Assessing Scrubbing Techniques for Xilinx SRAM-based FPGAs in Space Applications

Fredrik Brosser*, Emil Milh, Vilhelm Geijer[†], and Per Larsson-Edefors
Dept. of Computer Science and Engineering
Chalmers University of Technology, Gothenburg, Sweden
[†]RUAG Space, Gothenburg, Sweden
Email: fredrik@brosser.se, emil.milh@gmail.com

***Abstract*—SRAM-based FPGAs are becoming increasingly attractive for use in space applications due to their reconfigurability and signal processing capabilities, as well as their increasing speed and capacity. Traditional SRAM-based FPGAs, however, are highly sensitive to the ionizing radiation environment in space, making them prone to radiation-induced memory upsets. In this paper, we evaluate and compare scrubbing techniques for Xilinx SRAM-based FPGAs with respect to radiation-induced single event upsets. A test framework using an exchangeable payload is developed for this purpose and run on a Xilinx Virtex-5 FPGA. We show that recent SRAM-based FPGAs can constitute a cost-efficient alternative to radiation-hardened or antifuse FPGAs for non-critical space application such as satellite instruments.**

I. Introduction

The reprogrammable and reconfigurable capabilities of SRAM-based FPGAs, along with their suitability for signal processing applications and their increasing capacity, have made them increasingly attractive as alternatives to ASICs in space applications. Compared to radiation-hardened processors conventionally used, FPGAs constitute a highly versatile and high-performing alternative, especially considering their reprogrammability and great capability for parallelisation.

Operating in a space environment raises a number of issues that affect the design of a system, e.g., radiation can negatively impact the lifespan, performance and reliability of a digital system [1], [2]. As more advanced process nodes with smaller geometries are introduced, designing for fault tolerance is becoming an increasingly important aspect.

Traditionally, the space industry has avoided commercial off-the-shelf SRAM-based FPGAs due to the susceptibility of their configuration memory to radiation-induced bit upsets when subjected to the ionizing radiation environment in space. However, it is no longer feasible to disregard SRAM-based FPGA technology, especially when considering the high non-recurring engineering costs involved in developing custom ASIC designs, and the low production volumes typical of space applications. While alternatives such as flash- or antifuse-based FPGAs exist, these are less flexible and generally much smaller than their SRAM-based counterparts. There are also radiation-hardened versions of SRAM-based FPGAs available on the market, which have been proven in space missions. These, however, are generally far behind in performance and capacity compared to standard off-the-shelf FPGAs, as well as being overly expensive for many projects.

While it is true that SRAM-based FPGAs are sensitive to radiation-induced upsets in both their configuration and user memory [1], it is important to acknowledge that satellite FPGA applications range from instruments with relaxed timing and availability requirements, all the way up to mission-critical systems, requiring different levels of fault tolerance. The aim of this work is to investigate the feasibility of using Xilinx's SRAM-based FPGAs with triple modular redundancy (TMR) and scrubbing techniques for space applications.

This work was carried out in its entirety at and supported by RUAG Space Sweden. RUAG Space AB, Solhusgatan 11, SE-412 76 Gothenburg, Sweden.
* Author is now with ARM Ltd, Cambridge UK.

II. Related Work

Significant work on the subject has been done by the NASA Radiation Effects and Analysis Group and by Gaisler Research. As far as available soft-error mitigation techniques, prior studies have resulted in short [3] and more extensive [4] overviews of techniques that are possible to deploy. Sari and Psarakis presented a configuration-memory sensitivity analysis for a typical FPGA application with a soft-core processor [5]. Configuration-memory scrubbing is a well-explored area, and various scrubbing implementations have been introduced [6], [7]. Berg et al. compared external scrubber implementations with scrubber implementations internally on the FPGA [8]. Several previous papers have investigated TMR-implementation techniques for FPGAs, such as partial TMR or fine-grained TMR [9], [10], [11]. In their 2014 paper, Glein et al. present a method of using BRAM as radiation-level indicators [12]. Xilinx provide SEU-mitigation guideline documents [13], [14] as well as soft-error mitigation IP cores for Virtex series FPGAs [15], [16]. Xilinx's IDF and IVT tools can be applied to partition FPGA designs into logically isolated blocks and to verify designs for fault tolerance.

III. Method

A. Test Platform

In order to test and evaluate mitigation techniques, a test platform has been implemented. In designing the test platform, a few basic functions were identified. A platform needs to provide 1) reliable fault injection, 2) fault detection and logging, and 3) an exchangeable payload. Both the test framework and the payload are implemented on the same FPGA. By keeping the payload logically separate from the test framework, it can be easily exchanged. Furthermore, testing was facilitated by moving complexity from the FPGA to software on a host PC wherever possible.

The hardware platform is implemented on a Virtex-5 XC5VLX50-ff676 (-1) evaluation board. The Virtex-5 platform is chosen as it represents a realistic choice of FPGA for space applications, and there is a radiation-hardened version of the Virtex-5 on the market. Architecturally the FPGA test platform consists of three main parts: A bus structure, a test framework, and the payload itself (the device-under-test). A framework block diagram is given in Figure 1. The bus structure is used to provide DMA and configuration-register access. The test-framework block is made up of the SEU Controller, which is managing fault injection, the Fault Monitor, which is providing result checking and error logging, and the Reconfiguration Manager, which manages configuration-memory scrubbing in the cases where it is used.

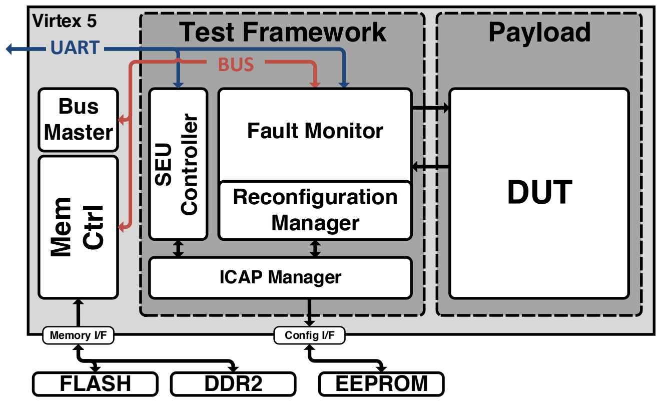

Fig. 1. Block diagram of FPGA test platform

The SEU Controller [15] is a soft-error-mitigation core provided by Xilinx. It implements SECDED functionality and fault injection capability through the use of the Frame ECC and ICAP primitives. Faults are injected by single bit flips in memory frames, accessed through ICAP. An UART interface enables communication with a PC through a set of commands and responses. The SEU Controller is able to detect and correct (single-bit) configuration-memory errors. It can also detect, but not correct, multi-bit errors.

The Fault Monitor provides an interface to the payload. An UART interface (separate from the SEU Controller) provides communication with the host PC. The set of commands interpreted by the Fault Monitor is separate from the set of commands for the SEU Controller. Application-specific test vectors are kept by the Fault Monitor and used as input to the payload. The result vectors produced by the payload application are compared to the expected output vectors. The Fault Monitor then stores a status vector to memory, indicating any errors detected in the payload output. Status vectors can be stored to off-chip flash memory or internally to BRAM, and retrieved via the UART channel. Minimising connections between the Fault Monitor and the payload ensures a good level of logical separation.

The Reconfiguration Manager manages configuration-memory scrubbing through the ICAP interface and interfaces to an off-chip platform Flash or EEPROM, which stores the reference bitstream. By placing the Reconfiguration Manager outside or inside the payload block of the test platform, it is possible to model an external or internal scrubber, respectively. As ICAP is used by both the SEU Controller and the Reconfiguration Manager, an ICAP controller is used to provide arbitration. ICAP is clocked using the system's master clock.

The payload is placed in a separate p-block, in a known frame interval in the configuration memory, using Xilinx PlanAhead. We do not make use of the Xilinx Essential Bits classification scheme. This allows logic separation between payload and framework, as well as a known range of frames in which to inject errors without risking upsetting configuration bits in the framework. PlanAhead produces a logic allocation (.ll) file containing information about the frame address mapping. We use this mapping when injecting faults in the payload.

B. Host PC Software

A Tcl application run on a host PC manages all test sequences through commands sent over UART to the test framework. To realistically simulate randomly occurring SEUs, a MATLAB program on the host PC is used in conjunction with an orbit profile to generate a sequence of SEU injections, based on the CREME96 model.

Two types of tests are used; isolation testing and continuous testing. In isolation testing, single-bit upsets are tested as isolated events. A single fault is randomly injected in the payload's configuration. The Fault Monitor then runs and checks a large number of test vectors, offering a good coverage of the payload application and minimising the risk of a fault going undetected. An error is logged, indicating what instance failed, when one or more of the test vectors cause the payload to produce an incorrect result. The fault is corrected before the next one is injected. This type of testing is useful for evaluating the masking capabilities of an SEU mitigation technique such as TMR. Each test run comprises 4,000 injected faults.

In continuous tests, errors are not corrected between each fault insertion. This allows the analysis of dynamic processes, such as error build-up, and the effect of scrubbing. From continuous testing, numbers for availability, Mean Time To Failure (MTTF), and performance over time can be obtained. A MATLAB script on the host PC is used to analyse the PC-to-FPGA communication logs to extract these statistics.

C. TMR Variants

Four TMR variants have been implemented, based on a thoroughly verified 128-bit AES-encryption block. The AES application was chosen because of the extensive documentation available on AES and because errors anywhere in the encryption chain are likely to propagate to the output. A large number of test vectors are available for the AES standard. We make use of the NIST AES Known Answer Test (KAT) Vectors [17]. The AES application used here does not make use of BRAM or DSP blocks. The different tested TMR variants are listed below.

1) **Reference**: A baseline payload without TMR.

2) **Single Voter TMR**: Module-level TMR with a single voter.
3) **Triple Voter TMR**: Module-level TMR where the voter is tripled.
4) **Synplify TMR**: TMR is applied using Synplify Premier.

D. Scrubbing Methods

1) **Blind Scrubbing**: This corresponds to a periodical, full-reconfiguration scrubbing which is based on an internal counter in the FPGA.
2) **CRC-based Scrubbing**: This type of scrubber performs a full reconfiguration upon detecting an error in the configuration-memory CRC. CRC error checking is done as a background task through the SEU Controller.
3) **Frame ECC-Based Scrubbing**: Using Frame-level ECC to detect errors, allowing single frames to be scrubbed using dynamic partial reconfiguration.
4) **SECDED Scrubbing**: This combination scrubbing makes use of the SEU Controller for correcting single bits, using SEU Controller's SECDED bit-toggling functionality. If a multi-bit error occurs, or if the SEU Controller is not able to correct an error, a full reconfiguration is triggered.

IV. Results

Table IV presents a summary of the isolation-testing results, giving an indication of how well the different TMR variants can mask errors. Each test comprises 4,000 injected faults, and a breakdown of the observed error types is made. A single error is defined as an error affecting the output of a single TMR branch. A bridge error is a single error that affects two separate TMR branches. We note that the single voter design has more failures, while the triple-voter design and Synplify-applied TMR perform roughly equally.

TABLE I
Isolation testing results for TMR

	Errors	**Single**	**Bridge**	**Voter**	**Failures (%)**
Reference	1084	1084	-	-	1084 (27.1)
Single Voter	697	663	14	20	34 (0.85)
Triple Voter	714	674	8	32	14 (0.35)
Synplify TMR	10[1]	10	-	-	10 (0.25)

[1]Synplify TMR has lower observability

Considering the resources used for each of the TMR variants, Table IV shows the relative resource usage on an XC5VLX50 FPGA. The designs are synthesised using Synplify Premier (H-2013.03).

TABLE II
FPGA resource usage for TMR variants

	Slices	**Regs**	**LUTs**	**LUT-DFF Pairs**	**Rel.**
Reference	291	408	1143	1143	1
Single Voter	1354	1345	3698	3698	3.24x
Triple Voter	1472	1603	4224	4224	3.87x
Synplify TMR	2275	1224	5436	5436	4.76x

Having a sense of the error-masking capabilities and the resource cost of the TMR variants, we move on to study the dynamic behaviour. For the purpose of evaluating blind scrubbing, the SEU rate (rate of fault-injections into the payload) and the scrubbing rate are varied. Figure 2 shows the resulting availability for systems using a blind-scrubbing approach for each of the TMR variants and the baseline. The SEU and scrubbing rates are varied to show their relation. The SEU period ($1/f_{SEU}$) is given in seconds, and the scrubbing rates in average number of reconfigurations per SEU. The bold line marks the maximum availability for each SEU scrubbing rate.

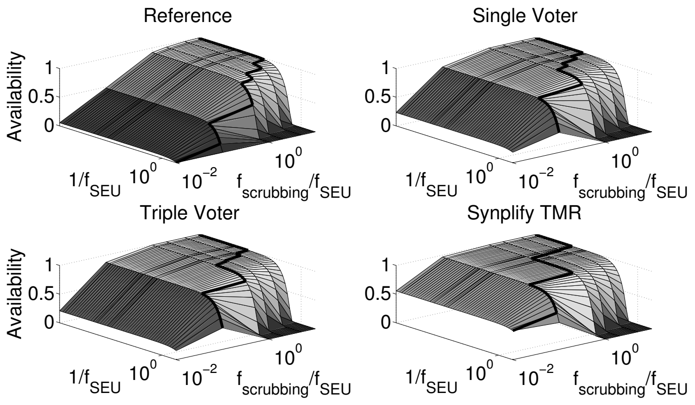

Fig. 2. Availability results for blind scrubbing implementations

It should be noted that a short SEU period combined with a high scrubbing rate results in very low availability. The system will be unavailable while it is reconfiguring. If the scrubbing rate is too high, the system will be busy with reconfiguration most of the time, leading to lower availability. Reconfiguration takes a certain amount of time regardless of the design, on our Virtex-5 approximately 500 ms. Figure 2 shows that scrubbing too infrequently or too often in relation to the expected SEU rate leads to lower availability when using blind scrubbing. The detection-based scrubbing methods tested provide a faster response time to errors compared to blind scrubbing.

Two realistic example scenarios were investigated: *a*) an 800-km, 98°inclination LEO and *b*) a -55°longitude GEO. Since no radiation testing results are available from the manufacturer for the XC5VLX50 FPGA used here, calculations will be made using the device characteristics published for another Virtex-5 series FPGA, the radiation-hardened Virtex-5QV FX130T (XQR5VFX130T) [18]. The test application is the same 128-bit AES application used in the previous tests, but scaled up to fill the entire FPGA. Table IV indicates how many functional D-flip-flops can fit in the FPGA for each TMR variant.

TABLE III
Test application resources

TMR	**Functional DFFs**	**AES Blocks**	**Device Usage**
Reference	28,968	71	99.1%
Single Voter	8,976	22	99.3%
Triple Voter	7,752	19	98.0%
Synplify TMR	6,120	15	99.5%

The purpose of scaling the application to fill as much of the FPGA as possible is that for a fixed number of injected faults, TMR variants with lower resource usage would otherwise experience a higher density of faults.

For space applications, considering MTTF or Failures-In-Time (FIT) is often more relevant than availability. FIT is a measurement of the number of failures in 10^9 hours. Figure 3 shows MTTF for our detection-based scrubbers (CRC, Frame-ECC and SECDED-based). We found that MTTF has a linear dependence on the SEU period. When comparing Frame-ECC-triggered scrubbing with CRC-triggered scrubbing, it is apparent that the triple-voter and Synplify TMR variants benefit more from having frame-level reconfigurability or SECDED than does single-voter TMR. The triple-voter and Synplify-TMR variants both have tripled voters, meaning that one of the voters can be down for reconfiguration while the other two still function, when using frame-level reconfiguration or SECDED.

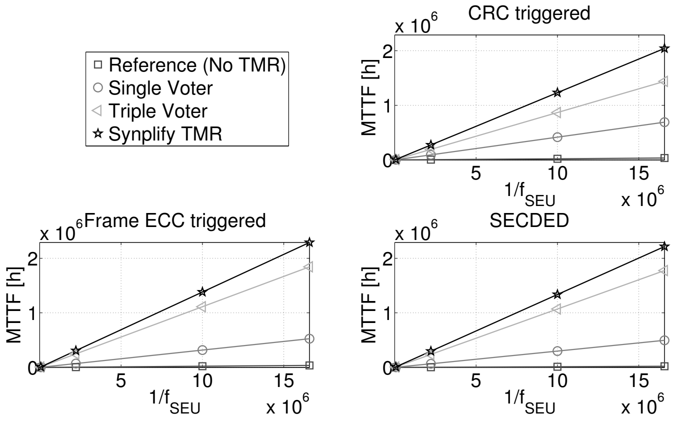

Fig. 3. Mean-time-to-failure for detection-based scrubbers

Calculating the radiation profiles from the orbit parameters and using the characteristics for the XQR5VFX130T, we extrapolate the results to assess the performance for the LEO and GEO examples, as shown in Figure 4. For a realistic estimate, we have included 5% of the available BRAM on the FPGA. This is done to show the impact of BRAM on the error rate. The resulting FIT values for LEO and GEO for the TMR payload are in the order of 10^5. The top performing techniques are triple-voter or Synplify TMR combined with SECDED or Frame-ECC based scrubbing. For LEO, FIT values are $\{1.3356, 1.0701, 1.2857, 1.0351\} * 10^5$, respectively.

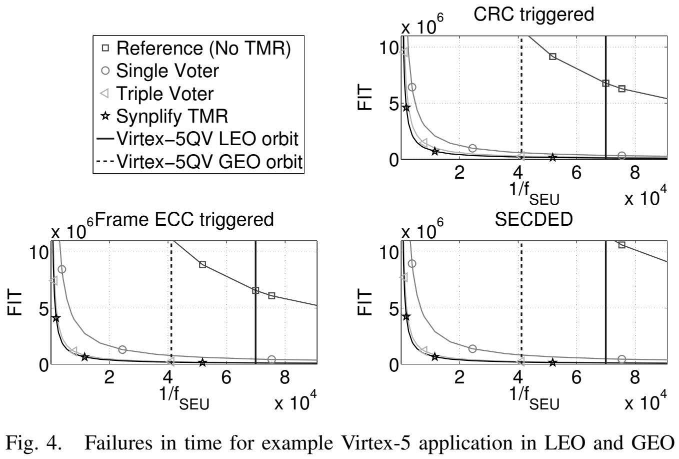

Fig. 4. Failures in time for example Virtex-5 application in LEO and GEO

V. Conclusion

The results presented in this paper suggest that standard, commercial SRAM-based FPGAs from Xilinx can indeed be used in some space applications. For suitable applications, such as non-critical instruments or communications, SRAM-based FPGAs can provide a very cost-to-area-efficient alternative. TMR needs to be combined with scrubbing in order to be efficient, or it will eventually suffer from error build-up. The results presented here apply to the selected test application, i.e., the 128-bit AES block. While the techniques discussed are general, the results will vary depending on the application. Applications with natural downtime windows can benefit from simple periodic scrubbing. Partial-reconfiguration in conjunction with TMR constitutes a very efficient fault-tolerance alternative, but there is significant resource overhead. The SECDED alternative based on the Xilinx SEU Controller represents a resource-efficient choice that can be combined with externally-triggered full reconfiguration. The example LEO and GEO scenarios discussed in this paper show that commercial SRAM-based FPGAs can be used in space applications, but both user data and configuration memory have to be protected.

References

[1] F. Sturesson, S. Mattsson, C. Carmichael, and R. Harboe-Sorensen, "Heavy ion characterization of SEU mitigation methods for the Virtex FPGA," in *6th European Conf. on Radiation and Its Effects on Components and Systems*, Sept. 2001, pp. 285–291.

[2] E. Petersen, *Single Event Effects in Aerospace*. Wiley, 2012.

[3] G.-H. Asadi and M. Tahoori, "Soft error mitigation for SRAM-based FPGAs," in *Proc. IEEE VLSI Test Symposium*, 2005, pp. 207–212.

[4] S. Habinc, "Suitability of reprogrammable FPGAs in space applications," Gaisler Research, Tech. Rep., 09 2002.

[5] A. Sari and M. Psarakis, "Scrubbing-based SEU mitigation approach for Systems-on-Programmable-Chips," in *Int'l Conf. on Field-Programmable Technology (FPT)*, Dec. 2011, pp. 1–8.

[6] J.-Y. Lee, C.-R. Chang, N. Jing, J. Su, S. Wen, R. Wong, and L. He, "Heterogeneous configuration memory scrubbing for soft error mitigation in FPGAs," in *Int'l Conf. on Field-Programmable Technology (FPT)*, Dec. 2012, pp. 23–28.

[7] I. Herrera-Alzu and M. Lopez-Vallejo, "Design Techniques for Xilinx Virtex FPGA Configuration Memory Scrubbers," *IEEE Trans. on Nuclear Science*, vol. 60, no. 1, pp. 376–385, Feb. 2013.

[8] M. Berg, C. Poivey, D. Petrick, D. Espinosa, A. Lesea, K. LaBel, M. Friendlich, H. Kim, and A. Phan, "Effectiveness of Internal Versus External SEU Scrubbing Mitigation Strategies in a Xilinx FPGA: Design, Test, and Analysis," *IEEE Trans. on Nuclear Science*, vol. 55, no. 4, pp. 2259–2266, Aug. 2008.

[9] F. Kastensmidt, L. Sterpone, L. Carro, and M. Reorda, "On the optimal design of triple modular redundancy logic for SRAM-based FPGAs," in *Proc. Design, Automation and Test in Europe*, 2005, pp. 1290–1295 Vol. 2.

[10] B. Pratt, M. Caffrey, J. Carroll, P. Graham, K. Morgan, and M. Wirthlin, "Fine-Grain SEU Mitigation for FPGAs Using Partial TMR," *IEEE Trans. on Nuclear Science*, vol. 55, no. 4, pp. 2274–2280, Aug. 2008.

[11] H. Quinn, K. Morgan, P. Graham, J. Krone, M. Caffrey, and K. Lundgreen, "Domain Crossing Errors: Limitations on Single Device Triple-Modular Redundancy Circuits in Xilinx FPGAs," *IEEE Trans. on Nuclear Science*, vol. 54, no. 6, pp. 2037–2043, Dec. 2007.

[12] R. Glein, B. Schmidt, F. Rittner, J. Teich, and D. Ziener, "A Self-Adaptive SEU Mitigation System for FPGAs with an Internal Block RAM Radiation Particle Sensor," in *IEEE 22nd Annual Int'l Symp. on Field-Programmable Custom Computing Machines (FCCM)*, May 2014, pp. 251–258.

[13] *SEU Strategies for Virtex-5 Devices*, Xilinx Inc., 2010.

[14] *Single-Event Upset Mitigation Selection Guide*, Xilinx Inc., 2008.

[15] *New Generation Virtex-5 SEU Controller*, Xilinx Inc., 2010.

[16] *LogiCORE IP Soft Error Mitigation Controller v4.1*, Xilinx Inc., 2014.

[17] *AES Known Answer Test (KAT) Vectors*, NIST CSRC, accessed: 2014-03-03. [Online]. Available: http://csrc.nist.gov/groups/STM/cavp/

[18] *DS192: Radiation-Hardened, Space-Grade Virtex-5QV Family Overview*, Xilinx Inc., 2012.