

A injeção de falhas acumuladas é, principalmente, uma variação na metodologia de teste, tendo pouco impacto sobre o módulo de injeção de falhas embarcado no FPGA.

A injeção acumulada foi desenvolvida com dois objetivos, que são acelerar a cam-

Figura D.6 – Comparativo entre partículas alfa, íons pesados e nêutros

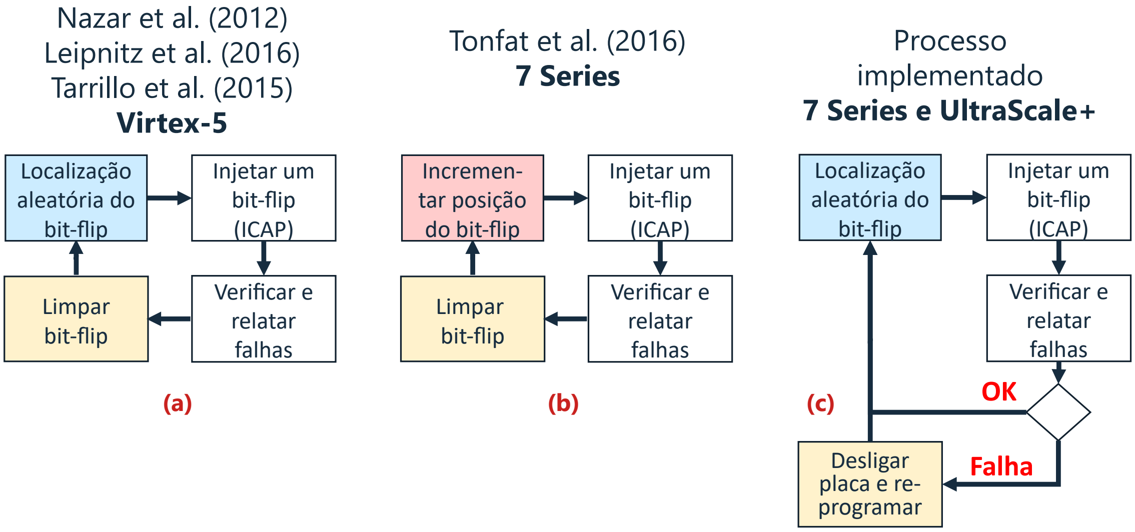
Tipo de memória	*Tipo de SEU*	*Exemplos*	*Partícula α*	*Íons pesados*	*Nêutrons*		
					14 MeV 0°	*14 MeV 180°*	*(Epi)Térmicos*
BRAM	SBU 1-1-1		100,0%	82,0%	93,4%	97,1%	95,4%
	MBU 2-1-2		—	16,2%	4,7%	2,9%	—
	MBU 1-2-2		—	—	—	—	4,5%
	Outras geometrias		—	1,8%	1,9%	—	0,1%
CRAM	SBU 1-1-1		97,6%	38,1%	76,7%	79,9%	78,1%
	MBU 2-2-2		2,4%	41,9%	16,9%	15,5%	0,0%
	MBU 2-1-2		0,0%	4,4%	3,5%	2,1%	0,0%
	MBU 1-2-2		—	—	0,3%	1,5%	17,8%
	MBU 2-2-3		—	3,0%	1,3%	0,5%	0,0%
	MBU 2-2-4		—	0,2%	—	0,5%	—
	MBU 2-3-4		—	8,3%	0,6%	—	0,0%
	MBU 2-3-5		—	0,6%	0,3%	—	0,0%
	Outras geometrias		—	3,4%	0,3%	—	4,1%

Fonte: O autor

panha de injeção de falhas e permitir a análise da confiabilidade na forma de uma curva de confiabilidade $R(t)$ do mesmo modo que se realiza a análise nos experimentos em radiação.

O diagrama da Fig. D.7 mostra as estratégias tipicamente utilizadas nas implementações anteriores do injetor de falhas e a nova metodologia implementada.

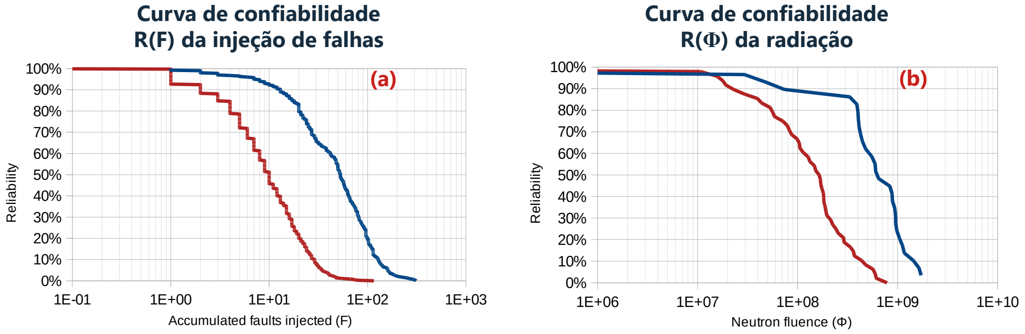
Figura D.7 – Metodologias de injeção de falhas: (a) amostragem aleatória, (b) exaustiva-sequencial, (c) acumulada-aleatória


Nazar et al. (2012)
Leipnitz et al. (2016)
Tarrillo et al. (2015)
Virtex-5
Localização aleatória do bit-flip
Injetar um bit-flip (ICAP)
Limpar bit-flip
Verificar e relatar falhas
(a)
Tonfat et al. (2016)
7 Series
Incremen-tar posição do bit-flip
Injetar um bit-flip (ICAP)
Limpar bit-flip
Verificar e relatar falhas
(b)
Processo implementado
7 Series e UltraScale+
Localização aleatória do bit-flip
Injetar um bit-flip (ICAP)
Verificar e relatar falhas
OK
Falha
Desligar placa e re-programar
(c)

Fonte: O autor

A Fig. D.8 mostra as curvas de confiabilidade obtida pela injeção acumulada e a curva obtida em ensaios de radiação com o mesmo circuito de teste.

Figura D.8 – Curva de confiabilidade: (a) injeção de falhas, (b) radiação



Fonte: O autor

Além de a nova metodologia permitir melhor comparação com os dados de radiação, ela reduziu o tempo de campanha, para o circuito de teste, de 142 horas no método de Tonfat et al. (2016) para 3,7 horas.

D.4.3 Injeção de falhas na presença de *scrubbing*

O mecanismo de *scrubbing* é um recurso de hardware disponível nos FPGAs Xilinx que realiza continuamente varreduras na memória CRAM do FPGA e identifica a ocorrência de SEUs pela comparação de códigos de deteção e correção de erros (ECC) inseridos na imagem de memória. No caso dos FPGAs Xilinx 7 Series, o mecanismo de scrubbing também tem a capacidade de corrigir os SEUs.

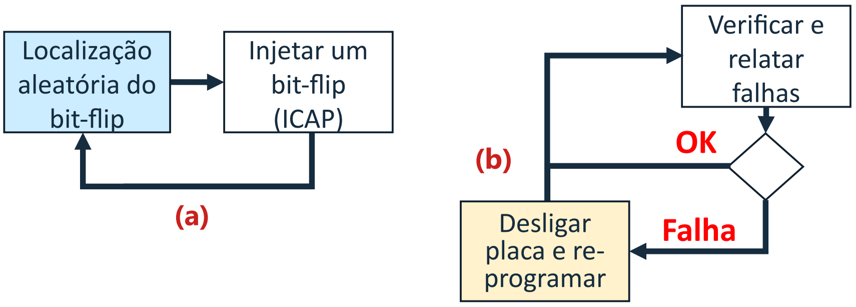
Em radiação, os SEUs podem ocorrer a qualquer momento durante o ciclo de processamento do circuito sob teste. Quando a correção pelo mecanismo de scrubbing está ativa, estes SEUs podem também serem corrigidos a qualquer momento ao longo do ciclo de processamento e, portanto, estão sujeitos a mascaramento temporal.

Esta é uma situação diferente da apresentada na Fig. D.7, onde os SEUs são considerados como persistentes e, portanto, não importa se ocorreram durante um ciclo de processamento anterior do circuito sob teste.

Para injetar falhas na presença de scrubbing, a principal modificação introduzida no injetor de falhas foi separar o processo apresentado na Fig. D.7c em dois processos independentes, como na Fig. D.9.

O subprocesso na Fig. D.9a opera de forma a produzir SEUs em posições aleatórias da memória CRAM do FPGA e em momentos aleatórios, a uma taxa média de

Figura D.9 – Injeção com scrubbing: (a) emulação de radiação, (b) diagnóstico do circuito em teste

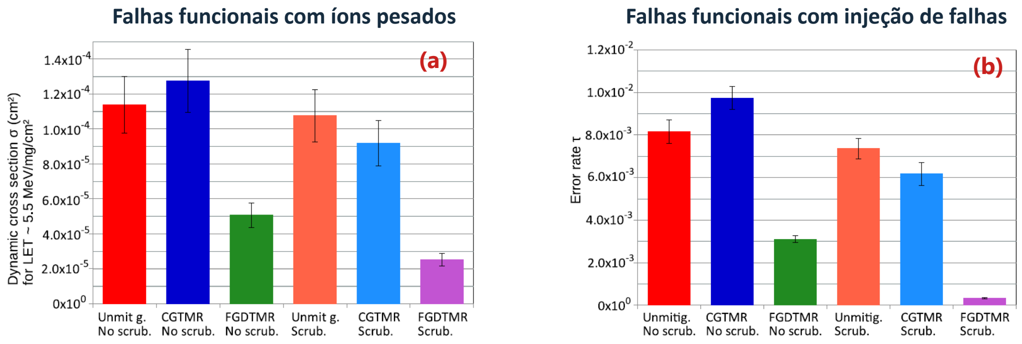

Localização aleatória do bit-flip
Injetar um bit-flip (ICAP)
(a)
Verificar e relatar falhas
OK
(b)
Falha
Desligar placa e re-programar

Fonte: O autor

ocorrência associada a um fluxo de radiação que está sendo emulando. Quando conveniente, a ocorrência se SEUs no subprocesso na Fig. D.9a pode ser modelada como uma distribuição de Poisson. Por sua vez, o subprocesso na Fig. D.9b, que monitora os resultados do circuito em teste, pode ser exatamente o mesmo processo utilizado nos experimentos em radiação. Uma vantagem adicional dessa estratégia é que o injetor de falhas pode ser utilizado também na validação dos scripts de monitoramente da campanha de radiação.

A Fig. D.10 mostra métricas de confiabilidade obtida para um circuito de teste em seis configurações diferentes, sob radiação e sob injeção de falhas. Em três destas configurações do circuito de teste o mecanismo de scrubbing em hardware Xilinx estava ativo e os resultados obtidos mostram que a ordem (*ranking*) entre as diferentes configurações, em termos de confiabilidade, é consistente entre radiação e injeção de falhas.

Figura D.10 – Métricas de confiabilidade obtidas em radiação (a) e em injeção de falhas (b)


Falhas funcionais com íons pesados

Falhas funcionais com injeção de falhas

Fonte: O autor

D.4.4 Injeção de falhas com múltiplos bit-flips

Como já citado anteriormente, o efeito dos SEUs na CRAM é o fator dominante na confiabilidade dos circuitos implementados em FPGA. Diversas técnicas de mitigação podem ser introduzidas no circuito para mascaras os efeitos dos SEUs a aumentar a confiabilidade ou o tempo de operação sob radiação.

Na presença destas mitigações, o modelo de falha simples adotado nas implementações anteriores do injetor de falhas passa a mostrar suas limitações. No exemplo da Fig. D.10, podemos observar que há maior divergência entre as métricas de confiabilidade quando as técnicas de mitigação de redundância modular distribuída (FGDTMR) e de scrubbing são combinadas. Neste cenário, a injeção de falhas mostra uma expectativa otimista, que não se concretiza nos ensaios em radiação.

O que ocorre é que em radiação, como já apresentado na Fig. D.6, uma parcela significativa dos SEUs envolvem múltiplos bit-flips (MBU). Estes MBUs podem facilmente prejudicar técnicas como FGDTMR onde os recursos redundantes não podem ser isolados. Além disso, a ocorrência de MBUs dentro de um mesmo frame de memória CRAM impede a correção pelo mecanismo de scrubbing que tem capacidade para corrigir apenas um bit-flip por frame.

Para ampliar a consistência entre radiação e injeção de falhas, a emulação dos efeitos de radiação pelo injetor de falhas precisa também emular a ocorrência de MBUs.

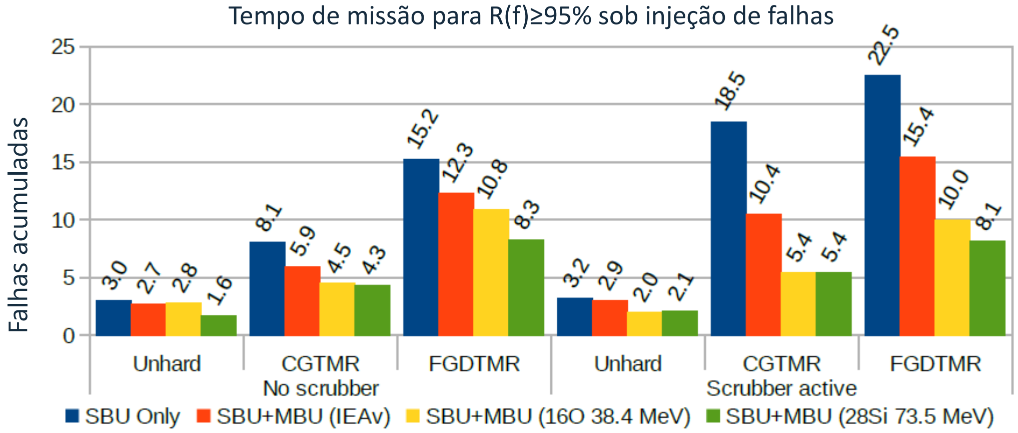
O desafio, todavia, é que em radiação os bit-flips associados a um mesmo SEUs ocorrem simultaneamente, enquanto na injeção de falhas cada um dos bit-flips precisa ser injetado separadamente para formar o MBU. Estes bit-flips injetados separadamente precisam ser observados tanto pelo circuito em teste quanto pelo mecanismo de scrubbing como sendo simultâneaos.

A solução adotada foi modificar o módulo de injeção de falhas para suspender temperariamente tanto a execução do circuito em teste, através de *clock gating*, quanto a execução do scrubber, através dos registradores de configuração do FPGA.

O protocolo de comando entre o script da campanha e o módulo injetor de falhas também foi modificado, para que se pudesse informar as posições dos diferentes bit-flips que formam o SEU.

A Fig. D.11 mostra métricas de confiabilidade para um circuito de teste com diferentes estratégias de mitigação, combinando redundância modular e scrubbing, em cenários de injeção de falhas emulando apenas SBUs e emulando SBUs e MBUs em geometrias e proporções observadas nos testes estáticos com nêutrons rápidos nas instalações do IEAv e com feixes de íons pesados oxigênio e silício nas instalações do LAFN. Estes diferentes ambientes de radiação produzem MBUs em diferentes taxas que prejudicam a eficiência das técnicas de mitigação em diferentes maneiras. Como se observa na Fig. D.11, a injeção de falhas incluindo a emulação de MBUs produz métricas de confiabilidade menos otimistas que a injeção apenas com SBUs, o que diminui a discrepância entre os resultados de radiação e injeção de falhas.

Figura D.11 – Métricas de confiabilidade obtidas com injeção de falhas



Fonte: O autor

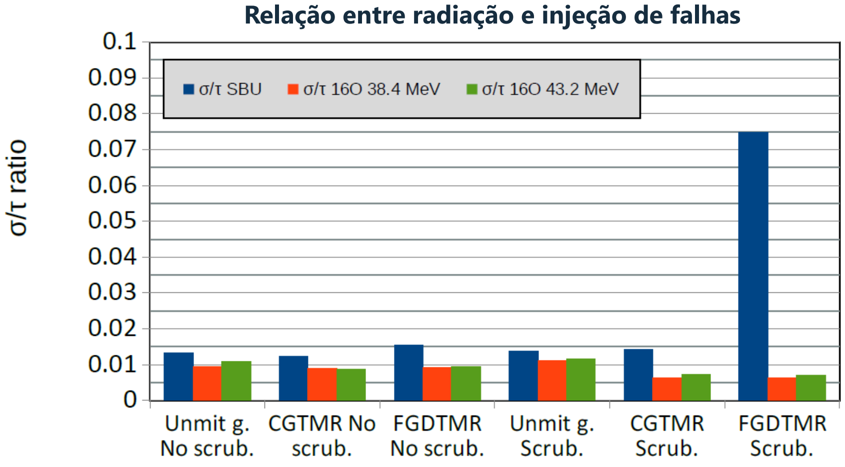
Para o mesmo circuito de teste discutido na Fig. D.10, a Fig. D.12 mostra a razão entre as métricas de confiabilidade obtidas em radiação e obtidas na injeção de falhas apenas com a emulação de SBUs e com a emulação de MBUs para feixes de íons pesados próximos do utilizado nos testes em radiação. Os resultados mostram que o desvio entre radiação e injeção de falhas para a configuração combinando as técnicas de mitigação é claramente menor quando se utiliza a injeção de falhas incluindo a emulação de MBUs.

D.5 Conclusão

Os experimentos de cartografia com laser e testes estáticos com radiação permitiram esclarecer aspectos da organização da memória do FPGA e construir modelos estatísticos da ocorrência de múltiplos bit-flips associados a incidência de uma mesma partícula.

Estas informações permitiram reproduzir mais fielmente o comportamenteo de radiação na injeção de falhas, especialmente na presença de técnicas de mitigação de falhas, que era onde havia maior desvio. Ao mesmo tempo, a injeção de falhas acumulada permitiu acelerar a execução das campanhas de injeção de falhas e produzir dados comparáveis

Figura D.12 – Relação entre radiação e injeção de falhas



Fonte: O autor

com radiação na forma da curva de confiabilidade.

O quadro da Fig. D.13 apresenta um resumo das funcionalidades e metodologias implementadas no injetor de falhas UFRGS para as duas famílias de FPGA objeto desta tese.

Figura D.13 – Maturidade das funcionalidades e metodologias implementadas

Técnica e metodologia	*Nível de maturidade*			
	7 Series		*UltraScale+*	
Exaustiva-sequencial (e.g. Tonfat et al. 2016)	**Alto**	Validado em laboratório e qualificado com experimentos de radiação	**Médio**	Prova de conceito. A funcionalidade está implementada no injetor.
Aleatório-acumulada	**Alto**	Validado em laboratório e qualificado com experimentos de radiação	**Médio**	Implementada no injetor e script de campanha. Demonstrado com estudo de caso.
Aleatório-acumulada + scrubbing	**Alto**	Validado em laboratório e qualificado com experimentos de radiação	**Baixo**	Conceito não foi testado amplamente. Algum esforço ainda é necessário para ajustar ao scrubber softcore.
Aleatório-acumulada + scrubber + MBU	**Médio**	Implementado no injetor e demonstrado comparando com dados de radiação.	**Baixo**	Conceito não foi testado. Necessário coletar dados de testes estáticos.
Injeção de falhas em BRAM	**Médio**	Implementação demonstrada. Identificadas limitações de viabilidade e aplicabilidade.	**Médio**	Prova de conceito. Pouca chance de o comportamento ser diferente de 7 Series.
Injeção de falhas em flip-flop	**Médio**	Implementação demonstrada. Identificadas limitações de viabilidade e aplicabilidade.	**Baixo**	Conceito não foi implementado ou testado. A reconfiguração parcial em UltraScale+ é completamente diferente de 7 Series e o método não pode ser diretamente portado.

Fonte: O autor

Quanto a futuras investigações e desenvolvimento, cabe observar que algumas das estratégias de injeção de falhas aplicadas apenas aos FPGAs Xilinx 7 Series ainda podem ser aplicadas nos FPGAs Xilinx UltraScale+. Novas investigações e testes com scrubber em UltraScale+ são especialmente importantes pois nesta família de dispositivos o mecanismo de scrubbing em hardware não inclui mais a capacidade de corrigir os bit-flips, passando a depender de um módulo adicional de correção implementando no próprio FPGA.

Apesar das mudanças arquiteturais na família UltraScale+ terem reduzido significativamente a vulnerabilidade do scrubbing a MBUs, ainda é conveniente investigar o impacto de MBUs sobre a redundância modular distribuída (FGDTMR), onde não é viável o partitionamento e isolamento físico dos recursos redundantes.

Finalmente, em desenvolvimentos futuros, todas as técnicas e metodologias de injeção de falhas implementadas merecem ser investigadas na nova família Xilinx Versal de FPGAs baseadas na tecnologia FinFET de 7 nm.