Master's Degree in
Computer Engineering

FPGA qualification for the LHC radiation environment

Developed at the European Organization for Nuclear Research

Supervisors
Prof. Luca Sterpone
PhD. Salvatore Danzeca - CERN
PhD. Rudy Ferraro - CERN

Candidate
Antonio Scialdone

Academic Year 2020/2021

Abstract

At CERN, many electronic systems need to operate in a radiation environment along with the Large Hadron Collider (LHC). Thus, the Radiation Hardness Assurance (RHA) of electronic components is fundamental to ensure the reliable functioning of accelerators and experiments. Because of their benefits in terms of costs, flexibility, and performances, Field Programmable Gate Arrays (FPGAs) are often the core of several electronic systems. However, their lifetime and performances are affected by radiation-induced effects such as Single-Event Effects (SEE) and Total Ionizing Dose (TID). This creates the necessity to perform many qualification tests to find a suitable FPGA to use in a specific system or experiment. Moreover, the upcoming upgrade of the LHC, the High-Luminosity LHC (HL-LHC), will bring many improvements to the present experiment, but radiation levels will increase. Therefore, more robust FPGAs are necessary, making the qualification process even more important. The current qualification procedure consists of testing the FPGA's internal components individually, retrieving a general overview of the FPGA sensitivity to radiations. Then, the specific application is tested to estimate the failure rate of the device when working inside the LHC. This procedure implies performing a radiation test for each application we want to run, which is time-consuming and very expensive. In this work, we present a new approach based on benchmark circuits that solves this problematic. These benchmark circuits reflect the workload of an actual application, allowing to perform standardized application-level testing that facilitates the comparison of results between different FPGAs and different experiments. Most importantly, the procedure allows us to estimate the device failure rate independently from the application we plan to use, thus reducing the number of radiation tests we need to perform. The benchmark application we used belongs to the ITC99 Benchmark suite developed in the CAD group at Politecnico di Torino. In addition, we developed an FPGA-based tester architecture to carry out radiation tests. The tester platform is based on an SoC, with an embedded processor connected to an FPGA. We developed Intellectual Property (IP) cores in VHDL implementing the different test routines to test the FPGA under radiation, and we developed the high-level APIs for the embedded processor to communicate with such IPs. With this approach, we greatly simplified the entire radiation process. Using this platform, we performed various radiation campaigns against protons and neutrons, qualifying two FPGAs for the CERN applications: the NG-Medium (SRAM-based) and the PolarFire (FLASH-based). We demonstrated how, we can easily compare the two FPGAs, even if they are based on two different technologies, and we estimated their failure rate for twelve years of operations inside the HL-LHC.

Acknowledgments

I would like to thank Rudy Ferraro who guided me throughout the entire development of this thesis.

Mammà ca me guarda rice l'aggio fatt'
Mammà è ancora ambresse, ma c'a sto pe' fa

Contents

CHAPTER 1

Introduction

At the European Organization for Nuclear Research (CERN), particle accelerators are used to run experiments to better understand the nature of our universe by recreating the conditions that existed after the big bang. The Large Hadron Collider (LHC) is the largest and most powerful particle accelerator in the world. It consists of a 27-kilometre ring allowing to accelerate particles up to TeV before they collide. Many electronic systems supporting the accelerators need to operate in a radiation environment. Therefore the Radiation Hardness Assurance (RHA) is a critical process to ensure the reliable operation of the LHC. Moreover, the upcoming update to the LHC, the High-Luminosity LHC (HL-LHC), will bring many improvements to the experiments by increasing the energy at which the particles collide. Consequently, we expect the radiation levels to increase, making the qualification process even more critical. Engineers often choose field Programmable Gate Arrays (FPGA) among the electronic systems because of their costs and performances. FPGAs are integrated circuits based on a matrix of Configurable Logic Blocks (CLBs), connected via programmable interconnects, which connect the blocks to implement the desired user logic. Inside an FPGA, we can usually find, in addition to the CLBs, hard blocks (arithmetic blocks or memories) and clocking resource (Phase-Locked Loop). However, radiation-induced effects could modify the behaviour of these components, degrading the lifetime and the performance of the FPGA, which could lead to the failure of the experiment. Therefore, it is necessary to perform many qualification tests to understand whether the FPGA can withstand the radiation levels of the environment where we are planning to use it.

The actual state of the art concerning FPGAs' radiation qualification process consists of studying the sensitivity of all the FPGA elements separately, using an ad-hoc design for each of them. On the one hand, this approach is helpful to understand how sensitive these elements are to radiation. On the other hand, it does not give a realistic overview of how the FPGA will work when deployed in the experiment. This limitation makes it quite complex to estimate the failure rate of a system during the LHC operation. Thus, it is mandatory to perform additional tests using actual applications to understand how the system will operate. However, if one performs this kind of test using a project-specific circuit, it would be impossible to compare the results among the different applications. We would need to test under radiation all the applications we plan to use.

Another critical aspect is the internal degradation of the device itself due to extended exposure to radiation, which can lead to a complete failure of the system. Usually, we monitor these effects by analyzing the current consumption of the FPGA during the radiation test. However, this mechanism only works when the internal degradation is already in an advanced state, which drastically changes the current consumption.
Finally, concerning the test procedure itself, there are no actual Automatic test equipment (ATE) like in the electronic manufacturing industry to perform radiation testing of FPGAs. This lack of tester equipment implies that every time we need to carry out a radiation test, we also need to develop an interface to communicate with the FPGA under test. In such a context, it is clear that a more reliable and robust approach to study FPGAs is needed to resolve the problems above.

In this work, we propose different improvements to the state-of-the-art qualification methodology. We will present a new procedure based on benchmark circuits since they reflect better the workload of an actual application, essential for estimating the device's failure rate in operation. They allow performing a single test to retrieve the FPGA's failure rate, avoiding performing a separate test for each application. Moreover, they allow standardized tests to repeat on different families of FPGAs, making the comparison easy. The benchmark circuits employed for this procedure belong to the ITC99' Benchmark Suite, developed at Politecnico di Torino. It contains circuits of different sizes, including various functionalities, from Finite-State Machines (FSM) to soft-core microprocessors.
We will study the internal degradation of the FPGA using a custom circuit that consists of many ring oscillators measuring the propagation delay of the FPGA routing interconnection. This mechanism allows monitoring the device degradation while evolving rather than when the device is about to fail.
Finally, we will present a TESTER platform based on an FPGA to ease and speed up the radiation test process. We developed many peripherals implementing different test routines, and we combined them, creating a single platform that we can use to interface with other FPGAs under test.

To demonstrate the efficiency of the new methodology and the efficiency of the TESTER architecture, we will show the results we obtained in the qualification process of two FPGAs: the NG-Medium and the PolarFire. These FPGAs are possible candidates for the upcoming radiation environment of the HL-LHC upgrade. The NG-Medium is a Radiation Hardened by Design (RDHB) SRAM-based FPGA, produced by NanoXplore. The PolarFire is FLASH-based, built on the state-of-the-art SONOS 28nm non-volatile technology from Microsemi. Using two different families of FPGAs, we will demonstrate the real benefits of our work. To summarise, the main contributions of this project are:

1. Introduction of a new test procedure for the qualification of FPGA that need to operate in the CERN radiation environment;

2. Development of a TESTER platform, based on FPGA, to facilitate the entire test procedure;

3. Complete qualification of two FPGAs.

The rest of the dissertation is organized as in the following. Chapter 2 gives an overview of the physics behind the radiation effects and describes the consequences of the radiations on electronic devices. Moreover, it presents the actual radiation environment at CERN and the expected

radiation levels of the upcoming upgrade, the HL-LHC. Chapter 3 details the entire work and the main contributions, describing the new test methodology, the test circuits we implemented to extrapolate the information, and finally, the TESTER architecture. Chapter 4 describes the results of the irradiation campaigns we carried out to test the two FPGAs. Finally, Chapter 5 concludes the project, presenting the possible next steps for future work.

CHAPTER 2

Radiations fundamentals and LHC Mixed Radiation Environment

The necessity to qualify FPGA for a radiation environment arises from the many radiation-induced effects affecting these devices when exposed to such an environment. This chapter gives a complete overview of the physics behind the radiation effects and the problems originating in electronic devices, FPGAs in particular. The layout of the LHC is also presented, along with its one-of-a-kind radiation environment.

2.1 Interactions between particles and matter

The material properties may change when exposed to radiations because of the interactions between the striking particles and the material itself. As shown in Fig. 2.1, the impinging particle can trigger various processes depending on its nature and energy.

In the case of a photon, three different mechanisms can ionize an atom:

- **Photoelectric effect**: when a photon interacts with an electron of the same energy. All the energy of the photon is transferred to the electron, which is then ejected from the atom;
- **Compton effect**: when the photon has higher energy than the binding energy of the electron. The photon gets scattered away, but part of its energy is transferred to the electron, which is ejected from the atom;
- **Pair production effect**: when a high-energy (about 1 MeV) photon passes near the nucleus of an atom. In this case, the photon may lose all of its energy and produce an electron/positron pair.

Neutrons instead, which are electrically neutral particles, can have only nuclear interactions. These are:

- **Neutron absorption**: it occurs when the striking neutron, in particular thermal neutrons (neutrons with an energy of about 25 meV), is absorbed by the nucleus, exciting it. The nucleus leaves the excitation state by emitting one or more photons;

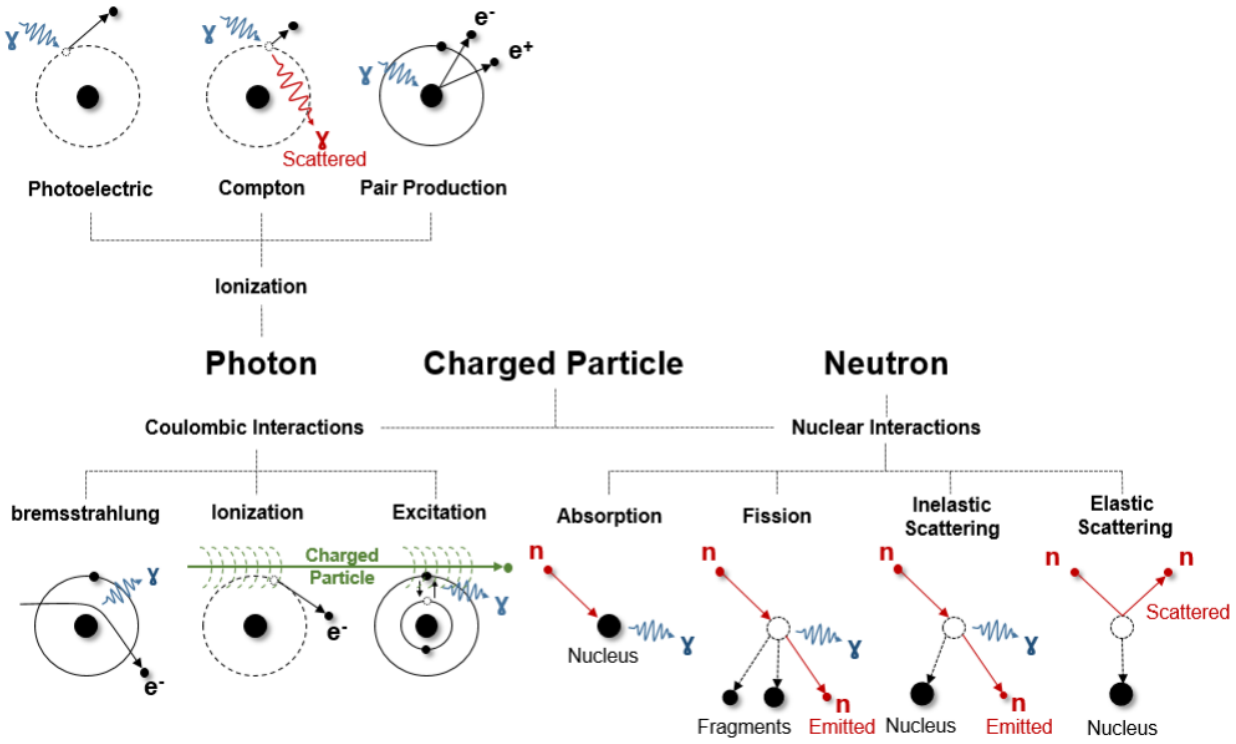


Figure 2.1: Particle interactions [11]

- **Elastic scattering**: which happens when the neutron is scattered after the collision with the nucleus, which in turn is also spread by the energy lost by the neutron;
- **Inelastic scattering**: It occurs when the incident neutron is absorbed by the recoiling nucleus, which emits a neutron with a lower kinetic energy that leaves in an excited state, and then emits one or more photons;
- **Fission reaction**: when a heavy nucleus splits into smaller ones, called fission fragments, after absorbing a neutron.

In addition to nuclear interactions, heavy charged particles are subject also to **Ionization** and **Excitation**, two Coulombian interactions. When a highly charged particle passes close to an orbital electron, if the energy is high enough, the electron might be ejected from its orbit (Ionization), or it may cause the electron to move between inner shells generating photons (Excitation).
Finally, poor energized particles are also subject to **bremsstrahlung**. When an electron passes close to the nucleus of an atom, the moving particle is deflected and loses energy which is converted to a photon.
When an atom is ionized through one of these interactions, and in case the interaction is generated directly by the striking particle, the process is called **direct ionization**. However, striking particles also generate secondary particles which in turn can trigger other interactions that may ionize the atom. This phenomena is called **indirect ionization**.

The consequence of these interactions is the loss of energy, which can be grouped into two main contributions:

- **Ionizing Energy Loss (IEL)**: includes the energy lost because of all the interactions causing ionization;
- **Non-Ionizing Energy Loss (NIEL)**: includes the energy lost due to all the nuclear interactions, causing Displacement Damage (DD).

The particles lose their energy until they are completely stopped. The rate of energy loss along the path is the **Linear Energy Transfer** (LET), described by Eq. (2.1).

$$LET = \frac{dE}{dx} \tag{2.1}$$

The LET is the average energy deposited per unit of length. Normalizing this quantity to the material density ρ, we obtained the **mass stopping power** (S), described in Eq. (2.2).

$$S = \frac{1}{\rho}\frac{dE}{dx} \tag{2.2}$$

The rate of energy loss dE/dx includes both the IEL and NIEL contributions. However, for energetic charged particles, the IEL dominates. Moreover, since MOSFETs are less affected by NIEL, only the IEL and its effects will be further analysed. Figure 2.2 shows an example of the LET of hydrogen in silicon. We extracted the data using the SRIM tool [1].

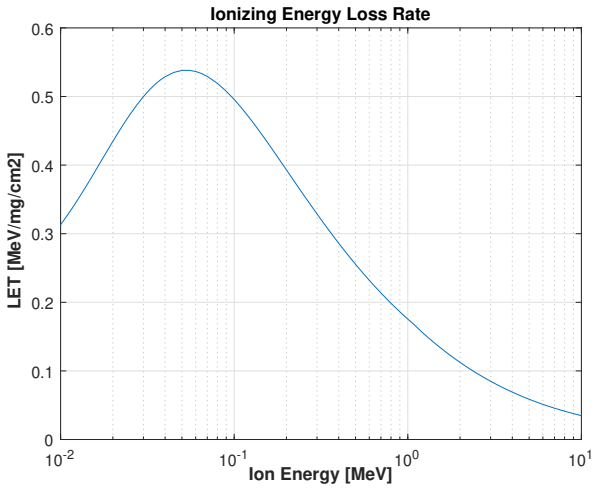


Figure 2.2: Electronic LET of hydrogen in silicon

The LET value depends on the nature of the particle and the material that it is traversing. Most of the energy is deposited when the particle is almost at the end of its path. Such a point is called **Bragg Peak**.
The energy lost by the particle and absorbed by the material is defined as the **Total Ionizing Dose** (TID) and its unit of measure is the Gray (Gy). It corresponds to one joule of energy absorbed by a kilogram of matter.
The rate of incident particles on a material is called **Flux**, and its integral over a specified period of time is the **Fluence** $\Phi(E)$ [p/cm^2]. Therefore, considering the LET and the fluence of a charged

particle, we define the Total Ionising Dose absorbed by the material using Eq. (2.3), where K is the unit conversion parameter whose value is equal to $1.6 \cdot 10^{-7}$ Gy·g·MeV^{-1}.

$$TID = K \int \frac{LET}{\rho} \Phi(\rho, E) dE \qquad \text{[Gy]} \tag{2.3}$$

2.2 Radiation effects on electronics

Independently from the type of interaction between particles and the material, two categories of effects alter the behaviour of electronic devices exposed in radiation environments:

- **Cumulative effects**: they take place during the whole lifetime of the device. As long as the device is exposed, damages induced by each particle lead to a gradual internal degradation of the device, for example, current consumption increase. The fluence accumulated will keep increasing until the device reaches its limit and stops working once the accumulated damages are too many. These effects are the consequences of DD and TID. Since the FPGAs are not subject to DD because they are based on MOSFET, we will discuss only TID effects;
- **Single Event Effects**: a single ionized particle strikes the device's sensitive region modifying its behaviour.

2.2.1 Total Ionizing Dose (TID) effects

Total Ionizing Dose Effects are a topology of cumulative effects caused by the ionization of the insulation material used in Integrated Circuit technology. In the case of MOSFETs, it is the silicon dioxide films. When a transistor is exposed to ionizing radiations, electron-hole pairs (e-h) are created in the oxide, which induce the buildup of charges, causing the degradation of the device [20].
To understand the impact of these effects on the behaviour of the device, let's consider an n-channel MOS device with a positive gate bias. Thus, an electric field is created along with the gate oxide, and an inversion layer is formed at the Si - SiO_2 interface. Figure 2.3 illustrates the response of such device to ionizing radiation.

We can identify four main processes:

1. Depending on the energy of the striking particle, electron-hole (e-h) pairs are generated within the silicon (Si) and dioxide dielectric layer (SiO_2). Table 2.1 lists the energy and the corresponding pair generation rate for these two materials.
   When electron-hole pairs are created, most of the electrons will flow toward the gate, whereas holes will flow toward the Si/SiO_2 interface. Before the electrons leave the oxide, some of them will recombine with holes, while others will escape. The fraction of electron-hole pairs escaping from the recombination constitutes the **charge yield**;
2. The holes which escape this initial recombination will move towards the interface by hopping through localized states in the oxide;
3. As the holes approach the interface, some of them will be trapped in the oxide forming a positive oxide-trap charge;

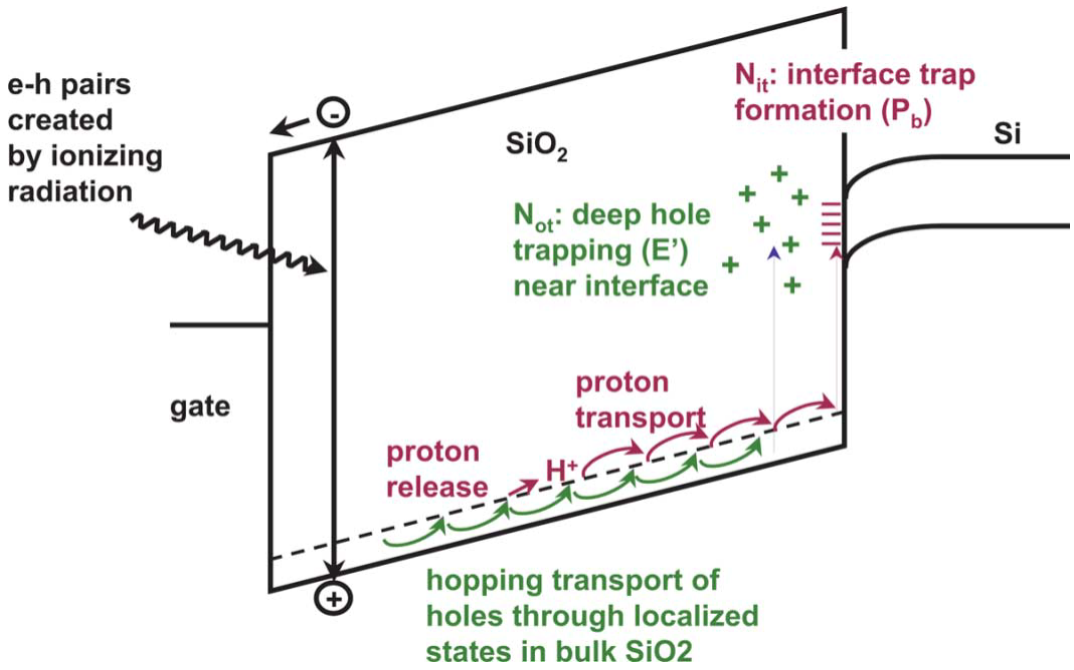


Figure 2.3: Radiation-induced charge generation for a MOS capacitor.

4. As the holes hop through the oxide, protons are released, which can also drift towards the interface. Here, they may react to form interface traps. These are positively charged for p-channel transistors and negatively charged for n-channel transistors.

Material	**Ionization Energy [eV]**	**Pair generation rate [e-h pairs · Gy^{-1}·cm^{-3}]**
Silicon (Si)	3.6	4×10^{15}
Silicon Dioxide (SiO2)	18	8.2×10^{14}

Table 2.1: e-h pair generation rate for silicon and silicon dioxide

The accumulation of oxide-trapped and interface-trapped charges make the threshold voltage of the device shift:

$$\Delta V_{TH} = -q\frac{1}{C_{ox}}\Delta N = V_{OT} \pm -q\frac{1}{C_{ox}}\Delta N = V_{IT} \tag{2.4}$$

In Eq. (2.4), ΔV_{OT} and ΔV_{IT} are the oxide-trapped and interface-trapped charge respectively, while C_{ox} is the oxide capacitance. The sign of the second term depends on the MOSFET type, negative for p-channel and positive for n-channel. Therefore, we have the following behaviours for PMOS and NMOS transistors:

- **PMOS transistor**: the buildup of positive charge will require a higher voltage to be applied to the gate to create an inversion channel. Thus, the absolute value of the threshold voltage is increased, and it will be harder to switch on;

- **NMOS transistor**: the buildup of charge makes it easier to create an inversion layer of electrons, resulting in a lower threshold voltage value, so it will be easier to switch it on.

Eventually, the threshold voltage can become negative for NMOS transistors, which will be permanently on, and can become higher than the voltage applied to the circuit in the case of PMOS transistor, which will be always off.

2.2.2 Single Event Effect (SEE)

When high energy particles strike a sensitive region of an electronic circuit it might produce a Single Event Effect (SEE) . Depending on various factors, this may cause no observable effect, a transient misbehaviour, a change of logic state, or permanent damage.
A particle passing through a material deposites part of its energy with direct or indirect ionization. In both cases, the charge deposited by a striking particle can then be collected inside the reverse-biased p-n junctions by the present electric field through drift processes, causing a transient current at the junction contact and the alteration of the electrostatic potential, called field funnel. Eventually, the funnelling effect can increase even more the collection of charges at the struck node. This charge collection could cause a Single Event Effect depending on a variety of parameters, i.e.affected technology. Still, generally, they can be classified as non-destructive [9], which can be usually removed by resetting the device or applying the correct signal, and destructive SEE [22], which may permanently damage the device. The destructive SEEs are:

- **Single Event Latch-up - (SEL)**: it is a potentially destructive event that occurs when a low-resistance path between the power supply and ground of a device is created. This event can be triggered in any parasitic structure, i.e the p-n-p-n in CMOS technology. In this structure, an increase in the PNP collector current gives an increase in the NPN base current. This in turn increases the NPN collector current, which then increases the PNP base current. Therefore, if a particle strikes the device turning on the NPN or PNP transistor, the latch-up is triggered, and the positive feedback will increase the current until power is removed from the device or it fails catastrophically;

- **Single-Event Burnout - (SEB)**: it occurs when the passage of a heavy-ion causes the MOSFET to enter a second breakdown that, if not removed, induces a high-current leading the device into a thermal runaway until the failure;

- **Single-Event Gate Rupture - (SEGR)** : it occurs when the passage of a heavy-ion through the neck region of the MOSFET creates a conducting path in the gate oxide. The charges created propagate up to the insulator interface, making the electric field across the dielectric very large. If this exceeds a certain value, a gate rupture can occur leading the device to fail.

The non-destructive SEEs are:

- **Single Event Upset - (SEU)**: when the energy deposited by the striking particle is enough to flip the logic state of a storage element, i.e. memory cell; in this case, the value stored will be wrong until the storage element is rewritten;

- **Multiple Event Upset**: because of the lateral diffusion, the charge deposited by the striking particle leads to the corruption of storage elements close to each other. It is also referred to as **Multiple Bit Upset - (MBU)**;

- **Single Event Transient - (SET)**: it is a temporary voltage spike that propagates in the system along with the other signals. Its effects are not always visible since they may be masked by other signals.

- **Single Event Functional Interrupt - SEFI**: it is caused by a single ion striking particle that brings the system into a temporary failing state. It does not damage the device, however, the device stays in the failing state until a reset is sent to resume the normal operation.

The experimental results we present in Chapter 4 concern non-destructive SEE, thus we provide a further description of these effects and their mechanism.

Single Event Upset - SEU

We can illustrate the mechanism leading to an SEU (MBU in case of multiple bits) using the typical structure of a CMOS memory cell inside an IC, as shown in Fig. 2.4. The memory cell is composed of two CMOS inverter. In stable conditions, it stores either a '0' or a '1', depending on which couple of transistors is switched on. The reverse-biased drain junction of the two 'off' transistors constitutes the sensitive location. When an energy particle strikes this region, the collected charge results in a transient current in the affected transistor. In Fig. 2.4, the "n-channel" transistor is struck, thus a transient current is generated because of the charge collected. The restoring transistor (on, p-channel) tries to balance the particle-induced current. However, the restoring transistor has a finite amount of current drive. The current flowing induces a voltage drop at the drain of the p-channel transistor, which in combination with the transient current may cause the state of the transistors to reverse. Eventually, the other inverter will toggle, and the stored value will change.

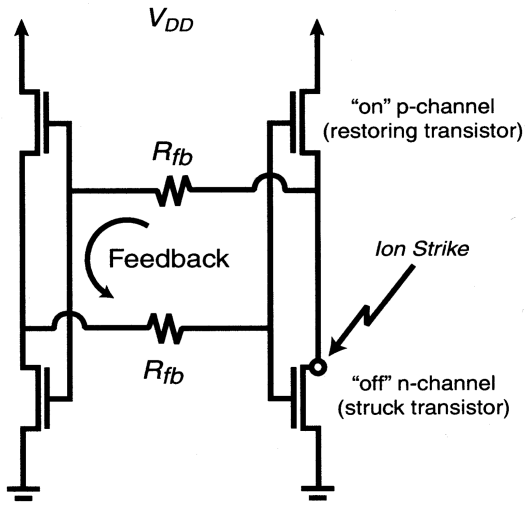


Figure 2.4: SRAM memory cell affected by a Single Event Upset

Single Event Transient - SET

When an energy particle strikes the sensitive region of a transistor of combinatorial logic, it may generate a voltage disturbance on that node. However, whether or not this upset is visible depends on different things: the existence of a path between the combinatorial logic and the subsequent latch/register; the time necessary to reach the storage element; the clock pulse at the input of the storage element. Since this voltage disturbance is transmitted together with the other signals in the circuit, it may be masked by the logic function implemented. On the other hand, if the signal propagates to the Flip-Flop data input, it could result in a wrong value stored inside the Flip-Flop, causing the same effects as if it were a true SEU.

2.2.3 Overview of mitigation techniques for radiation effects

Single-Event Upset and Single Event Transient are classified as non-destructive SEEs because their effect can be cancelled by simply rewriting the concerned storage element with the correct value. However, when considering a more extensive system/application, the SEE can lead to a wrong computation, driving the entire system to fail. For this reason, designers adopt mitigation techniques to mask these effects. We briefly report hereafter the most common approaches adopted for FPGAs to reduce the radiation-induced problems.

Triple Module Redundancy

The most common technique for SEU mitigation is **Triple-Module Redundancy** (TMR). Figure 2.5 shows its schematic implementation. It uses three identical blocks, receiving the same input and implementing the same function. Their output is not passed directly to the following block, instead, they are sent to a voting system that chooses the correct output based on the majority. Thus, in case one of the blocks is affected by an SEU, the voter will still produce the correct output to pass to the next block.

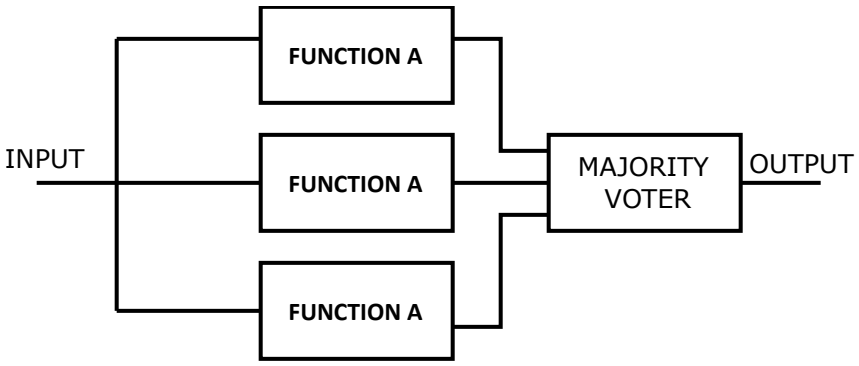


Figure 2.5: Triple Module Redundancy (TMR) implementation schema

The concept of TMR can be applied at different levels of the design process, depending on the resources available and the safety requirements. In the case of FPGAs, for example, the strategy could be to apply TMR to the entire design. In this case, the FPGA will contain three identical circuits, therefore this is suitable only in case the FPGA is big enough. In case it is not, partial TMR could be applied. This implies that only some parts of the circuit, usually sequential logic, are

triplicated. Finally, it can also be applied at system level. In this case, three FPGAs implementing the same design are employed, and their output is chosen using an additional hardware voting system.

Memory Scrubbing

The term scrubbing refers to a category of techniques that consist of restoring the content of the memory with the correct value to prevent the accumulation of errors. When an SEU happens inside a storage element, the following write operation cancels the effect. SRAM-based FPGAs use external SRAM memories to store their configuration. If an SEU affects this memory, the user needs to load again the design to rewrite the entire memory and thus remove the SEU. However, an SEU does not always correspond to the instantaneous failure of the design. For example, if the affected bit inside the memory corresponds to a portion of the FPGA where there is no user logic. However, the SEUs will accumulate until the circuit fails. In this kind of situation, memory scrubbing is particularly useful to prevent error build-up and avoid the system's failure. To implement memory scrubbing, one can store the original configuration in an additional memory and use it as a reference. An external configuration manager, the scrubber, can continuously read the memory, compare it with the reference, and repair it if there are differences.

SET mitigation

The particle charge deposited in the material may generate a transient pulse in combinatorial logic. Then, the pulse must propagate through the logic data path and be sampled into a storage element like a Flip-Flop to manifest as an error. Therefore, for this kind of SEE, the width of the pulse is crucial. Because of technology shrink, this kind of event is going to be a forthcoming issue in digital technology, which engineers need to assess, and therefore different mitigation techniques have been developed [18] for this purpose. The idea behind SET mitigation consists of filtering the generated transient pulse by using an additional *guard gate* (GG) placed at the input of the storage element. The GG will let pass just transient whose pulse width exceeds a specific value. All the other transient are filtered and will not reach the input of the storage element.

2.3 LHC mixed radiation field

CERN is the largest particle physics laboratory in the world. It was built in 1954 to research and study new particles to understand our universe's laws better. Since the foundation, the organization built different accelerators to increase the energy of particle beams. Every accelerator uses those already built as injectors and can send the beam to the next stage or experiments. Figure 2.6 shows the actual structure composed of six accelerators.
The very first circular accelerator is the Proton Synchroton Booster (PSB), whose purpose is to accelerate particles coming from the linear accelerator, LINAC 2, from 50 MeV up to 1.4 GeV.
The second stage of the chain is the Proton Synchrotron (PS). Before the construction of PSB, PS was receiving particles from LINAC 1. Now it can receive either the 1.4 GeV protons from PSB, or the 72 MeV ions delivered by the Low Energy Ion Ring (LEIR) and LINAC 3. PS can boost particles up to 24 GeV and then send them to the next accelerator stage, to the Antiproton

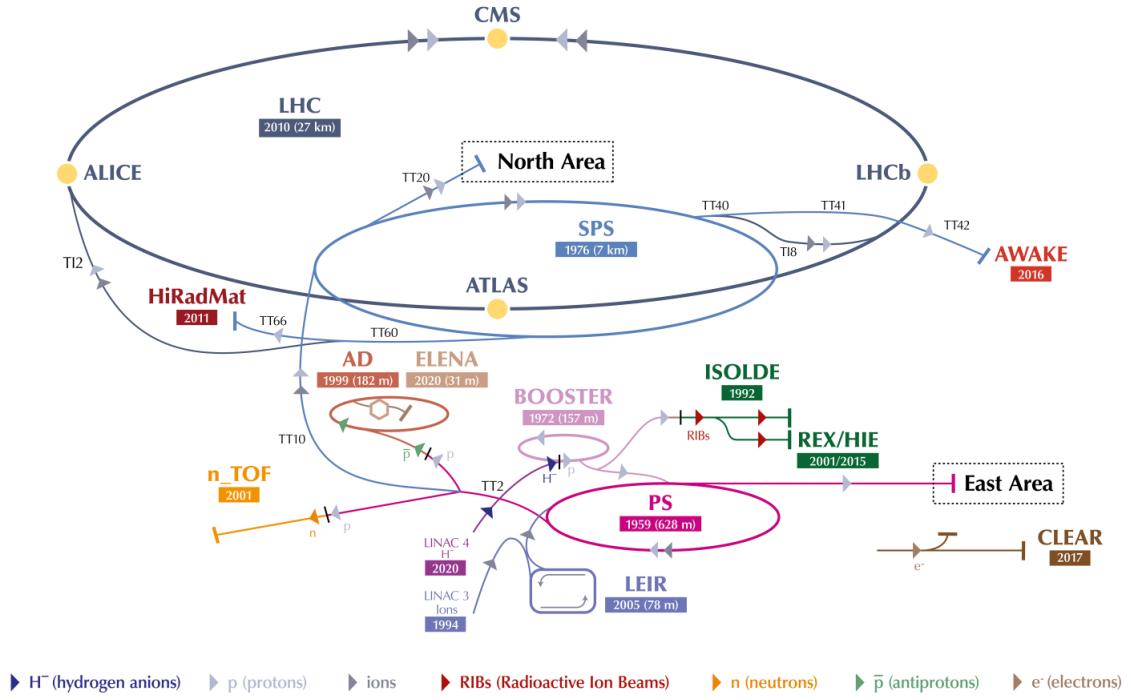


Figure 2.6: The CERN accelerator complex

Decelerator (AD), or it can be extracted for experiments (CHARM/IRRAD, nTOF).
The third stage is Super Proton Synchroton (SPS), a 7 km circular accelerator able to accelerate particles up to 450 GeV. The beam can then be sent to experiments (HiRadMat, AWAKE, CENF), or the next stage, that is the Large Hadron Collider (LHC).

2.3.1 The Large Hadron Collider

The LHC is, up to now, the biggest accelerator in the world. Its goal is to accelerate particles in two opposite directions up to an energy of 7 TeV and make them collide at 14 TeV in one of the four collision points: ATLAS, CMS, LHCb and ALICE. The two-particle beams are guided through the channels by a magnetic field generated by superconducting magnets. These are made of unique materials able to operate in a superconductive state when cooled down to -271.3 °C, allowing to conduct electricity without resistance or energy loss. Most of the machine is connected to a liquid helium distribution system to cool down the magnets.
The main research domain is discovering and researching new particles to understand which theories in the current scientific model are most likely to be correct. One of the main objectives was to confirm the existence of the Higgs boson, predicted by the standard model and finally discovered in 2012 [2] by two LHC experiments, ATLAS and CMS.

Layout

Figure 2.7 shows the layout of the LHC. It is divided into eight arcs and eight Insertion Regions (IR). Four IRs are used for experiments. IR1 for the ATLAS, IR2 for ALICE, IR5 for CMS and IR8

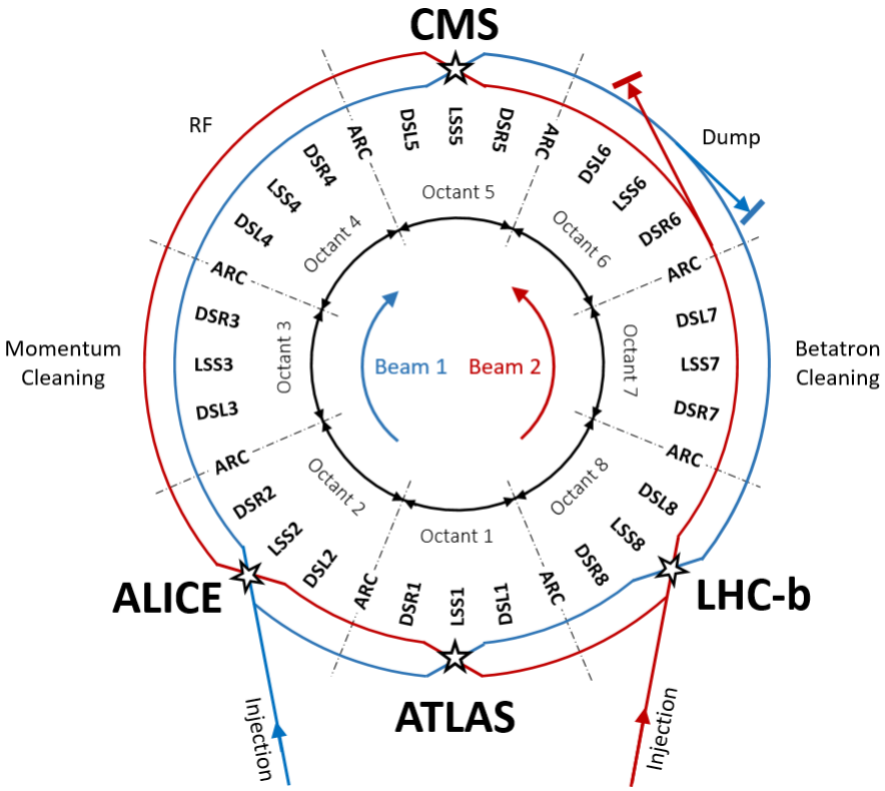


Figure 2.7: LHC layout

for LHCb. The remaining four are dedicated to specific operation: acceleration (IR4), collimation (IR3 and IR7) and dump (IR6). Each IR, whose layout is in Fig. 2.8, can be divided into two

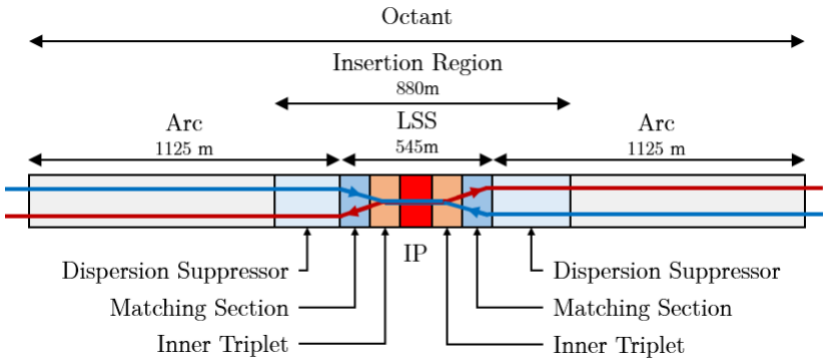


Figure 2.8: Layout of an LHC octant

parts: the Dispersion Suppressor (DS) and the Long Straight Section (LSS). The DS is a junction between the arc and the LSS, and it is necessary to reduce the beam dispersion before it enters the LSS, where the beam focalizes before colliding in the Insertion Point (IP).

Operation phases

Since the product of collisions is extremely important for particle physics, it is important to have as many interactions as possible. This is especially true when rare events with a small production cross-section σ_p are studied. The quantity that measures the ability of a particle accelerator to produce the required number of interactions is called **luminosity**, defined in Eq. (2.5) . It is the proportionality factor between the number of events per second *dR/dt* and the cross-section of the event σ_p. In the case of the LHC, it is in the order of 10^{34}.

$$\frac{dR}{dt} = \mathcal{L} \cdot \sigma_p \tag{2.5}$$

The number of useful collisions (events) that the accelerator can produce over a defined period of time is the integrated luminosity $\mathcal{L}_{int}$, defined by Eq. (2.6). It is expressed in femtobarn (fb^{-1}) and corresponds to $\sim 10^{14}$ proton-proton interactions.

$$\mathcal{L}_{int} = \int_0^T \mathcal{L}(t')dt' \tag{2.6}$$

Therefore, one can improve the efficiency of the LHC by increasing either the instant luminosity, as planned in the HL-LHC upgrade or the operation time. Three phases define the operation time of the LHC:

1. **Fill**: it consists of filling the two LHC pipes through the injections of protons by SPS;
2. **Ramp**: the two particles beam are accelerated up to 7 TeV;
3. **Stable beam**: in this phase, the two beams can collide in the different experiments.

However, during the stable beam phase, the instantaneous luminosity gradually decreases because of losses in collisions or other locations. When it becomes too low, a beam dump occurs, and the accelerator must go back to the *fill* phase to restart the collisions. Usually, the stable beam operation lasts around 15 hours, but it lasts 5 hours on average because of system failures. When a beam dump occurs, the system that failed needs to be repaired, reducing the accelerator's available time. Therefore, it is clear that the number of dumps must be kept as low as possible to increase the effective time of operation of the LHC.

Impact of radiations on LHC operation

Radiation-induced effects are among one of the primary sources of system failures causing the dump of the beam in the LHC. In 2011, 70 beam dumps happened, causing downtime of 400 hours. At the very beginning, when the radiation levels were unknown, the measures adopted in case of radiation-induced failures consisted of moving the components in the shielded area or replacing them with more robust parts. When engineers introduced the radiation monitoring system, it became possible to understand before the experiment where to put shielding or radiation-tolerant systems—these precautions allowed to reach 0.5 dumps per fb^{-1} in 2017. However, with the approach of the LHC upgrade, the HL-LHC, the expected integrated luminosity is 3000 fb^{-1} for ten years of operations. Therefore, the instantaneous luminosity is reduced below its maximum

performance to achieve a collision rate that the detectors can still process. However, reducing the instantaneous luminosity means that the desired integrated luminosity is reached only after a specific stable beam time. Engineers estimated that, in these conditions, to achieve the desired luminosity, only 0.1 beam dumps per fb^{-1} are tolerable. This estimation, combined with the higher radiation expected, makes the qualification process of electronics more important before deploying systems in the experiments.

2.3.2 The LHC radiation levels

When evaluating the use of an FPGA in a radiation area, an estimation of the SEE rate that could affect the device is necessary to accept or reject the candidate. Thus, we need to know the radiation environment where we are working. Hereafter, we present and analyze the LHC mixed radiation field, its radiation levels, and the expected levels of the HL-LHC upgrade to quantify the impact in terms of SEE effects.

Inside the LHC, we can identify three primary radiation sources:

- Proton-proton collision in the IPs generates a shower of secondary particles ranging from thermal energies up to GeV. Near the IPs, photons and pions are dominating the spectrum. While detectors intercept most of them, a small part of particles is still emitted and reaches the elements around the accelerators. In these areas, the spectrum of particles depends on the distance from the interaction point. Since it is proportional to the luminosity of the machine, it is said to be a luminosity-driven radiation source;

- Direct losses between the beam halo particles and the collimators placed around the experiments initiating particle showers. These losses correlate with the beam intensity, and so it is an intensity driven radiation source;

- Interaction between the beam and the residual gas density in the vacuum pipes of the ARCs. In this case, the intensity is related to the beam intensity and the quality of the pipes.

The difference between intensity-driven and luminosity-driven radiation sources is that the first is emitted during each operation cycle, whereas the latter emits only during the stable beam phase.

The radiation environment inside the CERN accelerators comprises different particles over a large spectrum of energies, from meVs up to GeV, whose distribution may vary significantly depending on the location. When exposed to these environments, TID and SEE effects have a substantial impact on FPGAs performances. The two main contributions to SEE inside the LHC are High Energy Hadrons (HEH) and Thermal neutrons (ThN). HEH are defined as all hadrons with kinetic energy above 20 MeV. In [12], the different areas inside the LHC and the corresponding radiation levels of the upcoming HL-LHC are analysed in detail. They are generally estimated using the FLUKA Monte Carlo code [3]. We can divide the LHC areas where to install COTS component into two main groups according to their radiation levels: the tunnel areas and the shielded areas. These last, like UJ (junction chamber) and UL (liaison gallery) located near the Interaction Points (IP), are heavily shielded, whereas RR, located between the IR (insertion region) and Dispersion Suppressors (DS), is lightly shielded. In the heavily shielded areas, the thermal neutron contribution is much higher because the shielding thermalizes high-energy neutrons while it attenuates high-energy particles. FPGAs are less sensitive to thermal neutrons than HEH. However, different

studies, like [25], [23], and [28], show that the thermal neutron contribution can affect a lot the sensitivity of recent technologies. Thermal neutrons can also have a non-negligible contribution to the total failure rate, adding more complexity to the Radiation Hardness Assurance [19] procedure requiring assessing the ThN sensitivity of FPGAs with smaller node technologies. Therefore, the radiation assurance team introduced a Risk-factor R_{th}, defined in Eq. (2.7) to identify locations with higher thermal neutron contribution.

$$R_{th} = \frac{\phi_{ThN}}{\phi_{HEH}} \tag{2.7}$$

To conclude, in Table 2.2, we summarise the expected annual radiation levels of the upcoming HL-LHC for both HEH and ThN as a function of the location. In the next chapter, we will use these values to estimate the failure rate of the FPGAs we qualified.

Location	**TID (Gy)**	**HEH Fluence (p/cm^2)**	**R_{th}**
UJ	10	5×10^9	48.6
UL	0.2	10^8	24.7
RR	6	3×10^9	0.77
DS	100	5×10^{10}	3.5
ARC	2	10^9	2.7

Table 2.2: Average expected radiation levels in the LHC areas for the HL-LHC upgrade

CHAPTER 3

FPGA qualification procedure and tester platform

We can identify three significant steps in the qualification process of an FPGA:

- Test topology identification;
- Circuit design;
- Data analysis

Once we identify the topology of the test to perform (1), i.e. characterize the sensitivity of a specific element of the FPGA against SEE, the corresponding circuit can be designed (2). In this phase, it is essential to design the circuit itself and an interface allowing communication with an external device, i.e. a host computer, to save the data produced by the circuit. After the irradiation, we can analyze the data retrieved to estimate the device degradation, damage, and error rate (3). Among these steps, the correct design of the circuit under test is crucial for successfully characterizing the device. Thus, NASA established some guidelines to help in the development process. Starting from these suggestions, we developed the basic circuits to perform the standard FPGA qualification, and we extended the process with our contribution. This chapter gives an overview of the general guidelines to follow when performing FPGA irradiation tests. We will explain the limitations and problems of this process, and we will describe the new solution we introduced for the FPGA qualification at CERN. Finally, we will describe the FPGA-based test platform we developed and its advantages regarding the other interface types.

3.1 Standard FPGA characterization

FPGAs are integrated circuits based on a matrix of Configurable Logic Blocks (CLBs), connected via programmable interconnects, which connect the blocks to implement the desired user logic. Inside an FPGA, we can usually find three types of components:

- Configurable Logic Blocks (CLBs): they consist of few logical cells. Each cell usually contains Look-Up-Tables (LUTs), Full Adders (FAs), and Flip-Flops (FFs). These are the essential elements that wired together implement the desired logic function;
- Hard blocks: these include a variety of components like Digital Signal Processing (DSPs), multipliers, high-speed I/O blocks, and embedded memories. These are usually built inside FPGAs using transistors instead of LUTs in order to offer more functionalities with high performances and to save the space that CLBs would use to implement such functionalities;
- Clocking resources: these include analog Phase-Locked loops (PLLs) and Delay-Locked loops (DLLs). They allow the synthesis of new clock frequencies and attenuation of the jitter.

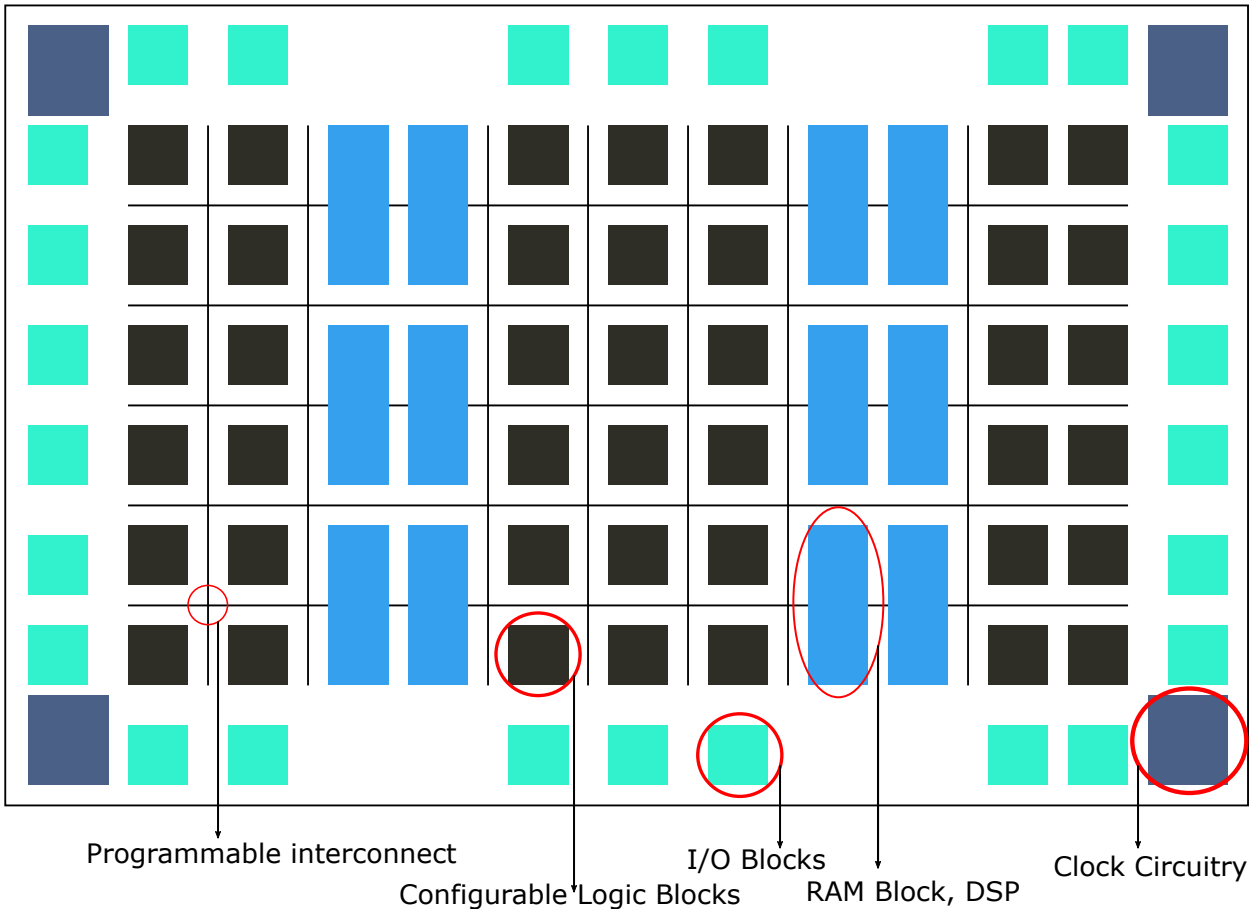


Figure 3.1: FPGA internal structure

Figure 3.1 shows the general schematic of an FPGA, including the elements mentioned above. Every one of these elements is susceptible to SEE, and they have a unique contribution to the overall failure rate of the device. Thus, the first step to accept a possible candidate for CERN application is to study the sensitivity of all these elements. Each category should have its specific test circuit to assist in the evaluation of its sensitivity. The NASA Goddard Radiation Effects and Analysis Group

(REAG) has developed a robust test and analysis methodology for evaluating FPGA sensitivity [5].

Hereafter, we report the salient point concerning the development of the test circuit for the DUT:

- It should use the same components of actual designs;
- It should avoid logic masking (i.e. when a change in a logic gate is not visible because the following gates mask it) to allow for functional visibility so that the user can identify and record all the possible upsets;
- It must allow obtaining many statistics to have good characterization. Thus, it should contain many instances of the same component;
- It should follow the synchronous design methodology used in an actual application to reflect better a real situation;
- It should allow the user to perform various type of tests, for example, dynamic, where the inputs change, and static, where they do not.

Based on the guidelines above, we built specific circuits to estimate the sensitivity of each FPGA element. In the following, we describe their implementation, giving an overview of their schematic and their functionalities. We implemented all the circuits using VHDL.

3.1.1 Flip-Flops

To retrieve the Flip-Flops (FFs) sensitivity, a shift-register (SR) would be a reasonable test structure. It could contain many elements, allowing to increase the statistics, and it is straightforward without any logic masking. However, this structure does not allow high-speed testing since the chain's output would switch at high frequencies, causing signal integrity issues that would prevent the tester interface from observing the error. To solve this issue, one could perform the test at a lower speed or using only a static input, but this will not accurately characterize the device. For these reasons, we used an improved version called **Windowed-Shift-Register (WSR)**, presented in Fig. 3.2. The structure consists of an SR of a given length, with a serial-to-parallel output window of

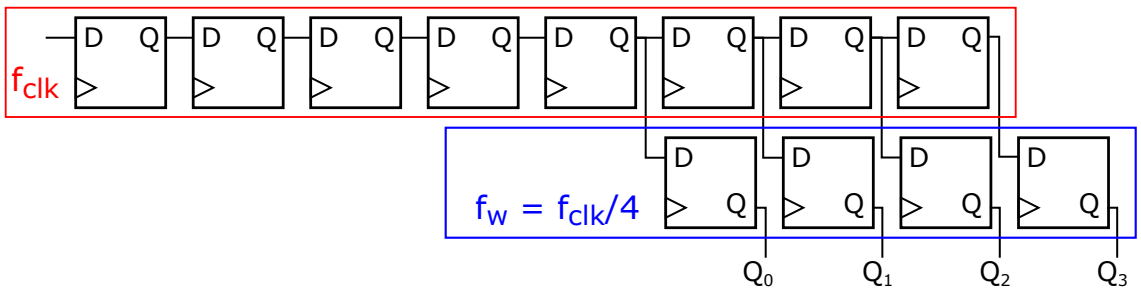


Figure 3.2: Windowed-Shift Register (WSR)

size N that captures the content of the last N Flip-Flops of the SR. A clock drives all the FFs with a nominal frequency f_{clk}. A load signal with a frequency $f_w = f_{clk}/N$ enables the Flip-Flops of the

output window instead. Therefore, if the system is working correctly, the output of the window will remain static which allows performing the test at higher frequencies without any signal integrity issue.

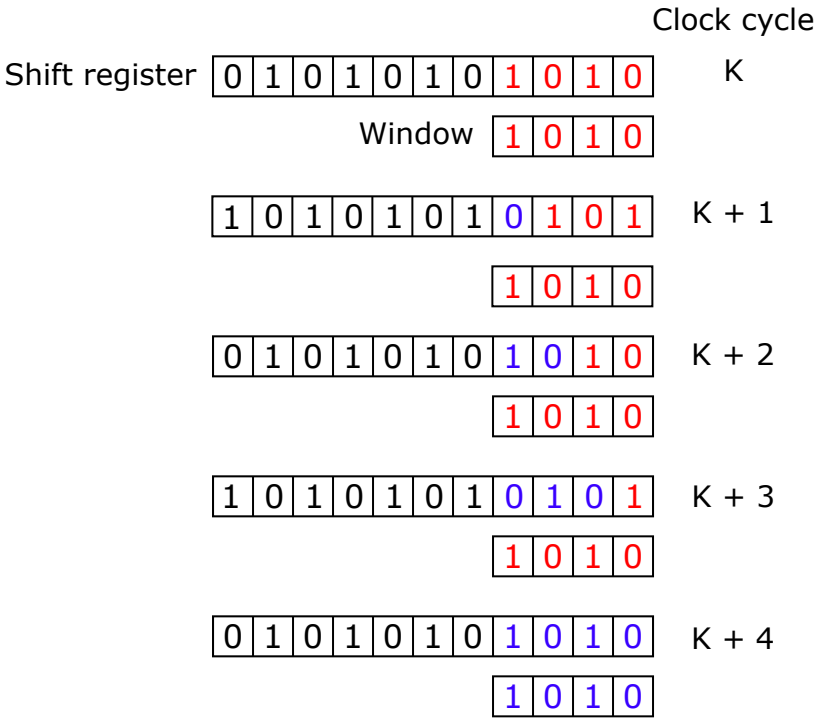


Figure 3.3: WSR working principle

Figure 3.3 shows an example of an SR chain of length 11 and a window of size 4. The window loads the value of the last four flip-flops at clock cycle K. For the following clock cycles, the bits shift inside the SR (blue). The output of the SR switches, but the window stays static. After four clock cycles, at K+4, the window loads the new values, which in normal condition are the same as those previously stored. Thus its content does not change. In this structure, the tester only needs to check that the window's content is always equal to a fixed pattern. If not, it means that an SEU affected one of the FF. By counting the number of events happening during the irradiation, we can retrieve an estimation of the Flip-Flop sensitivity against SEU.

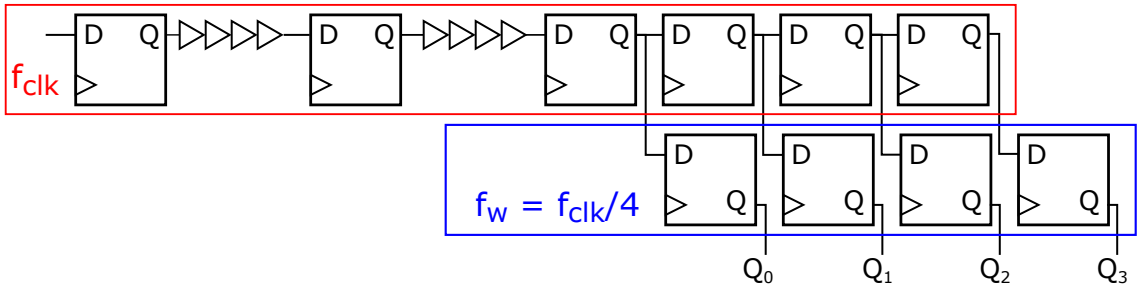


Figure 3.4: Windowed-Shift Register (WSR) adapted for SET measurement

A modified version allows estimating the SET sensitivity. Section 2.2.2 describes the SET effect in details. An SEU may be the result of a transient in the combinatorial logic that propagated

to the input of the FF, and that was long enough for the Flip-Flop to capture it. Figure 3.4 shows the modified version of the WSR, allowing the measurement of the SET sensitivity. We added between each FF of the SR a chain of NOT gates. If a SET pulse affects the combinatorial gates, and it is long enough to be sampled by the FF, it will become an SEU. By comparing the number of SEUs retrieved with the normal WSR to the number obtained with this structure, we can estimate the SET sensitivity.

When performing the characterization, it is helpful to understand how much a mitigation technique can improve the device's response. Thus, we applied TMR (Section 2.2.3) on both structures by triplicating each FF and adding a finale voter.

To sum up, we prepared four test structure to assess the FF's sensitivity, all based on the Windowed Shift Register (WSR) concept:

- WSR: Basic WSR for SEU estimation;
- WSR-SET: WSR with combinatorial logic between FFs for SET estimation;
- WSR-TMR: Basic WSR with TMR to estimate the advantages of the TMR mitigation technique;
- WSR-SET-TMR: The WSR-SET with TMR.

All the structures require at least one bit for the input and several output pins depending on the window size, limiting the number of structures instantiated inside the device under radiation. Since it is fundamental to obtain as many statistics as possible from a radiation run, we modified their interface to reduce the amount of I/Os required to place more structure. We connected each window to a comparator that checks whether the window's content equals a fixed pattern. This change reduces the number of output pins for each structure to one. Additionally, we implemented the possibility to perform both static and dynamic test. In the case of static test, we can fix the input either to 0 or 1, which means the window's content would be 0x0 and 0xF, respectively. In the dynamic test case, the input switches between 0 and 1, with the window's content fixed to 0xA or 0x5. In total, there are two pins for the input interface. Figure 3.5 shows the top-level view of

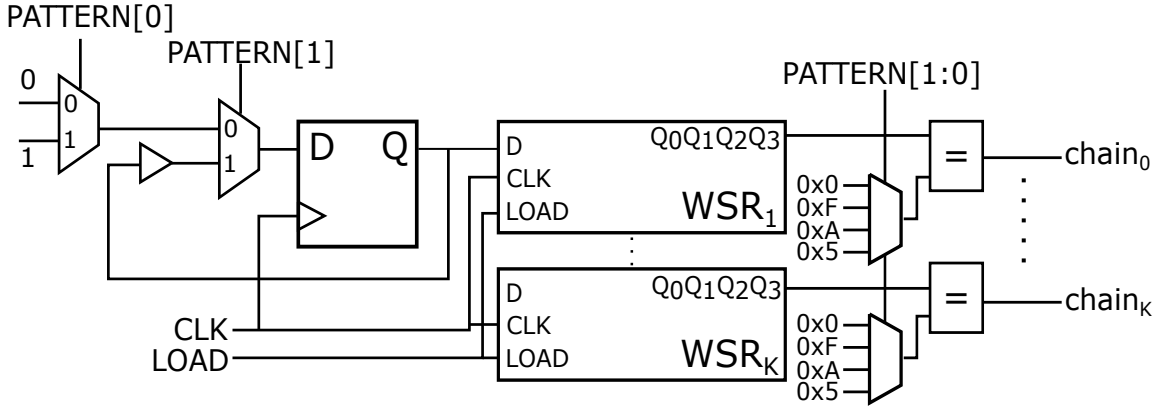


Figure 3.5: Circuit for Flip-Flop characterization

the circuit we used for the irradiation. Even if it is not shown in the picture, we triplicated all the

additional logic that is not part of the structures themselves to reduce the probability of retrieving error from those parts. As we can see, we have the CLK, LOAD and the Q output of the FF driving all the structures WSR_K. Each structure can be one of the structure we described above. Using the PATTERN signal we can choose which value we want to shift inside the chain, the type of test, and automatically change the golden reference. Each WSR_K has the associated $chain_K$ output signal that we can read to know if an error affected the chain.

3.1.2 DSPs

DSPs are those functional blocks that implement arithmetic operations like multiplication, addition, and a combination of them like multiply-add or multiply and accumulate. To retrieve the SEU sensitivity of these blocks, we need to perform an operation and then check that the output is correct by comparing it with a golden reference. In case it is not, an SEU occurred. Since these blocks operate on operands of multiple bits and we need to place many of them for good characterization, we adopted the following structure. We created different clusters, each one containing DSPs performing one operation. Each DSP has its comparator checking that the output equals the expected result. In case it is not, the comparator outputs a zero. Finally, an AND gate connects the comparators in a cluster. Thus, we used this signal as an active-low error detector. If an SEU affects one or more DSP in the cluster, the corresponding AND gate will be zero, signalling the error. Moreover, the device under irradiation contains both the operands and the expected output of the operation. To reduce the probability of an SEU occurring on them, we stored the operands and the references inside triplicated registers. With these choices, we drastically reduced the number of input pins. Figure 3.6a shows an example cluster, where the DSPs inside perform a multiplication.

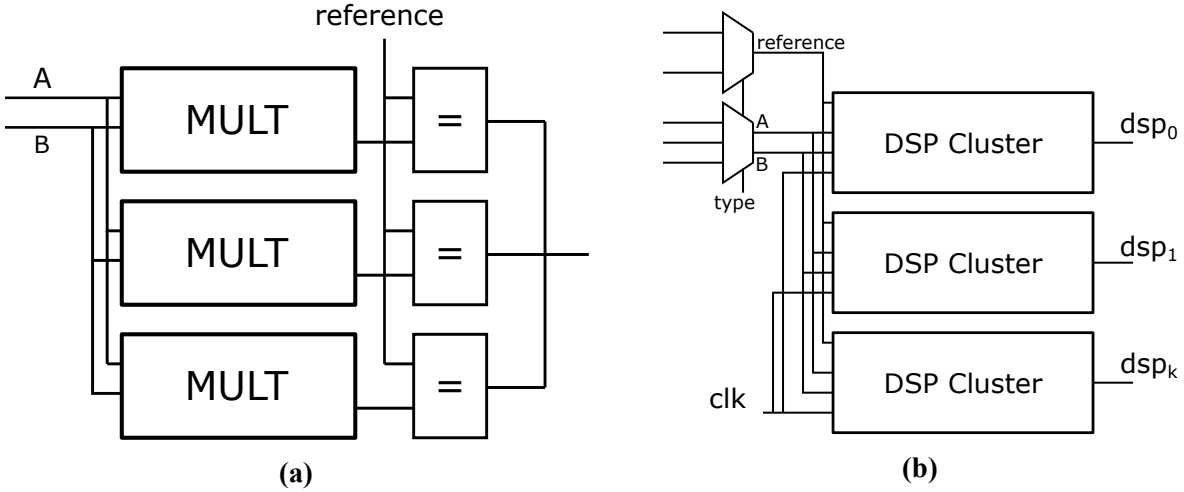


Figure 3.6: The circuitry for the DSPs qualification. Figure (a) shows the schematic of a cluster. Figure (b) shows the schematic of the top-level circuit.

Additionally, we implemented two types of test:

- Static: the DSPs perform their operation on fixed operands. In this situation, the output of the DSPs is always the same;

- Dynamic: the operands of the DSPs change every clock cycle. As a consequence, also their output changes.

Figure 3.6b shows the top-level view of the circuit we used for the irradiation. As we can see, we have very few input and output pins. In particular, we have the CLK signal driving all the clusters and a TYPE signal selecting the type of the test. Concerning the output, each cluster has its own output DSP_k, that we can monitor to understand when an SEU occurred.

Every FPGA has its own DSP implementation allowing to perform different kind of operation. Moreover, the DSP may include an internal register to store the intermediate results; therefore, the final implementation of this circuit really depends on the FPGA we are targeting. In our case, we implemented six type of clusters to cover all the possible implementations that the FPGA offered:

- M1: Multiplication
- M2: Same as M1 but using the internal register
- M3: Multiplication + Addition
- M4: Same as M3 but using the internal register
- M5: Same as M1 but with TMR
- M6: Same as M3 but with TMR

3.1.3 Phase-locked-loop

Phase-locked loops are blocks capable of generating output whose phase is related to the input signal phase. These blocks may differ depending on the FPGA manufacturer, but they always provide at least two outputs:

- Clock: it is the clock signal with the desired characteristics in terms of phase shift, jitter, frequency;
- Lock: it indicates when the clock signal is ready and stable, meaning the following blocks can use it.

An SEU may destabilize the clock signal. Consequently, the lock signal will de-assert (or assert if it is an active-low signal) to notify that the clock is not valid anymore. Thus, we can use this signal to retrieve the susceptibility of this element.

The structure we implemented for the irradiation is relatively straightforward. The only input is the clock signal, which is the input of all the PLLs of the FPGA under test. As for the output, we connected the lock signals to the FPGA IOs to observe them from our tester.

3.1.4 RAM blocks

FPGAs usually contain memory blocks to store data to avoid consuming many CLBs to implement storage functionalities. To retrieve their sensitivity, we need to write the memory with a certain pattern, and read it back during/after the irradiation. While reading, if we do not read exactly the same data, then it means an SEU affected the memory cell.

Usually, each FPGA contains memory blocks of different sizes. However, the circuit we can use to retrieve their sensitivity is the same. We can combine the blocks in a large memory, and then perform read/write operation on it. Therefore, the circuit is quite simple and does not require any additional logic. The input interface consists of a clock signal, R/W signals, data and address lines, which are the standard signals allowing to perform read and write operations on all the memories. The output interface instead is the memory output. However, to reduce the number of FPGAs pins, we implemented the data lines as a bidirectional bus. Thus, when we perform a write operation, the FPGA under test sets the data lines to high impedance so that an external device can drive them with the data we want to write. During a read operation, the FPGA under test drives the data lines to output the memory content at the desired address, and the external device must set them at high impedance.

3.1.5 TID degradation

In Section 2.2.1, we described in details the cumulative effects of radiations on electronic devices. When qualifying an FPGA for total ionizing dose effects, the process is usually different to SEE testing. TID effects mainly cause the gradual degradation of the device parameters, causing a shift in the MOSFET threshold voltage, an increase in the leakage current and an increase in the propagation delay of the FPGA, which eventually ends in complete failure of the device. The most common technique for testing TID effects is to measure the leakage current of the device during the irradiation using external equipment. The drawback of this method is that the current trend becomes visible only when the absorbed dose reaches a value near the failure threshold. However, since it is a gradual degradation, some internal faults may emerge even before the device fails, i.e. increasing the propagation delay leading to breaking the timing constraints of the circuit inside the FPGA. Thus, a run-time monitoring system is necessary to improve the reliability of the FPGA. It could provide warnings that can help understand when the device is degrading too much so that one can replace the system before it stops working completely.

The circuit we implemented for the TID qualification exploits the propagation delay increase. By measuring the propagation delay of the circuit during irradiation, we can monitor the TID effects on the FPGA dynamically. Therefore, we created a circuit whose propagation delay can be easily measured, that is, the **ring oscillator**. It is a circuit generating a periodic signal whose frequency is proportional to the propagation delay of its elements and routing connections. By measuring the frequency of this signal during the irradiation, we can retrieve the propagation delay increase. Thus it is the perfect circuit for propagation delay measurement. We implemented it using a chain of inverter and connecting the output of the last inverter to the input of the last inverter, creating a loop. By using an odd number of inverters, the output of the last inverter will continuously oscillate. Figure 3.7 shows an example with five inverters. The output frequency f depends on the number

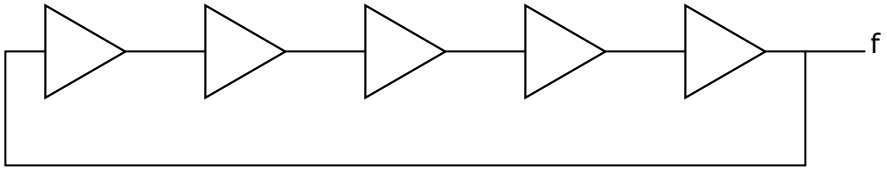


Figure 3.7: Ring oscillator with five inverters

of inverters (N) and their propagation delay (t), and is expressed in Eq. (3.1).

$$f = \frac{1}{2 \times N \times t} \tag{3.1}$$

With the dose accumulation, the propagation delay will change. Consequently, the signal will change its frequency. As for other circuits, we need to retrieve many statistics for an excellent characterization. Thus, it is essential to place many ring oscillators all over the FPGA to estimate how the degradation spreads, and the bigger the FPGA, the higher the number of ring oscillators needed. Measuring all these frequencies means implies connecting all the ring oscillators to the output of the FPGA. However, FPGAs have limited pins. Therefore we implemented a multiplexing mechanism to select only one ring oscillator to forward to the output of the FPGA. Figure 3.8 shows the circuit we implemented for our irradiation test from a functional point of view. The interface has

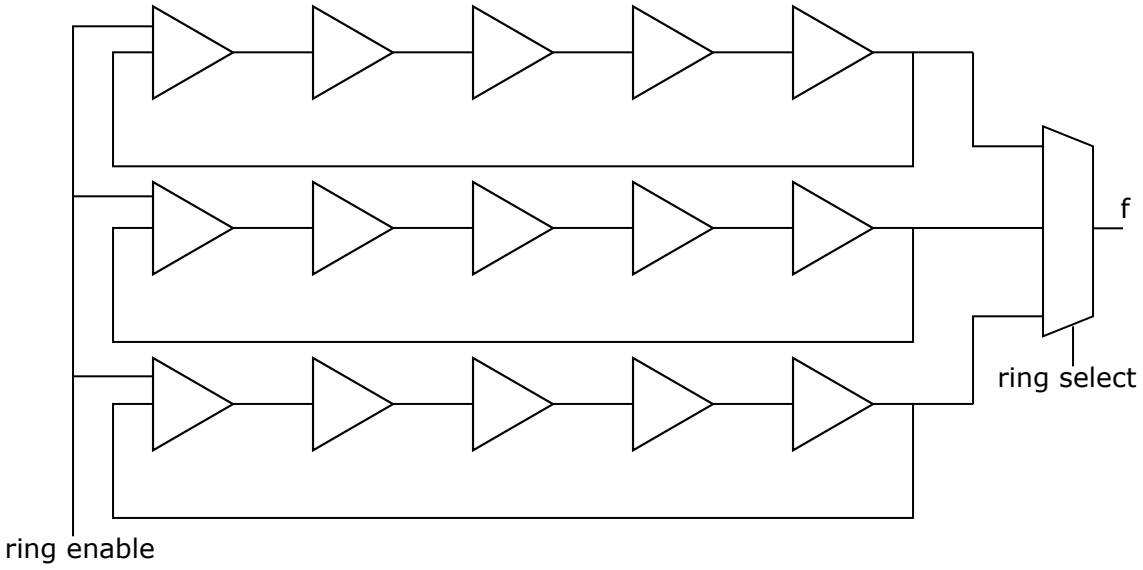


Figure 3.8: Circuit TID monitoring

an input signal that enables the ring oscillators and a selection signal to connect one ring oscillator to the output of the FPGA. The measuring system that measures the signal's frequency is part of the interface we describe in the next section.

3.2 Benchmark-based characterization

The standard approach for the qualification of FPGAs reveals much information about the sensitivity of each element. However, it does not give a realistic overview of how an actual design will

work in operation, which makes quite complex the estimation of the device's failure rate. When testing the Flip-Flops using the structure in Figure 3.5, an SEU leads to the chain's failure, but that only allows to retrieve the failure rate of that logic element. Using this data to estimate the device's failure rate when using an actual application is quite complicated, if not impossible. For example, an SEU in a Flip-Flop does not correspond to an error if the actual design does not use its content when it is wrong. In their guidelines and suggestions, the NASA RAEG affirms that performing an application-style test is mandatory to have a realistic behaviour of the system. Nonetheless, when using a project-specific application to retrieve the failure rate, we cannot use the results for other projects. On the contrary, we should perform a radiation test for all the designs we plan to use on that FPGA. Moreover, it would be impossible to compare the results with other organizations, slowing down research. For these reasons, we decided to follow a benchmark-based methodology. The approach uses benchmark circuits that reflect the workload of an actual circuit. It allows performing standardized application-level testing that makes it easier to compare results between the different organizations and between different FPGA families. Most importantly, it allows retrieving an estimation of the failure rate of the FPGA when in operation, independently from the application that we plan to use. We describe here the benchmark suite we employed and the circuit we implemented for our radiation tests. In the next chapter, we describe the results and show how we can retrieve the device's failure rate for the different areas of the LHC.

3.2.1 The benchmark suite

We decided to use the ITC'99 benchmarks [8] developed in the CAD Group at Politecnico di Torino for our qualification process. The suite contains circuits built starting from public VHDL files gathered and modified to behave synchronously. The result is a set of fully synthesizable circuits without any hardware/compiler-specific directive, allowing implementation quickly on different FPGA families. The circuits are syntactically correct, but they are not functionally meaningful due to their development process. The VHDL descriptions range from tiny circuits to larger ones and include various functionalities, from Finite-State Machines (FSM) to soft-core microprocessors; therefore, they are a good representation of the FPGA's workload in an actual context. In the literature, other reliability experiments like [14] and [17] used benchmarks belonging to this suite to validate the efficiency of the mitigation techniques.

3.2.2 Benchmark test structure

Among the various benchmarks available in the suite, we used the B13. Its original function was to act as an interface with a weather sensor. The circuit occupies relatively small resources (339 gates, 53 FFs, 20 I/O). It also contains a Finite-State Machine (FSM), which is quite common in an actual application. Being the circuit relatively small, we needed to replicate it many times on the FPGA under irradiation to retrieve many statistics for good characterization. Nevertheless, the circuit has ten inputs and ten outputs, posing a significant limitation on the number of circuits we can place because of the FPGA IOs. Thus, we developed a structure to work around this issue and have the same level of observability. We associated to every B13 a combinatorial comparator circuit. The comparator has two inputs: the output of the B13 and a golden reference. The B13 output equals the golden reference in normal condition, and the logic level at the output of the comparator is one. In case an SEU affects the B13, its output will differ from the golden reference. Thus the

output logic level of the comparator will be zero. By replicating this concept, we have an error bit associated with each B13 instance. Then, we wired all these error bits into a single bit that we used as an error flag, and we connected it to the output of the FPGA to monitor it:

- If zero, an SEU affected at least one of the B13;
- If one, all the B13 are working correctly.

We would not have much observability with this structure since we would not know which B13 is failing and how many output bits are wrong. Therefore we added a combinatorial circuit that produces as output the address of the failing B13. We connected this address to the output of the FPGA to monitor it. The size of this signal grows logarithmically with the number of B13 instances we place on the FPGA under test. Moreover, we use this address to select the faulty B13 and forward its output to the output of the FPGA. We applied TMR on this additional circuitry to reduce the probability of observing errors deriving from them rather than from the B13 instances. As a result, we have an interface with a high level of observability since it allows us to monitor:

- When an error occurs;
- Which circuit is failing;
- The value of the faulty output.

Figure 3.9 shows the top-level diagram of the circuit we used for the radiation test. We also realized a second version for the same circuit where we applied TMR on all the storage elements to study the possible improvements of this mitigation technique. The two versions, standard and TMR, have the same structure and interface, with the only difference being the number of B13 instances placed on the DUT. Concerning the input interface, in addition to the clock and reset signals, we connected all the B13 instances to the same 10-bits input, STIMULUS. Thus they will all produce the same result. As for the golden REFERENCE necessary for the comparison, we do not compute it inside the FPGA under radiation, but instead, we compute it outside, which accounts for ten other input pins. The output interface requires one pin for the ERROR DETECTED, ten pins for the FAULTY OUTPUT, and log(N) pins for the FAULTY B13 ADDRESS, where N is the number of instances. We describe the interface driving this test circuit in the next section.

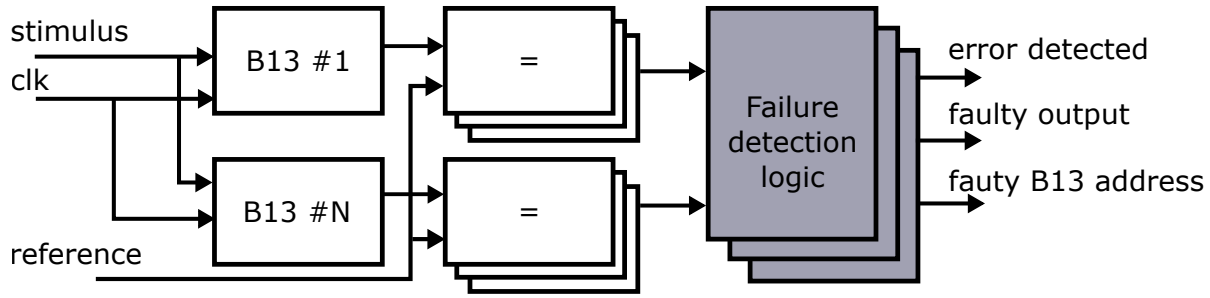


Figure 3.9: Benchmark testing circuit

3.3 FPGA-based testing platform

The various test circuits we presented in the previous section require an external tester driving their inputs and monitoring their outputs. Therefore, we need to control and monitor the Device under Test (DUT) using a host computer when performing radiation qualification tests. Moreover, we need to save the data since it is fundamental to carry on the data analysis to retrieve the device sensitivity, which is the main objective of the qualification process. We decided to develop our interface on a System-on-Chip (SoC) containing an FPGA since it brings many advantages in terms of flexibility, functionalities and speed, as we will describe later. This section describes the interface we developed, the technologies we used, and the motivations behind the choices we made.

3.3.1 Hardware platform

The interface, or better the system, we need for the radiation qualification, needs to provide different functionalities, the most important being the possibility to provide inputs to the DUT and monitor its output. It needs to perform high-speed testing and catch all the errors produced by the DUT. Most importantly, it needs to record all the data produced by the DUT to retrieve the device sensitivity. In addition to these functionalities, we developed our interface keeping in mind these key features:

- Visibility: to give access to as many signals as possible from the DUT;
- Compatibility: to control/monitor FPGA Development Kit (DevKit) of different manufacturers;
- Maintainability: to add/modify/delete any feature, test routine, or interface signal without requiring many efforts;
- Usability: to facilitate as much as possible the qualification process, including the real-time monitoring during the irradiation and the final data analysis.

Considering these features, we decided to base our interface on an SoC, including an embedded processor and an FPGA. This choice allows us to connect the two FPGAs directly without any communication protocol, avoiding the overhead we would have if connecting the FPGA to a host PC using any other kind of communication protocol, like UART or ETHERNET. Moreover, the user could not observe the DUT output pins' or drive the DUT input pins directly. Instead, he should format the commands/data as the protocol requires, adding overhead to the radiation process. On the contrary, by connecting the IOs of two FPGAs directly, the user does not need any protocol to exchange information since the TESTER FPGA can drive and observe the IOs of the DUT FPGA, even at high speed. Once we decided on the connection to the DUT, we also needed an interface to allow the user to control the TESTER FPGA. We decided to use the CPU embedded inside the SoC for this task, which allows the user to control the TESTER, thus the DUT, using high-level APIs. In the next section, we describe how we implemented this mechanism. Figure 3.10 gives a high-level representation of our system, showing the primary connections between the different components. We will now analyze all the components of the interface and discuss our contribution to each part.

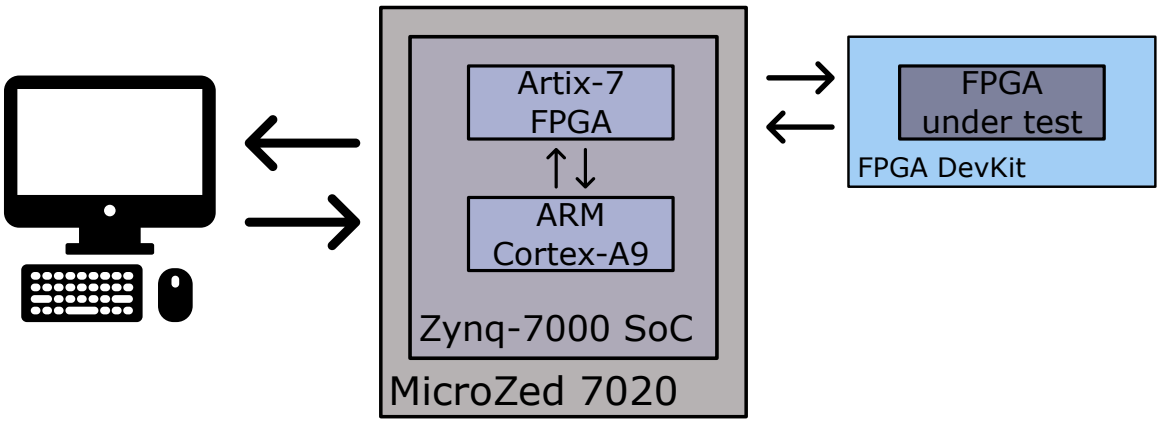


Figure 3.10: Tester platform architecture

The development board

The development board we adopted is the MicroZed™7020 from AVNET, shown in Figure 3.11. The board contains the Xilinx Zynq®-7000 SoC family, a family of products that integrate a dual-core ARM®Cortex™-A9 CPU (Processing System, PS) connected to a 28 nm Xilinx FPGA (Programmable Logic, PL) in a single device.

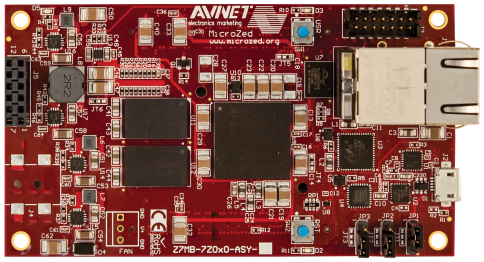


Figure 3.11: MicroZed™7020 from AVNET

The interconnection

We decided to adopt the FPGA Mezzanine Card (FMC) standard to interconnect the TESTER development board to the DUT development board. FMC is an ANSI/VITA 57.1 standard that defines the I/O mezzanine modules for an FPGA baseboard. The standard offers two sizes of connector: Low Pin Count (LPC), offering 68 user-defined signals, and High Pin Count (HPC), providing 160 user-defined signals. They both offer ten transceiver pairs and additional clock lines. The LPC and HPC use the same mechanical connectors, with the only difference being the signal connected. The standard brings many advantages because it decouples the I/O interfaces placed on the mezzanine module from the FPGAs located on the carrier card. With the increased capacity of FPGA and I/O pins, an engineer can design and update carrier cards and then utilize the different FMC-based external modules already developed. For these reasons, the FMC connector is now present on most FPGAs development kit board, and it represents a reasonable choice to use it as an interface. In our case, the MicroZed™7020 development board has its FMC-carrier card, as

Figure 3.12 shows, which connects the FPGA pins to the pins of the FMC connector. Finally, we can connect the TESTER and DUT using an FMC-to-FMC cable.

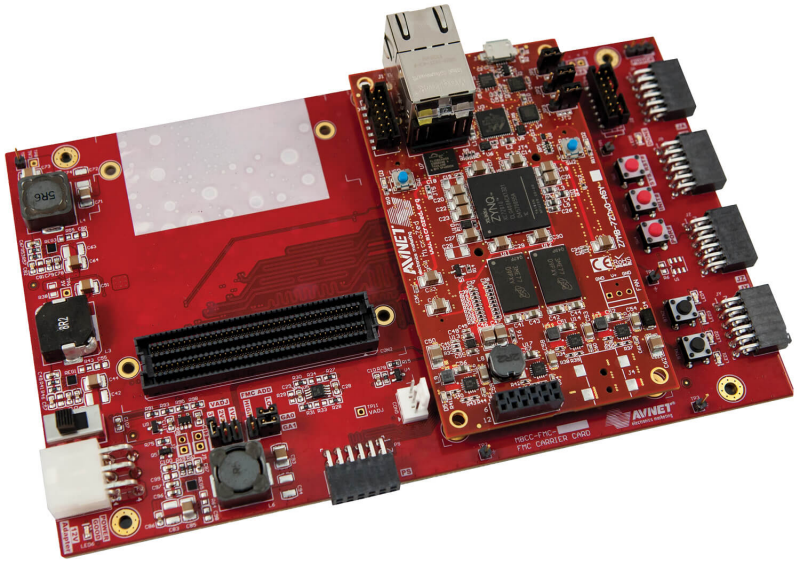


Figure 3.12: MicroZed™7020 on its carrier board

System architecture

The core of the TESTER interface consists of a combination of CPU and FPGA. In particular, we developed on the FPGA the peripherals driving/monitoring the circuits running on the DUT, and we used the CPU to communicate with these peripherals. In the previous section, we presented the logic circuits we developed to characterize the FPGA's sensitivity. Here, we discuss the Intellectual Property (IP) cores we built, using VHDL, to interface such circuits. Each IP acts as a tester for the corresponding circuit on the DUT. This choice gives the maximum flexibility because it allows using only the tester related to the circuit that we are testing. We developed the IPs focusing on the possibility to integrate them quickly into a broader system. Therefore, we decided to implement the most common interface between an embedded processor and a peripheral device used in any embedded system: control, status and data registers. The CPU communicates with the peripheral by writing the instructions inside the control registers. The peripheral communicates its status writing inside the status register that the CPU can read. Finally, the peripheral can pass data to the processor through the data registers. The peripherals registers are memory-mapped, and the CPU writes/reads these registers performing Memory-Mapped operations. Therefore, we can easily connect multiple peripherals to the CPU by allocating an address space for each one. Finally, we provided each IP with an Advanced eXtensible Interface (AXI) interface, a standard communication protocol widely used in System-on-chips. The AXI interface facilitates the usage of the IPs in more extensive systems, allowing to build platforms that are not related to a specific FPGA or a particular development board. Figure 3.13 shows the top-level architecture of this system..

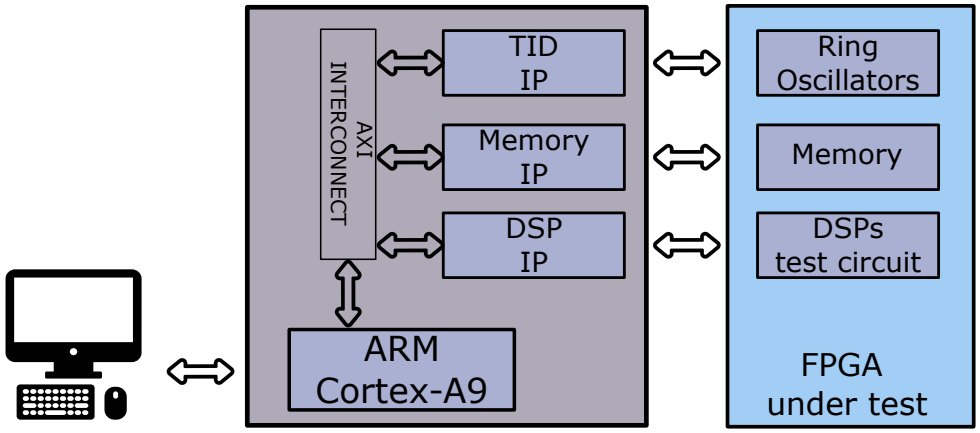


Figure 3.13: TESTER-DUT system top-level view

Advanced eXtensible Interface (AXI)

The AMBA AXI protocol is one of the specifications provided by AMBA, an open standard for the connection and management of functional blocks in a system-on-chip. There are three types of AXI4 interfaces:

- AXI4 - for a system requiring high-performance memory-mapped interfaces;
- AXI4-Lite - simple and low-throughput memory-mapped communication, i.e. the communication to/from control and status registers;
- AXI4-Stream - for high-speed streaming data.

This protocol brings many improvements and benefits in terms of Productivity, Availability, Flexibility and Compatibility:

- Flexibility: the types of interfaces provided by AXI allow to have always the correct type of protocol for the application;
- Productivity: the developer needs to know only a single communication protocol for all the IPs;
- Availability: By using an industry-standard protocol, the developer can access a variety of IPs from different vendors;
- Compatibility: The key characteristics are re-usability, which require the IP to support various SoC with different power, performance and area requirements. Using a widely-adopted protocol ensures compatibility and scalability between IP components from different teams and companies;

The AXI specifications describe an interface between a single master and a single slave, representing two IPs that exchange information. The interface, which is the same for both AXI4 and AXI4-Lite, consists of five different channels:

- Read address channel;

- Read data channel;
- Write address channel;
- Write data channel;
- Write response channel.

The different channels allow performing simultaneous, bidirectional data transfer. The R/W address channel carries the control information describing the nature of the data on the respective R/W data channel. The transfer size, called burst, can vary. In AXI4, the maximum size of a burst transaction is 256 transfers, with only one address phase, which means that only the first address is necessary as the protocol computes the others automatically based on the burst size. AXI4-Lite instead fixes the burst size to one. The two IPs can exchange data in two ways: (i) the master uses the write data channel to transfer data to the slave, which uses the write response channel to signal the completion of the transfer and (ii) the slave can send data through the read data channel to the master. Figure 3.14 and Figure 3.15 describes the two situations. AXI4-Lite is very similar to AXI4 with some exceptions, the two most notable of which are:

- Support for only 32-bit and 64-bit data bus;
- Fixed burst length of one.

These simplifications allow building a simple component interface, especially useful when the main objective is not a high-performance interface for data transfer but a simple memory-mapped communication between different entities, like in our case.

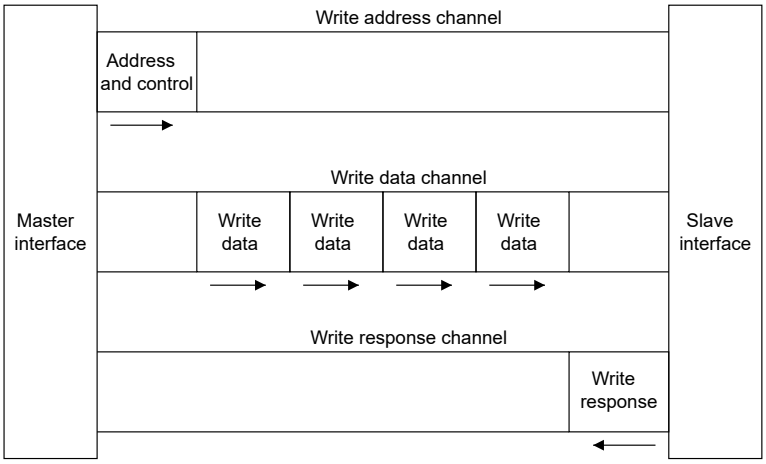


Figure 3.14: AXI4 channels architecture of a write transaction

3.3.2 FPGA peripherals

This section describes the peripherals that we developed to interface the various circuits for the FPGA qualification we described in the previous section. For each peripheral, we will describe the interface and its working principle.

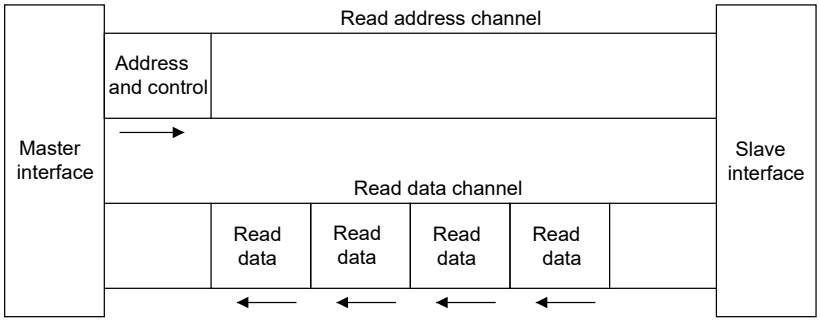


Figure 3.15: AXI4 channels architecture of a read transaction

Flip-Flop tester

The Flip-Flop tester manages the FF test circuit we described in Section 3.1.1.

Figure 3.16: Flip-Flop IP interface

Therefore, we developed an IP that can:

- Provide the structure with clock and load signals;
- Switch between static and dynamic tests, and select the input pattern;
- Monitor each chain's output signal and record the number of errors for every chain.

Figure 3.16 shows the IP interface, including the AXI4-Lite interface necessary for communication with the embedded CPU. Table 3.1 contains a description of each port. Internally, the IP has a register associated with each chain we are monitoring, and if a test is running, the IP stores inside each register the number of errors recorded for the corresponding chain.

The user can interact with the IP using the registers described in Tab. (3.2). The table contains the address space, a description of the function of each register, and the functionalities of the bits. The user can start the test using the CONTROL register. At any time, he can retrieve the number of errors by writing the chain number inside the ADDRESS register. The IP will copy the number of errors related to that chain inside the DATA register.

Port	I/O	Description
AXI4-Lite	INOUT	AXI4-Lite bus. It groups all the signal necessary for the AXI4-Lite communication.
clk	Input	Global Interface Clock: All signals on Interface must be synchronous to this clock.
rst_n	Input	Global reset: This signal is active-Low, it must be asserted at least for one clock cycle.
chains[n-1:0]	Input	The output lines of the chains on the Device Under Test.
pattern[1:0]	Output	This signal is used to select the type of test: - 00/11 = static 0 - 01/10 = dynamic
window_load	Output	This signal is used to load the flip-flops inside the window.

Table 3.1: Flip-Flop IP ports description

Name	Address	Description	Bit(s) description	
			Bit(s)	Functions
Control	0x0	Control the functionalities of the core	1-0	01 = Start the test 10 = Stop the test 11 = Reset the SEU counts
			3-2	00 = Static test with 0000 01 = Dynamic test with 0101 10 = Dynamic test with 1010 11 = Static test with 1111
Status	0x4	Check the status of the IP	0	0 = Idle 1 = Test running
Address	0x8	Select the Flip-Flop chain	-	
Data	0xC	Contains the number of errors detected on the chain specified in the Address register	-	

Table 3.2: Flip-Flop IP register definition

DSPs tester

Section 3.1.2 describes the circuit to retrieve the DSPs sensitivity and its IO interface. Therefore,

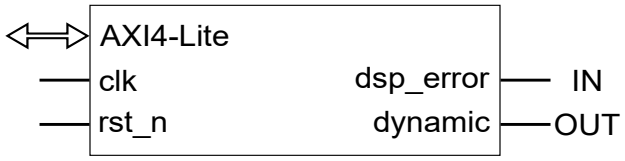


Figure 3.17: DSPs IP interface

we developed a DSPs tester that can:

- Provide the DSPs with a clock signal;
- Switch between static and dynamic tests;
- Monitor the output signal of each cluster of DSPs and record the number of errors for each of them.

Figure 3.17 shows the IP interface, including the AXI4-Lite interface necessary for communication with the embedded CPU. Table 3.3 contains a description of each port. Internally, the IP has a

Port	I/O	Description
AXI4-Lite	INOUT	AXI4-Lite bus. It groups all the signal necessary for the AXI4-Lite communication.
clk	Input	Global Interface Clock: All signals on Interface must be synchronous to this clock.
rst_n	Input	Global reset: This signal is active-Low, it must be asserted at least for one clock cycle.
dsp_error[n-1:0]	Input	The output lines of the DSP comparator on the Device Under Test.
dynamic	Output	This signal is used to select the type of test: - 0 = static - 1 = dynamic

Table 3.3: DSPs IP ports description

register associated with each cluster we are monitoring, and if a test is running, the IP stores inside each register the number of errors recorded for the corresponding cluster.

The user can interact with the IP using the control, status, and data register. Table 3.4 shows the address space and the functionalities of each bit. The user can start/stop the test in either static or dynamic mode using the CONTROL register. He can then retrieve the number of errors by writing the cluster number inside the ADDRESS register. The IP will copy the number of errors related to that DSP cluster inside the DATA register.

Name	Address	Description	Bit(s) description	
			Bit(s)	Functions
Control	0x0	Control the functionalities of the core	1-0	01 = Start the test 10 = Stop the test 11 = Reset the SEU counts
			2	0 = Static test 1 = Dynamic test
Status	0x4	Check the status of the IP	0	0 = Idle 1 = Test running
Address	0x8	Configure the DSP cluster to observe	-	
Data	0xC	Contains the number of errors detected on the cluster specified in the Address register	-	

Table 3.4: DSPs IP register definition

RAM tester

The circuit to retrieve the memory sensitivity does not contain any additional circuitry. Thus all the logic is in the tester.

Figure 3.18: Memory IP interface

Therefore, we developed an IP to:

- Provide the memory with a clock signal;
- Perform read/write operations on the memory;
- Check that the data received equals the data written. In case it is not, record the number of SEU and MBU.

Figure 3.18 shows the IP interface and Tab. 3.5 contains a description of each port. The user can interact with the IP using the registers described in Tab. 3.6, which shows the address space, and the functionalities of each register. Whenever the user issues a read/write operation, the IP performs a read/write on all the memory addresses in increasing order. Finally, during each read, the IP checks that the data corresponds to the data written during the write operation and updates the number of SEU and MBU if it is not. The user can access these values by reading the corresponding registers at any time.

Port	I/O	Description
AXI4-Lite	INOUT	AXI4-Lite bus. It groups all the signal necessary for the AXI4-Lite communication.
clk	Input	Global Interface Clock: All signals on Interface must be synchronous to this clock.
rst_n	Input	Global reset: This signal is active-Low, it must be asserted at least for one clock cycle.
read	Output	This signal is active-High, it is used to perform a read operation.
write	Output	This signal is active-High, it is used to perform a write operation.
data[N-1:0]	InOut	These are the data lines carrying data to/from the memory under test on the DUT.
address[K-1:0]	Output	These are the address lines indicating the memory address where to perform the R/W operation .

Table 3.5: Memory IP ports description

Name	Address	Description	Bit(s) description	
			Bit(s)	Functions
Control	0x0	Control the functionalities of the core	1-0	01 = Read memory 10 = Write memory 11 = Reset the SEU/MBU counts
			3-2	00 = Set pattern to all zeros 01 = Set checkerboard pattern 10 = Inverted checkerboard pattern 11 = Set pattern to all ones
Status	0x4	Check the status of the IP	0	0 = Idle 1 = Test running
SEU	0x8	Contains the total number of SEUs	-	
MBU	0xC	Contains the total number of MBUs	-	

Table 3.6: Memory IP register definition

PLL tester

The PLL tester is the IP we built to interface with th PLL test circuit presented in sec. 3.1.3. Figure 3.19 shows its interface and Tab. 3.7 contains a description of each port.

Figure 3.19: PLL IP interface

The IP can:

- Provide the PLLs with a reference clock;
- Monitor each PLL's Lock signal and record the number of errors for every PLL.

Port	I/O	Description
AXI4-Lite	INOUT	AXI4-Lite bus. It groups all the signal necessary for the AXI4-Lite communication.
clk	Input	Global Interface Clock: All signals on Interface must be synchronous to this clock.
rst_n	Input	Global reset: This signal is active-Low, it must be asserted at least for one clock cycle.
ref_clk	Output	This is the reference clock sent to all the PLLs under test.
locks[N-1:0]	Input	These are the LOCK signals of the PLL present on the DUT .

Table 3.7: PLL IP ports description

Internally, the IP has a register associated with each PLL we are monitoring, and if a test is running, the IP stores inside each register the number of times that the LOCK signal is low (which means the PLL is not locked). The user can interact with the IP using the registers described Table 3.8. The user can start/stop the test using the CONTROL register. He can retrieve at any time the number of errors by writing the PLL number inside the address register. The IP will copy the number of errors related to that PLL inside the data register.

Name	Address	Description	Bit(s) description	
			Bit(s)	Functions
Control	0x0	Control the functionalities of the core	1-0	01 = Start the test 10 = Stop the test 11 = Reset the SEU counts
Status	0x4	Check the status of the IP	0	0 = Idle 1 = Test running
Address	0x8	Set the PLL instance for the DATA register	-	
Data	0xC	Contains the number of errors detected on the PLL specified in the ADDRESS register	-	

Table 3.8: PLLs IP register definition

Benchmark tester

Section 3.2.2 describes the circuit to study the device sensitivity using a benchmark application.

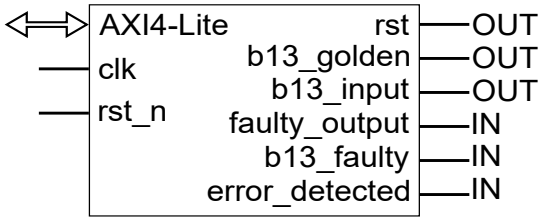


Figure 3.20: B13 IP interface

To interface this circuit, we developed an IP that can:

- Provide the design with a clock signal;
- Generate the input for the B13 instances;
- Provide the golden reference that the comparators can use to check the correctness of the B13 output;
- Monitor the error detected signal to understand when a B13 instance is failing. In this case, save the content of the faulty output and faulty address lines.

Figure 3.20 shows the IP interface, while Tab.3.9 contains a description of each port. The IP contains different registers described in Tab.3.10.

The IP contains a B13 instance and a random input generator (LFSR). The LFSR produces a 10-bit signal going to all the B13s, both on the tester and the DUT. Thus all the instances are in the same state. When the user starts the test, the IP sends the B13 output to the comparators on the DUT. All the B13 produce the same results in normal condition because they are all in the same state. In case of an error, the IP stores the output of the faulty B13 and its address inside an internal FIFO. The user can, at any moment, issue a read operation to the IP to read the values contained

Port	I/O	Description
AXI4-Lite	INOUT	AXI4-Lite bus. It groups all the signal necessary for the AXI4-Lite communication.
clk	Input	Global Interface Clock: All signals on Interface must be synchronous to this clock.
rst_n	Input	Global reset: This signal is active-Low, it must be asserted at least for one clock cycle.
rst	Output	This is the reset signal, active high, used to bring all the B13s in the same state.
b13_output[9:0]	Input	This is the output of the faulty B13.
b13_faulty[N - 1:0]	Input	This is the identifier of the faulty B13. The size depends on the number of B13 instances on the DUT.
error_detected	Input	Error detected signal active-high It is asserted when an error is detected
b13_input[9:0]	Output	The bus carrying the input bits for each B13

Table 3.9: Benchmark IP ports description

Name	Address	Description	Bit(s) description	
			Bit(s)	Functions
Control	0x0	Control the functionalities of the core	1-0	01 = Start the test 10 = Stop the test 11 = Reset the SEU counts
Status	0x4	Check the status of the IP	0	0 = Idle 1 = Test running
			1	0 = Fifo is empty 1 = Fifo not empty
Address	0x8	The address of the faulty B13	-	
Data	0xC	Faulty output produced by the B13 specified in the ADDRESS register	-	

Table 3.10: Benchmark IP register definition

inside the FIFO. When the user issues a read operation, the IP fetches a tuple of values from the FIFO and copy them inside the ADDRESS and DATA register, that the user can read. Moreover, when an SEU affects the DUT, the IP asserts a reset signal to bring all the instances back in the same state.

TID tester

The TID tester is the monitoring IP for the internal delay degradation of the FPGA that we measure using the circuit in Section 3.1.5. Its interface is shown in Fig. 3.21. The IP allows to:

- Select the ring oscillator's output that we want to measure;
- Measure the frequency of an incoming signal, as a frequency counter;

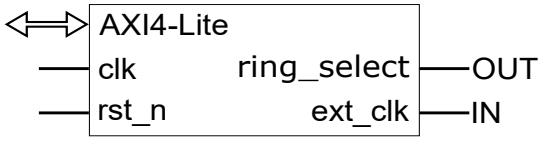


Figure 3.21: TID IP interface

Table 3.11 contains a description of each port.

Port	I/O	Description
AXI4-Lite	INOUT	AXI4-Lite bus. It groups all the signal necessary for the AXI4-Lite communication.
clk	Input	Global Interface Clock: All signals on Interface must be synchronous to this clock.
ext_clk	Input	Clock signal produced by the ring oscillator. It is the signal for which we want to measure the frequency
rst_n	Input	Global reset: This signal is active-Low, it must be asserted at least for one clock cycle.
ring_sel[N-1:0]	Output	This is the signal used to select the ring oscillator on the DUT.

Table 3.11: TID IP ports description

We implemented a reciprocal frequency counter to measure the frequency of the input signal. This mechanism consists of counting a reference clock and the external signal for a fixed time. Once it is over, we retrieve the pulses counted from the reference clock $pulses_{ref}$ and the pulses counted from the external signal $pulses_{ext}$. Using these two values and knowing the reference clock frequency f_{ref}, we can retrieve the frequency of the external signal using Eq. (3.2).

$$f_{ext} = \frac{pulses_{ext} * f_{ref}}{pulses_{ref}} \tag{3.2}$$

The IP contains different registers for the communication, described in Table 3.12. The user needs to select the ring oscillator he wants to monitor by writing his number inside the address register. Once selected, the user can set the measuring time, depending on the precision he needs, and then start the measurement. The status register indicates whether the measurement is running or not. Finally, the two data registers contain the number of pulses counted for the reference clock and the external signal, which the user can use to compute the final frequency.

3.3.3 CPU-FPGA interface

Thanks to the connection between CPU and FPGA offered by the Xilinx SoC, the user can write for each qualification test he wants to perform the code needed to communicate with the peripherals to start/stop the test and acquire the data. The most common programming languages for an embedded system are C and C++. In the Xilinx platforms, the Xilinx Runtime Library (XRT) enables developers to deploy applications in C/C++ on the embedded processors. The XRT is a software interface running on the host CPU (the embedded CPU in Xilinx SoC) that facilitates the

Name	Address	Description	Bit(s) description	
			Bit(s)	Functions
Control	0x0	Control the functionalities of the core	1	1 = Start the measurement 0 = Stop the measurement
Status	0x4	Check the status of the IP	0	0 = Idle 1 = Measurement running
Address	0x8	Set the ring oscillator to measure	-	
Interval	0x8	The interval (in clock cycles) to measure	-	
Reference	0xC	Number of pulses counted for the reference clock	-	
External	0xA	Number of pulses counted for the external signal	-	

Table 3.12: TID IP register definition

communication between the application code and the FPGA design.
On top of the XRT interface, we adopted another framework enabling Python usage to increase productivity and facilitate the qualification process. **PYNQ**, Python Productivity for Zynq, is an open-source project from Xilinx that aims to simplify the use of the Xilinx Hardware Platform. The main goal is to allow embedded systems engineer to exploit the capabilities of FPGAs using Python. PYNQ achieves this goal using three key features:

- The circuits designed for the PL are presented to the user as hardware libraries, called overlays. An engineer can select the functionality he needs by using the appropriate hardware library for its application. Moreover, the engineer can use the hardware capabilities offered by the overlay using Python APIs. Even though the logic circuit design is still required, once the design is complete, people can re-use it anytime. Therefore, the hardware libraries must be designed to be configurable and re-used as often as possible, like software libraries;
- Using Python for programming both the embedded processors found on the Xilinx SoC and thus to communicate with the overlays, which raises the level of programming abstraction and the programmer productivity;
- it is an open-source project that works on any computing platform and operating system because it adopts a web-based architecture. It incorporates the open-source Jupyter notebook to run an Interactive Python (IPython) kernel and a web server directly on the ARM processor of the Zynq device.

Using this framework, we enormously simplified the execution of the radiation qualification process. After the PYNQ installation on the embedded processor, the user can communicate with the development board through the internet using a browser and a Jupyter notebook. Once connected, the IP we developed, will be presented to the user as different python objects. In this way, the user can exploit the FPGA capability and carry out the radiation test as if he was writing a Python program.

CHAPTER 4

FPGA radiation test results

In Chapter 2, we described the radiation levels of the upcoming upgrade of the LHC, the HL-LHC. In the past, engineers at CERN qualified other FPGAs for the CERN environment. These are the ProASIC3, the SmartFusion2 [27] and the Artix7 [25]. The ProASIC3 has been used in the past developments, while nowadays, the SmartFusion2 and Igloo2 FPGA are being used to develop the majority of CERN applications. However, with the approach of the High-Luminosity LHC, and the increased radiation levels, engineers need more robust FPGAs able to withstand a higher fluence and ionizing dose. Therefore, we qualified two FPGAs to understand if they are a good candidate for CERN applications: the NG-Medium and the PolarFire. The NG-Medium is an SRAM-based FPGA produced by NanoXplore. It is Radiation Hardened by Design (RHBD), manufactured according to the STM C65 space process. The PolarFire, instead, is the fifth generation of non-volatile FPGA device from Microsemi built on the state-of-the-art SONOS 28nm non-volatile process technology. We studied these two FPGAs because of the TID levels expected for the HL-LHC. The NG-Medium, despite its cost, is an RHBD FPGA. Therefore it might be the only solution for systems in very harsh environments, especially where TID levels are high. The PolarFire, instead, belongs to the same family of devices whose performances have already been qualified for the LHC environments. Hence, we expected similar/better performances from this new FPGA, considering the addition of the SONOS technology. This chapter will present the entire qualification process we carried out. We will briefly describe the facilities where we performed the irradiation tests. Then, we will report and explain the results we retrieved from the standard, the benchmark-based, and the TID characterization. Finally, using the data retrieved from the benchmark-based methodology, we will estimate the failure rate of the FPGAs for the HL-LHC environment.

4.1 Irradiation facilities

In the last section of Chapter 2, we mentioned the two main contributions to SEE for an FPGA: High Energy Hadron (HEH) and Thermal Neutrons (ThN). To address the sensitivity of the devices against these two effects individually, we tested the FPGAs in two different facilities: the Paul

Scherrer Institute (PSI), in Switzerland, and the Institute Laue-Langevin (ILL) in France. The facility at PSI provides a proton beam of 200 MeV. We could use it to test the effects of HEH because according to the RHA procedure, all the HEH should have the same probability of inducing SEEs due to their similar nuclear interaction cross-section. We used the facility at ILL for the ThN test.

4.1.1 Paul Scherrer Institute - PSI

The standard facility that CERN groups use to perform radiation tests against protons on electronic components is the Proton Irradiation Facility (PIF) beam line at the Paul Scherrer Institute (PSI) [13], constructed by the cooperation between PSI and ESA. Space community and research groups in other disciplines use the facility extensively. The irradiation area of PIF is in the cyclotron-type accelerator COMET, dedicated to proton therapy, which provides the PIF area with a beam of 230 MeV. By using copper plates of different thicknesses, we can reduce this energy down to 10 MeV. The measurement and calibration system consists of an ionization chamber located downstream of the collimators. Before the test, a plastic scintillator placed in the DUT location (5 cm from the ionization chamber) is used to measure the flux and establish the conversion factors between the counts from the ionization chamber and the proton flux. The same scintillator can also be used to retrieve the beam profile by moving it horizontally and vertically. Finally, both the beam intensity and energy can be set from the measurement control room equipped with a PC showing the various parameters during the irradiation run. To position the DUT, the user can use a laser system. The maximum beam current is typically around 5 nA, corresponding to a flux of $2 \cdot 10^8$ p/cm^2/s at 230 MeV. The minimum is around 0.1 nA, after which the beam becomes unstable. Concerning the beam size, we can fix it using a collimator. In our tests, we fixed it to 2 cm so that we could irradiate only the FPGA.

4.1.2 Institute Laue-Langevin - ILL

The Institute Laue-Langevin (ILL) [4] is an international research centre providing facilities in neutron science. For the irradiation campaigns, we used the irradiation facility D50. It is a scientific instrument dedicated to industrial activity, including microelectronics. The ILL horizontal cold source produces the neutrons available on D50 and then transmits them along a 100m long n-guide. The captured flux (i.e. equivalent flux of 25meV neutron) delivered on D50 is adjustable from 0 to 10^{10} n/cm^2/s, with optimisation for flux range from 10^6 n/cm^2/s to 10^8 n/cm^2/s. The neutron spot size may be easily adapted for local irradiation (1mm2) to global irradiation of a homogeneous square section of about 5 000 mm^2 through motorized borated carbon (B4C) slits. During our tests, we covered with boron carbide all the parts of the board except the FPGA, so that we could protect them from thermal neutrons. The neutron flux is controlled with a dedicated 3He detector and periodical gold foil measurements to achieve reliable flux data. A sample changer makes possible successive irradiation of components and boards.

4.2 Evaluation metric

Once we identified the radiation environments where we will use the FPGA, the second essential factor to evaluate is its sensitivity to SEE. The standard metric we use to evaluate the sensitive area of a device to radiation is the **cross-section** σ. It estimates the probability that a specific event happens in response to a radiation effect. We can retrieve the cross-section experimentally through N different measurements x_i, where every measurement is the number of SEE counts m divided by the total particle fluence, as shown in Eq. (4.1).

$$x_i = \frac{m_i}{\phi_i} \rightarrow \sigma = \frac{\sum_{i=1}^{N} x_i}{N} = \frac{m}{N \times \phi_{tot}} \tag{4.1}$$

4.3 NG-Medium characterization

The NG-MEDIUM (NX1H35S) is an SRAM based FPGA, Radiation Hardened, fabricated using the STMicroelectronics C65-SPACE process technology (65nm rad-hard). The configuration memory cells are built with dedicated rad-hard layout to guarantee a very low probability of soft-errors. On top, there is the Configuration Memory Integrity Check (CMIC) engine that performs automatic verification and repair of the configuration memory. The FFs are also built with rad-hard layout. Finally the embedded Dual-Port RAMs are protected with Error Code Correction (ECC). According to the specifications, the device can withstand a TID of at least 1 kGy. The FPGA embeds 56 RAM Blocks of 48 kbits each, with ECC options, 112 embedded Digital Signal Processors (DSP), 168 register file blocks of 64×16 bits each and 4 PLLs.
We investigated the NG-Medium sensitivity because it could be a possible solution for the upcoming HL-LHC radiation environment thanks to its radiation-hardened manufacturing process combined with the CMIC engine. This engine is an example of the memory scrubbing mitigation technique. SRAM-based FPGAs use an additional memory, which is sensitive to radiation, to store their configuration. If SEUs start to accumulate, eventually the design will fail. Therefore, the CMIC prevents the accumulation of errors inside the FPGA configuration memory, reducing the device failure rate. However, it cannot correct double errors, and in case it finds one, it stops working.

In this work, we did not perform any test to characterize the standard elements of the FPGA because the CERN R2E group had already performed the characterization for this device [26]. Hence, we focus here on discussing the results we obtained for the benchmark circuit, the TMR mitigation technique, and the benefits of the CMIC engine. The results we present refers to the two proton irradiation campaign we carried out.

4.3.1 Test setup

The evaluation board we used to carry out the irradiation test campaigns is the NX1H35 evaluation kit. We used a test setup to have the minimum equipment near the beam. Apart from the evaluation kit that we installed in front of it, we placed the remaining instrumentation inside the room access corridor to minimize their radiation exposure and avoid activation. We can control all the instruments remotely using a computer inside the control room. Figure 4.1 shows a schematic of the setup. It consists of the NG-Medium FPGA (under test), connected to the MicroZed 7020 de-

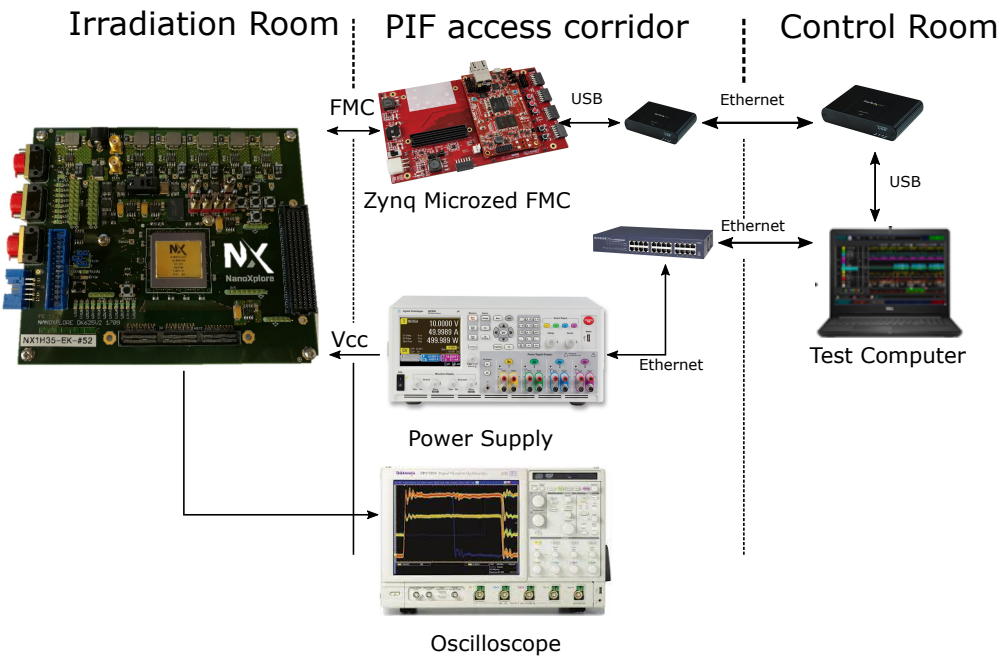


Figure 4.1: Test setup for the NG-Medium testing

velopment board (TESTER) with an FMC-to-FMC cable. We used an oscilloscope to monitor the benchmark application error signal. We connected the TESTER to the host PC in the control room, from where we could start/stop the test and monitor the SEEs. To program the DUT, we saved the bitstream configuration inside a flash memory connected to the DUT. Using the host PC, we could download the bitstream from the flash to the configuration memory of the FPGA. The board contains many components, but we could target only the FPGA thanks to the adjustable beam size. In this way, we could reduce the probability of SEE and SEL from other components, even though a susceptible component could have still been a problem. Figure 4.2 shows the development kit in position.

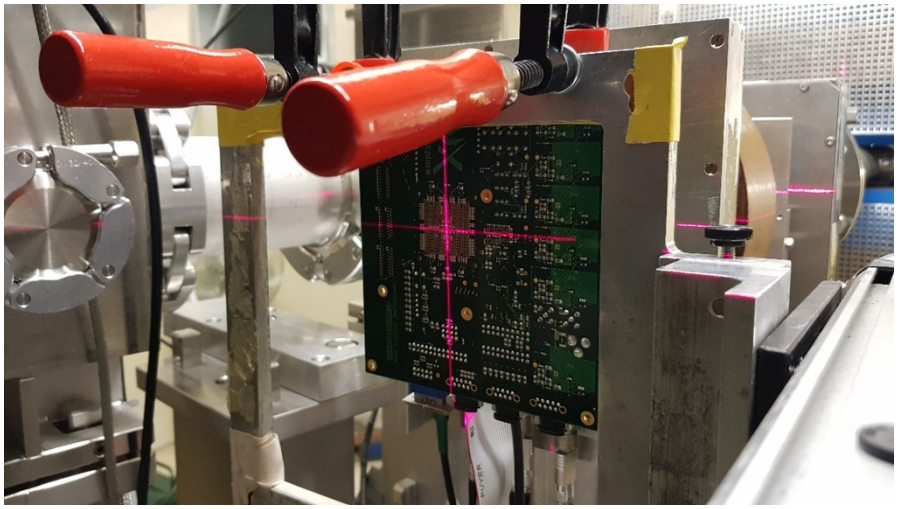


Figure 4.2: NG-Medium aligned with the beam

4.3.2 Evaluation metrics

During the irradiation campaigns we carried out, we could observe two kinds of events. In the first place, we could observe SEUs that we divided into two categories thanks to their error signature:

- SEUs that affect B13 circuits. We can recognize this kind of event because every time the output of a B13 is wrong, we reset all the instances, expecting a correct output at the next clock cycle. Thus, if the error lasts for a single cycle, we consider that the SEU affected the logic circuit itself (an SEU in a FF, or a captured SET). We refer to this event with its cross-section σ_{seu_b13}, which we express using Eq. (4.2), where N_{B13} refers to the number of B13 we placed inside the FPGA;

$$\sigma_{seu\_b13} = \frac{\#SEUs}{N_{B13} \times Fluence} \qquad [\text{cm}^2/\text{circuits}] \tag{4.2}$$

- SEUs that affect the configuration memory (CRAM). In this case, we observe that a B13 produces a wrong output for multiple clock cycles. Since this is not the behaviour we expect, we can imply that the SEU affected a CRAM's bit rather then the circuit logic, making the design fail continuously until the CMIC engine corrects the error. We refer to this event using its cross-section σ_{CMIC_b13}, expressed in Eq. (4.3). In this case, we compute the cross-section as a function of the number of circuits we placed.

$$\sigma_{CMIC\_b13} = \frac{\#SEUs}{N_{B13} \times Fluence} \qquad [\text{cm}^2/\text{circuits}] \tag{4.3}$$

  Moreover, in an SRAM FPGA, the user design occupies only a fraction of the total bits available, called 'essential bits'. However, not all of them lead to a failure. For example, there may be LUTs in the design that are not used at 100%, but are still included in the analysis. Thus, only a fraction of these bits, called *critical bits* can actually lead to the design failure. Hence, we retrieved the cross-section for the same event as a function of the design critical bits reported by the tool. We express it using Eq. (4.4). In this way, we could try to retrieve some information about the differences between the TMR and no-TMR design. The number of critical bits we will use is estimated by the NanoXplore tool that we used for the place and route.

$$\sigma_{CMIC\_bits} = \frac{\#SEUs}{\text{Critical bits} \times Fluence} \qquad [\text{cm}^2/\text{bits}] \tag{4.4}$$

Secondly, we could observe another kind of event not related to the design itself, but rather to the FPGA itself. We had a radiation-induced reset that triggered the reprogramming of the FPGA. We refer to this event with the cross-section σ_{reset}, and we compute it with Eq. (4.5). We computed it as a function of the device.

$$\sigma_{reset} = \frac{\#resets}{Fluence} \qquad [\text{cm}^2/\text{device}] \tag{4.5}$$

Finally, we also dumped the configuration memory of the FPGA before and after the irradiation. Thanks to the vendor tool, we could compare the two dumps and retrieve the number of bits that

were different, which represents the SEU affecting the CRAM. We refer to this event with the cross-section σ_{CRAM}, expressed in Eq. (4.6).

$$\sigma_{CRAM} = \frac{\#SEUs}{CRAMsize \times Fluence} \qquad [\text{cm}^2/\text{bits}] \tag{4.6}$$

When irradiating the device, we set a target fluence for the instrument, and when we reach it, the beam automatically stops. We refer to this cycle of operation, beam start-irradiation-beam stop, as a RUN. We save the number of SEEs registered for every run, and then we proceed with the next run. However, for simplicity, we will not report the results for all the single runs, but we will provide their combination. For each cross-section, we also report the values for the confidence interval we calculated according to [10].

4.3.3 Benchmark application

The first results refer to the benchmark circuit without the TMR mitigation technique. The circuit we used contains few replica (either 96 or 88), which is a relatively low number if we consider that the FPGA is much bigger. However, at the moment of the campaign, the NanoXplore tool used to generate the bitstream was affected by a bug that did not allow to use the entire FPGA, therefore we were limited to these numbers.

First, we performed some runs enabling the CMIC engine, and then we disabled it to study the differences.

CMIC enabled

Table 4.1 shows the events affecting the B13s and the corresponding cross-section σ_{seu_b13}. As we can see, we observed very few errors on the B13 instances, giving a total cross-section for this event of 6.37×10^{-14}.

Fluence	# B13s	# SEUs	Lower limit	$\sigma_{\text{seu\_b13}}$	Upper limit
$4.90 \cdot 10^{11}$	96	3	$1.27 \cdot 10^{-14}$	$6.37 \cdot 10^{-14}$	$1.86 \cdot 10^{-13}$

Table 4.1: Benchmark: SEUs cross-section

Concerning the CMIC events, we monitored the error detected signal with an oscilloscope, and we report in Fig. 4.3 the error profile of one of the CMIC events we had. As we can see, the device stays in a faulty state for multiple clock cycles, for some milliseconds, which corresponds to the time the CMIC needs to scan and correct the error. All the events had more or less the same duration. We report in Tab. 4.2, the cross-section for these events as a function of the critical bits, whereas in Tab. 4.3 as function of the circuits. We obtained very few SEUs on the B13 instances and more on the CRAM, as we can see from the tables. Even though we had many events for the configuration memory, the design did not fail permanently, thanks to the CMIC engine. The CMIC runs periodically (we used the minimum period of 5.3 ms), and it takes around 4 ms to perform a scan of the configuration memory. Therefore, the design can keep failing up to this time, after which the CMIC corrects the error. Nonetheless, we observed a permanent design failure that never recovered during one run of the first campaign. At the same time, we also observed an increase

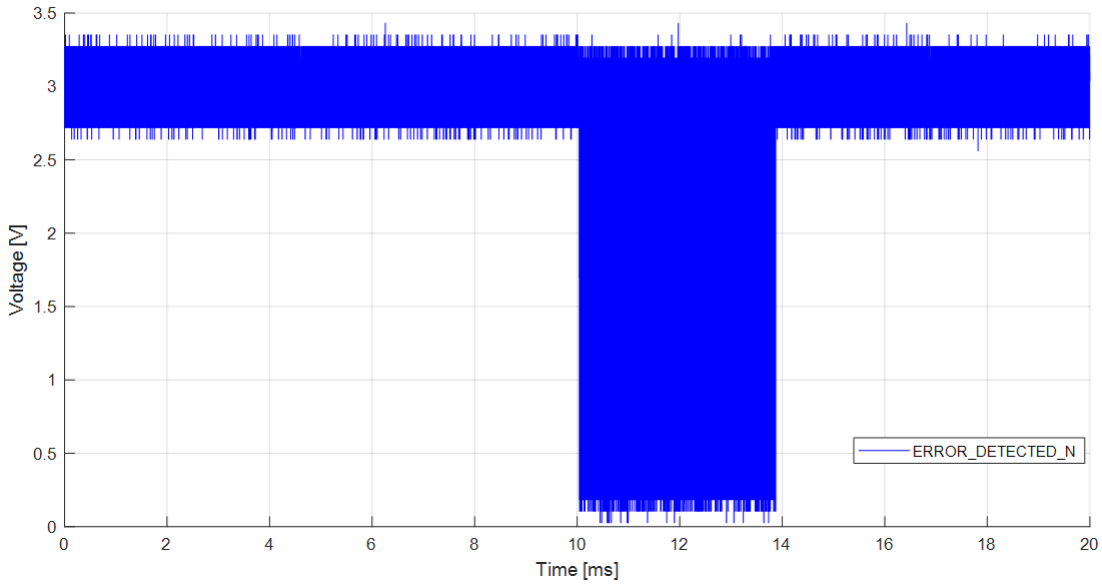


Figure 4.3: Error signal waveform captured with the oscilloscope

Fluence	Critical bits	CMIC	Lower limit	$\sigma_{\text{cmic\_bits}}$	Upper limit
$4.90 \cdot 10^{11}$	601247	20	$2.04 \cdot 10^{-18}$	$6.78 \cdot 10^{-17}$	$2.98 \cdot 10^{-17}$
$3.00 \cdot 10^{11}$	512796	15	$3.90 \cdot 10^{-18}$	$9.75 \cdot 10^{-17}$	$5.70 \cdot 10^{-17}$

Table 4.2: Benchmark: CMIC events cross-section as function of critical bits

in the device's current consumption. We can deduce that this behaviour is the consequence of a double error inside the CRAM, which stops the CMIC engine. When the CMIC stops working, and single-bit errors start to accumulate inside the CRAM, the device will never recover from a faulty state as it does when the CMIC is enabled. Table 4.4 reports the cross-section for this event that we indicated with $\sigma_{\text{permanent}}$ as a function of the critical bits and total CRAM size.

After the irradiation, we dumped the configuration memory of the FPGA and compared it to the golden reference saved before the irradiation. We could compare the two configurations thanks to the NanoXplore tool. We had many differences between the two configurations during the first campaign, which probably indicates that the dump failed. In the second campaign, instead, we performed a dump of the memory at the end of the test, while the FPGA was still operational. In that case we retrieved a single-bit of difference. Table 4.5 shows the cross-section for this event. The presence of this error indicates that the CMIC engine did not correct it; however, according to the datasheet, the CMIC only stops working when it detects a double error. This does not seem the situation, which suggests that probably the CMIC engine stopped for other reasons. In any case, the design was still working correctly.

	Fluence	# B13s	# SEUs	Lower limit	$\sigma_{\text{cmic\_b13}}$	Upper limit
	$4.90 \cdot 10^{11}$	96	20	$1.28 \cdot 10^{-1e}$	$4.25 \cdot 10^{-13}$	$1.87 \cdot 10^{-13}$
	$3.00 \cdot 10^{11}$	88	15	$2.27 \cdot 10^{-14}$	$5.68 \cdot 10^{-13}$	$3.32 \cdot 10^{-13}$
Total	$7.90 \cdot 10^{11}$	184	35	$4.13 \cdot 10^{-15}$	$2.41 \cdot 10^{-13}$	$6.04 \cdot 10^{-14}$

Table 4.3: Benchmark: CMIC events cross-section as function of the circuit

Fluence	Metric	Size	Lower limit	$\sigma_{\text{permanent}}$	Upper limit
$4.90 \cdot 10^{11}$	Critical bits	601247	$2.04 \cdot 10^{-18}$	$3.39 \cdot 10^{-18}$	$2.98 \cdot 10^{-17}$
	CRAM	6138096	$1.99 \cdot 10^{-19}$	$3.32 \cdot 10^{-19}$	$2.92 \cdot 10^{-18}$

Table 4.4: Benchmark: Permanent failure cross-section as function of the critical bits and CRAM size

CMIC disabled

We also performed two runs with the CMIC engine disabled. In this case, we did not observe any SEUs affecting the B13, but we observed only SEUs affecting the CRAM. However, as expected, as soon as one an SEU affects one of the critical bits inside the CRAM, it leads to the permanent failure, because there is no scrubbing that can restore its value. Thus, the only possibility would be to reload again the bitstream. During these two runs, then, we irradiated the device until it failed. Table 4.6 reports the cross-sections for this event. Being a corruption of the CRAM, we indicated it with $\sigma_{\text{cmic\_bits}}$. As we expected, the cross-section is comparable with the CMIC event cross-section we observed when the engine was enabled, reported in Tab. 4.2, since the two events are exactly the same. The only difference is that in this case there is no engine correcting it. In addition, at the time of the failure, we observed the same current behaviour we had when the design permanently failed and the CMIC was enabled.

4.3.4 Benchmark application - TMR

We tested the same benchmark application applying the TMR technique to investigate if it could reduce the cross-sections for the events we observed. Even in this case, we could not use the entire FPGA, because of the placing tool. Therefore, the number of circuits we placed is limited to 32 (in the TMR case, we consider a circuit as the combination of three B13 instances and their voter).

CMIC enabled

Starting with the CMIC enabled, we retrieved, unexpectedly, some events affecting the B13 circuits. The results are described in Table 4.7. As we can see, the use of TMR did not reduce much the cross-section for this event, considering that we obtained 6.37×10^{-14} in the no-TMR case. However, having the same cross-section as the non-TMR version implies that the probability of an SEU affecting three B13 instances simultaneously is the same as affecting a single instance, which is very unlikely. Therefore, most probably, the errors are due to an SET on the additional logic we introduced to interface with the TESTER, or in the voting logic we introduced by TMR.

Fluence	CRAM size	# CRAM SEUs	Lower limit	σ_{CRAM}	Upper limit
$3.00 \cdot 10^{11}$	6138096	1	$3.26 \cdot 10^{-19}$	$5.43 \cdot 10^{-19}$	$4.77 \cdot 10^{-18}$

Table 4.5: Benchmark: CRAM SEUs cross-section

Critical bits	Fluence to failure	Lower limit	$\sigma_{\text{permanent}}$	Upper limit
601247	$9.16 \cdot 10^{9}$	$1.09 \cdot 10^{-16}$	$1.82 \cdot 10^{-16}$	$1.59 \cdot 10^{-15}$
	$1.10 \cdot 10^{10}$	$9.07 \cdot 10^{-17}$	$1.51 \cdot 10^{-16}$	$1.33 \cdot 10^{-15}$
Total	$2.02 \cdot 10^{10}$	$4.94 \cdot 10^{-17}$	$1.65 \cdot 10^{-16}$	$7.23 \cdot 10^{-16}$

Table 4.6: Benchmark: CMIC event cross-section for the benchmark application

We could also observe the CMIC events, that were corrected thanks to the CMIC engine. We report in Table 4.8 the cross-section for this event as a function of the critical bits. From these results, we can see that the CMIC event cross-section computed as a function of the critical bits is lower than the no-TMR version we reported in Tab. 4.2. However, since this error is a consequence of an SEU in the configuration memory, the cross-section in the two cases should be comparable. Since it is not the case, it probably indicates that the number of critical bits estimated by the tool does not reflect reality, probably because the tool does not understand the nature of the TMR configuration. Thus, such a metric does not seem suitable for characterization.
We report int Tab. 4.9 the cross-section for the same event but as a function of the circuits. The total cross-section is not much different from the no-TMR version reported in Tab. 4.3. The reason is probably that an SEU affected the configuration bit coding the path of the single point of failure (SPF). When we apply TMR, we add a voter to the system and compare its output with a reference. Thus, it is clear that if an SEU affects the configuration bit of the voter's output, the design fails, and this is very likely the reason why we have two similar cross-sections with and without TMR. Mitigation techniques to reduce the errors coming from SPF when using TMR exist [7]. They consist of manually placing the elements inside the FPGA. However, at the time of writing, there was no possibility to apply these techniques on the NG-Medium FPGA because the tools were still undergoing a development phase. Even for the TMR version, we were monitoring the error signal using an oscilloscope, and we could verify that the duration of these errors is a few milliseconds, which is comparable with the time required by the CMIC engine to correct it.

In between the different runs, we dumped the configuration memory and compared it with the golden configuration. Table 4.10 shows the CRAM SEUs and their cross-section. Between run 54 and 56, we did not reprogram the FPGA; hence, the errors accumulated. However, even if the CMIC was enabled, we have many SEUs in the configuration memory not corrected, which means that the CMIC probably stopped working. By considering the average cross-section and the number of SEUs in run 55, we estimated the fluence at which the CMIC stopped working to be 8.25×10^{10}. This fluence corresponds to a cross-section of 1.84×10^{-18}, comparable with the cross-section for the double error reported by NanoXplore in its radiation test [16].

Fluence	#B13s	#SEUs	Lower limit	$\sigma_{\text{seu\_b13}}$	Upper limit
$1.32 \cdot 10^{12}$	32	4	$1.42 \cdot 10^{-14}$	$9.47 \cdot 10^{-14}$	$2.08 \cdot 10^{-13}$

Table 4.7: Benchmark-TMR: B13 SEUs cross-section

Fluence	Critical bits	#CMIC events	Lower limit	$\sigma_{\text{cmic\_bits}}$	Upper limit
$1.32 \cdot 10^{12}$	590846	5	$7.69 \cdot 10^{-19}$	$6.41 \cdot 10^{-18}$	$1.13 \cdot 10^{-17}$
$1.33 \cdot 10^{12}$	607344	6	$7.43 \cdot 10^{-19}$	$7.43 \cdot 10^{-18}$	$1.09 \cdot 10^{-17}$

Table 4.8: Benchmark-TMR: CMIC event cross-section as function of the critical bits

CMIC disabled

We also performed irradiation with the CMIC engine disabled. Table 4.11 shows the cross-section we retrieved, again similar to the non-TMR version. Being the CMIC disabled, the design entered a permanent failure, as expected. Moreover, we observed a current consumption increase simultaneously as the error.

Even in this case, we dumped the configuration memory after the runs and compared it with the golden reference. The results in Table 4.12 show that we obtain the same cross-section as in the non-TMR design, as we were expecting.

4.3.5 Radiation-induced reset

Concerning the unexpected resets, we observed them during different runs in both the campaigns we performed. We configured the FPGA to load the bitstream from the flash memory at every start-up. However, even during irradiation, we could see that the bitstream was being reloaded again, implying that a reset occurred. We could understand this behaviour because we monitored the current profile while programming the FPGA before the irradiation. Then, during the irradiation, we put in place an algorithm to look for the specific signature. Since the current profiles we observed before and during the irradiation were the same, we concluded that a reset occurred. Table 4.13 reports the cross-section for this event.

The source of this reset was not clear. After a discussion with the company, we understood that this reset might come from the SpaceWire module (one of the module available in the FPGA). Nonetheless, we disabled it during the second campaign, without any results leaving the cause unknown.

4.3.6 Final comments

As expected from SRAM-based FPGA, most SEUs are affecting configuration memory bits that, in case they are part of the user logic, lead to the design failure. However, thanks to the CMIC engine NanoXplore developed, the FPGA recovers from these errors after few milliseconds. In combination with the Radiation Hardened manufacturing process, this technique makes the reliability of this FPGA in harsh environment better compared to other kind of SRAM-based FPGA. Concerning the TMR mitigation technique, we observed values similar to the version without any

	Fluence	# B13s	CMIC Events	Lower limit	$\sigma_{\text{cmic\_b13}}$	Upper limit
	$1.32 \cdot 10^{12}$	32	5	$1.42 \cdot 10^{-14}$	$1.18 \cdot 10^{-13}$	$2.08 \cdot 10^{-13}$
	$1.33 \cdot 10^{12}$		6	$1.41 \cdot 10^{-14}$	$1.41 \cdot 10^{-13}$	$2.06 \cdot 10^{-13}$
Total	$2.65 \cdot 10^{12}$		11	$7.07 \cdot 10^{-15}$	$1.30 \cdot 10^{-13}$	$1.03 \cdot 10^{-13}$

Table 4.9: Benchmark-TMR: CMIC event cross-section as function of the circuits

Run #	CRAM size	Fluence	# CRAM SEU	Lower limit	σ_{CRAM}	Upper limit
54		$1.00 \cdot 10^{11}$	0	-	-	-
55		$1.00 \cdot 10^{11}$	97	$9.77 \cdot 10^{-19}$	$1.58 \cdot 10^{-16}$	$1.43 \cdot 10^{-17}$
56	6138096	$1.00 \cdot 10^{11}$	269	$9.77 \cdot 10^{-19}$	$4.38 \cdot 10^{-16}$	$1.43 \cdot 10^{-17}$
		FPGA RE-PROGRAMMING				
59		$1.72 \cdot 10^{11}$	56	$5.68 \cdot 10^{-19}$	$5.30 \cdot 10^{-17}$	$8.31 \cdot 10^{-18}$

Table 4.10: Benchmark-TMR: CRAM SEUs cross-section

mitigation, mainly because of the many SPF generated during the placement of the design, since the tools were still in development at the time this work was performed; thus, the placement/routing logic was very poor. However, we can expect better performances in case the tool would allow the manual placement of the design. Finally, we had different radiation-induced resets, that triggered an FPGA reprogramming. For our tests, we loaded the design inside an external flash memory, and we configured the FPGA to download the bitstream from the flash into the configuration memory. Nonetheless, in operation we can not not use a flash memory to store the user design because commercial flash memories have a short TID lifetime, which makes loose all the advantages of having an SRAM-based Rad-Hard FPGA.

4.4 PolarFire characterization

The PolarFire is a Flash-based, non-volatile FPGA. It is built on state-of-the-art 28nm non-volatile process technology. It is cost-optimized, delivering the lowest power at mid-range densities. The fabric is composed of logic elements, containing a 4-input LUT and a Flip-Flop, memory blocks, and Math blocks for arithmetic operations. Its configuration memory cells are SEU immune due to the flash memory technology, which cannot be affected by SEE. The device uses the SONOS technology which brings many advantages in terms of costs, size and reliability.
We performed four irradiation campaigns to understand if it is a good candidate for CERN applications: three at PSI for the protons qualification and one at ILL for the thermal neutrons qualification. We tested all the elements of the FPGA fabric, which are: FFs, Math blocks, memory blocks and PLL. Then, we used the benchmark circuit to compare it to the NG-Medium FPGA. Finally, we also measured the degradation due to TID effects using the TID monitoring circuit we developed.

Fluence to failure	Critical bits	Lower limit	$\sigma_{\text{cmic\_bits}}$	Upper limit
$9.87 \cdot 10^{10}$	590846	$1.03 \cdot 10^{-17}$	$1.71 \cdot 10^{-17}$	$1.50 \cdot 10^{-16}$
$4.68 \cdot 10^{10}$	607344	$2.11 \cdot 10^{-17}$	$3.52 \cdot 10^{-17}$	$3.09 \cdot 10^{-16}$

Table 4.11: Benchmark-TMR: CMIC event cross-section with CMIC disabled

Fluence	CRAM size	CRAM SEUs	Lower limit	σ_{CRAM}	Upper limit
$1.69 \cdot 10^{11}$	6138096	227	$5.78 \cdot 10^{-19}$	$2.19 \cdot 10^{-16}$	$8.46 \cdot 10^{-18}$
$5.35 \cdot 10^{10}$		153	$1.83 \cdot 10^{-18}$	$4.66 \cdot 10^{-16}$	$2.67 \cdot 10^{-17}$

Table 4.12: Benchmark-TMR: CRAM SEUs with CMIC disabled

4.4.1 Test setup

The evaluation board we used to carry out the qualification tests is the MPF300-EVAL-KIT. The board contains many components around the FPGA, but thanks to the adjustable beam size, we could focus the beam only on the FPGA itself, reducing the probability of having SEL from the other components. Figure 4.4 and 4.5 shows the development kit in position at PSI and Grenoble, respectively. The test setup we used for the campaigns consists of the TESTER platform, the MicroZed, and the other instruments, placed in the access corridor to avoid activation. The TESTER and other instruments are connected to the host computer in the control room, from where we can control every aspect of the test. Figure 4.6 shows an overview of the test setup.

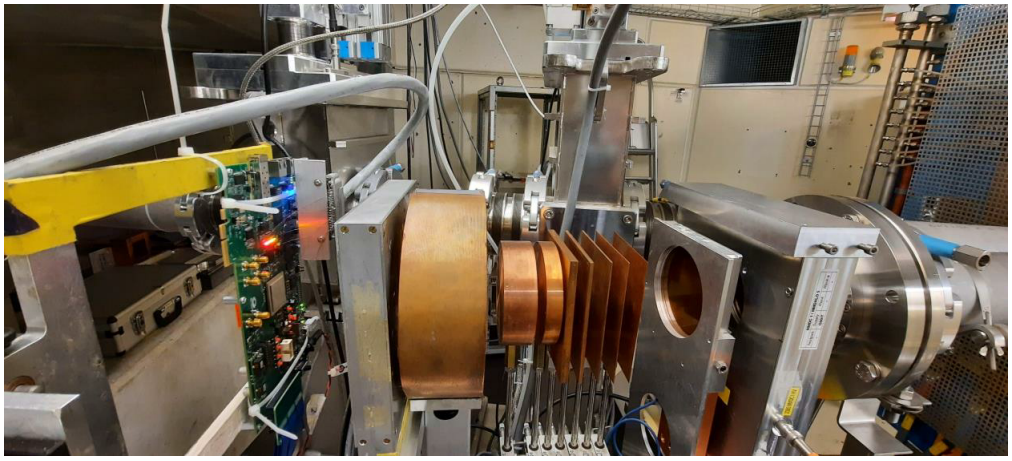


Figure 4.4: PolarFire FPGA in position in front of the beam at the PIF facility

4.4.2 Memory blocks sensitivity

The PolarFire comes with two types of memory blocks, the large SRAM (LSRAM) and micro SRAM (uSRAM), the main difference being the size of the embedded blocks: 20Kbits for the

	Fluence	# Resets	Lower limit	σ_{reset}	Upper limit
	$1.95 \cdot 10^{12}$	2	$3.08 \cdot 10^{-13}$	$1.03 \cdot 10^{-12}$	$4.50 \cdot 10^{-12}$
	$1.68 \cdot 10^{12}$	2	$3.57 \cdot 10^{-13}$	$1.19 \cdot 10^{-12}$	$5.22 \cdot 10^{-12}$
Total	$3.63 \cdot 10^{12}$	4	$1.65 \cdot 10^{-13}$	$1.10 \cdot 10^{-12}$	$2.42 \cdot 10^{-12}$

Table 4.13: Radiation-induced reset cross-section

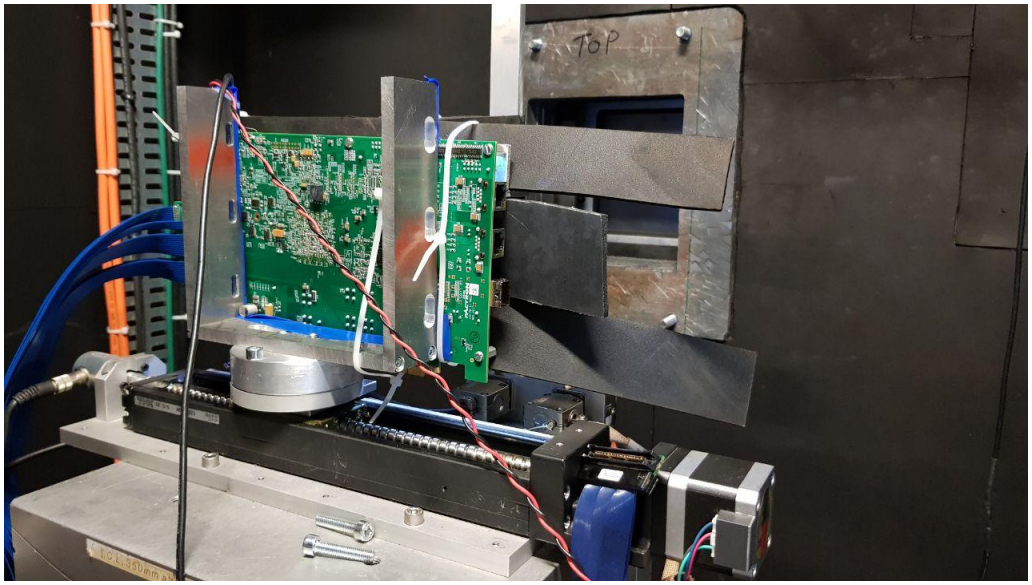


Figure 4.5: PolarFire FPGA in position in front of the beam at the ILL facility

LSRAM and 768 bits for the uSRAM. For our purpose, we combined multiple blocks into a single memory on which we could perform read and write operations using our Memory tester IP (Section 3.3.2). We created a 20x524288 memory using LSRAM blocks and a 12x131072 memory using uSRAM blocks. In the results, we show the fluence, the total memory size (in bits), and the pattern we used to perform the test.

Proton test

Type	Fluence	Size	Pattern	SEUs	MBUs	σ_{seu}	σ_{mbu}
LSRAM	$1,00 \cdot 10^{10}$	10485760	Static 0	1379	0	$1,32 \cdot 10^{-14}$	$< 9.54 \cdot 10^{-18}$
			Static 1	1428	0	$1,36 \cdot 10^{-14}$	
			Static 1	1508	0	$1,44 \cdot 10^{-14}$	
Total	$3,00 \cdot 10^{10}$			4315	0	$1,37 \cdot 10^{-14}$	$< 9.54 \cdot 10^{-18}$

Table 4.14: LSRAM SEUs cross-section against protons

Starting from the memory composed of LSRAM blocks without ECC, we report the results in

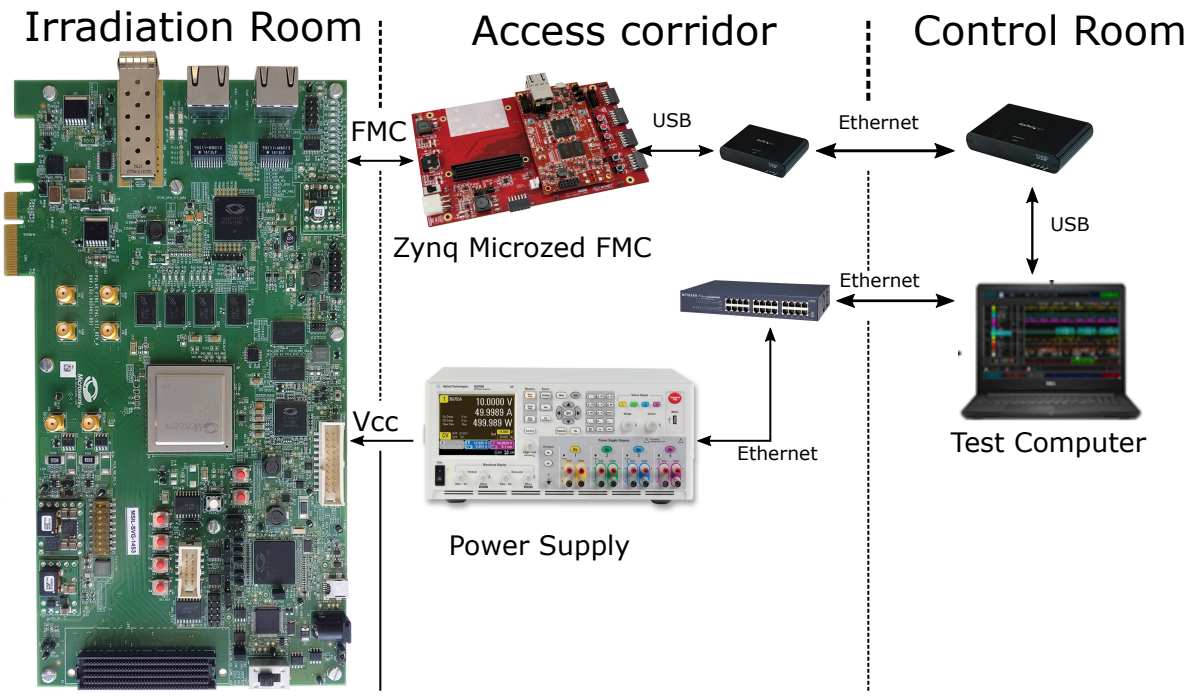


Figure 4.6: PolarFire FPGA test setup

Table 4.14. We did not observe any MBUs, resulting in a cross-section lower than $9.54 \cdot 10^{-18}$ cm^2. These results are compatible with those reported by Microsemi with a 10 MeV neutron field [15].

Type	Fluence	Size	Pattern	SEUs	MBUs	σ_{seu}	σ_{mbu}
uSRAM	$1,00 \cdot 10^{10}$	1572864	Static 0	188	11	$1.20 \cdot 10^{-14}$	$6.99 \cdot 10^{-16}$
			Static 1	492	63	$3.13 \cdot 10^{-14}$	$4.01 \cdot 10^{-15}$
			Static 1	89	0	$5.66 \cdot 10^{-15}$	$< 6.36 \cdot 10^{-17}$
Total	$3,00 \cdot 10^{10}$			769	74	$1.63 \cdot 10^{-14}$	$1.57 \cdot 10^{-15}$

Table 4.15: uSRAM SEUs cross-section against protons

On the contrary, the memory composed of uSRAM blocks did not show a consistent behaviour among the three runs we performed. We had MBUs on the first two runs and none on the third. Table 4.15 shows the average cross-section. During this test, we could not understand which addresses were failing, so the inconsistency in the MBU cross-section may be a consequence of a SEFI affecting entire blocks of the memory. We planned more tests to investigate this problem.

4.4.3 Flip-Flops sensitivity

To retrieve the Flip-Flops SEU cross-section, we instantiated multiple times the chains we described in3.1.1. Table 4.16 contains the description of the chains we used. During the test, we observed many errors suddenly affecting many structures. Since this error was also affecting the other circuit

Type	# Chains	# FFs per chain	Window size	# Inverters	TMR
WSR	4	8000	4	0	N
WSR-TMR	2	8000	1	0	Y
WSR-SET	4	8000	4	6	N
WSR-SET-TMR	2	8000	4	6	Y

Table 4.16: FFs chains characteristics used for the PolarFire characterization

we implemented (MACC and PLLs), we think they may result from a SEFI; therefore, we did not include them in the SEU analysis, and we will present them in a separate section.

Proton test

Concerning the proton tests, we performed the static and the dynamic test at different frequencies, ranging from 16 MHz to 225 MHz to understand if it impacted the final cross-section. We report in

Frequency [MHz]	Fluence	WSR		WSR-SET		WSR TMR		WSR-SET TMR	
		SEUs	σ	SEUs	σ	SEUs	σ	SEUs	σ
16	$2,00 \cdot 10^{11}$	34	$5,31 \cdot 10^{-15}$	30	$9,38 \cdot 10^{-15}$	0	-	1	$3,13 \cdot 10^{-16}$
32	$3,52 \cdot 10^{11}$	50	$4,44 \cdot 10^{-15}$	51	$9,06 \cdot 10^{-15}$	4	$3,55 \cdot 10^{-16}$	11	$1,95 \cdot 10^{-15}$
65	$1,15 \cdot 10^{12}$	163	$4,43 \cdot 10^{-15}$	170	$9,24 \cdot 10^{-15}$	8	$2,17 \cdot 10^{-16}$	43	$2,34 \cdot 10^{-15}$
112	$2,00 \cdot 10^{11}$	32	$5,00 \cdot 10^{-15}$	29	$9,06 \cdot 10^{-15}$	2	$3,13 \cdot 10^{-16}$	14	$4,38 \cdot 10^{-15}$
175	$2,00 \cdot 10^{11}$	38	$5,94 \cdot 10^{-15}$	48	$1,50 \cdot 10^{-14}$	13	$2,03 \cdot 10^{-15}$	31	$9,69 \cdot 10^{-15}$
225	$2,00 \cdot 10^{11}$	39	$6,09 \cdot 10^{-15}$	40	$1,25 \cdot 10^{-14}$	11	$1,72 \cdot 10^{-15}$	39	$1,22 \cdot 10^{-14}$
# FF		32000		16000		32000		16000	

Table 4.17: SEU cross-section against proton for PolarFire FFs in static test

Tab. 4.17 the results of the irradiation runs. As we can see, the standard WSR and the SET-adapted WSR showed only a relatively small increase of their cross-section with the frequency increase. Instead, the chains where we used TMR show a consistent increase in the SEU cross-section with the frequency increase, probably because increasing the frequency also increases the probability to capture SET. Moreover, the usage of TMR reduced the FF sensitivity by one order of magnitude.

Frequency [MHz]	Fluence	WSR		WSR-SET		WSR TMR		WSR-SET TMR	
		SEUs	σ	SEUs	σ	SEUs	σ	SEUs	σ
16	$1,00 \cdot 10^{11}$	74	$2,31 \cdot 10^{-14}$	57	$3,56 \cdot 10^{-14}$	10	$3,13 \cdot 10^{-15}$	13	$8,13 \cdot 10^{-15}$
32	$1,00 \cdot 10^{11}$	61	$1,91 \cdot 10^{-14}$	51	$3,19 \cdot 10^{-14}$	14	$4,38 \cdot 10^{-15}$	17	$1,06 \cdot 10^{-14}$
65	$1,00 \cdot 10^{11}$	85	$2,66 \cdot 10^{-14}$	47	$2,94 \cdot 10^{-14}$	5	$1,56 \cdot 10^{-15}$	17	$1,06 \cdot 10^{-14}$
112	$1,00 \cdot 10^{11}$	78	$2,44 \cdot 10^{-14}$	63	$3,94 \cdot 10^{-14}$	28	$8,75 \cdot 10^{-15}$	25	$1,56 \cdot 10^{-14}$
175	$1,00 \cdot 10^{11}$	93	$2,91 \cdot 10^{-14}$	39	$2,44 \cdot 10^{-14}$	31	$9,69 \cdot 10^{-15}$	25	$1,56 \cdot 10^{-14}$
225	$7,36 \cdot 10^{10}$	67	$2,84 \cdot 10^{-14}$	42	$3,57 \cdot 10^{-14}$	20	$8,49 \cdot 10^{-15}$	19	$1,61 \cdot 10^{-14}$
# FF		32000		16000		32000		16000	

Table 4.18: SEU cross-section against protons for PolarFire FFs in dynamic test

We can observe similar behaviour for the dynamic test, whose results are in Tab. 4.18. The two non-triplicated structures show a constant cross-section through all the frequencies. Moreover, it is five times and three times lower than the corresponding cross-section in static mode. Instead, the chains with TMR show a cross-section increasing with the frequency, as in the static test. However, the increasing factor is lower.

Thermal neutron test

Using the same chains composition, we performed the same tests for thermal neutrons. Table 4.19 and 4.20 contains the SEUs we retrieved for static and dynamic test mode, respectively.

Frequency [MHz]	**Fluence**	**WSR**		**WSR-SET**		**WSR TMR**		**WSR-SET TMR**	
		SEUs	σ	SEUs	σ	SEUs	σ	SEUs	σ
5	$7,26 \cdot 10^{11}$	44	$1,89 \cdot 10^{-15}$	25	$2,15 \cdot 10^{-15}$	0	-	1	$8,61 \cdot 10^{-17}$
30	$5,50 \cdot 10^{11}$	44	$2,50 \cdot 10^{-15}$	18	$2,05 \cdot 10^{-15}$	0	-	0	-
# FF		32000		16000		32000		16000	

Table 4.19: SEU cross-section against neutrons for PolarFire FFs in static test

Frequency [MHz]	**Fluence**	**WSR**		**WSR-SET**		**WSR TMR**		**WSR-SET TMR**	
		SEU	σ	SEU	σ	SEU	σ	SEU	σ
30	$5,96 \cdot 10^{11}$	88	$4,61 \cdot 10^{-15}$	34	$3,57 \cdot 10^{-15}$	2	$1,05 \cdot 10^{-16}$	3	$3,15 \cdot 10^{-16}$
# FF		32000		16000		32000		16000	

Table 4.20: SEU cross-section against neutrons for PolarFire FFs in dynamic test

As we can see from the table, we obtained a smaller cross-section for the chains where we applied triplication in both static and dynamic test mode.

4.4.4 Math blocks sensitivity

The PolarFire includes embedded math blocks optimized for Digital Signal Processing applications. The block includes a built-in multiplier, a pre-adder, and an adder, thus minimizing the external logic required to implement these operations. To retrieve the sensitivity of this block, we instantiated many clusters as described in Section 3.1.2 for a total of 80 MACC blocks. Table 4.21 describes the cluster we used. Microsemi did not provide any data concerning their sensitivity, and no other publications studied the MACC block of this FPGA. Therefore, we based ourselves on the SmartFusion 2 FPGA test performed at CERN, and we were expecting to retrieve many events with this number of MACC blocks.

Using the same circuit, we performed the irradiation test against protons and thermal neutrons, and we report in Tab. 4.22 the two cross-sections. As we can see, we retrieved very few events, and so it is impossible to establish an accurate cross-section for this kind of block. Therefore, other tests must be performed in the future to address the sensitivity. Nevertheless, we can say that the sensitivity is much lower than the SmartFusion2, representing a good point for CERN applications.

Type	# MACC	Register type	TMR
MULT	8	Internal	**N**
MULT	8	External	**N**
MULT ADD	8	Internal	**N**
MULT ADD	8	External	**N**
MULT	8	Internal	**Y**
MULT ADD	8	Internal	**Y**

Table 4.21: MACC cluster description

Particle	# Math blocks	Fluence	# SEUs	σ
Protons	80	$2.5 \cdot 10^{12}$	3	$1.5 \cdot 10^{-14}$
Neutrons		$1.15 \cdot 10^{12}$	2	$2.17 \cdot 10^{-14}$

Table 4.22: SEUs cross-section for MACC blocks

4.4.5 PLLs sensitivity

We studied the cross-section of the PLLs monitoring the LOCK signal thanks to the PLL tester IP. Even though we observed some events, they were not affecting only the PLLs but also the other circuits that we were testing, the Flip-Flops and Math blocks, which we were driving using another clock signal. Therefore, it is very likely that a SEFIs rather than an SEU occurred at that moment. Consequently, we can say that we did not observe any actual loss of locks on the eight PLLs we tested. We experienced this situation for both the neutron and proton test. This lack of events means that the SEU cross-section for the PLL might be lower than the SmartFusion PLL cross-section, estimated to be around $3.21 \cdot 10^{-14}$ cm^2/PLL, in the case of protons.

4.4.6 Benchmark application

We tested the benchmark application, both the normal and TMR version, at PSI; hence the results we present here are for protons only. For this circuit, we consider a failure whenever we detect an error, no matter the number of bits that are different from the golden reference. Even in this case, we observed that all the circuits were failing simultaneously during one irradiation run, and we considered this kind of event a SEFI, so we did not include them in this analysis.

Fluence	# B13s	TMR	# SEUs	σ_{b13}
$3.00 \cdot 10^{10}$	2048	NO	42	$6.84 \cdot 10^{-13}$
$4.60 \cdot 10^{11}$	1024	YES	2	$1.36 \cdot 10^{-13}$

Table 4.23: Benchmark application SEUs cross-section against protons

In Tab. 4.23 we present the SEU cross-section as a function of the B13 instances. The benchmark circuit without triplication shows an SEU cross-section of $6.84 \cdot 10^{-13}$ cm^2/B13, while the

TMR version has a cross-section 5-times lower. We can also notice how the difference between the two versions equals the difference observed between the non-TMR and TMR version of the Flip-Flop structure. This behaviour shows that the B13 cross-section is a good representation of the behaviour at the functional element level.

4.4.7 SEFIs

During the irradiation runs, we had many errors affecting all the structures. We attributed these errors to SEFIs, and we computed their cross-section separately from the SEU cross-section. In

Frequency [MHz]	Fluence [p.cm]	# SEFIs	σ
16	$3.00 \cdot 10^{11}$	5	$1.67 \cdot 10^{-11}$
32	$5.00 \cdot 10^{11}$	12	$2.40 \cdot 10^{-11}$
65	$1.25 \cdot 10^{12}$	31	$2.48 \cdot 10^{-11}$
112	$1.27 \cdot 10^{11}$	4	$3.15 \cdot 10^{-11}$
175	$1.00 \cdot 10^{11}$	2	$2.00 \cdot 10^{-11}$
225	$3.00 \cdot 10^{11}$	5	$1.67 \cdot 10^{-11}$
Total	$2.58 \cdot 10^{12}$	59	$2.29 \cdot 10^{-11}$

Table 4.24: SEFIs cross-section against protons for the standard circuits

Tab. 4.24 we show the SEFI cross-section we retrieved during the proton test of the standard circuits. In this case, we were testing Flip-Flops, Math blocks, and PLLs. However, the error was not affecting all the structures, but it was a stochastic event. If we assume that the error is a consequence of a SET affecting the global routing net, the SEFI affected only the structures close to the SET source because it could not reach farther structures because of the attenuation. Moreover, these errors happened only during the dynamic test mode, indicating that the problem's source was, most probably, the clock network. Nonetheless, the clock frequency does not seem to impact the SEFI occurrence since the cross-section is stable. In Tab. 4.25 we report instead the cross-section for the SEFIs we observed during the neutron test campaign.

Frequency [MHz]	Fluence	# SEFI	σ
5	$7.26 \cdot 10^{11}$	8	$1.10 \cdot 10^{-11}$
30	$5.50 \cdot 10^{11}$	4	$7.72 \cdot 10^{-12}$
30	$6.45 \cdot 10^{11}$	8	$1.24 \cdot 10^{-11}$
Total	$1.92 \cdot 10^{12}$	20	$1.04 \cdot 10^{-11}$

Table 4.25: SEFIs cross-section against neutrons for the standard circuits

We report in Table 4.26 the cross-section for the SEFIs events we observed when testing the benchmark circuits. In this case, we obtained events only for the TMR version, but with such low numbers, it is not easy to compare the two designs. Using the methodology expressed in [10], we computed the lower and upper limit.

TMR	Fluence	SEFIs	Lower limit	σ	Upper limit
NO	$3.00 \cdot 10^{10}$	0	-	$< 3.33 \cdot 10^{-11}$	$1.22 \cdot 10^{-10}$
YES	$1.00 \cdot 10^{11}$	1	$2.02 \cdot 10^{-13}$	$1.00 \cdot 10^{-11}$	$5.57 \cdot 10^{-11}$
YES	$3.60 \cdot 10^{11}$	2	$6.47 \cdot 10^{-13}$	$5.42 \cdot 10^{-12}$	$2.01 \cdot 10^{-11}$
Total	$4.90 \cdot 10^{11}$	3	$1.22 \cdot 10^{-12}$	$6.12 \cdot 10^{-12}$	$1.79 \cdot 10^{-11}$

Table 4.26: SEFIs cross-section against protons for the benchmark application

4.4.8 TID analysis

To qualify the FPGA against TID effects, we analyzed the programmability, the current consumption and the propagation delay increase.

Programmability

Concerning the programmability, we performed three different campaigns, where we reached doses of 500 Gy, 2kGy, and 3kGy, respectively, and we used the same evaluation board. During the first campaign, we reprogrammed the FPGA several times, and it failed only once. However, it worked the second trial. Then, we reprogrammed the FPGA at 171 Gy with success. We used the same evaluation board in the second campaign, which accumulated 360 Gy from the previous campaign and annihilated for eight months, without any problems. We could reprogram the FPGA several times. Finally, we used it without reprogramming up to 2 kGy. In the last campaign, after having accumulated a total dose of 3 kGy, we reprogrammed the board without any problems.

Current consumption

In the November campaign, we monitored the current profile of the board during the irradiation. The consumption was increasing depending on the clock frequency. Therefore, we report only the relative increase in current consumption in Fig. 4.7 to visualize the potential TID effects. The slow decrease and increase is most likely due to temperature effects. The two drops correspond to the two power cycles we performed, while the different spikes are correlated with the SEFIs we observed during the test. In summary, there is no significant change in the current consumption, which remained stable up to 2 kGy.

Propagation delay increase

Finally, we monitored the propagation delay increase using the TID monitoring circuit and the corresponding TID tester IP. We placed all the ring oscillators inside the FPGA manually to keep the consistency between the frequency of the ring oscillators. In total, we used 1952 ring oscillators, each one consisting of 47 inverters. We measured their frequencies before and after the irradiation. Figure 4.8 shows the histogram of the difference, in percentage, between the two measurements. As we can see, there is an average difference of 0.3%, meaning that TID did not impact the internal propagation delay of the FPGA, which represents an excellent result.

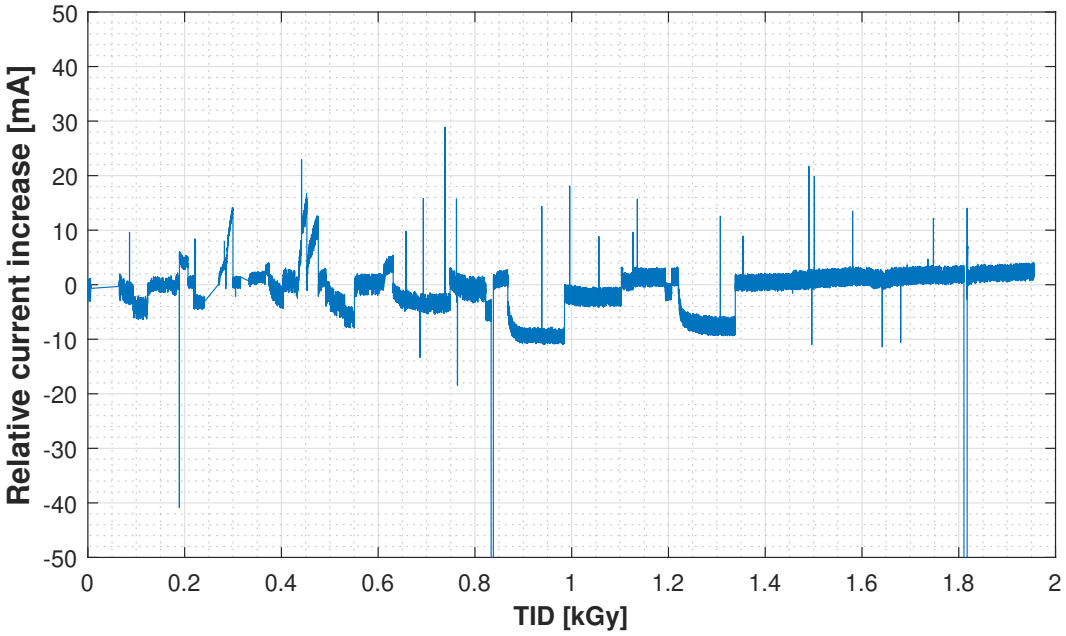


Figure 4.7: PolarFire relative current consumption variation during irradiation

4.4.9 Final comments

Comparing the PolarFire FPGA to its predecessor, that is the SmartFusion2, we can see that despite the smaller technology node, the Flip-Flops show in average a 2.5× lower cross-section. This difference increases even more, when we consider the Math blocks, considering that the equivalent block in the SmartFusion2 showed a cross-section two order of magnitude higher. In addition, we compare the thermal neutron and proton cross-section in Fig. 4.9. As we can see, for all the metrics, the thermal neutron cross-section is comparable to the proton cross-section. We discussed in Chapter 2 the importance of studying the thermal neutron sensitivity for of the smaller node technologies. These results confirm it and show how thermal neutrons may significantly impact the device when in operation.

4.5 Benchmark application analysis

Thanks to the benchmark applications, we can compare the two FPGAs obtaining an overview of their performances. In Fig. 4.10, we report the benchmark application cross-sections retrieved during our protons irradiation for both the NG-Medium and the PolarFire. Starting from the NG-Medium, we observed only SEUs affecting the configuration memory (CRAM), which caused the circuit design's corruption leading to its failure. However, we must remind that the NG-Medium comes with a memory scrubber, the CMIC, that periodically scans the memory and repairs single bit errors so that the design could recover from a faulty state. Given the Rad-Hard nature of the FPGA, we were not expecting to adopt any mitigation technique. However, we observed many SEUs affecting the CRAM; thus, we tested a TMR version of the same benchmark circuit. This technique did not improve the results, and further analysis demonstrated that it is probably due

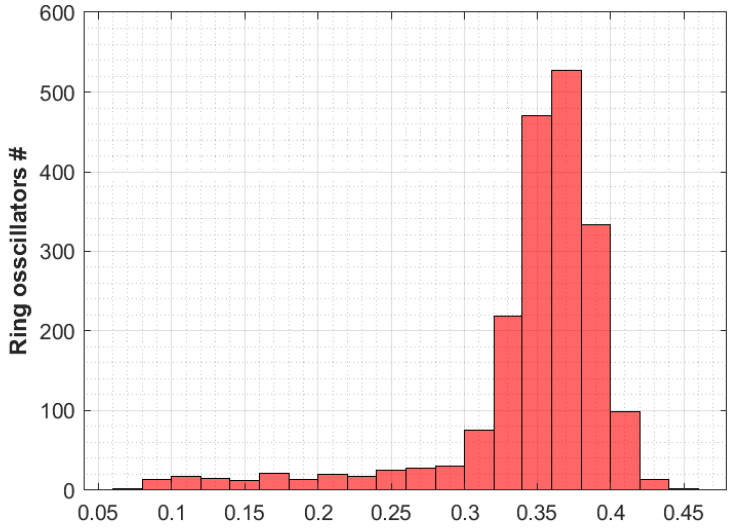


Figure 4.8: Frequency difference, in percentage, pre and post irradations of the FPGA internal logic

to the many SPF the placing/routing tool introduces, reducing the effectiveness of the mitigation technique. However, we can expect a high margin of improvements if the future updates of the tools allow a manual placement of the design.
On the other hand, the Polarfire is a flash-based FPGA; therefore, we do not have SEU affecting the configuration memory. In this case, the SEU may result from a bitflip inside the storage element or a SET on combinatorial logic captured by a storage element. Testing the benchmark on the PolarFire, we could observe many sudden failures, most probably related to SEFIs, that caused all the B13s instances to fail. We reduced the SEU sensitivity by a factor of five thanks to the TMR technique. However, even in this case, the total failure rate of the device is dominated by the SEFI cross-section. Thus, we need to investigate more on this error to mitigate its effect.
In addition to the separate analysis, the most important benefit from using the benchmark application is making a direct comparison. Looking at the two total cross-sections, it is clear that the NG-Medium exhibits better behaviour than the PolarFire. Nevertheless, mitigating the SEFIs effect on the PolarFire FPGA would lower its cross-section to the same level as the NG-Medium. This improvement would be an astonishing result, given that we are comparing a commercial FPGA with an RHBD FPGA.

It is important to stress that these conclusion would have been hard to get if we were using only the FPGA's components sensitivity, i.e. Flip-Flops, Memory Blocks. The total failure rate of the PolarFire is a combination of SEU and SEFIs, but the SEFI cross-section is different from the one we retrieved for the Flip-Flops. Moreover, we cannot estimate the impact of the CMIC operations at application level for the NG-Medium using the CRAM sensitivity because we do not know the number of critical bits. With this, we demonstrated why the standard characterization is not sufficient to evaluate whether an FPGA can work or not in a radiation environment, but a more realistic approach is necessary.

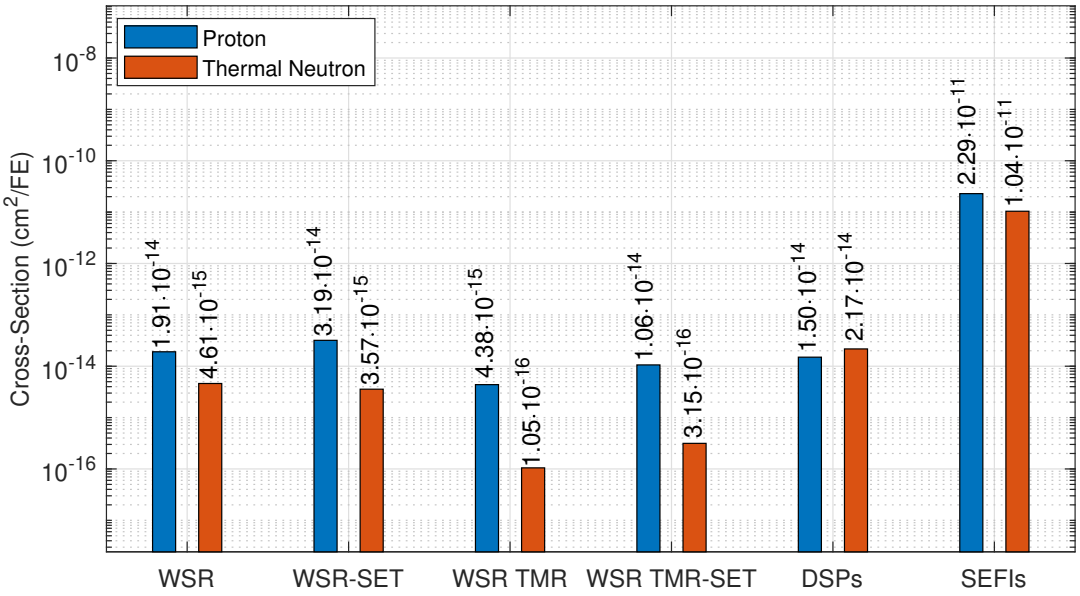


Figure 4.9: Comparison between proton and neutron cross-section for the PolarFire

4.6 HL-LHC failure rate estimation

Before using the device inside the CERN radiation environment, we need to estimate the device failure rate. Usually, the typical process requires testing the FPGA with the application we plan to use and to retrieve the device failure rate. However, this methodology entails a test for every application we plan to use. Thanks to the benchmark applications, we can avoid these steps and retrieve a failure rate correlated only to the device, without any dependencies with the application.

We use the Homogeneous Poisson process (HPP) [4.7] to estimate the probability for a specific number of failures to occur under a specific fluence to estimate the device's failure rate. The process considers a constant failure rate and that the failures are independent of each other. We express the probability using Eq. (4.7), where x is the number of failures over a period of time t, and λ is the failure rate.

$$f(x;\lambda;t) = \frac{(\lambda t)^x e^{-\lambda t}}{x!} \tag{4.7}$$

However, using the fluence instead of the time facilitates the correlation of the results to other locations at CERN where there are identical spectra of particles but different intensities. Therefore, we can map the time domain in the fluence domain by considering a constant flux as expressed in [24] and [6]. By replacing t with the fluence Φ and λ with 1/MFBF we can extract a curve that demonstrates the probability of having a specific number of failures under a given fluence, expressed in Eq. (4.8), where MFBF is the Mean Fluence Between Failure.

$$f(x;\lambda;\Phi) = \frac{(\frac{1}{MFBF}\Phi)^x e^{-\frac{1}{MFBF}\Phi}}{x!} \tag{4.8}$$

In Tab. 4.27, we summarise the relevant data we extrapolated during our tests for the NG-Medium and the PolarFire, using the benchmark application. We also report the corresponding

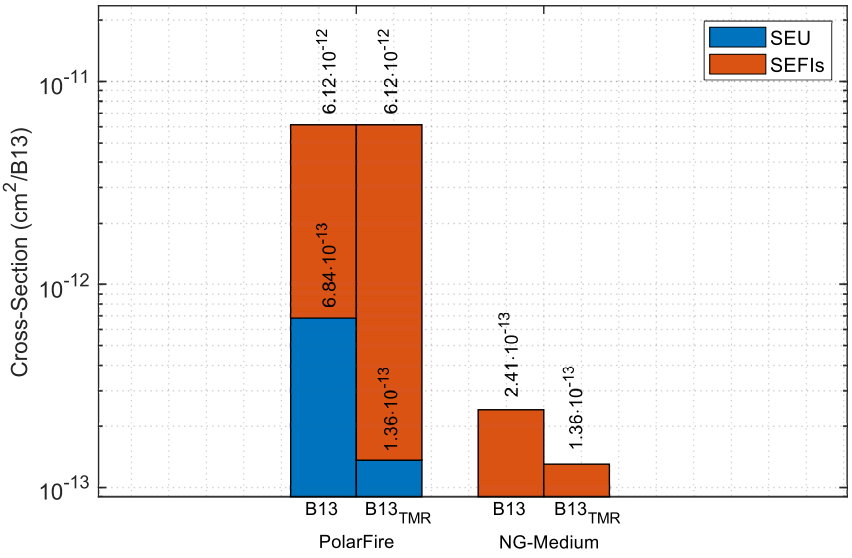


Figure 4.10: Comparison between PolarFire and NG-Medium cross-sections

Mean Fluence Between Failure for each event, corresponding to the inverse of the cross-section. Concerning the PolarFire, we considered the SEFI event a temporary failure because it resumes

FPGA	Type of event	# Events	σ	MFBF
PolarFire	Temporary failure – SEFI	3	6.12×10^{-12}	1.63×10^{11}
NG-MEDIUM	Temporary failure – CMIC	35	2.41×10^{-13}	4.15×10^{12}
	Temporary failure – CMIC TMR	11	1.30×10^{-13}	7.70×10^{12}
	Permanent failure – Reset	4	1.10×10^{-12}	9.09×10^{11}

Table 4.27: MFBF for the different events affecting the FPGAs

regular operation after its occurrence. As for the NanoXplore, instead, we took into account two types of events:

- CMIC event, which happens when an SEU affects the configuration memory. The design enters into a state of temporary failure until the CMIC engine finds and corrects the error inside the configuration memory. For this event, we have the data for the regular version and the TMR version;
- Radiation-induced reset: for board resets itself, causing the FPGA to reload the bitstream into the configuration memory. In our tests, we loaded the bitstream inside a flash memory that we connected to the FPGA. Thus, in the case of a reset, the board could restart to keep running the test. However, when in operation, we cannot use this solution because using a flash-based memory to store the user design will reduce the advantages of using an SRAM-based FPGA. Thus, we considered this event as a permanent failure.

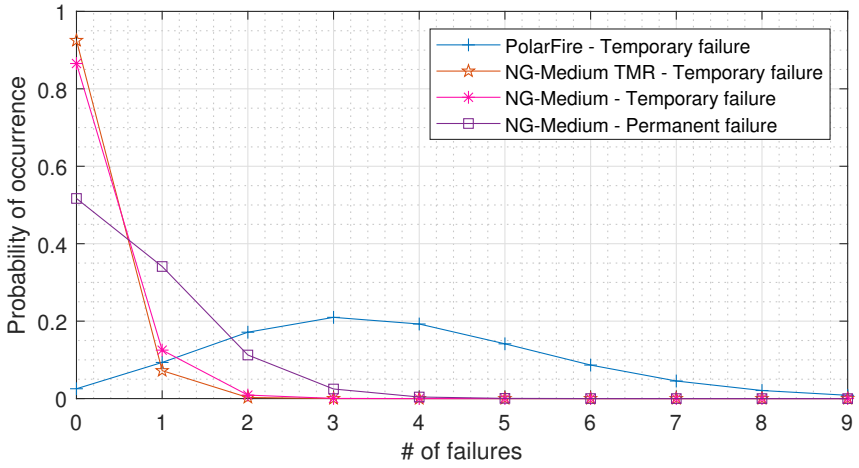
Using this information, we can now extract the probability of observing these failures in the LHC locations. Section 2.3.2 describes the radiation environment of the LHC in details, but we

Location	TID (Gy)	HEH Fluence (p/cm^2)	R_{th}
UJ	10	5×10^9	48.6
UL	0.2	10^8	24.7
RR	6	3×10^9	0.77
DS	100	5×10^{10}	3.5
ARC	2	10^9	2.7

Table 4.28: Average expected radiation levels in the LHC areas for the HL-LHC upgrade

report in Tab. 4.28 the most relevant information. We will consider the worst-case scenario, the DS. In this location, the HEH fluence is 5×10^{10}. The standard operating time of a device inside the tunnel is 12 years, which means that the accumulated fluence in the DS will reach 6×10^{11}. Thus, considering this fluence, we report in Fig. 4.11 the HPP curves for the various events.
We can extrapolate multiple information from this picture. First, we can see that the probability of having a temporary failure using the PolarFire is higher than having a temporary failure with the NG-Medium. We have a chance of 20% to have three temporary failures in 12 years of operation, compared to an almost null probability for observing the same events with the NG-Medium. Even though it is not a small probability, we could accept the PolarFire for those non-critical applications and use the NG-Medium for the more critical ones.
Regarding the NG-Medium, we have a 30% probability of observing a permanent failure in 12 years of operation. However, we must remember that we retrieved these curves by considering a single device, but hundreds of systems will use these FPGAs in actual operation. Thus, the final failure rate may not be negligible if we take into account all the systems since a reset would require stopping operation and reloading the design. On the contrary, the PolarFire does not have this kind of problems since it is not affected by permanent failures. A solution for this problem would be to use flash memories to store the design, such that in the case of a reset, the system can restart itself. However, this could be feasible inside shielded areas but not in the tunnel. Thus, both FPGAs have their pros and cons, and so the final decision will depend on the requirements of the specific area.

Figure 4.11: Failure rate of the FPGAs in the HL-LHC radiation environment



CHAPTER 5

Conclusion and future work

We described a novel approach for qualifying FPGA for the CERN radiation environment. The procedure provides for testing the standard blocks that we can find inside an FPGA, i.e. Flip-Flops and Memory blocks, and benchmark applications. Testing such applications to study the behaviour of the FPGA under radiation gives the possibility of repeating the test on multiple FPGAs and comparing the results. Standardized tests are fundamental if one wants to compare devices of different families and from various companies. Moreover, benchmark applications allow us to retrieve data that does not depend on a specif project. Therefore, we will not need to perform a radiation test for every single application we plan to use in operation.

We monitored the Total Ionizing Dose (TID) effects on the FPGA using a custom design made of multiple ring oscillators. The circuit allowed us to study the propagation delay degradation of the device during irradiation. Moreover, unlike other TID test techniques, i.e. current monitoring, this system allowed us to observe the gradual degradation of the device rather than its sudden failure. This aspect is essential because it gives a continuous overview of the device status, allowing engineers to take countermeasures when the system is about to fail. Moreover, it is also crucial to understand, before irradiation, the impact of TID effects on the internal propagation delay of the FPGA. Every design has its timing characteristics, and engineers usually consider some margin when setting the operating frequency to ensure that the application will work within a specific limit. However, in the case of radiation environments, TID may significantly impact the timing performances of the internal logic, which could also break the timing constraints making the design fail in advance. Therefore, understanding how much the logic delay is affected allow engineers to take a more considerable margin when designing the system or choose an FPGA with a better response.

In addition to the benchmark circuit to carry out the benchmark test we presented, we developed the test circuits to retrieve the FPGA sensitivity against Single Event Effects following the established guidelines in the radiation testing field. For each test circuit, we build the corresponding TESTER IP. It allows interfacing the logic circuit on the FPGA under radiation. We implemented such IPs on the FPGA embedded inside a System-on-Chip, the Zynq-7000. The user can communicate with these IPs through the embedded processor available on the SoC.

We adopted the PYNQ framework, allowing the user to exploit the FPGA capabilities by writing Python code inside a Juptyer notebook through a simple web browser. PYNQ also gives access to all the Python libraries for data analysis, which could speed up the data processing of the radiation test results.

Thanks to these implementations, we drastically facilitated the entire radiation test procedure. The IPs we developed in VHDL are presented to the user as Python objects thanks to the PYNQ framework. He can then exploit their capabilities by calling Python methods, which allows performing the radiation test of another FPGA by simply writing python code, hiding the complexity of the hardware design.

Thanks to our work, we performed many irradiation campaigns to qualify two FPGAs for the CERN radiation environment: the NG-Medium from NanoXplore and the PolarFire, from Microsemi. We followed the standard methodology and the benchmark-based procedure. Through the results we obtained, we demonstrated why the traditional characterization is not sufficient to accurately represent how the FPGA will behave when deployed in operation. Furthermore, we showed how, thanks to the benchmark application results, we can retrieve an estimation of the FPGA failure rate in different location of the LHC, which could not have been possible if we were using only the standard characterization.

We discussed all the results we retrieved in a scientific paper titled *"FPGA Qualification and Failure Rate estimation Methodology for LHC Environments Using Benchmarks Test Circuits"* [21] that has been accepted and will be published in the RADECS 2021 conference proceedings.

Finally, we set the foundation for building a complete TESTER platform for FPGA under radiation. Based on an SoC containing a processor and an FPGA, the platform could embed all the IPs handling the different test routines in a single design. Thus, every IP will act as an interface for a test circuit on the FPGA under test. In addition, we could create a user interface allowing the user to carry out the radiation testing without writing any code but just exploiting the IPs functionalities through a classic UI.

List of Figures

List of Tables

Bibliography

[1] *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.

[2] *Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC*, Physics Letters B, 716 (2012), pp. 1–29.

[3] G. Battistoni, F. Cerutti, A. Fasso, A. Ferrari, S. Muraro, J. Ranft, S. Roesler, and P. Sala, *The FLUKA code: description and benchmarking*, in AIP Conference Proceedings, vol. 896, 2007.

[4] J. Beaucour, J. Segura-Ruiz, B. Giroud, E. Capria, E. Mitchell, C. Curfs, J. C. Royer, M. Baylac, F. Villa, and S. Rey, *Grenoble Large Scale Facilities for Advanced Characterisation of Microelectronics Devices*, in 15th European Conference on Radiation and Its Effects on Components and Systems (RADECS), 2015, pp. 1–4.

[5] M. Berg, *Field Programmable Gate Array (FPGA) Single Event Effect (SEE) Radiation Testing*, 2012.

[6] M. Berg, *Characterization of system level single event upset (SEU) responses using SEU data, classical reliability models, and space environment data*, in Proc. NEPP Electron. Technol. Workshop (ETW), Jun. 2017.

[7] M. J. Cannon, A. M. Keller, C. A. Thurlow, A. Pérez-Celis, and M. J. Wirthlin, *Improving the Reliability of TMR With Nontriplicated I/O on SRAM FPGAs*, IEEE Transactions on Nuclear Science, 67 (2020), pp. 312–320.

[8] F. Corno, M. Reorda, and G. Squillero, *RT-level ITC'99 benchmarks and first ATPG results*, IEEE Design Test of Computers, 17 (2000), pp. 44–53.

[9] P. Dodd and L. Massengill, *Basic mechanisms and modeling of single-event upset in digital microelectronics*, IEEE Transactions on Nuclear Science, 50 (2003), pp. 583–602.

[10] ESA, *Single Event Effects test Method and Guidelines - Basic Specification*, 1995.

[11] R. Ferraro, *Development of Test Methods for the Qualification of Electronic Components and Systems Adapted to High-Energy Accelerator Radiation Environments*, PhD thesis, École doctorale Information, Structures et Système (I2S), 12 2019.

[12] R. García Alía, M. Brugger, F. Cerutti, S. Danzeca, A. Ferrari, S. Gilardoni, Y. Kadi, M. Kastriotou, A. Lechner, C. Martinella, O. Stein, Y. Thurel, A. Tsinganis, and S. Uznanski, *LHC and HL-LHC: Present and Future Radiation Environment in the High-Luminosity Collision Points and RHA Implications*, IEEE Transactions on Nuclear Science, 65 (2018), pp. 448–456.

[13] W. Hajdas, F. Burri, C. Eggel, R. Harboe-Sorensen, and R. de Marino, *Radiation effects testing facilities in PSI during implementation of the Proscan project*, in IEEE Radiation Effects Data Workshop, 2002, pp. 160–164.

[14] A. M. Keller, T. A. Whiting, K. B. Sawyer, and M. J. Wirthlin, *Dynamic SEU Sensitivity of Designs on Two 28-nm SRAM-Based FPGA Architectures*, IEEE Transactions on Nuclear Science, 65 (2018), pp. 280–287.

[15] Microsemi, *PolarFire Neutron Test Results*, 2018.

[16] NanoXplore, *RADIATIVE TEST Brave-FPGA*.

[17] H. Quinn, W. H. Robinson, P. Rech, M. Aguirre, A. Barnard, M. Desogus, L. Entrena, M. Garcia-Valderas, S. M. Guertin, D. Kaeli, F. L. Kastensmidt, B. T. Kiddie, A. Sanchez-Clemente, M. S. Reorda, L. Sterpone, and M. Wirthlin, *Using Benchmarks for Radiation Testing of Microprocessors and FPGAs*, IEEE Transactions on Nuclear Science, 62 (2015), pp. 2547–2554.

[18] S. Rezgui, R. Won, J. Tien, and J. George, *SET characterization and mitigation in 65-nm test structures*, in 12th European Conference on Radiation and Its Effects on Components and Systems, 2011, pp. 213–221.

[19] K. Roeed, M. Brugger, and G. Spiezia, *An overview of the radiation environment at the LHC in light of R2E irradiation test activities*, (2011).

[20] J. R. Schwank, M. R. Shaneyfelt, D. M. Fleetwood, J. A. Felix, P. E. Dodd, P. Paillet, and V. Ferlet-Cavrois, *Radiation Effects in MOS Oxides*, IEEE Transactions on Nuclear Science, 55 (2008), pp. 1833–1853.

[21] A. Scialdone, R. Ferraro, R. García Alía, L. Sterpone, S. Danzeca, and A. Masi, *FPGA Qualification and Failure Rate estimation Methodology for LHC Environments Using Benchmarks Test Circuits*, in European Conference on Radiation and Its Effects on Components and Systems, RADECS, 2021.

[22] F. Sexton, *Destructive single-event effects in semiconductor devices and ICs*, IEEE Transactions on Nuclear Science, 50 (2003), pp. 603–621.

[23] B. D. Sierawski, J. A. Pellish, R. A. Reed, R. D. Schrimpf, K. M. Warren, R. A. Weller, M. H. Mendenhall, J. D. Black, A. D. Tipton, M. A. Xapsos, R. C. Baumann, X. Deng, M. J. Campola, M. R. Friendlich, H. S. Kim, A. M. Phan, and C. M. Seidleck, *Impact of Low-Energy Proton Induced Upsets on Test Methods and Rate Predictions*, IEEE Transactions on Nuclear Science, 56 (2009), pp. 3085–3092.

[24] L. Tambara, E. Chielle, F. Kastensmidt, G. Tsiligiannis, S. Danzeca, M. Brugger, and A. Masi, *Analyzing the impact of radiation-induced failures in flash-based APSoC with and without fault tolerance techniques at CERN environment*, Microelectronics Reliability, 76-77 (2017), pp. 640–643.

[25] G. Tsiligiannis, S. Danzeca, R. García Alía, A. Infantino, A. Lesea, M. Brugger, A. Masi, S. Gilardoni, and F. Saigné, *Radiation Effects on Deep Submicrometer SRAM-Based FPGAs Under the CERN Mixed-Field Radiation Environment*, IEEE Transactions on Nuclear Science, 65 (2018), pp. 1511–1518.

[26] G. Tsiligiannis, C. Debarge, J. Le Mauff, A. Masi, and S. Danzeca, *Reliability analysis of a 65nm rad-hard sram-based fpga for cern applications*, in RADECS, 2019.

[27] G. Tsiligiannis, R. Ferraro, S. Danzeca, A. Masi, M. Brugger, and F. Saigné, *Investigation on the sensitivity of a 65nm Flash-based FPGA for CERN applications*, in 2016 16th European Conference on Radiation and Its Effects on Components and Systems (RADECS), 2016, pp. 1–4.

[28] S. Wen, R. Wong, M. Romain, and N. Tam, *Thermal neutron soft error rate for SRAMS in the 90NM–45NM technology range*, in 2010 IEEE International Reliability Physics Symposium, 2010, pp. 1036–1039.