Chapter 1
Radiation Effects and Fault Tolerance Techniques for FPGAs and GPUs

Fernanda Kastensmidt and Paolo Rech

Abstract This book introduces the concepts of soft errors in FPGAs and GPUs. The chapters cover radiation effects in FPGAs, fault-tolerant techniques for FPGAs, use of COTS FPGAs in aerospace applications, experimental data of FPGAs under radiation, FPGA embedded processors under radiation, and fault injection in FPGAs. Since dedicated parallel processing architectures such as GPUs have become more desirable in aerospace applications due to high computational power, GPU analysis under radiation is also discussed.

1.1 Introduction

Field Programmable Gate Array (FPGA) components are very attractive for aerospace applications, as well for many applications at ground level that require a high level of reliability, as automotive, bank servers, processing farms, and others. The high amount of resources available in programmable logic devices can be applied to add flexibility to the on-board computer in satellites and to the automotive industry, for example. As FPGAs can be configured in the field, design updates can be performed until very late in the development process. In addition, new applications and features can be configured after a satellite is launched, or updated in hash environments. Modern FPGAs are System-on-Chip (SoC) composed of variety of soft and hard processors, embedded DSP and memories and a large number of complex configurable logic blocks able to customized to implement the user's design.

Graphics Processing Units (GPUs), traditionally employed to accelerate graphics rendering in personal computers or portable devices. In multimedia applications reliability is not a concern as the probability of failure is pretty low and a given number of errors are tolerated, as human eye could not distinguish them. Nevertheless, lately GPUs start to be employed also in applications in which reliability matters. Thanks to their efficiency, computing capabilities, and low power

F. Kastensmidt (✉) • P. Rech
Federal University of Rio Grande do Sul, Porto Alegre, Brazil
e-mail: fglima@inf.ufrgs.br; prech@inf.ufrgs.br

F. Kastensmidt, P. Rech (eds.), *FPGAs and Parallel Architectures for Aerospace Applications*, DOI 10.1007/978-3-319-14352-1_1

consumption compare to traditional CPUs, GPUs are in fact part of projects in the aerospace and automotive field. GPUs parallel capabilities could be exploited to compress images on satellites, to limit the bandwidth required to send them to ground. Additionally, GPUs are used to implement the Advanced Driver Assistance Systems (ADAS) that helps the driver to avoid accidents. Finally, GPUs are heavily employed as accelerators in High Performance Computing (HPC) centers. A large HPC center has thousands of GPUs that work in parallel, increasing significantly the probability of having at least one GPU corrupted by radiation.

Unfortunately both FPGAs and GPUs have been found to be very sensitive to radiation, mainly as they are fabricated in nanometric process technologies. It is fundamental to experimentally measure the soft error rate of the available resources, as well as the output error rate of specific applications, to evaluate if they meet the project reliability requirements. The experimental characterization of those programmable components and GPU are mandatory to sustain its applicability under transient faults. The test methodology and characterization of FPGAs and GPUs under radiation is needed to appropriate select and evaluate fault tolerant techniques to make those components more resilient to radiation. Radiation experiments, although complex and costly, are the only known and certified way to precisely measure the probability of failure in modern integrated circuits.

1.2 Radiation Effects

Integrated circuits operating in radiation environment are sensitive to transient faults caused by the interaction of ionizing particles with silicon. A particle is considered ionizing if it has the capability of dividing a quite atom into ions. Ionizing radiation generates failures in electronic devices as the deposited charge may perturb a transistor state. The charge may be deposited directly (if the ionizing particle is charged) or indirectly. Neutrons impact, for instance, generates secondary particles (alpha particles, ions, protons), which are charged and then may perturb a transistor. The interaction of the ionizing particles with the transistors may provoke transient and permanent effects depending on the location and amount of charge transferred (directly or indirectly) to the material as a consequence of the particle collision with the silicon.

The effects that are caused by a single event interaction are called Single Event Effects (SEE) and they can be transient or permanent [1]. When the SEE has a transient behavior, it is called a Soft Error, as the device is not permanently damaged. Examples of Soft Errors are Single Event Upset (SEU) and Single Event Transient (SET). An SEU is a bit-flip that occurs when the ionizing particle hitting a transistor of a memory cell deposits enough change to revert the state of the cell. The memory cell still works perfectly in the sense that a write or read operation is performed normally, but the stored information is corrupted. When the ionizing particle hits a logic cell, it generates a voltage spike that, if latched, leads to a SET. Again, the logic cell is not damaged in the sense that a new operation will eventually be

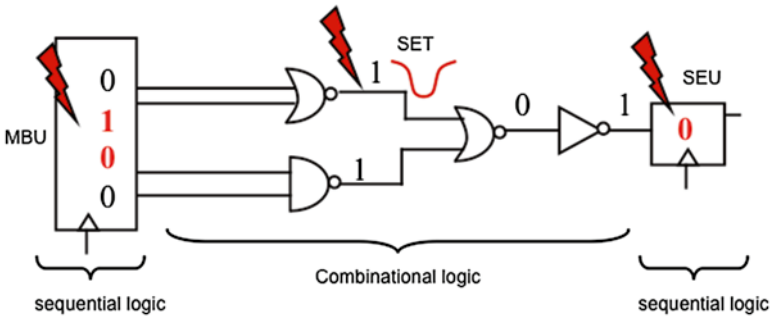

Fig. 1.1 SEU and MBU in the sequential logic and SET in the combinational logic

correctly performed. It is worth noting that the fact of being Soft does not reduce the severity of radiation-induced errors. On the contrary, the propriety of being transient and stochastic makes Soft Errors extremely hard to be identified and corrected. A permanent fault in a memory cell simply marks the cell as unused, while the possibility of having SEU makes the whole memory array as possible faulty. It is worth noting that with the shrink of transistor dimensions it is possible, for one single impinging particle, to interact with more than one transistor, generating a Multiple Cell Upset (MCU) in memory arrays. If the corrupted bits belong to the same memory word the MCU is called Multiple Bit Upset (MBU). MBU are particularly critical as they undermine the effectiveness of Error Correcting Codes (ECC). Figure 1.1 exemplifies SEU, MBU and SET in integrated circuits.

Radiation can generate also permanent faults as Single Event Latchup (SEL), Single Event Gate Rupture (SEGR), or Single Event Burnout (SEB). Finally, the accumulation of particle interactions causes an effect named Total Ionizing Dose (TID) and it represents degradation in the performance of the transistors as it modifies the threshold voltage and leakage current.

The radiation environment is composed of various particles generated by sun and stars activity [2]. The space is full of galactic cosmic rays, which are heavy ions produced by explosion of supernovas or collisions among celestial bodies. The atoms released, wondering around the universe, loses protons or electrons and, thus, gain charge. Interacting with the magnetic fields of planets and stars those ions are accelerated, reaching energies in the order of GeV. The sun produces a flux of protons and electrons, which reach the earth with low energies as they do not have sufficient time to be accelerated.

The particles can be classified as two major types: (1) energetic particles such as neutrons, electrons, protons and heavy ions, and (2) electromagnetic radiation (photons), which can be X-ray, gamma ray, or ultraviolet light. The main sources of energetic particles that contribute to radiation effects are protons and electrons trapped in the Van Allen belts, heavy ions trapped in the magnetosphere, galactic cosmic rays and solar flares. The charged particles interact with the silicon atoms causing excitation and ionization of atomic electrons.

At the ground level, neutrons are the most frequent cause of upset. Neutrons are created by cosmic ion interactions with the oxygen and nitrogen in the upper atmosphere. It is worth noting that while the solar wind is trapped in the Van Allen belts due to its low energy, galactic cosmic rays are so energetic to pass the belts and hit the upper level of the terrestrial atmosphere. The neutron flux is strongly dependent on key parameters such as altitude, latitude and longitude. There are high-energy neutrons that interact with the material generating free electron hole pairs and low energy neutrons. Those neutrons interact with a certain type of Boron present in semiconductor material creating others particles. Alpha particles are secondary types of particles emitted from interactions with radioactive impurities present in the device itself or in the packaging materials and they are the greatest concern. Materials aim to minimize the emission of alpha particles. However, it does not eliminate the problem completely.

As an energetic particle traverses the material of interest for instance a reverse-biased n+/p junction, it deposits energy along its path, as detailed explained in [3]. This energy is measured as a linear energy transfer (LET), which is defined as the amount of energy deposited per unit of distance traveled, normalized to the material's density. It is usually expressed in MeV-cm^2/mg. The total number of charges is proportional to the LET of the incoming particle. Depending on the fabrication details and the electrical characteristics of each sensitive node such as resistance and capacitance, different amplitude and duration of the transient voltage pulse are generated.

1.3 Soft Errors in FPGAs

Field-Programmable Gate Arrays (FPGAs) are configurable integrated circuit based on a high logic density regular structure, which can be customizable by the end user to realize different designs. The FPGA architecture is based on an array of logic blocks and interconnections customizable by programmable switches. Several different programming technologies are used to implement the programmable switches. There are three types of such programmable switch technologies currently in use: SRAM, where the programmable switch is usually a pass transistor or multiplexer controlled by the state of a SRAM bit (SRAM based FPGAs); Antifuse, when an electrically programmable switch forms a low resistance path between two metal layers (Antifuse based FPGAs); and EPROM, EEPROM or FLASH cell, where the switch is a floating gate transistor that can be turned off by injecting charge onto the floating gate.

Customizations based on SRAM are volatile. This means that SRAM-based FPGAs can be reprogrammed as many times as necessary at the work site and that they loose their contents information when the memories are not connected to the power supply. The antifuse customizations are non-volatile, so

they hold the customizable content even when not connected to the power supply and they can be programmed just once. Each FPGA has a particular architecture. Programmable logic companies such as Xilinx, MicroSemi, Aeroflex (licensed for Quicklogic FPGAs), Atmel and Honeywell (licensed for Atmel FPGAs) offer radiation tolerant FPGA families. Each company uses different mitigation techniques to better take into account the architecture characteristics.

1.3.1 Single Event Effects on SRAM-Based FPGAs

The SRAM-based FPGA is composed of an array of configurable logic blocks (CLB), a complex routing architecture, an array of embedded memories (Block RAM), an array of digital signal processing components (DSP) and a set of control and management logic. The CLBs are composed of Look-up Table (LUT) that implements the combinational logic, and flip-flops (DFF) that implements the sequential elements. The routing architecture can be very complex and composed of millions of pre-defined wires that can be configured by multiplexers and switches to build the desirable routing.

The configuration of all CLBs, routing, Block RAMs, DSP blocks and IO blocks is done by a set of configuration memory bits called bitstream. According to the size of the FPGA device, the bitstream can contain millions of bits. The memory bits that store the bitstream inside the FPGA is composed of SRAM memory cells, so they are reprogrammable and volatile. When an SEE occurs in the configuration memory bit of an SRAM-based FPGA, it can provoke a bit-flip. This bit-flip can change the configuration of a routing connection or the configuration of a LUT or flip-flop in the CLB. This can lead to catastrophic effects in the designed circuit, since an SEE may change its functionality.

SEE in the configuration memory bits of an SRAM-based FPGA has a persistent effect and it can only be corrected when a new bitstream is loaded to the FPGA [4]. In the combinational logic, the effect of an SEE is related to a persistent fault (zero or one) in one or more configuration bits of a LUT. Figure 1.2 exemplifies an SEU occurrence in a LUT configuration bit and in a bit controlling a routing connection. SEE in the routing architecture can connect or disconnect a wire in the matrix. This is also a persistent effect and its effect can be a modification in the mapped circuit, as a logic change or a short circuit in the combinational logic implemented by the FPGA. It can take a great number of clock cycles before the persistent error is detected and recovery actions are initiated, as the load of a faulty-free bitstream. During this time, the error can propagate to the rest of the system.

Bit-flips can also occur in the flip-flop of the CLB used to implement the user's sequential logic. In this case, the bit-flip has a transient effect and the next load of the flip-flop will correct it.

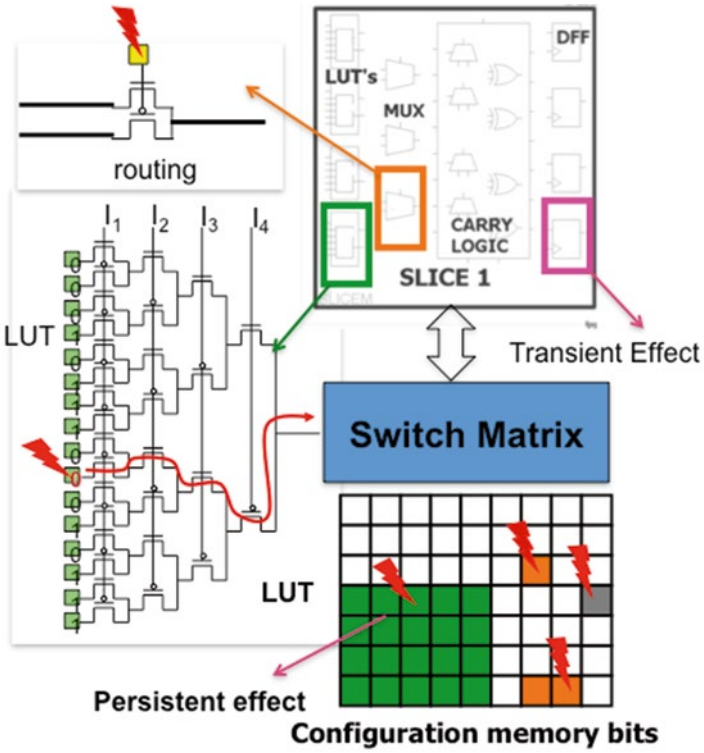

Fig. 1.2 Example of an SEU occurrence in a LUT and in the routing of an SRAM-based FPGA

1.3.2 Single Event Effects on Flash-Based FPGAs

Flash-based FPGAs have a reconfigurable array composed of VersaTiles and routing resources that are programmable by turning ON or OFF switches implemented by floating gate (FG) transistors (NMOS transistor with a stacked gate) [5]. The FG switch circuit is a set of two NMOS transistors: (1) a sense transistor to program the floating gate and sense the current during the threshold voltage measurement and (2) a switch transistor to turn ON or OFF a data-path in the FPGA (Fig. 1.3). The two transistors share the same control gate and floating gate. The threshold voltage is determined by the stored charge in the FG. Figure 1.3 illustrates VersaTiles used to implement some common logic gates. The VersaTiles are connected through a four-level hierarchy of routing resources: ultra-fast local resources; efficient long-line resources; high-speed, very-long-line resources; and the high-performance VersaNet networks.

Each VersaTile can implement any 3-input logic functions, which is functionally equivalent to a 3-inputs Lookup Table (3-LUT). But it is important to highlight that the electrical implementation of the VersaTile is totally different than the electrical

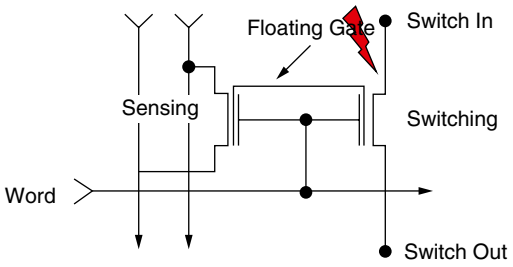

Fig. 1.3 SET in the Flash-based FPGA programmable switch

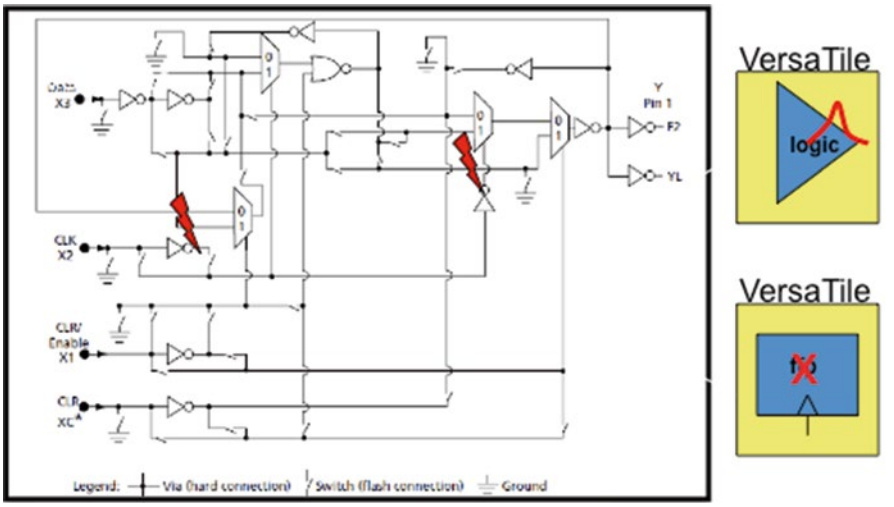

Fig. 1.4 SET and SEU in the Flash-based FPGA VersaTile

implementation of a Lookup Table (LUT). Hence, the VersaTile may have a different electrical behavior to variability effects with respect to a 3-inputs LUT. The VersaTile can also implement a latch with clear and reset, or D flip-flop with clear or reset, or enable D flip-flop with clear and reset by using the logic gate transistors and feedback paths inside the VersaTile block. For each configuration in the VersaTile block, the number of FG switches and transistors in the critical path changes. Single Event Transient (SET) pulses can hit the drain of the transistor at OFF state as presented in Fig. 1.3 provoking a transient pulse in the configuration switches. Or it can hit the sensitive nodes of the transistors in the VersaTile provoking SET or bit-flip according to the customization of the tile (Fig. 1.4). **Chapter 11** is focused on the evaluation of radiation-induced error in 65 nm Flash-Based FPGAs. **Chapter 14** gives an overview of the effects induced by neutrons in Mixed-Signal Flash-based FPGAs.

Table 1.1 Summary of SEU and SET effects in FPGAs

FPGA	SEU/SET in the logic of the configuration basic block	Routing connections	Configurable switches
SRAM-based	persistent	persistent	persistent
Flash-based	transient	no	no
Antifuse-based	transient	no	no

1.3.3 Single Event Effects on Antifuse-Based FPGAs

Antifuse-based FPGAs consists of a regular matrix composed of combinational (C-cells) and sequential (R-cells) surrounding by regular routing channels. All the customizations of the routing and the C-cells and R-cells are done by an antifuse element (programmable switch). Results from radiation ground testing have shown that programmable switches either based on ONO (oxide-nitride-oxide) or MIM (metal- insulator-metal) technology are tolerant to ionization and total dose effect [6]. Therefore, the customizable routing is not sensitive to SEU, only combinational logic and the flip-flops used to implement the design user sequential logic are sensitive to SEE.

Another well known antifuse-based FPGA is from Aeroflex and QuickLogic. Its architecture is composed of a regular matrix of configurable logic cells used to implement the combinational logic and flip-flops, surrounding by a regular routing matrix. Programmable switches called ViaLink connector are used to do all the customizations.

In order to summarize the SEU and SET effects in FPGAs, Table 1.1 shows the susceptible parts of the architectures and classifies the effects as transient or persistent, when it is needed reconfiguration to correct the fault.

1.4 Soft Errors on GPUs

Graphics Processing Units are complex parallel computing systems that dispose of large memory structures as L2 and L1 caches or register files, efficient Arithmetic Logic Units (ALU), and tasks schedulers and dispatchers.

Radiation can produce Single Event Upset as well as Multiple Bit Upset in the memory structures of a GPU. If radiation corrupts a register the process using that register for computation is likely to produce a wrong output. The peculiarity of being parallel makes errors in the caches to be more critical for GPUs than for traditional CPUs. In fact, the L1 cache is shared among all the parallel processes in a Steaming Multiprocessor (SM) while the L1 is shared among all the SMs. So, an error in the L1 cache may, in the worst case, propagate to all the parallel processes assigned to the struck SM. Similarly, an error in the L2 cache may affect all the processes running on the GPU [7].

When the impinging particle hit a logic gate, it may produce a Single Event Transient. As for SEU, the criticality and the overall effect on the output of a SET depends on the struck node. If the SET affects a logic gate inside a single core, the thread assigned to that core for computation will probably produce a single failure in the output. However, if the SET corrupts the parallel processes scheduler or dispatcher, it could affect the computation of several processes, as well as induce an application crash or system hang [8].

To have an exhaustive evaluation of GPU sensitivity is it then not sufficient to measure the radiation sensitivity of the single resources like memories or logic gates. It is also necessary to analyze how those resources are used in computation. To do so, radiation experiments can be performed on a representative set of applications, to have sufficient data to extend to other algorithms. An alternative is to calculate the program Architectural Vulnerability Factor (AVF), i.e. the probability for the corrupted resource to generate an output failure, as done in [9]. **Chapter 20** details the possible radiation effect on GPUs and presents possible way to evaluate GPUs behaviors under radiation.

1.5 Fault Tolerance Techniques

Fault-tolerance is defined as a set of techniques to provide a service capable of fulfilling the system function in spite of (a limited number of) faults. Fault-tolerance on semiconductor devices has been meaningful since upsets were first experienced in space applications several years ago. Since then, the interest in studying fault-tolerant techniques in order to keep integrated circuits (ICs) operational in such hostile environment has increased, driven by all possible applications of radiation tolerant circuits, such as space missions, satellites, high-energy physics experiments and others. Spacecraft systems include a large variety of analog and digital components that are potentially sensitive to radiation and therefore fault-tolerant techniques must be used to ensure reliability.

1.5.1 Resilience Techniques for FPGAs

Different fault tolerance techniques can be applied to FPGAs according to their type of configuration technology, architecture and target operating environment. Techniques can be implemented by the user at hardware description language (HDL) before the design is synthesized into the FPGA. In this book, authors focus on techniques that can be applied by the user at the HDL design.

The main techniques are either based on spatial redundancy or temporal redundancy [10]. Spatial redundancy is based on the replication of n times the original module building n identical redundant modules, where outputs are merged into a majority voter. Usually n is an odd number. The voter decides de correct output by

choosing the majority of the equal output values. The most common case of n-modular redundancy (nMR) is when n is equal to 3, where it is called Triple Modular Redundancy (TMR). In this case, a majority voter is used that is able to vote out 2 out of 3 values that are fault free. The TMR can be implemented in different ways by using large grain TMR, or breaking into small blocks and adding extra voters. There is local TMR when only the flip-flops are triplicated, or global TMR, also known as XTMR, where all the combinational and sequential logic is triplicated. Also Diverse TMR (DTMR) can be used, where each redundant module may present a different architecture implementation.

When dealing with the routing, different techniques can be chosen to increase or decrease fan-out, delay and set of connections, which may have a different impact in the SEE sensitivity. In addition, for those FPGAs programmable by SRAM, reconfiguration is mandatory to correct upsets in the configuration bitstream. The continuously blind full reconfiguration is called scrubbing and it is responsible to fully reconfigure the FPGA by a golden bitstream. Partial reconfiguration can also be used.

For embedded processors, one can use different mitigations based on software redundancy, or processor redundancy like lock-step and recomputation. Software-based fault tolerance techniques exploit information redundancy, control flow analysis and comparisons to detect errors during the program execution. For that purpose, software-based techniques use additional instructions in the code area, either to recompute instructions or to store and to check suitable information in memory elements. In the past years, tools have been implemented to automatically insert such instructions into C or assembly code, reducing significantly the hardening costs.

Time redundancy is based on capturing a value twice or three times in time to vote out a transient fault. The values are shifted by a delay [11]. The idea is to be able to capture 2 out of 3 upset free values to be able to mask the fault.

Each of these techniques can protect SEU or SET, or both, as shown in Table 1.2 and they will be addressed in the chapters of this book.

Very often, System-on-Chip (SoC) implemented in FPGAs use a set of the forehead mentioned mitigation techniques. **Chapters 2 and 3** present a System on Chip (SoC)

Table 1.2 List of mitigation techniques that can be applied by the user in designs targeting FPGAs

Mitigation technique	Abstraction level	SET	SEU
Local TMR	HDL		X
Global TMR or XTMR	HDL	X	X
Large grain TMR	HDL	X	X
Diverse TMR (DTMR)	HDL	X	X
Voter insertion	HDL	X	X
Reliability-oriented place and route algorithm	FPGA Flow	X	X
Temporal redundancy	HDL	X	
Embedded processor redundancy	HDL/software-based	X	X
Scrubbing/partial reconfiguration	System		X

designs using SRAM-based FPGA with embedded processor cores for satellite applications where a set of mitigation techniques is employed. **Chapter 6** details a failure detection, isolation, and recovery framework that takes advantage of the resources available in heterogeneous systems. **Chapter 7** proposes a novel scrubbing strategy for the configuration memory of FPGAs. **Chapter 8** evaluates the power requirements of n-modular redundancy, **Chapter 9** presents a fault-tolerant manager core for dynamic partial reconfiguration in FPGAs. **Chapter 12** proposes the use of C-Slow retiming for safety-critical applications. **Chapter 13** proposes a more efficient implementation of EDAC function in Radiation-Hardened FPGAs. **Chapter 15** presents hardening techniques for embedded processors, while **Chaps. 16 and 19** propose hardening techniques for soft-core processors. **Chapters 17 and 18** study how to reduce the overheads of common hardening solutions for circuits and processors.

1.5.2 Resilience Techniques for GPUs

As GPUs were initially designed to accelerate graphic rendering, the reliability research on GPUs is in its infancy. Most of the available GPUs does not offer any reliability solutions, preferring performances to fault tolerance. Only lately some of the GPUs produced for the High Performance Computing market include Error Correcting Codes in their major memory structures (L1 and L2 caches and internal registers). The available ECC is a Single Error Correction Double Error Detection (SECDED) one. It is then capable of correcting SEU and only detecting MBU. Experimentally, it was measured that about 30 % of the radiation induced failures in modern GPUs memory structures are actually multiple failures. Thanks to memory interleaving (i.e. logic bits belonging to the same word are physically separated), only the 5 % of errors are multiple errors affecting bits in the same word. Moreover, an MBU with more than 2 bits corrupted was never observed experimentally. Thus, the SECDED ECC seems sufficient to guarantee high reliability. Nevertheless, logic resources are computing structures and schedulers are left unprotected and internal flip-flops and queues are not covered by ECC. As a result, the ECC may not guarantee high levels of reliability [12].

Lately, some software-based hardening solutions for parallel codes have been proposed. The basic idea is to try to duplicate the parallel tasks to identify failures or to add coding-encoding procedures to detect and, eventually, correct, failures. Duplication With Comparison (DWC) is extremely easily implemented in a GPU, as the whole programming philosophy of the device is voted to parallelism [12]. Even if DWC seems promising and efficient to detect errors, it introduces a non-negligible computing overhead. As a result, redundancy may be non applicable to HPC applications or embedded systems with strict power consumption constraints. It is also essential to duplicate wisely the parallel processes, avoiding threads belonging to the same domain to be executed on the same Streaming Multiprocessor, as they would share the same cache. An error in a shared location will then propagate to both copies and remain undetected. Another hardening philosophy applied to parallel codes is the Algorithm Based Fault Tolerance (ABFT) one. ABFT is based

on the encoding of input data, the modification of the algorithm to be executed on coded data and, finally, the decoding of the output with error detection and correction. ABFT is algorithm-specific, and requires great algorithm analysis and code implementations efforts to be implemented. At the moment, the only algorithms for which an ABFT strategy is available are matrix multiplication and Fast Fourier Transform [7, 13]. **Chapter 20** provides an overview of the available hardening strategies to apply to modern parallel processors.

1.6 Characterizing FPGAs and GPUs Radiation Sensitivity

1.6.1 Fault Injection

In FPGAs, one very important step of the design flow is the validation of the fault tolerance technique that is usually done by fault injection. The original bitstream configured into the FPGA can be modified by a circuit or a tool in the computer by flipping one of the bits of bitstream, one at a time. This flip emulates a SEU in the configuration memory cells. The output of the design under test (DUT) can be constantly monitored to analyze the effect of the injected fault into the design. If an error is detected, this means that the fault tolerant technique implemented is not robust for that specific fault (SEU) in that target configuration memory bit.

It is possible to inject faults in all the configuration bits and to analyze the most critical parts of the design [14]. This can help to guide designers in early stages of the development process to choose the most appropriated fault tolerant design, even before any radiation ground testing. The entire fault injection campaign can spend from few hours to days depending on the amount of bits that are going to be flipped and the connection to the fault injection control circuit. When the entire system (fault injection control+DUT+golden designs) is implemented at the hardware level (board), avoiding the communication with the computer, the process is speeded up in orders of magnitude.

Chapters 4 and 5 present some techniques for fault injection in SRAM-based FPGAs. **Chapter 10** presents a fault injection framework that reproduce multiple and accumulation of upsets collected from real radiation experiments.

However, fault injection on GPUs has several limitations. Only few resources of the GPU are accessible by the user and to access those resources to inject fault it is necessary to change the flow of the algorithm, introducing artificial behavior. There is one fault injector for GPU available, the GPU-Qin [15], which allows the user to insert faults only on instantiated values.

1.6.2 Radiation Test Methodologies to Predict and Measure SER in FPGAs and GPUs

The test of FPGAs under radiation depends on a test plan developed for each type of FPGA and design architecture. Here we will detail the radiation test for SRAM-based FPGAs. There are two types of tests: the static test and the dynamic test. The static test can be done in SRAM-based FPGAs for instance, where the experiment

consists on configuring the FPGA with a *golden* bitstream containing the test-design and then constantly read back the FPGA configuration memory with the Xilinx iMPACT tool through the JTAG interface. In the experiment control computer, the *golden* bitstream is compared against the *readback* bitstream. If differences are found, the FPGA is reconfigured with the *golden* bitstream and the differences are stored in the computer. Faults are defined as any bit-flip in the configuration memory detected by the *readback* procedure. In this case, it is possible to calculate the upset rate in the configuration memory bits for that specific particle flux.

The cross-section per bit shows the sensitive area of a device and it is used to compare radiation sensitivity between devices. It is calculated as defined in Eq. 1.1.

$$\sigma_{SEU-bit} = \frac{N_{SEU}}{\Phi_{neutron} \times N_{bits}} \tag{1.1}$$

Where N_{SEU} is the number of SEU in the configuration memory bits, $\Phi_{neutron}$ is the neutron fluence and N_{bits} is the number of bits of the device. The fluence is measured by neutron per cm^2, and it is calculated by multiplying the neutron flux by the time the device has been exposed to that flux.

The dynamic test analyzes the design output mapped into the FPGA. In this case, the expected error rate is much lower than the static test. In case of SRAM-based FPGAs, based on the Xilinx Reliability Report [16], in average it is necessary 20 upsets in the configuration memory bits to provoke one error in the design output. This relation may of course vary according to the logic density, mapping, routing and the chosen architecture for the design. In case of using redundancy such as TMR or n-MR, the number of accumulated upsets in the bitstream without provoking functional error can increase significantly. In case of Flash-based FPGAs and antifuse based FPGAs, the soft error rate comes from the susceptibility of the configurable logic to the SET and SEU (bit-flips) only as the programmable cells (antifuse and flash cells) are normally not susceptible to transient upsets.

The static test of GPUs follows the same philosophy as the FPGA one. Basically a known pattern is loaded into the main memory structures of the device and then read back. There is not a special port to access the memory structures, so the test should be engineered to take advantage of normal GPU processes to write the pattern and read it back. The dynamic test of a GPU requires the selection of proper benchmarks to run on the device. It is worth noting that for being useful the benchmark must be representative of a given workload of application. Otherwise results would be valid only for the particular configuration tested. Normally the benchmark is executed with a pre selected input vector and results are checked with a pre computed golden copy of the output. When a mismatch is detected, it should be counted as an error. To evaluate the cross section it is necessary to evaluate the fluence hitting the device only when the code is being executed, and not during results check. Alternatively, one can calculate the cross section diving the observed error rate (errors/s) by the average flux provided by the facility during the test (particles/(cm^2 s)).

There are only few facilities in the world that provides good fluxes and spectrum of energies to ease the scale of experimental result to the expected natural error rate. Examples of neutron facilities are LANSCE, in Los Alamos, NM, USA, TSL, Uppsala, Sweden, TRIUMF, Vancouver, Canada, and ISIS, Didcot, UK.

In this book results were gathered from experiments performed at Los Alamos National Laboratory's (LANL) Los Alamos Neutron Science Center (LANSCE) Irradiation of Chips and Electronics House II and in the VESUVIO beam line in ISIS, Rutherford Appleton Laboratories, Didcot, UK. As shown in [17], both of these facilities provide a white neutron source that emulates the energy spectrum of the atmospheric neutron flux. The ISIS spectrum has a lower component of high-energy neutrons with respect to the LANSCE and the terrestrial one. The relationship between neutron energy and modern devices cross section is still an open question. Nevertheless, ISIS beam has been empirically demonstrated to be suitable to mimic the LANSCE one and the terrestrial radiation environment [17].

Figures 1.5 and 1.6 show the setup of experiments under neutron at ISIS Facility in United Kingdom and Los Alamos, respectively, composed of many different types of FPGAs and GPU performed in parallel.

Fig. 1.5 Neutron experiment Setup in ISIS for FPGAs and GPUs

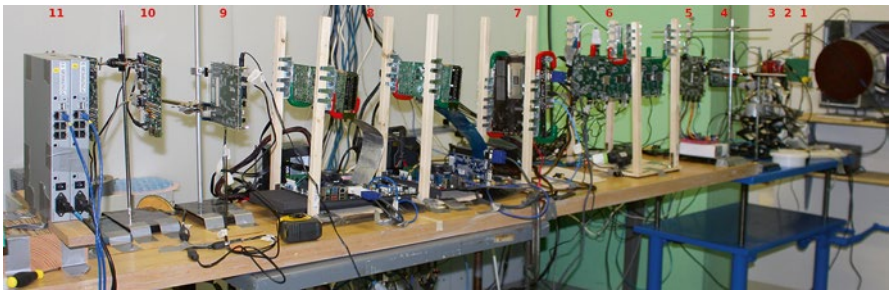

Fig. 1.6 Neutron experiment Setup in Los Alamos for FPGAs and GPUs

References

1. Nicolaidis M (2011) Soft errors in modern electronic systems. Springer, New York, p 318
2. Stassinopolous EG, Raymond JP (1988) The space radiation environment for electronics. Proc IEEE 76:1423–1442
3. Dodd PE, Massengill LW (2003) Basic mechanisms and modeling of single-event upset in digital microelectronics. IEEE Trans Nucl Sci 50(3):583–602
4. Kastensmidt FL, Reis R, Carro L (2006) Fault-tolerance techniques for SRAM-based FPGAs (frontiers in electronic testing). Springer, New York
5. Microsemi. ProASIC3, IGLOO and SmartFusion flash family FPGAs datasheet. www.microsemi.com
6. Rezgui S, Louris P, Sharmin R (2010) SEE characterization of the new RTAX-DSP (RTAX-D) antifuse-based FPGA. IEEE Trans Nucl Sci 57(6):3537–3546
7. Rech P, Aguiar C, Frost C, Carro L (2013) An efficient and experimentally tuned software-based hardening strategy for matrix multiplication on GPUs. IEEE Trans Nucl Sci 60(4):2797–2804
8. Rech P, Pilla L, Navaux POA, Carro L (2014) Impact of GPUs parallelism management on safety-critical and HPC applications reliability. In: Proceeding IEEE international conference on dependable systems and networks (DSN), June 2014, pp 455–466
9. Mukherjee SS, Emer J, Reinhardt SK (2005) The soft error problem: an architectural perspective. In: High-performance computer architecture, 2005. HPCA-11. 11th international symposium on, 12–16 Feb 2005, pp 243–247
10. Schrimpf RD, Fleetwood DM (2004) Radiation effects and soft errors in integrated circuits and electronic devices. Word Scientific, Singapore
11. Anghel L, Alexandrescu D, Nicolaidis M (2000) Evaluation of a soft error tolerance technique based on time and/or space redundancy. In: The Proceedings of symposium on integrated circuits and systems design, SBCCI, 13, pp 237–242
12. Oliveira DAG, Rech P, Pilla LL, Navaux POA, Carro L (2014) GPGPUs ECC efficiency and efficacy. In: International symposium on defect and fault tolerance in VLSI and nanotechnology systems
13. Pilla LL, Rech P, Silvestri F, Frost C, Navaux POA, Sonza Reorda M, Carro L (2014) Software-based hardening strategies for neutron sensitive FFT algorithms on GPUs. IEEE Trans Nucl Sci 61(4):1874–1880
14. Sterpone L, Violante M (2007) A new partial reconfiguration-based fault-injection system to evaluate SEU effects in SRAM-based FPGAs. IEEE Trans Nucl Sci 54(4):965–970
15. Fang B, Pattabiraman K, Ripeanu M, Gurumurthi S (2014) GPU-Qin: A methodology for evaluating the error resilience of GPGPU applications. In: Proceedings of the IEEE international symposium on performance analysis of systems and software (ISPASS)
16. Xilinx, Inc. (2013) Device reliability report third quarter 2013. http://www.xilinx.com/support/documentation/user_guides/ug116.pdf
17. Violante M, Sterpone L, Manuzzato A, Gerardin S, Rech P, Bagatin M, Paccagnella A, Andreani C, Gorini G, Pietropaolo A, Cargarilli G, Pontarelli S, Frost C (2007) A new hardware/software platform and a new 1/E neutron source for soft error studies: testing FPGAs at the ISIS facility. IEEE Trans Nucl Sci 54(4):1184–1189