Mitigation of Radiation Effects in SRAM-based FPGAs for Space Applications

FELIX SIEGLE, University of Leicester
TANYA VLADIMIROVA, University of Leicester
JØRGEN ILSTAD, European Space Agency
OMAR EMAM, Airbus Defence and Space

The usage of Static Random Access Memory (SRAM)-based Field Programmable Gate Arrays (FPGAs) in harsh radiation environments has grown in recent years. These types of programmable devices require special mitigation techniques targeting the configuration memory, the user logic as well as the embedded RAM blocks. This article provides a comprehensive survey of the literature published in this rich research field during the past ten years. Furthermore, it can also serve as a tutorial for space engineers, scientists and decision makers who need to get introduced to this topic.

Categories and Subject Descriptors: B.8.1 [**Reliability, Testing, and Fault-Tolerance**]

General Terms: Design, Reliability

Additional Key Words and Phrases: Dynamic Partial Reconfiguration, Fault Injection, FPGA, Hot Redundancy, Information Redundancy, Radiation Effects, Scrubbing, Single Event Upsets, Spatial Redundancy, Triple Modular Redundancy

ACM Reference Format:
Felix Siegle, Tanya Vladimirova, Jørgen Ilstad, and Omar Emam, 2014. Mitigation of Radiation Effects in SRAM-based FPGAs for Space Applications. *ACM Comput. Surv.* [redacted]
DOI:http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Electronics on board modern spacecraft comprise a considerable number of Field Programmable Gate Array (FPGA) devices. While in the early days these devices were mainly used to implement rudimentary glue logic, they enable far more complex operations today. Regardless of the application, e.g. science, Earth observation or military surveillance, a trend to ever increasing payload data volumes can be observed. Thus, data processing in space can be essential for some missions as payload data downlinks can be too slow to transmit these growing data volumes, even if data compression techniques are applied.

Many payload data processing applications benefit from an efficient implementation in hardware using programmable logic devices. Modern SRAM-based FPGAs offer huge amounts of logic resources, allow fast clocking and can be quickly reconfigured which makes them ideal platforms for the implementation of such algorithms. They are, however, prone to radiation effects in space because the state of their memory cells

This work is supported by the European Space Agency under the NPI Programme, Airbus Defence and Space UK, and University of Leicester.
Author's addresses: F. Siegle, and T. Vladimirova, Embedded Systems Group, University of Leicester, UK; J. Ilstad, ESA-ESTEC, Noordwijk, The Netherlands; O. Emam, Airbus Defence and Space, Stevenage, UK.

Table I. FPGA Usage Example: Sentinel-2 mission [Gardenyes 2012].

IC TYPE	QUANTITY PLATFORM	QUANTITY PAYLOAD
FPGA	**118**	**37**
Custom ASIC	72	6
Microcontrollers	23	0
Standard ASIC	10	0

can be flipped due to single and multiple event upsets caused by radiation. Hence, design techniques to mitigate radiation effects must be applied to these devices. In the past ten years, research on such mitigation methodologies established its own rich field, which we thoroughly survey in this article.

The article is structured as follows. Section 2 answers the question as to why SRAM-based FPGAs are so popular in space engineering and discusses radiation effects in space as well as their effects on SRAM-based FPGAs. Then, in Section 3, terminology, failure modes and mitigation techniques are outlined.

In Section 4, mitigation techniques applied during run-time operation are reviewed dividing this huge field into three areas: techniques aimed at the (i) configuration memory, (ii) user logic and (iii) embedded RAM blocks. A brief survey of methodologies that can be applied during design-time is then given in Section 5. Section 6 covers the simulation and emulation of radiation effects, including accelerated radiation testing and fault injection. Section 7 is dedicated to purpose-built hardware platforms, which have been used in research projects on SRAM-based FPGAs for space applications. Section 8 provides a summary of the reviewed techniques as well as design recommendations. Finally, Section 9 concludes the paper.

2. BACKGROUND

2.1. Why SRAM-based FPGAs in Space?

Nowadays, FPGAs are commonly used on board spacecraft. The importance of these devices for space applications is illustrated by the figures given in Table I, which show that the vast majority of Integrated Circuits (ICs) on board the Sentinel-2 spacecraft, a current mission of the European Space Agency, are FPGAs. As evidenced by Table I, while Application-Specific Integrated Circuits (ASICs) and microcontrollers still play an important role for platform applications, payload processing applications are mainly implemented with FPGAs.

Today, three main types of space qualified FPGA technologies are employed in commercial products. The most common technology is antifuse, which is used by one-time programmable FPGAs. One advantage of these devices is their natural tolerance against radiation effects because the hardware configuration is fixed. In principle, these devices can still suffer from Single Event Upsets (SEUs) in user logic and embedded RAM cells. However, radiation tolerant versions are available, which offer hardened user flip-flops by design, e.g. the RTAX and RTSX devices by Microsemi.

The second technology is based on SRAM memory, i.e. the configuration of the FPGA is stored in volatile memory cells. An obvious benefit of these devices is the possibility to reconfigure the hardware in later design or even mission stages. Furthermore, some of these devices like the newer Virtex-4 and Virtex-5 FPGAs by Xilinx offer high performance and a large amount of logic and embedded memory resources as well as dedicated Digital Signal Processing (DSP) blocks. In contrast to their antifuse counterparts, SRAM-based FPGAs are in principle more susceptible to SEUs because the hardware configuration can be altered by radiation effects (e.g. Virtex-4QV and earlier devices by Xilinx). FPGAs with radiation hardened configuration memory are also available, however, e.g. Virtex-5QV devices by Xilinx or ATF280 devices by Atmel.

Recently, flash memory based FPGAs are also being considered for use in space projects, e.g. the ProASIC3 device by Microsemi. Similar to SRAM-based FPGAs they can be reconfigured and offer good performance. The usage of such devices on long space missions is, however, problematic due to their rather low immunity to the Total Ionising Dose (TID) effect and Single Event Latchups (SELs) [Microsemi 2012].

Internal studies at the Jet Propulsion Laboratory (JPL) estimate that the raw, uncompressed data captured from spectroscopy instruments on board recently proposed U.S. missions could reach 1 to 5 Terabytes per day [Norton et al. 2009].

It is therefore recommended to drastically reduce the data volume which must be stored on board and later transmitted to Earth by transforming the raw measurements of payload instruments into intermediate results performing on-board processing. In a technology assessment NASA scientists also found that Xilinx FPGAs are best suited for such high-performance tasks due to their flexibility and their embedded DSP blocks compared to single board computers and DSP processors. Apart from the increase in performance, SRAM-based FPGAs offer the capability of being reconfigured - a feature not to be underestimated for space projects. Pingree describes the typical problem of one-time programmable FPGAs in [Pingree 2010]. For one of the instruments on the NASA Juno spacecraft to Jupiter the engineers had to design and program the FPGA design two years before launch. Since the FPGA was one-time programmable, it could not be changed or improved without an high impact to the project cost and schedule. Furthermore, as the spacecraft travels for five years to Jupiter, instrument calibration activities may be required during that time, in which the FPGA design cannot be changed too. With SRAM-based FPGAs, however, hardware updates could be easily applied in later design stages or even in-flight.

Most recent publications are concerned with the SRAM-based Xilinx Virtex-4QV and Virtex-5QV FPGAs, as they are currently the only fast SRAM-based FPGAs, which are available in space-qualified versions. Therefore, in the following the focus is on these devices.

2.2. Radiation Effects in SRAM-based FPGAs for Space

2.2.1. Sources of Radiation Effects. The space radiation environment comprises a large range of energetic particles with energies from several keV up to GeV and beyond. The main elements are [Holmes-Siedle and Adams 1993; ECSS 2008b]:

— Trapped radiation: Energetic electrons and ions are magnetically trapped in the so-called Van Allen radiation belts which extend from 100 km to 65,000 km and consist mainly of electrons up to a few MeV and protons of up to several hundred MeV energy. The Earth's magnetic field is not symmetrical, leading to local distortions. One important distortion is known as the South Atlantic anomaly. Spacecraft passing this area are exposed to an increased level of radiation.
— Galactic cosmic rays: High-energy charged particles which enter the solar system from outside and which are composed of protons, electrons and fully ionized nuclei.
— Solar energetic particles during solar flares: High-energy particles which are encountered in interplanetary space and close to Earth and which are seen in short bursts associated with other solar activity. The duration of such bursts can be a few hours up to several days. They consist of protons, electrons and heavy ions in the energy range of a few tens of keV to GeV and beyond.

In addition, secondary radiation is generated by the interaction of energetic particles with materials. One example is *bremsstrahlung*, a high-energy electromagnetic radiation that is caused by the deceleration of a charged particle in materials.

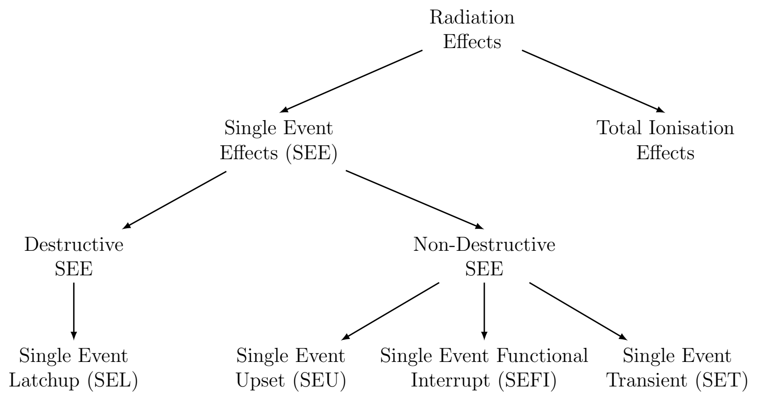
Radiation Effects
Single Event Effects (SEE)
Total Ionisation Effects
Destructive SEE
Non-Destructive SEE
Single Event Latchup (SEL)
Single Event Upset (SEU)
Single Event Functional Interrupt (SEFI)
Single Event Transient (SET)

Fig. 1. Common radiation effects that must be mitigated in SRAM-based FPGAs.

2.2.2. Radiation Effects. An overview of common radiation effects that must be mitigated in SRAM-based FPGAs is given in Figure 1. The main effects are:

— TID Effect: Ionisation of electronic components is caused by electrons, protons and bremsstrahlung and leads to a degradation due to increasing leakage currents and other effects [ECSS 2008a]. Processes that cause ionisation are based on photon interaction and include the photoelectric effect, compton effect and pair production, all leading to the production of free electrons and hole-electron pairs [Messenger and Ash 1992]. The accumulation of these effects is called TID and is usually measured in krad with $1\,\mathrm{rad} = 10^{-2}\,\mathrm{Gy} = 6.24 \cdot 10^{7}\,\frac{\mathrm{MeV}}{\mathrm{g}}$ [Dierker 2007]. For space-qualified Virtex-4QV and Virtex-5QV devices, the TID is of no concern since the dose is guaranteed to be 300 krad for Virtex-4QV devices [Xilinx 2010], respectively 1 Mrad for Virtex-5QV devices [Xilinx 2012a].
— SEL: A potentially destructive Single Event Effect (SEE) that can trigger parasitic PNPN thyristor structures in a device [ECSS 2008a]. Similar to the TID effect, SELs are of no concern for Virtex-4QV and Virtex-5QV devices since both devices have a guaranteed latchup immunity to $\mathrm{LET} > 100\,\mathrm{MeV} \cdot \mathrm{cm}^2 \cdot \mathrm{mg}^{-1}$ [Xilinx 2010; 2012a].
— SEU: This class of SEE is a *soft error* that changes the state of a bistable element. It is triggered by heavy ions and protons and results from ionisation by a single energetic particle or the nuclear reaction products of an energetic proton. The ionisation induces a current pulse in a p-n junction whose charge may exceed the critical charge which is required to change the logic state of the element. As a result, the value of a memory bit can be flipped [Holmes-Siedle and Adams 1993]. SEU is the most common effect for SRAM-based FPGAs as it may affect the configuration memory as well as memory cells that are used as part of the user logic (flip-flops, embedded RAM).
— Single Event Functional Interrupt (SEFI): This class of SEE interferes with the normal operation of the FPGA and is thus typically used to classify failures that affect the circuits needed to operate the FPGA. So far, six types of SEFIs have been identified for Virtex-4QV devices [Allen et al. 2008]:
  — Power-On-Reset (POR) SEFI: results in a global reset of all internal storage cells and the loss of all program and state data.
  — SelectMAP (SMAP) SEFI: results in loss of either read or write capability through the SelectMAP interface.
  — Frame Address Register (FAR) SEFI: results in the frame address register continuously incrementing.

Table II. Example SEFI and SEU rates for a Virtex-4QV SX55 [Allen et al. 2008].

COMPONENT	UNIT	LEO	GEO
All SEFIs	$\text{Device}\cdot\text{Years}/\text{Events}$	36	103
User flip-flops	$\text{Upsets}/\text{Device}\cdot\text{Day}$	0.0702	0.0387
Block RAM Memory	$\text{Upsets}/\text{Device}\cdot\text{Day}$	4.05	4.49
Configuration Memory	$\text{Upsets}/\text{Device}\cdot\text{Day}$	7.56	4.28

  - Global Signal SEFI: results in disruption of global signals like Global Write Enable, Global Drive High etc.
  - Readback SEFI: occurs when a portion of readback data has been upset and can lead to a false-positive detection of a SMAP SEFI.
  - Scrub SEFI: causes corruption of the data stream being downloaded to the device.
- Single Event Transient (SET): This class of SEE is a momentary voltage or current disturbance which may propagate through subsequent circuitry and eventually manifests as SEU once it reaches a latch or other memory elements [Dodd et al. 2004].

2.2.3. Single Event Effects Rates. SEU rates in FPGAs depend on the particular device component, in which they occur (see Section 3.2 below). The mitigation strategy for Virtex-4QV FPGAs must mainly take into account single event effects, as these devices are tolerant to accumulated ionisation and SELs. In contrast, Virtex-5QV devices are radiation hardened by design. This was achieved by replacing the configuration memory and flip-flop cells by dual-node counterparts that require charge collection in at least two active nodes before an upset can occur. Furthermore, all flip-flop inputs are now protected by SET filters and TMR is applied to control circuitry and registers [Swift and Allen 2012].

Static and dynamic cross-sections for most FPGA blocks with regards to the Virtex-4QV family, can be found in [Allen et al. 2008; Allen 2009]. Using these cross-sections, SEU and SEFI rates can be calculated for a particular design and orbit. For European space projects the necessary calculation methods are standardised in [ECSS 2008b] and [ECSS 2008a]. A tool that greatly simplifies the SEU prediction according to these standards is OMERE which was developed by the French company TRAD with support from the French space agency CNES [TRAD 2014].

In [Allen et al. 2008], SEU and SEFI rates for several orbits in quiet solar maximum conditions were calculated using the CREME96 model. For illustration, rates for two orbits are given in Table II. The first one is a Low Earth Orbit (LEO) at 800 km altitude with an inclination of 22.0°, the second one is a Geostationary Earth Orbit (GEO) at 36,000 km. The FPGA type is a XQR4VSX55 and it is assumed that all memory cells are used, i.e. the upset rates per bit-day are scaled to the whole device. It can be seen that the likelihood of SEFIs is low, with approximately one SEFI every 36 years in LEO and every 103 years in GEO.

Assuming that all flip-flop cells are used, the chance of an upset in these elements is far below 0.1 upsets per device-day. In contrast, if a design heavily utilises Block RAM (BRAM) blocks (in this example all blocks are used), the probability of an upset is more than 400 times higher than for a flip-flop upset due to the high ratio of BRAM cells to flip-flop cells. For the configuration memory cells the ratio is even larger: In LEO more than 7.5 upsets can occur per device-day. It is, however, assumed that all configuration memory cells are utilised which is unrealistic for a real design.

The results above show that mitigation techniques must mainly focus on configuration memory and Block RAM upsets. Although SEFIs occur only rarely, they can necessitate an undesired full reconfiguration and must be therefore mitigated as good as possible. In contrast, a mitigation strategy for flip-flops may not be necessary for some applications.

In 2012, Quinn et al. presented first on-orbit results for Virtex-4QV FPGAs [Quinn et al. 2012], collected from an experimental payload launched by Los Alamos National Laboratory. The system comprises two Virtex-4 FPGAs running the same digital signal processing application. The mitigation strategy is based on Triple Modular Redundancy (TMR) in combination with scrubbing. Using fault injection experiments and the CREME96 model, an observable output error rate of approximately one in 15 to 25 days was predicted before launch. This rate is based on a calculated configuration memory upset rate of 68 to 89 $^{\text{SEU}}/_{\text{device}\cdot\text{day}}$.

The on-orbit results are surprising: First, the measured upset rate per unit is much lower than predicted (19 $^{\text{SEU}}/_{\text{unit}\cdot\text{day}}$). Secondly, the only two measurable output errors were triggered by SEUs in bit locations which could not be predicted by fault injection before. Thirdly, a SelectMAP SEFI was observed, although such a failure should only occur very rarely according to worst case predictions. Finally, the authors were able to observe atypical events where many bits in one single frame were corrupted all at the same time.

The authors assume that the measured upset rate is artificially low due to the shielding of the spacecraft and the duty cycle of the device. It was further found that 8.42% of SEU events are actually Multiple Bit Upsets (MBUs), although the vast majority has a size of only two bits. 78% of the SEUs occurred in Configurable Logic Blocks (CLBs), followed by ca. 15% in BRAM interconnect and ca. 6% in Input Output Blocks (IOBs).

3. OVERVIEW OF RADIATION MITIGATION TECHNIQUES FOR SRAM-BASED FPGAS

3.1. Terminology

Several techniques can be applied during the design process to mitigate soft errors in digital circuits. A classification of these techniques is presented in Section 3.3 below, which makes use of the terminology introduced in the NASA Fault Management Handbook [NASA 2012]. Although targeting flight systems in general, this terminology proves to be well suited to describing soft error mitigation techniques for FPGAs too.

A common terminology to describe an abnormal state of a system includes the three terms: fault, error and failure. Although several standards define these terms slightly differently, a fault is usually understood as the cause of an error and an error as the cause of a failure. For instance, in functional safety standard ISO 26262, a fault is defined as an "abnormal condition that can cause an element or an item to fail". The error is defined as the "discrepancy between a computed, observed or measured value or condition, and the true, specified, or theoretically correct value or condition". Finally, the failure is defined as the "termination of the ability of an element, to perform a function as required".

According to the Fault Management Handbook, failures can be either prevented or tolerated. In the first case, actions are taken to avoid failures either at design time or run time. The Design-Time Fault Avoidance includes "design function and FM [fault management] capabilities to minimize the risk of a fault and resulting failure" while Operational Failure Avoidance "predicts that a failure will occur in the future and takes action to prevent it from happening". With failure tolerance, failures are either accepted or mitigated. Failure Masking techniques "allow a lower level failure to occur, but mask its effects so that it does not affect the higher level system function". Failure Recovery techniques "allow a failure to temporarily compromise the system function, but respond and recover before the failure compromises a mission goal". Finally, Goal

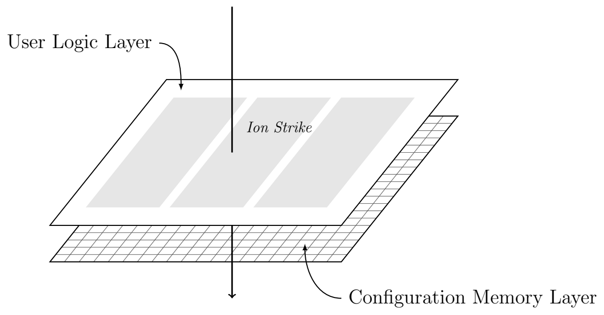

Fig. 2. Model of an SRAM-based FPGA.

Change strategies "allow a failure to compromise the system function, and respond by changing the system's goals to new, usually degraded goals that can be achieved".

In the following, erroneous FPGA output is seen as a failure. Although the failure is always caused by a fault, a fault does not necessarily lead to a failure. In an FPGA circuit design, such a fault could be for example a flipped bit in a flip-flop or a re-programmed logical operation due to a falsified look-up table. In any case, only if the faulty resource is actually used in the design the associated fault will finally lead to a failure.

3.2. Failure Modes in SRAM-based FPGAs

An FPGA model, commonly found in literature [Padovani 2005], which is suitable for illustration of different fault and failure modes of SRAM-based FPGAs is shown in Figure 2. SRAM-based FPGAs comprise a configuration memory layer that stores the configuration of the FPGA in SRAM memory cells and a user logic layer on which the actual circuit is implemented. A typical circuit utilises sequential and combinational logic elements and often accesses embedded BRAM and/or DSP blocks. While the user flip-flops and other user memory resources as well as the DSP blocks are physically present, combinational logic gates are realised with Look-Up Tables (LUTs) within CLBs.

The configuration bits on the configuration memory layer control the resources on the user logic layer, including the wiring between the resources, the content of the LUTs and the configuration of the CLB, BRAM, DSP and IOB blocks.

If an ion hits the FPGA, it can affect memory resources (i) on the configuration memory or (ii) on the user logic layer. In both cases, upsets can be seen as faults which *may* lead to a failure. And in both cases, the system can fortunately recover from such failures because affected memory cells can be updated with correct values. Since the configuration bits control "really everything" [Padovani 2005], the configuration memory is the main concern of most mitigation strategies. However, although more than 60 percent of the configuration bits are used to control routing resources, only 10 to 20 percent of routing resources are used in a typical design [Carmichael and Tseng 2009]. The ratio between used configuration bits and user flip-flops bits is, however, usually still so high that flip-flop upsets account for only a few percent of all upsets. And obviously, configuration bit upsets can lead to much more unpredictable behavior than flip-flop upsets. In contrast to user flip-flops, Block RAM upsets can be as much of a concern as configuration memory upsets if large amounts of these resources are utilised in a design.

A fault in the configuration memory may lead to a failure in case the affected configuration bit controls a resource which is utilised by the design. In Xilinx terminology,

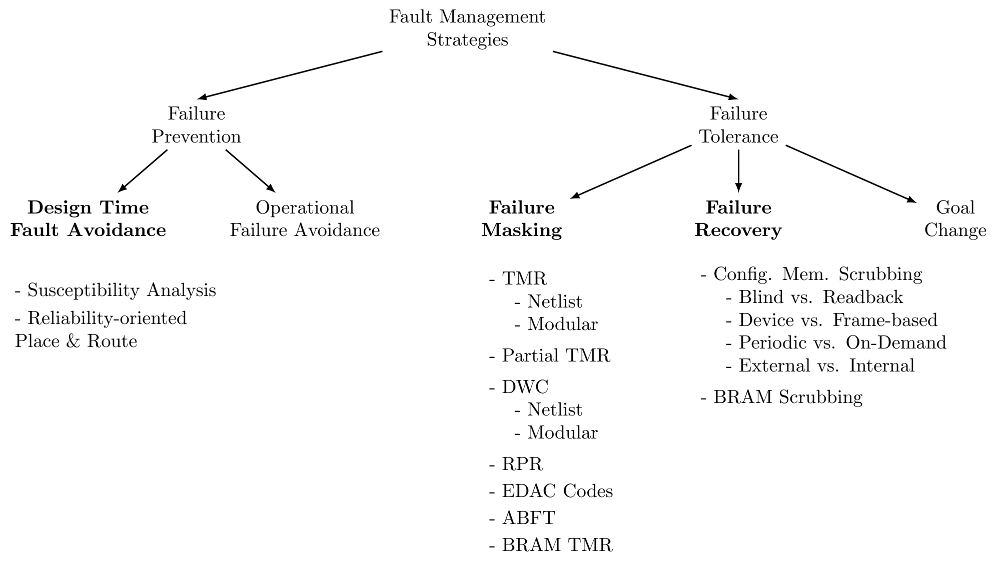

Fig. 3. Classification of fault management strategies and corresponding mitigation techniques.

configuration bits can be classified as Essential and Critical Bits [Xilinx 2012b]. Essential Bits are the subset of configuration bits which are responsible for resources of the design. Thus, a fault affecting an Essential Bit *may* lead to a failure. Because not every resource of a design is used by the application non-stop, only faults in a subset of the Essential Bits, also referred to as Critical Bits, are guaranteed to manifest as failure.

A fault in a user flip-flop can lead to a failure if its value is used by subsequent circuitry. Although the failure can propagate through the system until it becomes measurable at the output, it is often only of transient nature. If the flip-flop is used in state-dependent logic, however, a failure can be 'trapped' in a feedback loop until the logic is reset to a known (initial) state. For instance, if a bit of a counter register is flipped, the counter 'jumps' and the output is permanently falsified.

A fault in a Block RAM cell can lead to a failure with the next read access. Often, the memory is not immediately accessed and the manifestation of the failure is delayed.

3.3. Classification of Mitigation Techniques for Spaceborne SRAM-based FPGAs

Figure 3 shows an overview of fault management strategies, classified according to the aforementioned terminology, together with the corresponding mitigation techniques surveyed in this article.

During run-time, failure masking techniques can be used to tolerate failures. Failure masking is usually achieved by redundancy. Most commonly, spatial redundancy is applied, for instance TMR, Partial TMR, Duplication with Compare (DWC) or Reduced Precision Redundancy (RPR). Alternatively, information redundancy techniques can be used to detect and mask failures in certain types of circuits, for example Error Detection and Correction (EDAC) codes or Algorithm Based Fault Tolerance (ABFT).

Aside from failure masking, failure recovery techniques can be used during run-time too. Failure recovery is usually done by refreshing the memory, which is often referred to as scrubbing.

During design-time, several techniques can help to avoid faults in advance, including tools for the susceptibility analysis (e.g. for the quantification of sensitive configuration bits) and tools which place and route a circuit design in a reliability-oriented way.

4. MITIGATION DESIGN TECHNIQUES AIMED AT RUN-TIME FAILURE TOLERANCE

In this section a review of the rich field of failure tolerance techniques that can be applied during run-time, targeting both the configuration memory and the user logic is presented. Different scrubbing approaches are outlined with regards to the configuration memory. Besides implementation specific differences (Blind vs. Readback Scrubbing, Device vs. Frame-Oriented Scrubbing, External vs. Internal Scrubbing), two fundamentally different concepts, commonly found in literature, are discussed too. The first concept combines periodic scrubbing with a low-level redundancy approach while the second concept implements an FDIR approach in which the configuration memory is only repaired once a failure has been detected in user logic. For the user logic, different redundancy concepts are surveyed. Again, it turns out that the concepts presented in literature can be roughly divided into two categories. The first type of spatial redundancy is applied to the netlist of the circuit and is thus a quite low-level approach. The second type is a modular redundancy approach in which whole hardware blocks are operated in hot redundancy.

4.1. Configuration Memory

Single and multiple bit upsets in the configuration memory of SRAM-based FPGAs can be mitigated by periodically writing a known to be correct bitstream to the device. This technique is often referred to as scrubbing and several types of implementations can be found in research literature and application notes. In the following, the different methodologies and architectures are classified using a similar terminology as introduced in [Heiner et al. 2009; Herrera-Alzu and Lopez-Vallejo 2013] including:

— Blind vs. Readback Scrubbing.
— Device vs. Frame-Oriented Scrubbing.
— Periodic vs. On-Demand Scrubbing.
— External vs. Internal Scrubbing.

4.1.1. Blind vs. Readback Scrubbing. The most basic methodology is blind scrubbing where the configuration memory is periodically updated with a known to be good copy of the original bitstream. This copy which is sometimes referred to as the 'golden copy', is stored in an external, radiation hardened memory. An external or internal configuration controller controls the download of the bitstream via one of the configuration interfaces of the FPGA. Using the classification shown in Figure 3, blind scrubbing can be described as an operational failure avoidance methodology because faults are handled in a preventive manner without any knowledge about the current health state of the system.

One concern that is sometimes raised in connection with blind scrubbing is the fact that the configuration controller gains write access to the configuration memory even if there is no need for scrubbing. Since the configuration interface is prone to SEFIs, a bitstream download can be affected by radiation effects, potentially leading to a corrupted design. Therefore, Xilinx recommends a SEFI detection before each write access that includes a FAR check and a status and control register check [Carmichael and Tseng 2009].

To further minimise the risk of a corrupted bitstream download, the readback feature of SRAM-based FPGAs can be utilised for scrubbing. Using one of the configuration interfaces, bitstreams cannot be only written to the device but also read back during operation. With this capability, unnecessary write accesses to the configuration memory can be avoided during scrubbing: Before writing a correct bitstream to the device, the current bitstream is read and checked for upsets. Only if upsets are detected, the correct bitstream is eventually written. Such a scrubbing methodology can be identified as a failure recovery technique. Two possible detection mechanisms are commonly used: The first one is based on comparison and relies on golden bitstream copies. The current bitstream is read back from the FPGA and compared to the golden copy, either by bit-wise comparison or simpler, by calculating a Cyclic Redundancy Check (CRC) checksum during readback which can then be compared with the CRC value of the golden copy (later referred to as CRC readback scrubbing). If a mismatch is detected, the golden copy is used to overwrite the current bitstream.

The second detection mechanism is based on information redundancy and uses the Error-Correcting Code (ECC) bits which are embedded into each configuration frame. This Single Error Correction and Double Error Detection (SECDED) code allows the detection of single and double bit upsets and the correction of single bit upsets. For multiple bit upsets with more than two wrong bits, the syndrome value is indeterminate [Xilinx 2009]. During readback, a *syndrome* value is calculated by an ECC logic that must be initiated as a user primitive called `FRAME_ECC_VIRTEX4` for Virtex-4 devices, and respectively `FRAME_ECC_VIRTEX5` for Virtex-5 devices [Xilinx 2012c]. The syndrome value does not only identify upsets but can also localise single upsets. Hence, two possible failure recovery methodologies can be combined with the ECC logic: Either the erroneous bit is flipped and the corrected bitstream is written back to the device (later referred to as ECC readback scrubbing) or the whole bitstream is overwritten with a golden copy from memory.

A methodology that allows the detection and correction of multiple bit upsets using a custom-built EDAC core is presented in [Lanuzza et al. 2010]. The authors divide a configuration frame into several data segments and interleave the bits of these data segments. Then, an EDAC check code is calculated for each segment. Since adjacent memory cells are distributed over several data segments, multiple bit upsets can be detected and corrected. A recent work that advances this concept is proposed in [Rao et al. 2014]. Here, the process of detecting multiple bit upsets and correcting them is separated. The detection is done using a novel lightweight error detection coding technique called *Interleaved Two Dimensional Parity* while the correction utilises so-called *erasure codes* as can be found in reliable storage devices and similar applications.

Starting with the Virtex-5 architecture, an internal readback CRC logic allows a continuous and automatic readback in the background [Xilinx 2012c]. In the first readback round, a golden CRC checksum is calculated which is later used to compare the CRC values of the subsequent rounds to. Once a mismatch has been detected, dedicated user logic can initiate a reconfiguration of the device or a bitstream repair using the ECC logic [Chapman 2010]. A summary of Blind and Readback Scrubbing approaches is given in Table III.

4.1.2. Device vs. Frame-based Scrubbing. All scrubbing methodologies mentioned in the last paragraph can use different bitstream sizes. However, the configuration memory is typically scrubbed with a full bitstream or on a frame by frame basis. The first case, sometimes also referred to as device-based scrubbing, requires a rather simple implementation. Except of the modified header information, the bitstream can be directly downloaded from a memory to the configuration interface. One drawback of this solution is the susceptibility of the configuration interface to SEFIs. If such an upset occurs during download, the whole design is likely to become corrupted. Frame-based scrubbing requires a more complex configuration controller implementation because each frame must be prepared before download. But, the benefit of this approach is the possibility to isolate the effects of a SEFI to a single frame. Aside from the in-

Table III. Summary: Blind Scrubbing vs. Readback Scrubbing.

METHODOLOGY	ATTRIBUTES
Blind Scrubbing	— Simple implementation.
	— Robust for MBU mitigation.
	— Unnecessary write accesses.
Readback & Comparison	— Comparator necessary.
	— Robust for MBU mitigation.
	— Reduced risk of corrupted download.
Readback & EDAC	— Rather complex implementation.
	— Cannot cope with more than two upsets per frame.
	— Reduced risk of corrupted download.

Table IV. Summary: Device-based vs. Frame-based Scrubbing.

METHODOLOGY	ATTRIBUTES
Device-based	— Simple implementation.
	— Scrubbing SEFIs may affect the whole design.
Frame-based	— Increased implementation complexity.
	— Decreased scrubbing speed.
	— Scrubbing SEFIs can only affect one frame.

creased complexity of implementation, the scrubbing speed is decreased too. First, a SEFI check must be done before downloading each frame. Secondly, each frame bitstream comes with an overhead due to its header. Finally, after each frame a dummy frame must be written to flush the pipeline [Carmichael and Tseng 2009].

In some applications increased scrubbing speed is desired. This is especially true for applications in which scrubbing is used as the only mitigation technique. In [Sari and Psarakis 2011], such a 'low-cost' strategy, based on an idea presented in [Asadi and Tahoori 2005], is proposed. The authors point out that many configuration frames are scrubbed although they contain no or only a small number of essential bits. As a consequence, they propose to constrain the placement of the design in such a way that the number of frames with essential bits is minimised. Then, the frame-based scrubber must only take this subset of frames into account. A summary of Device and Frame-based Scrubbing approaches is given in Table IV.

4.1.3. Periodic vs. On-Demand Scrubbing. In many designs, the scrubbing process is independent of other mitigation techniques. Then the configuration memory is periodically scrubbed, respectively scanned for upsets with a fixed scrubbing rate.

Alternatively, the scrubbing process can also be triggered by a failure detection mechanism. Such a methodology can be advantageous in systems where continuous scrubbing is unwanted. Eventually, the availability of a system depends on the time a faulty component remains unrepaired. This time can be minimised by either increasing the scrubbing frequency or by implementing a mechanism that can trigger a repair

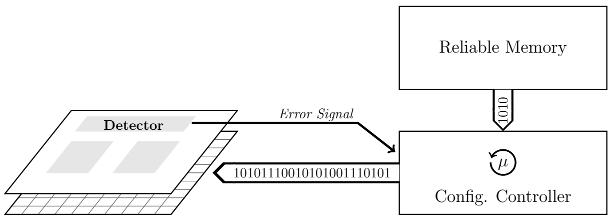

Fig. 4. On-Demand Scrubbing: A failure detection mechanism, e.g. a majority voter, flags an error code to the configuration controller once a faulty component has been detected. As a consequence, the configuration controller initiates a scrubbing process.

process immediately after failure detection. With the aid of stochastic models, Siegle et al. showed in [Siegle et al. 2013] that in principle, on-demand scrubbing always maximises the availability.

Depending on the implementation, on-demand scrubbing can be power saving because the scrubbing logic is only active if required. Becker et al. investigated in [Becker et al. 2007] the power consumption of Virtex-II devices during reconfiguration. Although the authors use a LZSS decompression core during reconfiguration, the power consumption is increased by only 95 mW. Nevertheless, the avoidance of additional resource overhead is always beneficial, especially for systems where failure detection mechanisms are implemented anyway.

In literature, on-demand scrubbing is mainly mentioned in connection with systems utilising dynamic partial reconfiguration. In many proposed systems, spatial redundancy, e.g. TMR, is implemented by using redundant reconfigurable modules and a majority voter within the static area. Due to the physical separation of the reconfigurable modules, on-demand scrubbing can be advantageous here. The majority voter can be easily designed as a failure detection mechanism which is able to identify a faulty module and which can trigger a scrubbing process on-demand, targeting only the faulty component.

Paulsson et al. presented such a system in [Paulsson et al. 2006] for Virtex-II devices. The authors use so-called Dynamic TMR or Hardware Test Benches as failure detection mechanisms and reconfigure a partition only if a failure has been detected. Researchers at University of Arizona proposed similar mechanisms for their SCARS system that is based on Virtex-5 devices [Sreeramareddy et al. 2008]. Here, a faulty reconfigurable module is only scrubbed after a failure has been detected by software routines. Jacobs et al. propose a similar approach in [Jacobs et al. 2012b]. Again, failures in reconfigurable modules are detected by voters or comparators and scrubbing is triggered only for the faulty module on-demand. Straka and his colleagues at University of Brno also work on a fault tolerant framework for SRAM-based FPGAs. Very similar to the already mentioned approaches, a so-called Generic Partial Reconfiguration Controller receives error signals from reconfigurable modules and triggers on-demand scrubbing if required [Straka et al. 2010b]. Azambuja et al. also use majority voters as failure detection mechanisms and scrub a faulty reconfigurable module only after a failure has been detected [Azambuja et al. 2008; Azambuja et al. 2009]. The authors emphasise the increased repair speed compared to a full reconfiguration. In [Iturbe et al. 2009], Iturbe et al. propose a fault management strategy in which they combine on-demand blind scrubbing, triggered by a majority voter as failure detection mechanism, with ECC readback scrubbing.

A methodology that aims at speeding up the on-demand scrubbing process is presented in [Nazar et al. 2013]. The authors analyse the statistical distribution of sensitive bits within a partial bitstream. Instead of starting the scrubbing process from the

Table V. Summary: Periodic vs. On-demand Scrubbing.

METHODOLOGY	ATTRIBUTES
Periodic	— Mean Time To Recover (MTTR) depends on scrubbing rate. — No failure detection mechanisms on user logic layer necessary. — Possibly increased power consumption.
On-Demand	— MTTR is minimised. — Partial scrubbing is possible. — Access to configuration memory is minimised. — Failure detection mechanisms on user logic layer necessary.

first byte position of the bitstream, it is started from the one frame for which the authors calculated that this start position minimises the Mean Time to Recover (MTTR). For a set of benchmark circuits, an average MTTR reduction of 30% was achieved. A methodology with a similar aim is presented in [Bolchini et al. 2011]. The authors partition a circuit design and apply a specific redundancy scheme (like DWC or TMR) to each partition. If one of these partitions is detected to be faulty, it is scrubbed by an external reconfiguration controller on demand. The authors developed an algorithm which optimises the floorplanning of the different partitions to find an optimal solution in terms of reconfiguration time, area and performance overhead. Results for a set of example circuits suggest that the reconfiguration time can be heavily reduced although this reduction is at the cost of an increased area and performance overhead. A summary of Periodic and On-Demand Scrubbing approaches is given in Table V.

4.1.4. External vs. Internal Scrubbing. The scrubbing logic can be implemented internally or externally. From the available configuration interfaces, the SelectMAP interface is commonly used for external scrubbing due to its high throughput rates. ICAP, the internal counterpart to SelectMAP, can be used if the scrubbing logic is implemented on user logic layer. Internal scrubbing is sometimes seen as a ’low-budget’ solution because it does not necessitate an external configuration controller and a memory for the golden bitstream copies. It can be argued, however, that in most space applications a radiation hardened supervisor as well as reliable memory for the initial bitstream configuration is available anyway.

External scrubbing via the SelectMAP interface is commonly seen as the more robust approach and is also recommended by Xilinx [Carmichael and Tseng 2009]. Melanie Berg and other researchers at NASA come to similar conclusions in [Berg et al. 2008] where the authors compare an external blind scrubber to an internal ECC readback scrubber by Xilinx. The internal scrubber is based on a PicoBlaze microcontroller and its design was published in the not longer available application note [Jones 2007]. Using heavy-ion SEE radiation testing, it was found that the external scrubber was always recoverable without the need for a reset or power cycle whereas the internal scrubber was never recoverable. Thus, the internal scrubber consistently reached a state where it could not operate anymore, either because of MBUs which cannot be handled by the scrubber or because the scrubber itself was hit by ions.

Heiner et al. from Brigham Young University improved the fault tolerance of the same Xilinx scrubber design by applying TMR and Block RAM scrubbing [Heiner et al. 2008]. In radiation tests it was found that the improved scrubber performs much better but still, in more than 45% of all tests the design failed at some point, requiring a sub-

Table VI. Summary: External vs. Internal Scrubbing.

METHODOLOGY	ATTRIBUTES
External	— Robust. — Radiation hardened external controller and memory needed.
Internal	— In case of ECC readback scrubbing, MBUs cannot be repaired. — No external controller and no golden bitstream copies necessary.

sequent full reconfiguration of the device. The authors assume that the missing ability of ECC readback scrubbers to repair MBUs was the main reason for this behaviour.

Ebrahim et al. from University of Edinburgh also work on a fault-tolerant ICAP controller in the course of their R3TOS system [Ebrahim et al. 2012]. The controller is based on Xilinx' `XPS_HWICAP` core. The controller is not only used for scrubbing but also for partial reconfiguration. Similar to the ICAP controller described above, the scrubber is an ECC readback scrubber. To improve its fault-tolerance the authors apply spatial redundancy but instead of applying TMR to the whole controller, only a so-called Recovery Module is triplicated. This module, on the other hand, is able to gain access to the ICAP interface only for the sake of recovering the controller from failures. A summary of External and Internal Scrubbing approaches is given in Table VI.

4.1.5. Integration with Dynamic Partial Reconfiguration. Most scrubbing approaches described in literature assume a static user design. If dynamic partial reconfiguration is used as part of the normal operation, however, e.g. to time-share chip area by swapping different modules during runtime, the reconfiguration and scrubbing process must be somehow orchestrated because only one of them can gain access to the configuration interface at the same time. Furthermore, if blind scrubbing or CRC readback scrubbing is used, the golden bitstream must be kept updated after each partial reconfiguration to mirror the currently running design.

One approach to overcome these problems is described by Heiner et al. in [Heiner et al. 2009]. The authors use a CRC readback scrubber as described earlier. Instead of downloading the bitstream of a reconfigurable module and updating the golden bitstream afterwards, the authors suggest to simply integrate the bitstream of the reconfigurable module into the golden bitstream. During the next scrubbing cycle the scrubber detects a discrepancy between the golden bitstream and the bitstream which has been read back from the device due to mismatching CRC sums. As a consequence, it will then 'repair' the bitstream by writing the updated frames to the device.

4.2. User Logic

While failure recovery takes mainly place on configuration memory layer, failure masking is implemented on user logic layer using some kind of redundancy. Most commonly, spatial redundancy is used but also information and temporal redundancy can be found for specific components.

4.2.1. Spatial Redundancy. By far the most common form of spatial redundancy is TMR. In this approach, all components of a circuit are triplicated as depicted in Figure 5 and a majority voter is placed at the end which chooses the correct output.

To decrease the susceptible area the circuit can be further partitioned by adding additional voters as can bee seen in Figure 6. The possible increase of availability is discussed by McMurtrey et al. in [McMurtrey et al. 2008] using Markov chains. The

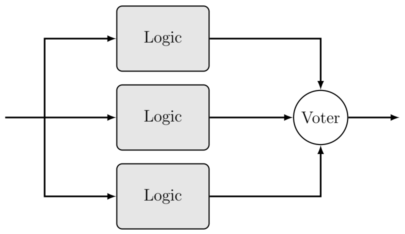

Fig. 5. Triple Modular Redundancy.

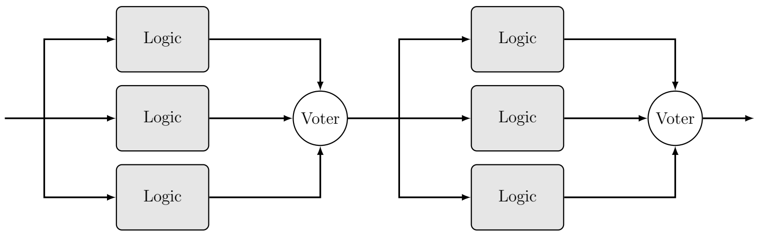

Fig. 6. Triple Modular Redundancy with two partitions.

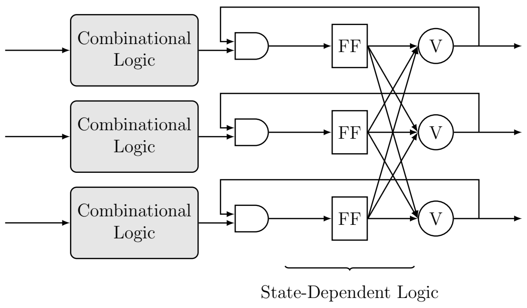

Fig. 7. X - Triple Modular Redundancy: Voters are placed in the feedback paths of state-dependent logic to allow automatic re-synchronisation.

authors show that the reliability is indeed increased because the area of each circuit stage and therefore the chance that more than two redundant circuit stages fail is decreased.

Since the voter is a single point of failure it is usually triplicated too. As mentioned earlier, upsets affecting feedback loops, e.g. counter or state machines, can be problematic because the failure is trapped in the loop. To overcome this problem voters can be placed inside feedback loops. This technique, sometimes also referred to as XTMR (Xilinx TMR) [Bridgford et al. 2008; Adell and Allen 2008], synchronises the flip-flops automatically after repair, see Figure 7.

Most commonly, TMR is applied to the netlist of a circuit using automatic insertion. Several commercial and academic software tools are available, including the TMRTool by Xilinx [Xilinx 2014], Precision Hi-Rel by Mentor Graphics [Mentor Graphics 2014] and Synplify Premier by Synopsys [Synopsis 2012]. A notable free collection of tools is the BYU EDIF Tool suite, developed at Brigham Young University [Brigham Young University 2014].

Researchers at Politecnico di Torino found analytically that TMR protected circuits are still prone to SEUs because in some cases one single configuration bit upset can

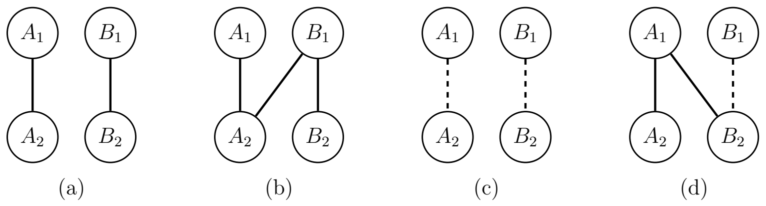

Fig. 8. Modes of possible SEU induced effects [Sonza Reorda et al. 2005b].

lead to multiple failures on user logic layer, invalidating the TMR approach [Sonza Reorda et al. 2005b]. Logic blocks inside the fabric of the FPGA are interconnected via switch boxes which are built from Programmable Interconnect Points (PIPs). The authors found that one single configuration bit can control two or more PIPs and they identified three possible modifications caused by one SEU, as depicted in Figure 8. Given a pair of connections (a), a short between the connections can occur (b), both connections can be opened (c) or a bridge between the connections can be created (d). If the connections belong to two redundant circuits of a TMR system, the voter will choose a wrong or falsified output. For Virtex-II devices, this failure mode which is sometimes also referred to as Domain Crossing Error (DCE), was partly confirmed by Quinn et al. [Quinn et al. 2007c] by fault injection experiments, although the authors point out that SEU induced "DCEs are possible when TMR is incompletely applied to a design, but they appear to be rare otherwise". However, the authors found that TMR can be even defeated by multiple bit upsets with two or more bits. Using a stochastic model, they predict a worst case probability for DCEs of 0.36% for Virtex-II devices and up to 1.2% for Virtex-5 device.

One obvious drawback of TMR is the large area and thus power overhead that can exceed more than 200%. To decrease the overhead of TMR, several alternatives were proposed in literature. One example is *Partial TMR* as discussed by Pratt et al. in [Pratt et al. 2006; Pratt et al. 2008]. The basic idea is to apply TMR only to feedback paths and optionally to their inputs to avoid so-called persistent errors [Morgan et al. 2005] in state-dependent logic. By doing so, only failures with a transient nature can occur. The authors demonstrated for a DSP Kernel design that the number of persistent bits decreased by 63% if only the feedback is triplicated at the cost of 26% hardware overhead. By applying Partial TMR to feedback paths and their inputs, the persistent bits were reduced by two orders of magnitude at the cost of 40% hardware overhead. This Partial TMR approach is part of the already mentioned TMR Tool by Brigham Young University.

Another drawback of TMR is its strong impact on the performance of a circuit, especially if the circuit contains many TMR partitions. For instance, Kastensmidt et al. analysed the performance of a digital FIR filter design in [Kastensmidt et al. 2005]. While the implementation without TMR could achieve a performance of 154 MHz, the performance of the TMR version with a maximum number of possible partitions dropped down to 123 MHz.

A less common form of spatial redundancy is DWC where a circuit is duplicated and the output of the redundant circuits is compared by a comparator. Naturally, this mechanism is only able to detect failures instead of masking them. It can be useful for systems which allow a downtime but need to implement fail-silent behaviour or it can also be used as a failure detection mechanism that triggers scrubbing on-demand. Johnson et al. investigated DWC in detail [Johnson et al. 2008]. By means of fault injection experiments and radiation tests, the authors found that DWC can detect approximately 99.85% of all failures at the cost of ca. 200% hardware overhead.

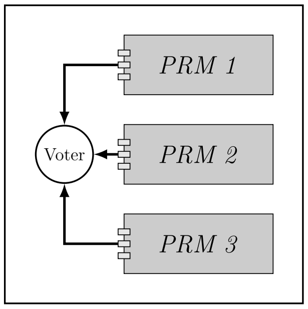

Fig. 9. A system utilising dynamic partial reconfiguration with three redundant reconfigurable modules (PRM) and a voter in the static area.

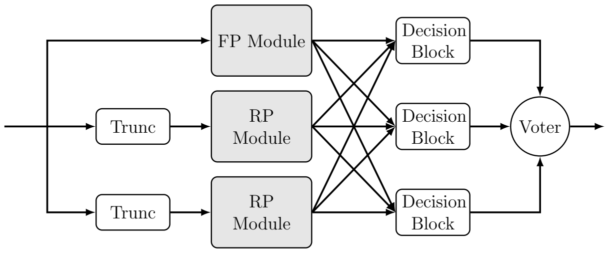

Fig. 10. Basic Principle of Reduced Precision Redundancy with one full-precision (FP) module and two reduced-precision (RP) modules.

An extension to DWC is proposed by Anderson et al. in [Anderson et al. 2010]. The authors use DWC as failure masking technique by taking advantage of the probabilistic distribution of a circuit's output. The authors use a system comprising five cascaded half-band filters whose output is characterised by a distinct, non-uniform distribution. Once the comparator detects a mismatch, it selects the output with the higher probability by checking a stored histogram. Due to the discrete bins of the histogram, correct detection percentage can be bad for lower significant bits. Therefore, the authors combine this approach with an additional history buffer filled with the last decisions.

Instead of applying spatial redundancy to the netlist of a circuit, the whole circuit can also be seen as a module which is then duplicated or triplicated. This approach is easy to implement [Habinc 2002] but lacks the automatic re-synchronisation after repair which can be achieved by netlist approaches like XTMR. However, a benefit of modular redundancy is the physical separation of the modules which allows a partial (on-demand) scrubbing [Azambuja et al. 2008].

Today's usage of modular redundancy is often driven by systems that utilises dynamic partial reconfiguration and in which the design is broken up into physically separated partitions anyway. Thus, it is no surprise that the earlier mentioned systems proposed by Paulsson [Paulsson et al. 2006], Jacobs [Jacobs et al. 2012b] and Straka [Straka et al. 2010a] are all based on modular redundancy. All these systems have in common that one or more voters and/or comparators are placed in the static area, similar to the scheme depicted in Figure 9. The failure detection mechanism monitors the output of redundant modules and triggers an on-demand scrubbing process once a failure has been detected. An interesting aspect of such a system is its adaptability as pointed out by Jacobs et al. in [Jacobs et al. 2012b]: Since redundant modules can be added and removed on-demand, the system availability can be tuned according to external constraints in terms of area and power overhead.

Another technique that can be understood as a modification of modular TMR is RPR. The idea of applying RPR to FPGAs in space systems goes back to several works at the U.S. Naval Postgraduate School, Monterey [Snodgrass 2006; Sullivan 2008; Sullivan et al. 2009; Gavros et al. 2011] and was later followed up by Bratt et al. [Pratt et al. 2011; Pratt et al. 2013]. Instead of using three redundant copies of a module, one module processes data with full precision while the two other modules process the data with reduced precision. Hence, RPR is suitable for algorithms that process data which is represented by a block of bits ordered in increasing or decreasing importance, e.g. fixed-point numerical problems [Sullivan 2008]. A decision block determines if a failure has occurred as follows [Pratt et al. 2011]:

```
if(($|FP_{Out} - RP1_{Out}| > T_h$) and ($RP1_{Out} = RP2_{Out}$))
then $output \Leftarrow RP2_{Out}$ else $output \Leftarrow FP_{Out}$
end if
```

The full precision output FP_{Out} is always chosen if no failure has been detected or if the reduced precision modules disagree. The decision further depends on a threshold level T_h: The full precision module is assumed to be correct if its output differs less than T_h from the reduced precision module output $RP1_{Out}$. For an FIR filter design, Bratt et al. showed in [Pratt et al. 2011] that the failure rate can be improved by ca. 200 times compared to an unmitigated design at the cost of ca. 70% hardware overhead. For the same circuit, a full TMR mitigation approach improves the failure rate by ca. 1200 times at the cost of 208% hardware overhead.

4.2.2. Information Redundancy. Although information redundancy techniques are mainly applied to memory and communication channels, several circuits can profit from them too. Information redundancy techniques add redundant bits to data to be able to detect or even correct falsified information. An example for the first case is the CRC code while error correction can be achieved for instance by Hamming codes.

EDAC techniques are often applied to state machines. The states can be encoded using different coding schemes, e.g. binary, one-hot or Grey. In addition, parity bits can be added to achieve a Hamming code which enables the detection or correction of bit upsets. In [Burke and Taft 2004], the robustness of state machines with binary, one-hot, Hamming with a distance of 2 (H2) and Hamming with a distance of 3 (H3) codes was tested using synchronous fault injection. According to the authors, H3 encoding can fully handle single errors and is least affected by double bit errors. State machines with H2 encoding have less overall errors than state machines with one-hot encoding and about half the error rate of state machines with binary encoding. Due to the hardware overhead and the decreased performance, the author concludes that H2 encoding is the best compromise in terms of size, speed and fault-tolerance. For SRAM-based FPGAs, however, these results should be regarded with care because the influence of configuration memory upsets is not taken into account. Using fault injection, Morgan et al. found in [Morgan et al. 2007] that the additional required logic can "potentially add more unreliability than the reliability it adds to the original circuit". An actual implementation of a fault-tolerant state machine that uses Hamming codes is described for instance in [Skliarova 2005].

An interesting application of information redundancy for the sake of error detection and correction is ABFT which goes back to the work of Huang and Abraham in 1984 [Huang and Abraham 1984]. ABFT is used to implement fault tolerant matrix operations. Recently, Jacobs et al. investigated in [Jacobs et al. 2012a] the overhead and reliability of ABFT in FPGA systems. The authors use a Multiply and Accumulate (MAC) unit where the inputs are fed from Block RAM and where the output data is written back to Block RAM. One of the implementations uses a second MAC unit that

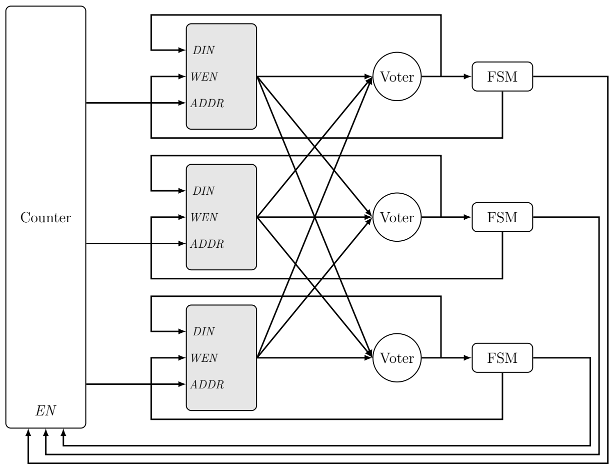

Fig. 11. Simplified diagram of a Block RAM to which TMR and scrubbing is applied.

generates and validates the checksums. Compared to TMR, the hardware overhead is as follows: 21% LUT Overhead (TMR: 148%), 24% Flip-flop Overhead (TMR: 84%), 0% Block RAM Overhead (TMR: 200%) and 25% DSP48 Overhead (TMR: 200%). From 100,000 injected faults, 1,216 errors occurred in the unmitigated design, 351 errors in the ABFT design and 42 errors in the TMR design.

4.3. Block RAM

The embedded RAM blocks in Virtex devices need special care regarding the mitigation because very similar to the configuration memory, upsets in Block RAM can accumulate, leading to an ever decreasing reliability of the memory. Although the Block RAM content can be read out via the configuration interface, external scrubbing is not possible during operation because the RAM can not be accessed by configuration and user logic at the same time [Xilinx 2009].

The recommended mitigation approach by Xilinx [Miller et al. 2008] includes TMR for the Block RAM (including triplicated voter) and a memory scrubbing engine implemented on user logic layer, similar to the scheme depicted in Figure 11. This method can only be applied to single-port RAMs because eventually, they must be replaced by dual-port counterparts as the second port is required for scrubbing. A counter auto-increments the address of the second port and once the voter detects a failure, the counter is stopped, the voted and thus corrected output is written back to memory and the counter is started again.

Rollins et al. present a comprehensive comparison of fault-tolerant memories in SRAM-based FPGAs in [Rollins et al. 2010]. The study covers the TMR and scrubbing approach as described above plus several information redundancy techniques, including duplication with error detection codes and error detection and correction, applied in different configurations (with and without triplicating the logic, with and without memory scrubbing). The fault-injection results are similar to what Morgan observed when applying information redundancy to state machines [Morgan et al. 2007]: First, the area overhead of the information redundancy techniques sometimes exceeds the TMR approach and secondly, the failure rates are always even worse than the rate for the unmitigated design. Only if Block RAM is used as ROM, some of the information redundancy techniques perform slightly better than the unmitigated design but always worse than TMR.

5. MITIGATION DESIGN TECHNIQUES AIMED AT DESIGN-TIME FAULT AVOIDANCE

Another group of mitigation techniques, which can be best described as fault avoidance techniques applicable during design-time are discussed in this section. This group includes analytic approaches that are aimed at analysing the sensitivity of circuits but also at reducing this sensitivity, for instance, by re-routing the circuit design.

As already mentioned in Section 4.2, researchers at Politecnico di Torino found that the TMR approach can be invalidated by SEUs because a single configuration bit upset can lead to multiple failures on the user logic layer. By observing and analysing this fault mechanism, a set of tools has been developed which allow the avoidance of these faults already at design time.

In [Sonza Reorda et al. 2005a], a *Static Analyzer* tool (STAR) is presented. Based on the researcher's knowledge about the proprietary bitstream format, the tool is able to determine the critical configuration bits of a circuit design. If TMR is applied to this circuit, the tool also determines the bits which can invalidate the TMR approach as described before.

Based on STAR, a *Reliability-Oriented Place and Route* (RoRA) algorithm has been developed [Sterpone et al. 2005; Sterpone and Violante 2006]. By re-routing the circuit, RoRA can avoid the problem of single upsets attacking the TMR approach. The authors were able to demonstrate that RoRA minimises the number of wrong answers of circuitry to which TMR or XTMR is applied drastically. Furthermore, the authors point out that RoRA can identify critical configuration bits much faster than fault injection experiments. However, compared to the original TMR version, the re-routing decreases the performance of the circuit.

To increase the performance of circuits to which TMR is applied, a tool called V-Place was then presented in [Sterpone and Battezzati 2008] and it was shown that this tool can optimise the circuit's frequency up to 44%.

The STAR tool was later updated to STAR-LX. The main advantages are the reduction of the analysis time of more than five times as well as the ability to analyse the dynamic evaluation of the design under the presence of SEUs [Sterpone and Violante 2007]. A modification which can analyse the effects of MBUs called STAR-MCU is presented in [Sterpone and Violante 2008]. To mitigate the effects of MBUs, a new placement algorithm called PHAM was presented in [Sterpone and Battezzati 2010].

For engineers and scientists who are interested in building their own netlist analysis and CAD tools, researchers from Brigham Young University present an interesting JAVA toolkit called *RapidSmith* [Lavin et al. 2011]. The toolkit offers a rich Application Programming Interface (API) to parse, analyse and manipulate XDL files (which can be easily created from Xilinx netlists). A very recent work that makes use of this toolkit is presented in [Sari et al. 2014]. In this paper, the authors estimate the susceptibility of an FPGA design. To determine the number of sensitive bits that are responsible for the different SEU induced effects, as discussed in Section 4.2.1 and shown in Figure 8, the authors conduct the post-routing analysis using appropriate API functions of RapidSmith.

6. SIMULATION AND EMULATION OF SINGLE EVENT EFFECTS

This section gives a brief overview of techniques for simulation and emulation of single event effects, including accelerated radiation testing and fault injection. These techniques are necessary to validate any mitigation methodology applied to the design.

6.1. Accelerated Radiation Testing

Although first in-flight data for Virtex-4 devices have been published [Quinn et al. 2012], the common way to gain reliable static and dynamic cross-sections for these devices is by means of accelerated radiation testing.

To simulate high-energy galactic cosmic rays and solar event heavy ions on ground, the FPGA is exposed to low energy ions available in particle accelerators. The quality of the simulation can be evaluated by the amount of energy lost per unit length of track, also referred to as Linear Energy Transfer (LET). Because the SEE sensitive region is rather thin, ions with lower energies are sufficient for simulation as long as the LET is similar to the one of galactic cosmic rays and solar event heavy ions. The typical energy range used for simulation is of the order of several $^{\mathrm{MeV}}/_{\mathrm{A}}$ and the penetration range is between 30 and 100 μm. In general, the estimation of the SEU sensitivity using this concept is rather conservative [Barth et al. 2004]. The machine most commonly used for heavy ion SEU testing is the cyclotron. Several accelerators can be found across Europe, for instance the facilities GANIL and IPN in France, SIRAD and LNS in Italy, GSI in Germany and the HIF in Belgium [ESA/ESCIES 2010].

Single event phenomena can also be induced by protons. Linear accelerators and cyclotron accelerators are capable of generating protons with sufficient energy to simulate solar flare and proton belt conditions [Holmes-Siedle and Adams 1993].

Regarding Virtex devices, most test result data has been collected at the cyclotron at Texas A&M University and/or at the cyclotron at Lawrence Berkeley National Laboratory and published by Los Alamos National Laboratory, NASA Goddard Space Flight Center and the Xilinx Radiation Test Consortium. Quinn et al. present in [Quinn et al. 2005] results regarding radiation-induced MBUs in Virtex, Virtex-II and Virtex-4 devices and discuss the general reliability concerns of Virtex FPGAs in [Quinn et al. 2007a]. In [Quinn et al. 2007c] they present the results regarding circuitry to which TMR is applied and discuss the problem of domain crossing errors. In [Quinn et al. 2007b], first results for Virtex-5 are published. In 2009, results regarding the SEU-susceptibility of logical constants were presented [Quinn et al. 2009a]. In the same year, a paper describing their methodology for static and dynamic testing was published too [Quinn et al. 2009b]. The upset characterisation of embedded PowerPC cores is presented by Allen in [Allen et al. 2007] and the more general characterisation for Virtex-4QV FPGAs by Swift in [Swift et al. 2008], finally leading to the summary report published by NASA and Xilinx [Allen et al. 2008]. One year later, a report summarising the results gathered from dynamic testing and from testing of mitigated designs was published by Allen [Allen 2009]. Recently, the static SEU characterisation of Virtex-5QV was presented in [Swift and Allen 2012].

6.2. Fault Injection

As an alternative to accelerated radiation testing, upsets in the configuration memory can also be emulated by fault injection.

Although many different fault injection implementations have been presented in literature, the basic structure is similar for many systems, see Figure 12. Two devices are simultaneously fed by test vectors. One of the DUTs is used as a 'golden' reference while faults are injected into the second DUT. Fault injection is based on bitstream manipulation: Often, a configuration frame is read back from the FPGA via one of the configuration interfaces, one or more bits are flipped and the frame is written back to the device. The outputs of both FPGAs are compared by some mechanism which detects if the fault injection led to a failure. Alternatively, only one DUT can be used and its response is compared to 'golden' answers during the fault injection campaign. In the following, two exemplary European systems are presented.

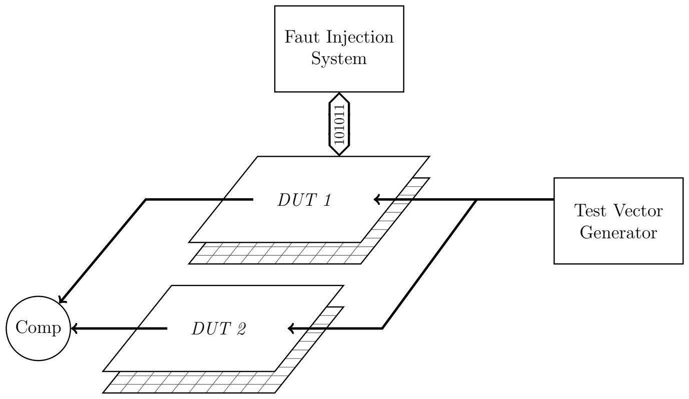

Fig. 12. Example of a fault injection system.

A fault injection system with a long history in Europe is FLIPPER, developed by Alderighi et al. under ESA funding [Alderighi et al. 2007]. The system comprises one Virtex-II Pro FPGA as controller that can be connected to a DUT board hosting the FPGA under test. The controller communicates with a software running on a host PC via USB. In contrast to the example depicted in Figure 12, only one DUT is used and its response is compared to stored, known to be correct answers. Test vectors can be converted from testbench stimuli and are fed into the DUT after one or more faults were injected into the bitstream. Results from FLIPPER were compared to acceleration testing results [Alderighi et al. 2010] and the authors conclude that FLIPPER is an effective tool to evaluate different mitigation techniques due to its capability to predict failure rates, provided that raw configuration bit upset rates of the target environment are known. However, the authors also point out that the failure rate might be underestimated because SEUs in flip-flops, SETs and MBUs are not emulated. FLIPPER was also compared to and used for the analysis of the STAR/RoRA tool as described in Section 5 [Alderighi et al. 2008].

The second fault injection system developed under ESA funding is FT-UNSHADES from University of Seville. Compared to FLIPPER, its initial aim was the emulation of SEUs and MBUs that originate from SETs and thus manifest in flip-flop cells rather than the emulation of configuration memory upsets. The system is based on a Virtex-II and the circuit under test is duplicated and compared within one FPGA. In each campaign, the application is driven to the desired fault injection time, the clock is stopped, the fault is injected into the desired flip-flop(s) of one of the circuits, the clock is restarted and the outputs of both circuits are compared to detect any mismatch [Aguirre et al. 2007a; Aguirre et al. 2007b]. Later, the injection system was updated (FT-UNSHADES-uP) to allow a more in-depth analysis of microcontrollers. The system only uses one circuit under test whose output is compared to the theoretically correct output and it is now able to also inject faults into Block RAMs, LUTs and SRL16s [Napoles et al. 2008; Guzman-Miranda et al. 2008]. More recently, FT-UNSHADES2 has been developed [Mogollon et al. 2011]. The system is based on Virtex-5, and all data management is processed in hardware, leading to much higher fault injection rates. Now, the system can also be used to inject faults into the configuration memory and the usability was increased due to a simplified design flow and a web browser based user interface.

Table VII. Comparison: Research Platforms for SRAM-based FPGAs in space.

	SYSTEM ARCHITECTURE
Braunschweig	Macro-Pipeline Multiprocessor
Brno	Hardwired hardware blocks
Bielefeld/Paderborn	General Purpose SoC
Edinburgh/IKERLAN	Reconfigurable Computer
Florida	General Purpose SoC
Arizona	Several FPGAs hosting single task processors
	COMMUNICATION MECHANISM
Braunschweig	Network on Chip
Brno	Hardwired
Bielefeld/Paderborn	Embedded Macro Bus Structure
Edinburgh/IKERLAN	via ICAP
Florida	PLB Bus
Arizona	OPB Bus
	KEY FEATURES
Braunschweig	NoC used to isolate modules during the reconfiguration process.
Brno	Reconfiguration controller implemented in hardware.
Bielefeld/Paderborn	Embedded Macro to allow flexible placement of modules.
Edinburgh/IKERLAN	Modules are handled as hardware tasks. Communication via ICAP to allow flexible placement of modules.
Florida	Adaptive fault-tolerance.
Arizona	Two-level healing methodology.
	FDIR STRATEGY
Braunschweig	Scrubbing.
Brno	Different redundancy modes and on-demand scrubbing.
Bielefeld/Paderborn	Scrubbing.
Edinburgh/IKERLAN	Scrubbing due to continuous task reconfiguration. Fault-aware task allocator (hard errors). Fault-tolerant ICAP controller.
Florida	Adaptable, modular redundancy and on-demand scrubbing.
Arizona	Software-based fault detection. Partial on-demand scrubbing. Cold module redundancy. Task allocation to another FPGA.

7. RESEARCH PLATFORMS FOR SRAM-BASED FPGAS IN SPACE

This section presents an overview of several research platforms that comprise SRAM-based FPGAs. It summarises concepts, which could possibly be applicable to future spacecraft data processing systems.

Flight heritage for Virtex-4 and Virtex-5 is relatively rare and many of the payloads serve only as technology demonstrators so far. From the publicly available information, it seems that none of the platforms utilises dynamic partial reconfiguration as a functional feature. Although dynamic partial reconfiguration may offer benefits for some projects, it can be assumed that in-flight experience for this unique capability of SRAM-based FPGAs is still a long way off.

However, in the research community, this topic is actively investigated. In the following, six platforms and frameworks are presented that comprise Virtex devices which utilise dynamic partial reconfiguration and which specifically target space applications. A summary of the systems is given in Table VII.

The platforms can be coarsely classified by the way the reconfigurable modules are used. Most of the platforms implement a System on Chip (SoC) in which the reconfigurable modules are connected to a soft Central Processing Unit (CPU) core, i.e. the reconfigurable modules are used as hardware accelerators that can be installed on demand. The other group of platforms uses reconfigurable modules as processors that can process data streams independently and thus without interaction of a CPU.

7.1. Reconfigurable System on Chip

A demonstrator platform called Dynamically Reconfigurable Processing Module (DRPM) is under development at University of Bielefeld and Paderborn, Germany [Hagemeyer et al. 2012]. It is based on a prototype platform called RAPTOR-X64. Aside from a communication module, the system comprises two processing modules, each module including a Virtex-4 FPGA. The reconfiguration controller is part of the FPGA. It is not only used to reconfigure the FPGA via the ICAP interface but also for scrubbing of the configuration memory. The partial reconfigurable modules are connected using so-called *Embedded Macros* which embed a bus structure into tiles. The main motivation for such a structure is given in [Koester et al. 2011]: By dividing a partial reconfigurable area into atomic units called tiles, modules of different sizes can be more efficiently placed at run-time. It was found that an embedded bus structure with shared signals supports the flexible placement of the modules optimally due to its homogeneity. The tool STARECS from Politecnico di Torino [Sterpone et al. 2011] is used to analyse the SEU effects on the system and at present, work regarding the fault-tolerant communication via the Embedded Macros is in progress.

Researchers at University of Edinburgh, UK are also working towards a partial reconfigurable system on Virtex-4 FPGAs. Main objective of the work is a Reliable Reconfigurable Real-Time Operating System (R3TOS), introduced in [Iturbe et al. 2010]. Reconfiguration is done by R3TOS internally through the ICAP port. In the course of the research on R3TOS, an Area-Time Response Balancing Algorithm (ATB) for scheduling real-time hardware tasks was proposed in [Iturbe et al. 2010] and a task allocator in [Hong et al. 2011], which is able to deal with spontaneously occurring faults. The paradigm followed in Edinburgh is that hardware tasks are handled like normal threads in a higher programming language (e.g. POSIX threads). To 'call' a hardware task, the reconfigurable module needs an appropriate interface, which is proposed in [Iturbe et al. 2011a]. In the same paper, an interesting approach for inter-task communication is presented: Instead of utilising a network with a high resource overhead, the data is simply copied from the output buffer of one module to the input buffer of another module by reading the data through ICAP and writing it back. One benefit of avoiding an on-chip communication is the fact, that less wires need to cross the partial reconfigurable modules which increases the flexibility regarding the module placement. Built on the ICAP-based communication, a second task allocator called *Snake* is presented in [Iturbe et al. 2011b] and because R3TOS is heavily making use of the ICAP port, a fault-tolerant ICAP controller is introduced in [Ebrahim et al. 2012].

Researchers at University of Florida, USA are working towards a framework for the usage of Commercial off-the-shelf (COTS) FPGAs in space applications. The system also utilises dynamic partial reconfiguration while one main aspect of their work is adaptable fault-tolerance that is achieved by adding and removing redundant reconfigurable modules depending on external constraints in terms of availability and power [Jacobs et al. 2012b; Yousuf et al. 2011]. The basic structure of the proposed framework is depicted in Figure 13. Several reconfigurable partitions are connected to a controller that comprises failure detectors. The controller itself is connected via a PLB bus to an on-chip Microblaze softcore. The controller, the bus and the softcore, as well as its peripherals, are placed in the static area of the FPGA. The static area is hardened against SEUs by applying TMR to the netlist of the design. The researchers also investigated the suitability of ABFT for such systems [Jacobs et al. 2012a] and presented their own fault injection system [Cieslewski et al. 2010].

Researchers at University of Arizona, USA, are working on a Virtex-5 based partial reconfigurable system called SCARS which is introduced in [Sreeramareddy et al.

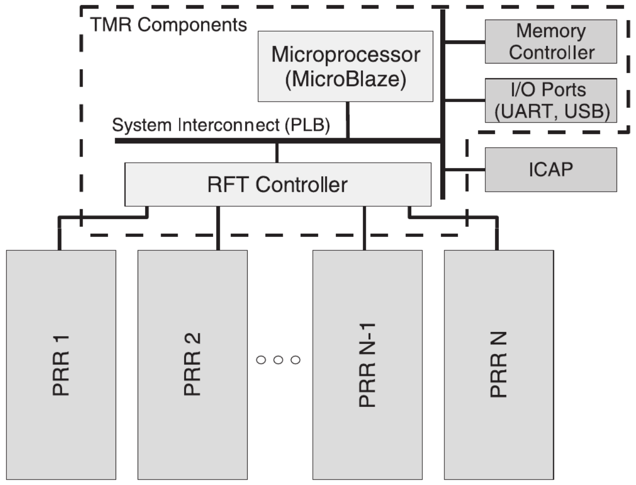

Fig. 13. Framework as proposed by Jacobs et al. (University of Florida). [Jacobs et al. 2012b]

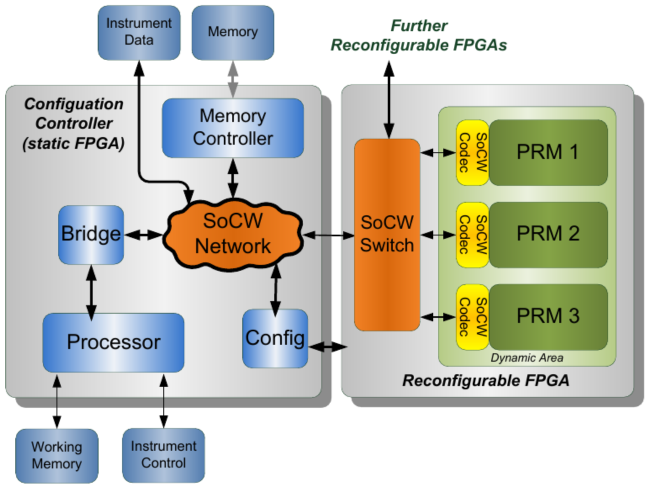

Fig. 14. DRPM by TU Braunschweig/Astrium Ltd. [Michel et al. 2011]

2008] and based on a two-level healing methodology. The system comprises five Virtex-5 FPGAs, each including a Microblaze soft core which is responsible for the self-healing. The partial reconfigurable modules are partitioned together with redundant copies into so-called slots and connected to the Microblaze processor bus. The software running on the Microblaze is responsible for the detection of faulty modules. If a fault is detected, the module is scrubbed through the ICAP interface. If the failure persists, it is seen as an hard error and another redundant module in the slot is activated. The five FPGAs are connected to a master node in a wireless network. Once all modules in a slot are faulty, the task which was running in the faulty slot is moved by the master node to another FPGA.

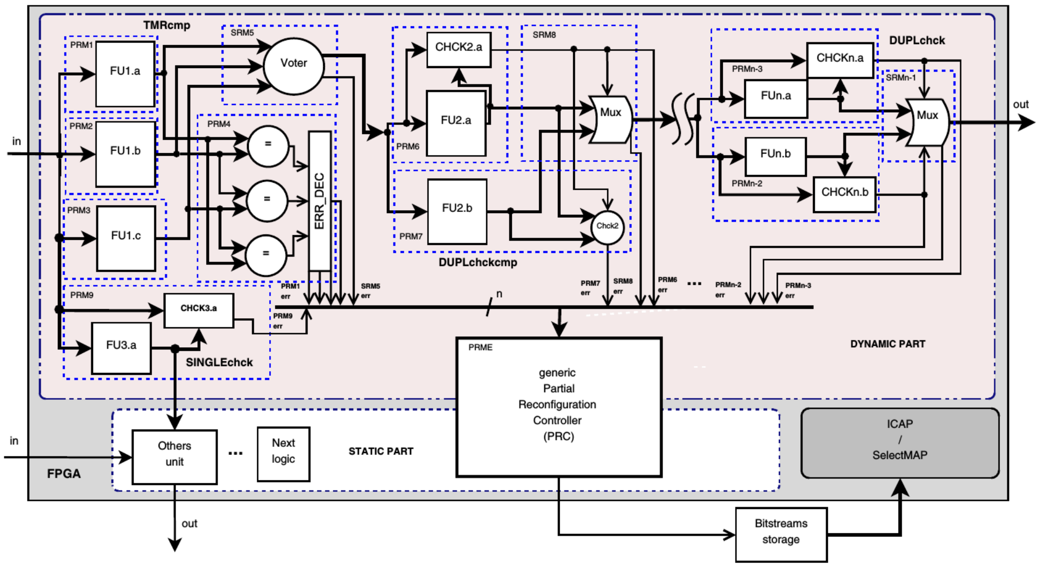

Fig. 15. Framework as proposed by Straka et al. (University of Brno) [Straka et al. 2010a]

7.2. Reconfigurable Stream Processors

The work on reconfigurable FPGAs at the University of Braunschweig, Germany, goes back to a Data Processing Unit for a camera on-board the Venus Express mission. In 2007, an update of this architecture was proposed which also allows in-flight partial reconfiguration [Osterloh et al. 2007]. To interconnect the partial reconfigurable modules, a NoC called SoCWire was proposed in [Osterloh et al. 2008] which is heavily based on SpaceWire, the only space-qualified point-to-point network architecture. The main motivation to use a NoC approach can be found in [Osterloh et al. 2009]: During the reconfiguration process, glitches can occur since frames become directly active during the write cycle. To qualify such a system for space applications, the partial reconfigurable module must be isolated from the host system which can be optimally achieved by a NoC approach. The successful isolation and thus protection against such glitches was proofed in [Osterloh et al. 2010]. SoCWire became part of a demonstrator platform called DRPM, developed in cooperation with the European Space Agency and Astrium Ltd., UK. The demonstrator comprises one or more modules, each module with a radiation hardened reconfiguration controller and two Virtex-4 devices. In 2011, an Advanced Microcontroller Bus Architecture (AMBA) to SoCWire bridge was presented in [Michel et al. 2011] and recently a higher protocol called SoCP, which is also based on a SpaceWire protocol, was introduced in [Michel et al. 2012]. The basic structure of the DRPM can be seen in Figure 14. The reconfiguration controller, depicted on the left hand side of Figure 14, is implemented on a reliable antifuse FPGA. It comprises a SoC with a LEON3 CPU and several peripherals, e.g. memory controllers. The Virtex-4 FPGAs, one of them depicted on the right hand side, are divided into reconfigurable partitions which are interconnected via a SoCWire routing switch.

A more theoretical framework that deals with Fault Detection, Isolation and Recovery (FDIR) for SRAM-based FPGAs is proposed by researchers at University of Brno, Czech Republic. The basic structure of this framework is depicted in Figure 15. Several hardware blocks are arranged in a hardwired processing pipeline. For each hardware block, a different redundancy mode can be applied, e.g. TMR or DWC. The output of the failure detectors is connected to a bus and the health status of the hardware blocks is reported via this bus to a reconfiguration controller. In contrast to other approaches presented here, the reconfiguration controller is implemented in hardware. In the course of this research, the design of online failure checkers was first proposed in [Straka et al. 2007] and later extended to the overall framework [Straka et al. 2010a]. The reconfiguration controller is described in [Straka et al. 2010b] and a fault injection system is presented in [Straka et al. 2012]. Finally, a dependability analysis for the framework is described in [Kastil et al. 2012].

8. SUMMARY OF EXISTING MITIGATION TECHNIQUES

Research on mitigating SRAM-based FPGAs for space applications has engendered a very large number of publications in this research field. As it was shown in Section 3.3 the proposed methodologies can be split into just a few main categories targeting (i) the configuration memory, (ii) the user logic or (iii) the Block RAMs. Regrettably there are not enough design details in the open literature in order to compare the existing methods fairly. In addition, on-board designs are very much dependent on mission objectives and constraints. In most cases, the decision on the use of a particular technique will be based on a trade-off between power, area and performance overheads as well as achievable system availability. In this section a summary of the reviewed mitigation methods is presented which is illustrated by an example decision strategy on selecting the right mitigation technique. It is hoped that the proposed decision strategy can serve as a guidance to researchers and engineers who are novices in the field. However, it is expected that designers will exercise their own judgment and draw their own conclusions taking into account the specifics of their projects when considering our recommendations below.

Figure 16 exemplifies the main steps, which a decision process on selecting a particular mitigation technique for SRAM-based FPGAs on board spacecraft might involve. Considering the fact that space engineering projects usually require strict verification procedures, it might be a wise decision to choose a solution with the lowest possible implementation complexity to meet the given constraints.

If the FPGA design is often reconfigured, for instance because the chip area is shared by several applications, it could be decided not to apply any mitigation technique at all. This is because every time the system is reconfigured it is brought back into a safe initial state, i.e. possible faults in configuration memory and user memory are removed too. This simple solution has no additional power, area or performance overhead at all. If it does not lead to a satisfactory system availability, however, one may add periodic scrubbing which is able to remove faults in the configuration memory during the time the system is running. Still, failures can be 'trapped' in state-dependent user logic but fortunately, the ratio of user memory elements to sensitive configuration bits is often small enough to gain a significant increase in system availability anyway.

If frequent full reconfigurations are not part of the normal operation, one must consider to add a combination of mitigation techniques. Applying spatial redundancy without implementing a repair strategy (like scrubbing) is not recommended because it only extends the time span until the system becomes unreliable. On the other hand, a strategy solely based on scrubbing is not an ideal solution either because faults can manifest as permanent failures within the user memory logic. Thus, a failure detection and/or failure masking technique is usually combined with a failure recovery technique.

Most payload data and imaging applications could be classified as either being of a *processor* type or a *stream* type. The first type comprises all kinds of microcontrollers and -processors or custom-built processors used for data acquisition and similar tasks. These types of applications are never or rarely reset or restarted and may contain a

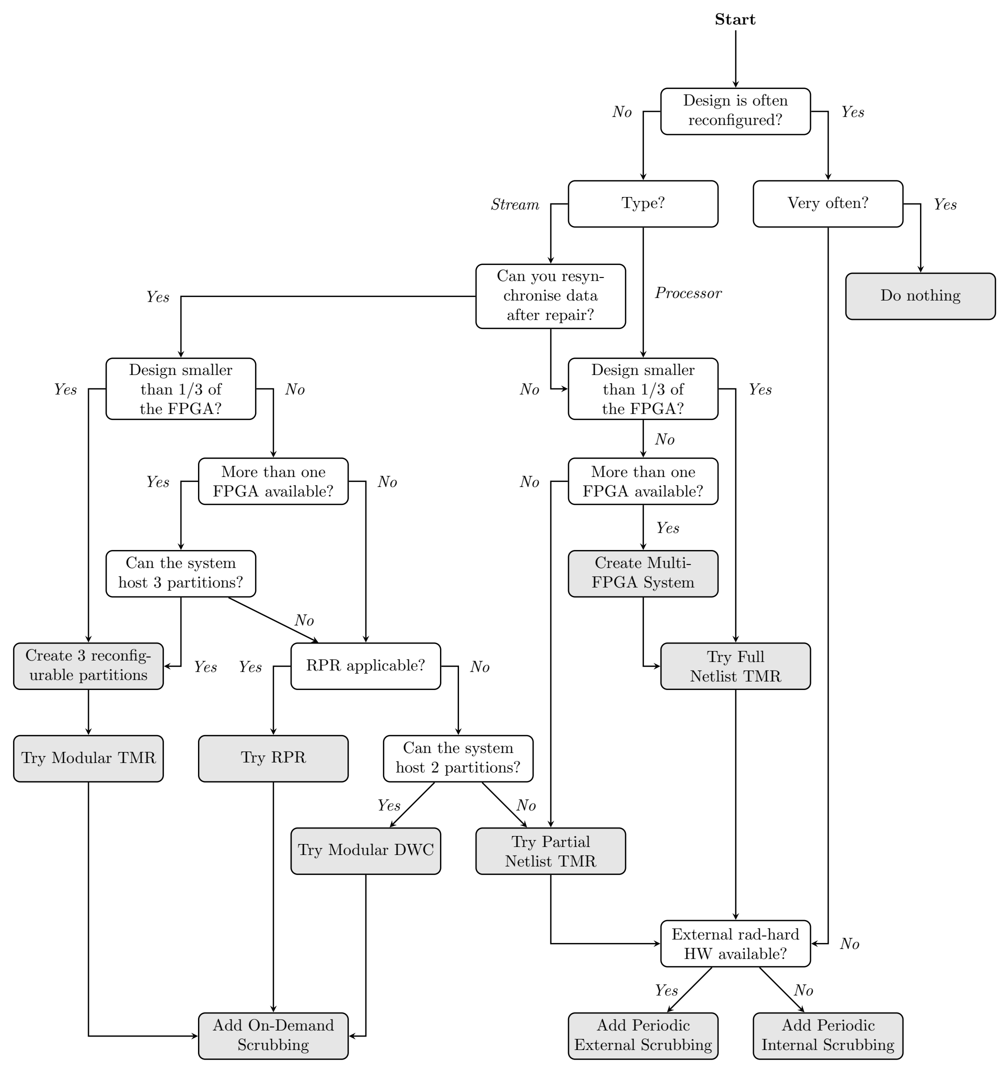
Start
Design is often reconfigured?
No
Yes
Type?
Very often?
Yes
Stream
Processor
Do nothing
Can you resynchronise data after repair?
Yes
No
Design smaller than 1/3 of the FPGA?
Yes
No
Design smaller than 1/3 of the FPGA?
No
Yes
More than one FPGA available?
Yes
No
More than one FPGA available?
No
Yes
Can the system host 3 partitions?
No
Yes
Create Multi-FPGA System
Create 3 reconfigurable partitions
RPR applicable?
Yes
No
Try Full Netlist TMR
Try Modular TMR
Try RPR
Can the system host 2 partitions?
Yes
No
Try Modular DWC
Try Partial Netlist TMR
External rad-hard HW available?
No
Yes
No
Add On-Demand Scrubbing
Add Periodic External Scrubbing
Add Periodic Internal Scrubbing

Fig. 16. Example Decision Flow.

large state space and a huge amount of state variables. Embedded RAM is often used to store data over a long period of time. The second type comprises circuits that can be mainly found in payload data processing applications, for tasks like data compression, encryption or filtering. These types of applications process data block-wise, e.g. image by image, and often, the state space is traversed with each data block. Typically, the number of state variables is low and embedded RAM is mainly used for FIFO buffers.

Two possible mitigation strategies can be followed:

(1) Spatial redundancy (Partial TMR or Full TMR) is applied to the netlist of the circuit and the configuration memory is periodically scrubbed.

(2) The whole circuit is duplicated or triplicated (modular redundancy) and a majority voter, respectively comparator is used as failure detector. Then, the failure recovery can be triggered on demand.

The spatial redundancy mitigation strategy goes well with the *processor* type of application because this low-level redundancy approach allows automatic data resynchronisation after repair. This strategy is also simple to apply because commercial tools exist which automate the insertion. However, one must carefully verify that the used toolchain does not optimise the inserted redundancy away. If the design is small enough and the power budget relaxed, Full TMR leads to the best possible system availability. If the design is too large to apply Full TMR, one could either use a Multi-FPGA system or apply Partial TMR. With Partial TMR, only the feedback loops are typically protected but not the data path within the user logic. As a consequence, errors will become visible as transient failures at the output of the FPGA. No matter which kind of TMR is applied, one must implement periodic scrubbing too. If external radiation-hardened hardware is available or can be afforded, external scrubbing is the more reliable approach. If either blind or readback scrubbing should be used depends on the particular application but blind scrubbing is surely the solution with the lowest implementation complexity.

The modular redundancy mitigation strategy goes well with the *stream* type of application because the user logic can be brought back to a safe initial state after each data block. One drawback of this mitigation approach is the fact that state variables must be synchronised between redundant instances after repair. But since data is processed block-wise and the number of state variables is usually low, technical solutions can be found to execute the data resynchronisation between the data blocks. Despite the increased implementation complexity, this mitigation approach offers some real benefits. For instance, the configuration memory must only be repaired after a failure has been detected, which can maximise the system availability and minimise the power consumption compared to the periodic scrubbing approach. Often, the *stream* type of application does not require maximum possible system availability. For instance, one may tolerate a short downtime with each upset as long as the system is fail-silent. In this case, it is sufficient to only duplicate the circuit and to use a comparator as failure detector. If downtime is not an option, modular TMR can be applied. If not enough chip area is available to triplicate the circuit, one can investigate if Reduced Precision Redundancy (RPR) is applicable. Then, failures can be masked but the output of the circuit might be degraded in precision until the faulty module is repaired. Modular redundancy goes also well with Multi-FPGA systems because redundant instances can be distributed over several FPGAs.

9. CONCLUSIONS

SRAM-based FPGAs are well suited for modern approaches to spacecraft data processing since these devices offer high performance computing capabilities as well as a large amount of logic and memory resources. However, even if some manufacturers offer radiation hardened devices, most systems will demand a methodology to (further) mitigate soft errors triggered by radiation effects.

This article is meant as literature survey and a tutorial for scientists and engineers who need to get a quick yet thorough overview of this topic. A comprehensive coverage of all aspects of radiation effects design mitigation for SRAM-based FPGAs on board spacecraft is given. In addition, design guidelines for the right choice of mitigation techniques are provided too. Despite that the final choice may heavily depend on the project constraints, the proposed recommendations can still serve as a starting point in what is a very difficult and multi-facetted design process.

ACKNOWLEDGMENTS

Sponsorship from ESA under the NPI Programme, Airbus Defence and Space, UK and the University of Leicester is gratefully acknowledged.

REFERENCES

P. Adell and G. Allen. 2008. *Assessing and Mitigating Radiation Effects in Xilinx FPGAs*. JPL Publication 08-9 2/08. NASA Jet Propulsion Laboratory.

M. A. Aguirre, V. Baena, J. Tombs, and M. Violante. 2007a. A New Approach to Estimate the Effect of Single Event Transients in Complex Circuits. *IEEE Transactions on Nuclear Science* 54, 4 (2007), 1018–1024. DOI:http://dx.doi.org/10.1109/TNS.2007.895549

M. A. Aguirre, J. N. Tombs, F. Muoz, V. Baena, H. Guzman, J. Napoles, A. Torralba, A. Fernandez-Leon, F. Tortosa-Lopez, and D. Merodio. 2007b. Selective Protection Analysis Using a SEU Emulator: Testing Protocol and Case Study Over the Leon2 Processor. *IEEE Transactions on Nuclear Science* 54, 4 (2007), 951–956. DOI:http://dx.doi.org/10.1109/TNS.2007.895550

M. Alderighi, F. Casini, S. D'Angelo, M. Mancini, D.M. Codinachs, S. Pastore, C. Poivey, G.R. Sechi, G. Sorrenti, and R. Weigand. 2010. Experimental Validation of Fault Injection Analyses by the FLIPPER Tool. *IEEE Transactions on Nuclear Science* 57, 4 (2010), 2129–2134. DOI:http://dx.doi.org/10.1109/TNS.2010.2043855

M. Alderighi, F. Casini, S. D'Angelo, M. Mancini, D.M. Codinachs, S. Pastore, G. Sorrenti, L. Sterpone, R. Weigand, and M. Violante. 2008. Robustness analysis of soft error accumulation in SRAM-FPGAs using FLIPPER and STAR/RoRA. In *European Conference on Radiation and Its Effects on Components and Systems (RADECS)*. 157–161. DOI:http://dx.doi.org/10.1109/RADECS.2008.5782703

M. Alderighi, F. Casini, S. D'Angelo, M. Mancini, S. Pastore, and G.R. Sechi. 2007. Evaluation of Single Event Upset Mitigation Schemes for SRAM based FPGAs using the FLIPPER Fault Injection Platform. In *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT)*. 105–113. DOI:http://dx.doi.org/10.1109/DFT.2007.45

G. Allen. 2009. *Virtex-4QV Dynamic and Mitigated Single Event Upset Characterization Summary*. JPL Publication 09-4 01/09. NASA Jet Propulsion Laboratory.

G. Allen, G. Swift, and C. Carmichael. 2008. *Virtex-4QV Static SEU Characterization Summary*. JPL Publication 08-16 4/08. NASA Jet Propulsion Laboratory, Xilinx.

G. Allen, G.M. Swift, and G. Miller. 2007. Upset Characterization and Test Methodology of the PowerPC405 Hard-Core Processor Embedded in Xilinx Field Programmable Gate Arrays. In *IEEE Radiation Effects Data Workshop (REDW)*. 167–171. DOI:http://dx.doi.org/10.1109/REDW.2007.4342559

J.-P. Anderson, B. Nelson, and M. Wirthlin. 2010. Using statistical models with duplication and compare for reduced cost FPGA reliability. In *IEEE Aerospace Conference*. 1–8. DOI:http://dx.doi.org/10.1109/AERO.2010.5446660

G.-H. Asadi and M.B. Tahoori. 2005. Soft error mitigation for SRAM-based FPGAs. In *23rd IEEE VLSI Test Symposium (VTS)*. 207–212. DOI:http://dx.doi.org/10.1109/VTS.2005.75

J. R. Azambuja, C. Pilotto, and F.L. Kastensmidt. 2008. Mitigating soft errors in SRAM-based FPGAs by using large grain TMR with selective partial reconfiguration. In *European Conference on Radiation and Its Effects on Components and Systems (RADECS)*. 288–293. DOI:http://dx.doi.org/10.1109/RADECS.2008.5782729

J. R. Azambuja, F. Sousa, L. Rosa, and F.L. Kastensmidt. 2009. Evaluating large grain TMR and selective partial reconfiguration for soft error mitigation in SRAM-based FPGAs. In *15th IEEE International On-Line Testing Symposium (IOLTS)*. 101–106. DOI:http://dx.doi.org/10.1109/IOLTS.2009.5195990

J. L. Barth, K. A. LaBel, and C. Poivey. 2004. Radiation assurance for the space environment. In *International Conference on Integrated Circuit Design and Technology (ICICDT)*. 323–333. DOI:http://dx.doi.org/10.1109/ICICDT.2004.1309976

J. Becker, M. Hubner, G. Hettich, R. Constapel, J. Eisenmann, and J. Luka. 2007. Dynamic and Partial FPGA Exploitation. *Proc. IEEE* 95, 2 (2007), 438–452. DOI:http://dx.doi.org/10.1109/JPROC.2006.888404

M. Berg, C. Poivey, D. Petrick, D. Espinosa, A. Lesea, K.A. LaBel, M. Friendlich, H. Kim, and A. Phan. 2008. Effectiveness of Internal Versus External SEU Scrubbing Mitigation Strategies in a Xilinx FPGA: Design, Test, and Analysis. *IEEE Transactions on Nuclear Science* 55, 4 (2008), 2259–2266. DOI:http://dx.doi.org/10.1109/TNS.2008.2001422

C. Bolchini, A. Miele, and C. Sandionigi. 2011. A Novel Design Methodology for Implementing Reliability-Aware Systems on SRAM-Based FPGAs. *IEEE Trans. Comput.* 60, 12 (2011), 1744–1758. DOI:http://dx.doi.org/10.1109/TC.2010.281

B. Bridgford, C. Carmichael, and C. W. Tseng. 2008. *Single-Event Upset Mitigation Selection Guide*. Application Note XAPP987. Xilinx.

Brigham Young University. 2014. BYU EDIF Tools Home Page. (2014). Retrieved June 2014 from http://reliability.ee.byu.edu/edif/

G. Burke and S. Taft. 2004. Fault Tolerant State Machines. In *Military and Aerospace Programmable Logic Devices Workshop (MAPLD)*. Jet Propulsion Laboratory.

C. Carmichael and C. W. Tseng. 2009. *Correcting Single-Event Upsets in Virtex-4 FPGA Configuration Memory*. Application Note XAPP1088. Xilinx.

K. Chapman. 2010. *SEU Strategies for Virtex-5 Devices*. Application Note XAPP864. Xilinx.

G. Cieslewski, A. D. George, and A. Jacobs. 2010. Acceleration of FPGA Fault Injection Through Multi-Bit Testing. In *International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA)*. 218–224.

C. Dierker. 2007. *Fehlertolerante Instrumentenrechner fuer kompakte Kameras auf Raumsonden*. Ph.D. Dissertation. TU Braunschweig, Braunschweig, Germany.

P. E. Dodd, M.R. Shaneyfelt, J.A. Felix, and J.R. Schwank. 2004. Production and propagation of single-event transients in high-speed digital logic ICs. *IEEE Transactions on Nuclear Science* 51, 6 (2004), 3278–3284. DOI:http://dx.doi.org/10.1109/TNS.2004.839172

A. Ebrahim, K. Benkrid, X. Iturbe, and Chuan Hong. 2012. A novel high-performance fault-tolerant ICAP controller. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. 259–263. DOI:http://dx.doi.org/10.1109/AHS.2012.6268660

ECSS. 2008a. *Methods for the calculation of radiation received and its effects, and a policy for design margins*. Standard ECSS-E-ST-10-12C. European Space Agency.

ECSS. 2008b. *Space Environment*. Standard ECSS-E-ST-10-04C. ESA-ESTEC.

ESA/ESCIES. 2010. Radiation Test Facilities. (2010). Retrieved June 2014 from https://escies.org/webdocument/showArticle?id=230&groupid=6

R. B. Gardenyes. 2012. *Trends and patterns in ASIC and FPGA use in space missions and impact in technology roadmaps of the European Space Agency*. Master's thesis. TU Delft, Delft, The Netherlands.

A. Gavros, H.H. Loomis, and A.A. Ross. 2011. Reduced Precision Redundancy in a Radix-4 FFT implementation on a Field Programmable Gate Array. In *IEEE Aerospace Conference*. 1–15. DOI:http://dx.doi.org/10.1109/AERO.2011.5747459

H. Guzman-Miranda, M.A. Aguirre, and J. Tombs. 2008. A non-invasive system for the measurement of the robustness of microprocessor-type architectures against radiation-induced soft errors. In *IEEE Instrumentation and Measurement Technology Conference Proceedings (IMTC)*. 2009–2014. DOI:http://dx.doi.org/10.1109/IMTC.2008.4547378

S. Habinc. 2002. *Suitability of reprogrammable FPGAs in space applications*. Feasibility report. Gaisler Research.

J. Hagemeyer, A. Hilgenstein, D. Jungewelter, D. Cozzi, C. Felicetti, U. Rueckert, S. Korf, M. Koester, F. Margaglia, M. Porrmann, F. Dittmann, M. Ditze, J. Harris, L. Sterpone, and J. Ilstad. 2012. A scalable platform for run-time reconfigurable satellite payload processing. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. 9–16. DOI:http://dx.doi.org/10.1109/AHS.2012.6268642

J. Heiner, N. Collins, and M. Wirthlin. 2008. Fault Tolerant ICAP Controller for High-Reliable Internal Scrubbing. In *IEEE Aerospace Conference*. 1–10. DOI:http://dx.doi.org/10.1109/AERO.2008.4526471

J. Heiner, B. Sellers, M. Wirthlin, and J. Kalb. 2009. FPGA partial reconfiguration via configuration scrubbing. In *International Conference on Field Programmable Logic and Applications (FPL)*. 99–104. DOI:http://dx.doi.org/10.1109/FPL.2009.5272543

I. Herrera-Alzu and M. Lopez-Vallejo. 2013. Design Techniques for Xilinx Virtex FPGA Configuration Memory Scrubbers. *IEEE Transactions on Nuclear Science* 60, 1 (2013), 376–385. DOI:http://dx.doi.org/10.1109/TNS.2012.2231881

A. Holmes-Siedle and L. Adams. 1993. *Handbook of Radiation Effects*. Oxford University Press.

C. Hong, K. Benkrid, X. Iturbe, A.T. Erdogan, and T. Arslan. 2011. An FPGA task allocator with preliminary First-Fit 2D packing algorithms. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. 264–270. DOI:http://dx.doi.org/10.1109/AHS.2011.5963946

K.-H. Huang and J.A. Abraham. 1984. Algorithm-Based Fault Tolerance for Matrix Operations. *IEEE Trans. Comput.* C-33, 6 (1984), 518–528. DOI:http://dx.doi.org/10.1109/TC.1984.1676475

X. Iturbe, M. Azkarate, I. Martinez, J. Perez, and A. Astarloa. 2009. A novel SEU, MBU and SHE handling strategy for Xilinx Virtex-4 FPGAs. In *International Conference on Field Programmable Logic and Applications (FPL)*. 569–573. DOI:http://dx.doi.org/10.1109/FPL.2009.5272410

X. Iturbe, K. Benkrid, T. Arslan, I. Martinez, and M. Azkarate. 2010. ATB: Area-Time response Balancing algorithm for scheduling real-time hardware tasks. In *International Conference on Field-Programmable Technology (FPT)*. 224–232. DOI:http://dx.doi.org/10.1109/FPT.2010.5681494

X. Iturbe, K. Benkrid, T. Arslan, R. Torrego, and I. Martinez. 2011a. Methods and Mechanisms for Hardware Multitasking: Executing and Synchronizing Fully Relocatable Hardware Tasks in Xilinx FPGAs. In *International Conference on Field Programmable Logic and Applications (FPL)*. 295–300. DOI:http://dx.doi.org/10.1109/FPL.2011.60

X. Iturbe, K. Benkrid, A. Ebrahim, Chuan Hong, T. Arslan, and I. Martinez. 2011b. Snake: An Efficient Strategy for the Reuse of Circuitry and Partial Computation Results in High-Performance Reconfigurable Computing. In *International Conference on Reconfigurable Computing and FPGAs (ReConFig)*. 182–189. DOI:http://dx.doi.org/10.1109/ReConFig.2011.82

X. Iturbe, K. Benkrid, A.T. Erdogan, T. Arslan, M. Azkarate, I. Martinez, and A. Perez. 2010. R3TOS: A reliable reconfigurable real-time operating system. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. 99–104. DOI:http://dx.doi.org/10.1109/AHS.2010.5546274

A. Jacobs, G. Cieslewski, and A.D. George. 2012a. Overhead and reliability analysis of algorithm-based fault tolerance in FPGA systems. In *22nd International Conference on Field Programmable Logic and Applications (FPL)*. 300–306. DOI:http://dx.doi.org/10.1109/FPL.2012.6339222

A. Jacobs, G. Cieslewski, A.D. George, A. Gordon-Ross, and H. Lam. 2012b. Reconfigurable Fault Tolerance: A Comprehensive Framework for Reliable and Adaptive FPGA-Based Space Computing. *ACM Trans. Reconfigurable Technol. Syst.* 5, 4 (2012), 21:1–21:30. DOI:http://dx.doi.org/10.1145/2392616.2392619

J. Johnson, W. Howes, M. Wirthlin, D.L. McMurtrey, M. Caffrey, P. Graham, and K. Morgan. 2008. Using Duplication with Compare for On-line Error Detection in FPGA-based Designs. In *IEEE Aerospace Conference*. 1–11. DOI:http://dx.doi.org/10.1109/AERO.2008.4526470

L. Jones. 2007. *Single Event Upset (SEU) Detection and Correction using Virtex 4 Devices*. Application Note XAPP714. Xilinx.

F. L. Kastensmidt, L. Sterpone, L. Carro, and M.S. Reorda. 2005. On the optimal design of triple modular redundancy logic for SRAM-based FPGAs. In *Design, Automation and Test in Europe (DATE)*, Vol. 2. 1290–1295. DOI:http://dx.doi.org/10.1109/DATE.2005.229

J. Kastil, M. Straka, L. Miculka, and Z. Kotasek. 2012. Dependability Analysis of Fault Tolerant Systems Based on Partial Dynamic Reconfiguration Implemented into FPGA. In *15th Euromicro Conference on Digital System Design (DSD)*. 250–257. DOI:http://dx.doi.org/10.1109/DSD.2012.40

M. Koester, W. Luk, J. Hagemeyer, M. Porrmann, and U. Ruckert. 2011. Design Optimizations for Tiled Partially Reconfigurable Systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 19, 6 (2011), 1048–1061. DOI:http://dx.doi.org/10.1109/TVLSI.2010.2044902

M. Lanuzza, P. Zicari, F. Frustaci, S. Perri, and P. Corsonello. 2010. Exploiting Self-Reconfiguration Capability to Improve SRAM-based FPGA Robustness in Space and Avionics Applications. *ACM Trans. Reconfigurable Technol. Syst.* 4, 1, Article 8 (2010), 22 pages. DOI:http://dx.doi.org/10.1145/1857927.1857935

C. Lavin, M. Padilla, J. Lamprecht, P. Lundrigan, B. Nelson, and B. Hutchings. 2011. RapidSmith: Do-It-Yourself CAD Tools for Xilinx FPGAs. In *21th International Workshop on Field-Programmable Logic and Applications (FPL)*.

D. McMurtrey, K. S. Morgan, B. Pratt, and M. J. Wirthlin. 2008. Estimating TMR Reliability on FPGAs Using Markov Models. contentdm.lib.byu.edu/cdm/ref/collection/IR/id/612

Mentor Graphics. 2014. Precision Hi-Rel Technology Overview. (2014). Retrieved June 2014 from http://www.mentor.com/products/fpga/

G.C. Messenger and M.S. Ash. 1992. *The Effects of Radiation on Electronic Systems* (2nd edition ed.). Van Nostrand Reinhold, New York.

H. Michel, A. Belger, F. Bubenhagen, B. Fiethe, H. Michalik, W. Sullivan, A. Wishart, and J. Ilstad. 2012. The SoCWire protocol (SoCP): A flexible and minimal protocol for a Network-on-Chip. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. 1–8. DOI:http://dx.doi.org/10.1109/AHS.2012.6268631

H. Michel, F. Bubenhagen, B. Fiethe, H. Michalik, B. Osterloh, W. Sullivan, A. Wishart, J. Ilstad, and S.A. Habinc. 2011. AMBA to SoCWire network on Chip bridge as a backbone for a Dynamic Reconfigurable Processing unit. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. 227–233. DOI:http://dx.doi.org/10.1109/AHS.2011.5963941

Microsemi. 2012. *Radiation-Tolerant ProASIC3 Low Power Spaceflight Flash FPGAs with Flash*Freeze Technology*. Datasheet Rev. 5. Microsemi.

G. Miller, C. Carmichael, and G. Swift. 2008. *Single-Event Upset Mitigation for Xilinx FPGA Block Memories*. Application Note XAPP962. Xilinx.

J. M. Mogollon, H. Guzman-Miranda, J. Napoles, J. Barrientos, and M.A. Aguirre. 2011. FTUNSHADES2: A novel platform for early evaluation of robustness against SEE. In *12th Euro-*

pean Conference on Radiation and Its Effects on Components and Systems (RADECS). 169–174. DOI:http://dx.doi.org/10.1109/RADECS.2011.6131392

K. Morgan, M. Caffrey, P. Graham, E. Johnson, B. Pratt, and M. Wirthlin. 2005. SEU-induced persistent error propagation in FPGAs. *IEEE Transactions on Nuclear Science* 52, 6 (2005), 2438–2445. DOI:http://dx.doi.org/10.1109/TNS.2005.860674

K. Morgan, D.L. McMurtrey, B.H. Pratt, and M.J. Wirthlin. 2007. A Comparison of TMR With Alternative Fault-Tolerant Design Techniques for FPGAs. *IEEE Transactions on Nuclear Science* 54, 6 (2007), 2065–2072. DOI:http://dx.doi.org/10.1109/TNS.2007.910871

J. Napoles, H. Guzman-Miranda, M. Aguirre, J.N. Tombs, J.M. Mogollon, R. Palomo, and A.P. Vega-Leal. 2008. A Complete Emulation System for Single Event Effects Analysis. In *4th Southern Conference on Programmable Logic (SPL)*. 213–216. DOI:http://dx.doi.org/10.1109/SPL.2008.4547760

NASA. 2012. *Fault Management Handbook. Draft 2.* Handbook NASA-HDBK-1002. National Aeronautics and Space Administration.

G.L. Nazar, L.P. Santos, and L. Carro. 2013. Accelerated FPGA repair through shifted scrubbing. In *23rd International Conference on Field Programmable Logic and Applications (FPL)*. 1–6. DOI:http://dx.doi.org/10.1109/FPL.2013.6645533

C.D. Norton, T.A. Werne, P.J. Pingree, and S. Geier. 2009. An evaluation of the Xilinx Virtex-4 FPGA for on-board processing in an advanced imaging system. In *IEEE Aerospace conference*. 1–9. DOI:http://dx.doi.org/10.1109/AERO.2009.4839460

B. Osterloh, H. Michalik, B. Fiethe, and F. Bubenhagen. 2007. Enhancements of reconfigurable System-on-Chip Data Processing Units for Space Application. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. 258–262. DOI:http://dx.doi.org/10.1109/AHS.2007.47

B. Osterloh, H. Michalik, B. Fiethe, and F. Bubenhagen. 2010. Architecture verification of the SoCWire NoC approach for safe dynamic partial reconfiguration in space applications. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. 1–8. DOI:http://dx.doi.org/10.1109/AHS.2010.5546220

B. Osterloh, H. Michalik, B. Fiethe, and K. Kotarowski. 2008. SoCWire: A Network-on-Chip Approach for Reconfigurable System-on-Chip Designs in Space Applications. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. 51–56. DOI:http://dx.doi.org/10.1109/AHS.2008.43

B. Osterloh, H. Michalik, S.A. Habinc, and B. Fiethe. 2009. Dynamic Partial Reconfiguration in Space Applications. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. 336–343. DOI:http://dx.doi.org/10.1109/AHS.2009.13

R. Padovani. 2005. Reconfigurable FPGAs in Space - Present and Future. Presentation at MAPLD Conference. (2005).

K. Paulsson, M. Hubner, and J. Becker. 2006. Strategies to On-Line Failure Recovery in Self-Adaptive Systems based on Dynamic and Partial Reconfiguration. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. 288–291. DOI:http://dx.doi.org/10.1109/AHS.2006.67

P. J. Pingree. 2010. Advancing NASA's on-board processing capabilities with reconfigurable FPGA technologies: Opportunities and implications. In *IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW)*. DOI:http://dx.doi.org/10.1109/IPDPSW.2010.5470824

B. Pratt, M. Caffrey, J.F. Carroll, P. Graham, K. Morgan, and M. Wirthlin. 2008. Fine-Grain SEU Mitigation for FPGAs Using Partial TMR. *IEEE Transactions on Nuclear Science* 55, 4 (2008), 2274–2280. DOI:http://dx.doi.org/10.1109/TNS.2008.2000852

B. Pratt, M. Caffrey, P. Graham, K. Morgan, and M. Wirthlin. 2006. Improving FPGA Design Robustness with Partial TMR. In *44th IEEE International Reliability Physics Symposium (IRPS)*. 226–232. DOI:http://dx.doi.org/10.1109/RELPHY.2006.251221

B. Pratt, M. Fuller, M. Rice, and M. Wirthlin. 2013. Reduced-Precision Redundancy for Reliable FPGA Communications Systems in High-Radiation Environments. *IEEE Trans. Aerospace Electron. Systems* 49, 1 (2013), 369–380. DOI:http://dx.doi.org/10.1109/TAES.2013.6404109

B. Pratt, M. Fuller, and M. Wirthlin. 2011. Reduced-Precision Redundancy on FPGAs. *International Journal of Reconfigurable Computing* 2011 (2011).

H. Quinn, G.R. Allen, G.M. Swift, Chen Wei Tseng, P.S. Graham, K.S. Morgan, and P. Ostler. 2009a. SEU-Susceptibility of Logical Constants in Xilinx FPGA Designs. *IEEE Transactions on Nuclear Science* 56, 6 (2009), 3527–3533. DOI:http://dx.doi.org/10.1109/TNS.2009.2033925

H. Quinn, P. Graham, J. Krone, M. Caffrey, and S. Rezgui. 2005. Radiation-induced multi-bit upsets in SRAM-based FPGAs. *IEEE Transactions on Nuclear Science* 52, 6 (2005), 2455–2461. DOI:http://dx.doi.org/10.1109/TNS.2005.860742

H. Quinn, P. Graham, K. Morgan, Z. Baker, M. Caffrey, D. Smith, and R. Bell. 2012. On-Orbit Results for the Xilinx Virtex-4 FPGA. In *IEEE Radiation Effects Data Workshop (REDW)*. 1–8. DOI:http://dx.doi.org/10.1109/REDW.2012.6353715

H. Quinn, P.S. Graham, M.J. Wirthlin, B. Pratt, K.S. Morgan, M.P. Caffrey, and J.B. Krone. 2009b. A Test Methodology for Determining Space Readiness of Xilinx SRAM-Based FPGA Devices and Designs. *IEEE Transactions on Instrumentation and Measurement* 58, 10 (2009), 3380–3395. DOI:http://dx.doi.org/10.1109/TIM.2009.2025469

H. Quinn, K. Morgan, P. Graham, J. Krone, and M. Caffrey. 2007a. A review of Xilinx FPGA architectural reliability concerns from Virtex to Virtex-5. In *9th European Conference on Radiation and Its Effects on Components and Systems (RADECS)*. 1–8. DOI:http://dx.doi.org/10.1109/RADECS.2007.5205533

H. Quinn, K. Morgan, P. Graham, J. Krone, and M. Caffrey. 2007b. Static Proton and Heavy Ion Testing of the Xilinx Virtex-5 Device. In *IEEE Radiation Effects Data Workshop (REDW)*. 177–184. DOI:http://dx.doi.org/10.1109/REDW.2007.4342561

H. Quinn, K. Morgan, P. Graham, J. Krone, M. Caffrey, and K. Lundgreen. 2007c. Domain Crossing Errors: Limitations on Single Device Triple-Modular Redundancy Circuits in Xilinx FPGAs. *IEEE Transactions on Nuclear Science* 54, 6 (2007), 2037–2043. DOI:http://dx.doi.org/10.1109/TNS.2007.910870

P. M. B. Rao, M. Ebrahimi, R. Seyyedi, and M. B. Tahoori. 2014. Protecting SRAM-based FPGAs Against Multiple Bit Upsets Using Erasure Codes. In *51st Annual Design Automation Conference (DAC)*. ACM, New York, NY, USA, Article 212, 6 pages. DOI:http://dx.doi.org/10.1145/2593069.2593191

N. Rollins, M. Fuller, and M.J. Wirthlin. 2010. A Comparison of fault-tolerant memories in SRAM-based FPGAs. In *IEEE Aerospace Conference*. 1–12. DOI:http://dx.doi.org/10.1109/AERO.2010.5446661

A. Sari, D. Agiakatsikas, and M. Psarakis. 2014. A Soft Error Vulnerability Analysis Framework for Xilinx FPGAs. In *ACM/SIGDA International Symposium on Field-programmable Gate Arrays (FPGA)*. ACM, New York, NY, USA, 237–240. DOI:http://dx.doi.org/10.1145/2554688.2554767

A. Sari and M. Psarakis. 2011. Scrubbing-based SEU mitigation approach for Systems-on-Programmable-Chips. In *International Conference on Field-Programmable Technology (FPT)*. 1–8. DOI:http://dx.doi.org/10.1109/FPT.2011.6132703

F. Siegle, T. Vladimirova, O. Emam, and J. Ilstad. 2013. Adaptive FDIR framework for payload data processing systems using reconfigurable FPGAs. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. 15–22. DOI:http://dx.doi.org/10.1109/AHS.2013.6604221

I. Skliarova. 2005. Self-correction of FPGA-Based control units. In *2nd International Conference on Embedded Software and Systems (ICESS)*. Springer-Verlag, Berlin, Heidelberg, 310–319.

J.D. Snodgrass. 2006. *Low-Power Fault Tolerance for Spacecraft FPGA-based Numerical Computing.* Ph.D. Dissertation. Naval Postgraduate School, Monterey, CA, U.S.A.

M. Sonza Reorda, L. Sterpone, and M. Violante. 2005a. Efficient estimation of SEU effects in SRAM-based FPGAs. In *11th IEEE International On-Line Testing Symposium (IOLTS)*. 54–59. DOI:http://dx.doi.org/10.1109/IOLTS.2005.26

M. Sonza Reorda, L. Sterpone, and M. Violante. 2005b. Multiple errors produced by single upsets in FPGA configuration memory: a possible solution. In *European Test Symposium (ETS)*. 136–141. DOI:http://dx.doi.org/10.1109/ETS.2005.29

A. Sreeramareddy, J.G. Josiah, A. Akoglu, and A. Stoica. 2008. SCARS: Scalable Self-Configurable Architecture for Reusable Space Systems. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. 204–210. DOI:http://dx.doi.org/10.1109/AHS.2008.77

L. Sterpone and N. Battezzati. 2008. A Novel Design Flow for the Performance Optimization of Fault Tolerant Circuits on SRAM-based FPGA's. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. 157–163. DOI:http://dx.doi.org/10.1109/AHS.2008.59

L. Sterpone and N. Battezzati. 2010. A new placement algorithm for the mitigation of multiple cell upsets in SRAM-based FPGAs. In *Design, Automation Test in Europe (DATE)*. 1231–1236.

L. Sterpone, F. Margaglia, M. Koester, J. Hagemeyer, and M. Porrmann. 2011. Analysis of SEU effects in partially reconfigurable SoPCs. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. 129 –136. DOI:http://dx.doi.org/10.1109/AHS.2011.5963926

L. Sterpone, M.S. Reorda, and M. Violante. 2005. RoRA: a reliability-oriented place and route algorithm for SRAM-based FPGAs. In *Research in Microelectronics and Electronics, 2005 PhD*, Vol. 1. 173–176. DOI:http://dx.doi.org/10.1109/RME.2005.1543031

L. Sterpone and M. Violante. 2006. A new reliability-oriented place and route algorithm for SRAM-based FPGAs. *IEEE Trans. Comput.* 55, 6 (2006), 732 –744. DOI:http://dx.doi.org/10.1109/TC.2006.82

L. Sterpone and M. Violante. 2007. Static and Dynamic Analysis of SEU Effects in SRAM-Based FPGAs. In *12th IEEE European Test Symposium (ETS)*. 159–164. DOI:http://dx.doi.org/10.1109/ETS.2007.37

L. Sterpone and M. Violante. 2008. A New Algorithm for the Analysis of the MCUs Sensitiveness of TMR Architectures in SRAM-Based FPGAs. *IEEE Transactions on Nuclear Science* 55, 4 (2008), 2019–2027. DOI:http://dx.doi.org/10.1109/TNS.2008.2001858

M. Straka, J. Kastil, and Z. Kotasek. 2010a. Fault Tolerant Structure for SRAM-Based FPGA via Partial Dynamic Reconfiguration. In *13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD)*. 365–372. DOI: http://dx.doi.org/10.1109/DSD.2010.12

M. Straka, J. Kastil, and Z. Kotasek. 2010b. Generic partial dynamic reconfiguration controller for fault tolerant designs based on FPGA. In *The Nordic Microelectronics Event (NORCHIP)*. 1–4. DOI: http://dx.doi.org/10.1109/NORCHIP.2010.5669477

M. Straka, L. Miculka, J. Kastil, and Z. Kotasek. 2012. Test platform for fault tolerant systems design properties verification. In *IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*. 336–341. DOI: http://dx.doi.org/10.1109/DDECS.2012.6219084

M. Straka, J. Tobola, and Z. Kotasek. 2007. Checker Design for On-line Testing of Xilinx FPGA Communication Protocols. In *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT)*. 152–160. DOI: http://dx.doi.org/10.1109/DFT.2007.21

M.A. Sullivan. 2008. *Reduced Precision Redundancy Applied to Arithmetic Operations in Field Programmable Gate Arrays for Satellite Control and Sensor Systems*. Master's thesis. Naval Postgraduate School, Monterey, California.

M. A. Sullivan, H.H. Loomis, and A.A. Ross. 2009. Employment of Reduced Precision Redundancy for Fault Tolerant FPGA Applications. In *17th IEEE Symposium on Field Programmable Custom Computing Machines (FCCM)*. 283–286. DOI: http://dx.doi.org/10.1109/FCCM.2009.53

G. Swift and G. Allen. 2012. *Virtex-5QV Static SEU Characterization Summary*. Technical Report. NASA Jet Propulsion Laboratory, Xilinx.

G. Swift, G.R. Allen, Chen Wei Tseng, C. Carmichael, G. Miller, and J.S. George. 2008. Static Upset Characteristics of the 90nm Virtex-4QV FPGAs. In *IEEE Radiation Effects Data Workshop (REDW)*. 98–105. DOI: http://dx.doi.org/10.1109/REDW.2008.25

Synopsis. 2012. Synplify Premier. Fast, Reliable FPGA Implementation and Debug. (2012). Retrieved June 2014 from http://www.synopsys.com/Tools/Implementation/FPGAImplementation/CapsuleModule/syn_prem_ds.pdf

TRAD. 2014. OMERE Software. (2014). Retrieved June 2014 from http://www.trad.fr/OMERE-Software.html

Xilinx. 2009. *Virtex-4 FPGA Configuration Guide*. User Guide UG071. Xilinx.

Xilinx. 2010. *Space-Grade Virtex-4QV Family Overview*. Datasheet DS653. Xilinx.

Xilinx. 2012a. *Radiation-Hardened, Space-Grade Virtex-5QV Family Overview*. Datasheet DS192. Xilinx.

Xilinx. 2012b. *Soft Error Mitigation Using Prioritized Essential Bits*. Application Note XAPP538. Xilinx.

Xilinx. 2012c. *Virtex-5 FPGA Configuration Guide*. User Guide UG191. Xilinx.

Xilinx. 2014. TMRTool. (2014). Retrieved June 2014 from http://www.xilinx.com/ise/optional_prod/tmrtool.htm

S. Yousuf, A. Jacobs, and A. Gordon-Ross. 2011. Partially reconfigurable system-on-chips for adaptive fault tolerance. In *International Conference on Field Programmable Logic and Applications (FPL)*. 1–8. DOI: http://dx.doi.org/10.1109/FPT.2011.6132708

[redacted]