Software Techniques for Soft Error Resilience: the ASTRAEUS project

Federico Reghenzani*, Davide Baroffio*, Emilio Corigliano*, William Fornaciari*, Giancarlo S. Gajani*, Paolo Maffezzoni*, Antonino Catanese†, Alessandro Balossino†, Marco Giuliani†

*DEIB – Politecnico di Milano, Milano, Italy. email:{name.surname}@polimi.it
†Argotec, San Mauro Torinese, Torino, Italy. email:{name.surname}@argotec.it

***Abstract*—ASTRAEUS project aims to improve the use of Commercial-Off-The-Shelf (COTS) devices in space telecommunication applications. The goal is to develop specialized radiation mitigation techniques for both hardware and software components with a special focus on the latter. The aim is to demonstrate the feasibility and reliability of these techniques, enabling their future use in telecommunication payload processing units. The SIHFT (Software Implemented Hardware Fault Tolerance) approach will be enforced, where some proper modification to a conventional compilation toolchain, will make possible the identification of temporary fault and the adoption of fault tolerant solutions. The successful implementation of this project will de-risk the use of software-based radiation mitigation techniques and foster the adoption of high-performance while cost-effective COTS electronics in space applications.**

***Index Terms*—Fault tolerance, Space Applications, Radiation hardening**

I. Introduction

This goal of the project is to investigate, develop, and test radiation-hardening methods for using Commercial-Off-The-Shelf (COTS) SoC in space applications, specifically targeting their sensitivity to radiation. This sensitivity is a significant barrier to the widespread adoption of COTS components in the space industry. However, appropriate countermeasures, which can be either fully software-based or a combination of software and hardware functions, have shown promise in mitigating these effects. Moreover, this project aligns with recent advancements in satellite communications driven by Software Defined Radio (SDR), which utilizes software to process various signals, offering flexibility and enhanced processing power compared to traditional hardware components. To address the combination of high performance required by telecom applications and the possibility of implementing software radiation mitigation techniques, the present research will focus on the implementation of a mixed solution which combine flexibility and performances from programmable logic technologies and reliability and stability from hard-wired processors that, with applied mitigation techniques, can achieve the required dependability for space applications. To achieve this, the project develops prototype architectures incorporating fault-mitigation functions in both hardware and software. Representative telecommunication applications will be ported to these mitigated processors, and the prototypes will undergo validation through fault injection and radiation tests. The prototype will not only demonstrate the feasibility of these techniques but also provide critical reliability, availability figures, and performance measurements to support the future reuse of this solution in telecom payload processing units. The realization of this system will be the result of the partnership between Politecnico di Milano and Argotec. Politecnico di Milano is the largest technical university in Italy. It participates in this project through the HEAP laboratory[1] of the Dipartimento di Elettronica, Informazione e Bioingegneria, which is one of the largest ICT departments in EU. Argotec, based in Torino, Italy, is a leader in the development of small satellites for Deep Space missions, whose Avionics department has a productive background in the exploitation of mixing programmable logic and software layer management devices in its missions, along with previous experience on telecom satellite payloads.

The proposed work approach follows a linear structure articulated in four technical workpackages (WPs): WP1, WP2, WP3 and WP4, globally spanning over a two-year timeframe. After the kick-off, WP1 activities focus on requirements, specification, and gap analysis and terminate with the filing of the System Requirements. These outputs are input for the design activities in WP2 to select the technologies and demo platform that will be adopted for the final implementation. Then, WP3 implements the identified solutions, and it is followed by the Test Readiness Review. Finally, WP4 tests them and confirms the satisfaction of the specifications. Once the testing and verification WPs have been completed, a final review by ESA (European Space Agency) concludes the activities.

The remainder of the paper is organized as follows. Section II contains a review of the state of the art to create the appropriate background and positioning of ASTRAEUS objectives within the fault-tolerance landscape. In the following, Section III presents the characteristics of the application that will be improved during the execution of the project, while in Section IV the criteria and a preliminary analysis of the platforms for hardware implementation are reported. The potential benefits of the approach are reported in Section V,

The project ASTRAEUS was funded by ESA (European Space Agency), ESA Contract No. 4000147102/25/NL/RA, under the ARTES 4.0 Generic Programme Line “Core Competitiveness” Component A “Advanced Technology” (AT) Activity Reference ARTES AT 5C.490

[1]https://heaplab.deib.polimi.it/

while the different stages of the verification plan are discussed in Section VI. Some conclusions are drawn in Section VII.

II. State-of-the-Art

Software mitigation techniques are known in scientific community under the umbrella term Software-Implemented Hardware Fault Tolerance (abbreviated SIHFT or SIFT). SIHFT approaches are not a novel concept even if they have not found a widespread use in industry due to the still low TRL (Technology Readiness Level). An introductory reference on SIHFT is the book by Golubeva et al. [14]. The SIHFT techniques related to soft faults are able to detect and/or recovery from hardware errors by exploiting different mechanisms depending on the specific application and considered computational platform. in general, we can categorize the SIHFT recovery procedures in two categories [30]:

- *Space redundancy*: these techniques resemble the traditional hardware fault tolerance techniques by replicating the workload over different resources (for example, cores). This category includes N-modular redundancy (NMR) and reconfigurable duplication (also called standby redundancy).
- *Time redundancy*: these techniques replicate the workload on different time periods, possibly on the same resource. This category includes, for instance, re-execution, checkpoint/restart, recovery blocks, and forward error recovery.

SIHFT approaches for fault detection and fault isolation aim to determine the presence of an error in the system and to identify its location and characteristics, respectively. The seminal paper by Oh et al. presented the algorithm Error Detection by Duplicated Instructions (EDDI) [15]. This algorithm provides a description of how to implement instruction-level redundancy as an automatic transformation of the assembly code. Since that work, multiple approaches have tried to reduce the overhead of EDDI by introducing some mechanisms for selective checking and even selective duplication, using multiple heuristics to determine the most crucial variables, code sections, and functions [12] [17] [6] [35]. EDDI and its extensions provide data protection; however, the execution flow of the program is not protected. For this reason, multiple approaches for Control-Flow Checking (CFC) aim to protect the Control-Flow Graph (CFG) of the program [22] [36] [37] [38] [16]. EDDI and CFC techniques are often combined to provide a more extensive protection of the system. To relieve developers from having to manually insert SIHFT, researchers have developed compiler tools to automate the process [9] [11] [13] [32] [34].

Recently, POLIMI concluded an ESA OSIP project [23] that analyzed, in addition to mixed-criticality aspects, SIHFT with promising results. In the context of this project and other past EU projects, the software named ASPIS[2] has been developed by POLIMI. ASPIS is a LLVM-based tool that automatically injects SIHFT approaches into the Intermediate-Representation of the code. The tool is still in development but preliminary papers have been published [7] [8]. This tool is a suitable candidate to automatically inject SIHFT techniques in the software in the context of this project. Indeed, ASPIS is agnostic from the programming language and the hardware architecture, and an experimental validation preliminary showed the potential ability to fulfill the performance/reliability requirements of this project. Another, different, OSIP project also analyzed the possibility to use mixed software/hardware techniques [24]. Almost all of the aforementioned articles on software mitigation techniques dealt with SEU or MCU/MBU. SET and SDC manifest at software-level as SEU, MCU, or MBU so they are implicitly tolerated. Other higher events may be detected (like SEFI and SEL) but their recovery requires higher solutions. To the best of our knowledge, no academic or industrial works targeted higher events than SEU/MCU/MBU with pure software techniques. Concernig standards, the handbook ECSS-E-HB-20-40A has an entire chapter (14) dedicated to SIHFT. The handbook emphasizes the importance of SIHFT in the context of COTS hardware to reduce the costs but maintain reliability to SEEs. This chapter describes some basic SIHFT techniques and suggests manual code instrumentation. Quality standard ECSS-Q-ST-80C describes the requirements to be used for the development and maintenance of space software. It is highlighted the need to verify that the software correctly handles hardware failures without causing system-level failures. In other safety-critical standards, such as automotive software ISO-26262, railway software EN-, and aviation software DO-178C, SIHFT methodologies are usually a suggested practice as an extra measure to react to SEEs. All the standards assume that the software reliability introduced by SIHFT is not quantifiable; however, this is a common misunderstanding as highlighted by recent scientific articles [28] [31]. De-risking the SIHFT technology will lead to a path that may allow the quantification of the software reliability improvement obtained by SIHFT.

III. Use Case Scenario

The idea behind the proposed design architecture is to consider the key points in the digital section of a multicarrier regenerative processor and to integrate a set of suitable radiation mitigation techniques in the architecture to respond to the specific system specifications. The Defined System Scenario will be modelled on a satellite communication payload onboard processor (OBP), specifically designed for a multicarrier regenerative processor, utilizing Software Defined Radio (SDR) architectures. This processor architecture is selected due to its relevance in modern satellite communication systems, where flexibility and adaptability are crucial. The design will address both hardware and software aspects required to achieve the desired functionality, with a particular emphasis on radiation mitigation, a critical factor in spaceborne systems. The Defined System Scenario aims to represent a realistic onboard payload processor, thus reflecting the real-world operational conditions and challenges. This approach aligns with the development of a comprehensive technology demonstrator, incorporating essential aspects of a multicarrier regenerative processor, re-

[2]https://github.com/HEAPLab/ASPIS

ported in Figure 1. The objective is to integrate appropriate radiation mitigation techniques into the architecture to meet the specifications. To build a representative design that fully mirrors the main elements of a geostationary telecom satellite, the design process will consider the core building blocks of an SDR-based regenerative processor.

An SDR has been evaluated as a robust reference solution by the consortium due to its flexibility and capability to adapt to various signal processing tasks. SDR serves as an ideal reference solution, providing a robust framework for implementing the regenerative processing functions. In order to be a useful evaluation of the full system, the system must satisfy specific functional capabilities, summarized in Figure 2.

The first module of the proposed test architecture implements a polyphase filter channelizer to demultiplex parallel Frequency Division Multiplexing (FDM) channels, allowing for efficient processing of multiple signal carriers simultaneously. Polyphase filters have become very common in satellite telecommunication applications thanks to the fact that they reduce the price and complexness of the filter reducing the multiplication per input sample. Each FDM channel is then demodulated and Low-Density Parity-Check (LDPC)-decoded. The decoded data streams are subsequently formatted and LDPC-encoded, ensuring data integrity and robustness. Once the signal passes through the entire regenerative process, the performance of the system can be assessed using Frame Error Rate (FER) or Bit Error Rate (BER) values, which are indicative of the system's ability to handle errors and maintain data integrity helping in assessing the overall reliability and robustness of the system in space environments identifying any anomalies and assess the performances. The presented architecture includes modules that require large device resources/area to build a meaningful architecture from the radiation mitigation point of view. Moreover, other function blocks could be integrated in order to ensure carrier synchronization, filtering, and symbol timing recovery. The overall functional block diagram of the designed system is shown in Figure 3.

Note that the potential applications for both GEO and LEO orbits are broad, offering opportunities to develop and refine multiple stages of the proposed architecture. This design proposal not only supports a wide range of satellite communication scenarios but also enables further advancements and adaptations, ensuring the system's flexibility and robustness across multiple fields of applications.

IV. Platform requirements and selection process

The final configuration of the system is influenced by numerous parameters that must be carefully integrated to achieve compliance with all specified requirements. We already performed some preliminary trade-offs to initially identify the most appropriate configuration and/or devices. During the design phase, these analyses will be further refined, and all parameters will be thoroughly evaluated. Among the others, it will be crucial to conduct a thorough trade-off analysis to determine the best device that meets the system's specific requirements. Additionally, a careful evaluation of the Overhead vs. Fault Detection trade-off will be essential for the software mitigation aspect. This involves balancing the need to minimize system overhead with the effectiveness of fault detection mechanisms, ensuring that the chosen solution optimizes both performance and reliability. By carefully weighing these factors, the goal is to select a device and software strategy that together deliver optimal system functionality while addressing potential vulnerabilities. At this stage, no limitations or non-compliances have been identified that could prevent the project from meeting its objectives. However, some requirements still require further investigation:

- Full-mitigation of destructive SEE (especially catastrophic ones) is not possible in software and necessarily requires full-hardware protection and usually redundancy.
- Heavy-ion testing is still *tentative* depending on if ESA can provide access to an external facility.

The initial set of devices under consideration includes the Zynq Ultrascale+, Virtex Ultrascale+, Versal AI Edge, and PolarFire SoC. These devices are chosen for their advanced features and capabilities, but each of these devices offers unique strengths. The Zynq Ultrascale+ [5] and the Versal AI Edge [3], with their advanced ARM processors, provide robust processing capabilities suitable for applications requiring high computational power and efficiency. The Kintex Ultrascale [2] and the Virtex Ultrascale+ [4] stand out with the higher number of LUTs and FFs, making it ideal for applications needing extensive programmable logic resources. Meanwhile, the PolarFire SoC [25] offers a more balanced approach with a focus on power efficiency, thanks to its 28nm fabric technology, making it suitable for applications where power consumption is critical. In summary, the choice of device will depend heavily on the specific requirements of the application. If high computational performance and processing flexibility are prioritized, the Versal AI Edge and Zynq Ultrascale+ are excellent choices due to their powerful ARM cores and advanced capabilities. For applications demanding extensive programmable logic resources, the Kintex Ultrascale or the Virtex Ultrascale+ are the most suitable. Finally, for power-sensitive applications, the PolarFire SoC provides a compelling balance of performance and efficiency. The consortium's evaluation of these devices aims to identify the optimal solution that provides the best combination of performance, reliability, and dependability for the intended application.

V. Expected Benefits

Reliable and robust systems already exist and are widely utilized in the aerospace market. Several processors and FPGAs have been brought to a state of radiation tolerance or even hardened, sometimes by design, to ensure they are impervious to radiation-related issues. Notable examples include the LEON3 [18] or LEON4 [19] processors from Gaisler, as well as the newer, even if less powerful, SAMRH [27] processors from Microchip. For programmable logic, among the others, can be mentioned the European NanoXplore [20] [21], the emerging Certus NX [10] hardened in collaboration

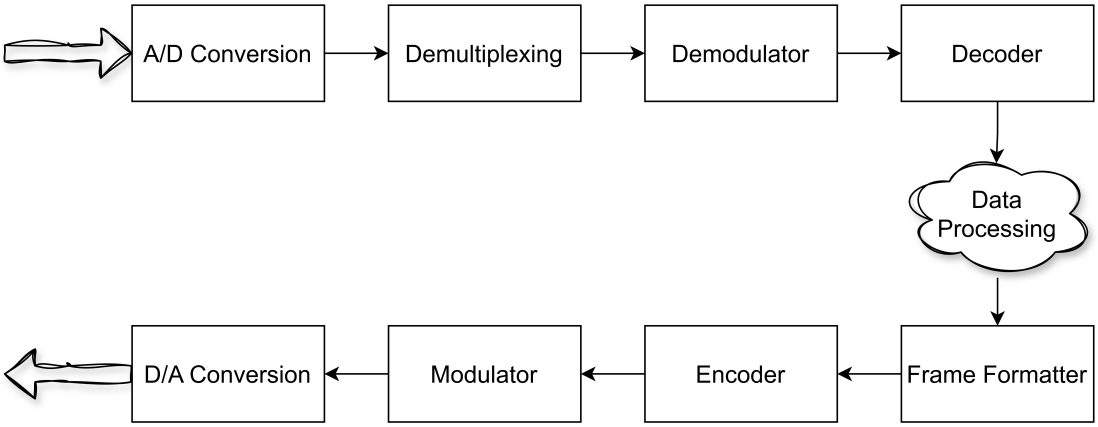

Fig. 1. General Architecture of a Digital Regenerative On-Board Processor.

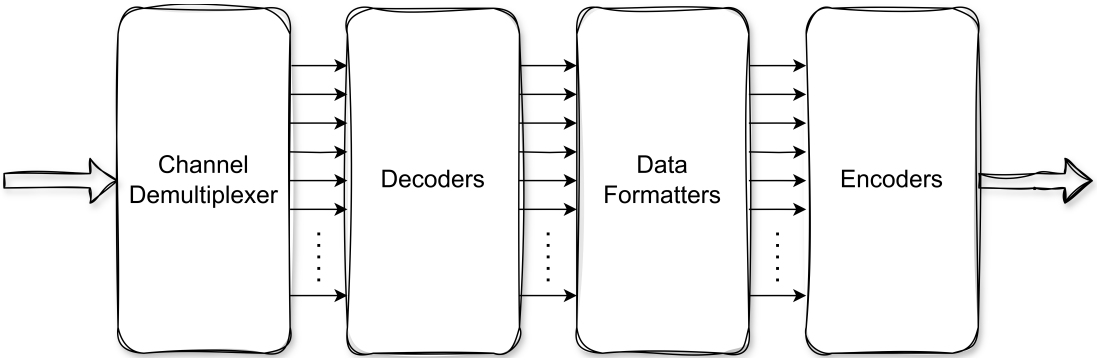

Fig. 2. Demonstrator Block Diagram.

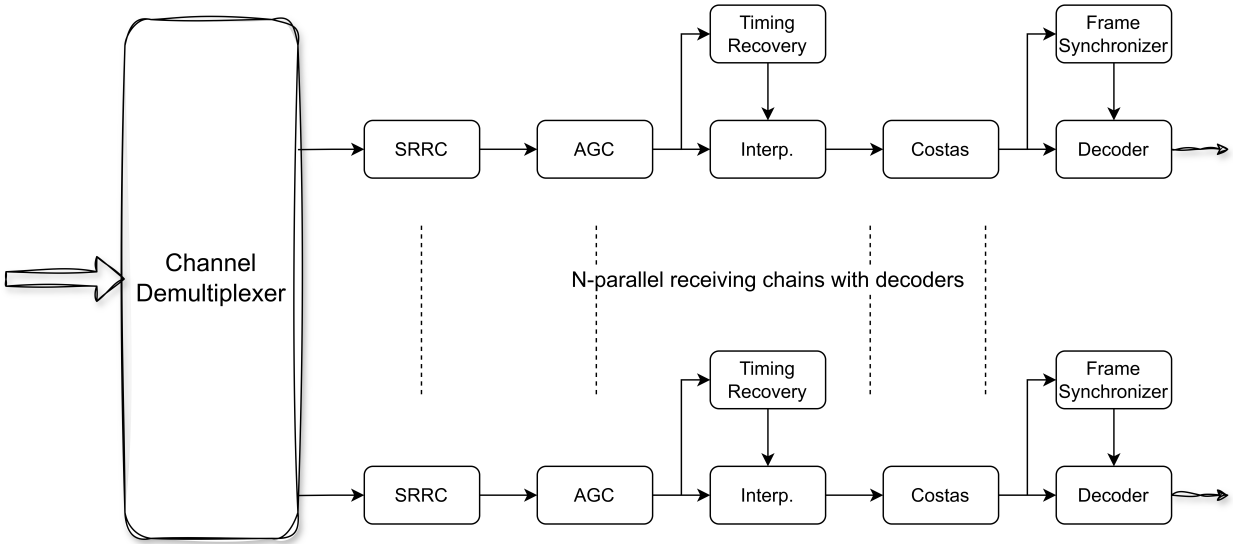

Fig. 3. Overall system: functional block diagram.

with Frontgrade, Microchip's flash-based RTG4 [33], and the older ProAsic [26]. Devices that combine both processing and programmable logic capabilities within the same SoC are less common in space applications. The latest Versal [3] and the emerging NanoXplore NG Ultra [20] are among the few that integrate both functionalities. These complex devices, especially those integrating both processing and programmable logic, tend to be extremely expensive and have long lead times. On the other hand, the use of commercial off-the-shelf (COTS) devices offers the advantage of reducing purchase costs and lead times. It also allows for the exploitation of emerging technologies without waiting for them to be qualified for space environments. However, using COTS devices necessitates the implementation of various mitigation and redundancy techniques to address and mitigate potential issues. Specifically, this project aims to focus on mitigation techniques for the processors side, with the goal of developing software mitigation strategies that, combined with a minimal set of hardware mitigations and reinforcements, will ensure that the system requirements are met successfully.

More in detail, the use of radiation-hardened processors is justified by the fact that COTS hardware is not designed to be radiation-tolerant and therefore not adequate for mission- or safety-critical operations. However, COTS platforms are attractive for two reasons: they offer high-performance capabilities and are order of magnitudes less expensive than specialized processors. The computational power is especially important for emerging applications that require large amount of computational power. The cost represents a critical issue in low-end satellites (like CubeSats) or large constellations in which the cost of the computational unit cannot be ignored. It should also be noted that COTS hardware tends to have better software support and many information/libraries/examples are available (often for free), making the software development process considerably easier. Software mitigation solutions can enable the use of COTS platforms in the space domain by introducing the missing reliability. Recent scientific works highlighted that SIHFT solutions are able to detect 99.8 % of the SEUs occurring in a computing platform [8]. Comparable results have been obtained by an ESA OSIP project in a laser-based and alpha-radiation based campaign [23]. State-of-the-art software mitigation techniques have two drawbacks:

- They require the developer to manually implement them, which has a negative effect from a software engineering standpoint, increasing the development time, reducing the code maintainability, and preventing the use of compiler optimizations
- They introduce a non-negligible overhead to run the detection and recovery mechanisms that are detrimental for timing properties, especially in hard real-time systems

The first issue has been solved in scientific community by providing a compiler mechanism that introduces SIHFT without the need of developer intervention [8]. The second issue has been also subject of several scientific works (see [30] for a comprehensive survey) and the problem is composed by the fact that COTS devices have a considerable larger amount of computational power. Enabling the use of COTS hardware opens new possibilities to pick the computational platform, including SoCs. SoCs including a CPU, and an FPGA accelerator are particularly interesting for the scenario described in Section 5.1.1. The CPU+FPGA SoC will be able to satisfy the target 10x-100x performance increase while guaranteeing sufficient fault tolerance metrics. For the sake of the example, let us consider the following embedded SoC: Freescale i.MX 95. This COTS SoC has a performance of 40 000 MIPS (MFLOPS) compared to the rad-hard processor LEON3 that has only 200 MIPS (MFLOPS). In addition, the FPGA module can further accelerate the workload. For example, a study in 2018 showed how an FPGA-implemented SDR can reach four times the computing power of a 16-core processor [1]. In this project, we seek to implement software-only mitigations techniques possibly assisted by the FPGA, while hardware-only solutions will not be considered because it would not achieve the goal of using COTS for space environment. The software techniques must be architecture neutral, so that such techniques can be applied with a minimal effort on different architectures.

VI. Verification Plan

The testing and verification processes driven by WP4 will be performed according to the following preliminary plan, organized in several consecutive steps:

1. Functional verification of the Test Application and of the selected benchmarks. The related software tasks, FPGA accelerator circuits, and their integration will be tested to verify their logic correctness, i.e., correctness of the output. Specifically designed tests will be developed for the Test Application as well as for selected benchmarks (in case they do not already provide output correctness verification).

2. Functional verification of the mitigation solutions. The implemented solutions will be tested by simulating, via software, the triggering of fault detection and fault recovery routines. After the instrumentation of the original code of Test Application and benchmarks, the step 1 must be redone to verify that the software mitigation solutions did not introduce any bug.

3. Performance assessment. The Test Application and the selected benchmarks are run on the real platform to measure the execution time with and without the mitigation techniques in order to assess the introduced overhead. The execution time will be measured in terms of average, worst-case observed, best-case observed, and variability. In addition to the time overhead, additional metrics will be derived, such as memory consumption and object file size.

4. Theoretical reliability assessment. A preliminary, theoretical, study on the reliability of the introduced SIHFT will be performed, depending on the Test Application characteristics and the available hardware metrics. It is unlikely that such study could allow to derive a perfect reliability metrics, but it will give an approximation of the expected result of the following fault injection campaign.

5. Simulated fault injection. Multiple fault-injection verification campaign will be performed via the use of software-based or hardware-based fault injection. For instance, the use of a debugger will allow to alter the state of memory cells or CPU registers to simulate the introduction of an error. The behavior of the system in terms of detection, recovery and final reliability metrics will allow to quantify the reliability introduced by the employed mitigation mechanisms. Both the SDR use case and benchmark suites will be used to assess the reliability performance.

6. Laser-based fault injection (optional). Laser-based fault-injection can be employed to preliminary test the device before the radiation campaign [29]. The laser-based fault injection, while not representative of a mission profile, can still trigger effects not visible with the simulated fault injection. Moreover, conversely to radiation testing, it enables the testing of only specific areas of the device, to better debug and understand the effect of the implemented SIHFT approaches.

7. Preliminary radiation-based fault injection (optional). Testing with radiation sources such as Californium-252 (alpha and neutron source) available at ESTEC would provide a preliminary evaluation that can reduce the risk of unsatisfactory implementations during proton-based or heavy-ion-based testing.

8. Proton-based fault injection. Proton-based facility will be selected, and proper proton-based fault injection experiments planned to obtain the reliability metrics. The proton test parameters (energy, flux, fluence, etc.) will be properly defined.

VII. Conclusions and Future Works

The expected outputs of the ASTRAEUS project could be beneficial for both space and avionics applications, and, in general, for any systems hosting critical applications, exploiting COTS components. The key objectives of ASTRAEUS are the following:

- Developing and prototyping fault mitigation functions for selected high-performance processors embedded in System on Chip (SoC) components.
- Validating these prototypes through fault injection and radiation tests, ensuring their dependability.
- Enabling a significant increase in computing performance and advanced features for commercial components compared to current space-specific data processing technology.

The final configuration of the system is influenced by numerous parameters that will be carefully integrated to achieve compliance with all specified requirements. At the current stage of development, we already performed some preliminary trade-offs to identify the most appropriate configuration and/or devices. During the following design phase, these analyses will be further refined, and all parameters will be thoroughly evaluated. Among the others, it will be crucial to conduct a thorough trade-off analysis to determine the best device that meets the system's specific requirements. Additionally, a careful evaluation of the Overhead vs. Fault Detection trade-off will be essential for the software mitigation aspect. This involves balancing the need to minimize system overhead with the effectiveness of fault detection mechanisms, ensuring that the chosen solution optimizes both performance and reliability. By carefully weighing these factors, the goal is to select a device and software strategy that together deliver optimal system functionality while addressing potential vulnerabilities. At the end of the project, it is planned the release of an improved version of an open source tool, named ASPIS, capable to augment the software code with identification and tolerance to soft transient faults.

References

[1] Akeela, R., Dezfouli, B.: Software-defined radios: Architecture, state-of-the-art, and challenges. Computer Communications **128**, 106–125 (2018). https://doi.org/https://doi.org/10.1016/j.comcom.2018.07.012, https://www.sciencedirect.com/science/article/pii/S0140366418302937

[2] Kintex UltraScale FPGAs Datasheet, https://docs.amd.com/v/u/en-US/ds892-kintex-ultrascale-data-sheet

[3] Versal Architecture and Product Data Sheet, https://docs.amd.com/v/u/en-US/ds950-versal-overview

[4] Virtex UltraScale+ FPGA Datasheet, https://docs.amd.com/v/u/en-US/ds923-virtex-ultrascale-plus

[5] Zynq UltraScale+ MPSoC Data Sheet, https://docs.amd.com/v/u/en-US/ds891-zynq-ultrascale-plus-overview

[6] Arasteh, B., Bouyer, A., Pirahesh, S.: An efficient vulnerability-driven method for hardening a program against soft-error using genetic algorithm. Computers & Electrical Engineering **48**, 25–43 (2015). https://doi.org/https://doi.org/10.1016/j.compeleceng.2015.09.020, https://www.sciencedirect.com/science/article/pii/S0045790615003419

[7] Baroffio, D., Reghenzani, F.: Compiler-injected sihft for embedded operating systems. In: Proceedings of the 20th ACM International Conference on Computing Frontiers. p. 337–343. CF '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3587135.3589944, https://doi.org/10.1145/3587135.3589944

[8] Baroffio, D., Reghenzani, F., Fornaciari, W.: Enhanced compiler technology for software-based hardware fault detection. ACM Trans. Des. Autom. Electron. Syst. **29**(5) (Sep 2024). https://doi.org/10.1145/3660524, https://doi.org/10.1145/3660524

[9] Bohman, M., James, B., Wirthlin, M.J., Quinn, H., Goeders, J.: Microcontroller compiler-assisted software fault tolerance. IEEE Transactions on Nuclear Science **66**(1), 223–232 (2019). https://doi.org/10.1109/TNS.2018.2886094

[10] Certus™-NX-RT and CertusPro™-NX-RT FPGAs Product Brief, https://www.frontgrade.com/sites/default/files/documents/advanced_product_brief_for_frontgrade_certus_fpgas_v2.pdf

[11] Didehban, M., Shrivastava, A.: nzdc: a compiler technique for near zero silent data corruption. In: Proceedings of the 53rd Annual Design Automation Conference. DAC '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2897937.2898054, https://doi.org/10.1145/2897937.2898054

[12] Feng, S., Gupta, S., Ansari, A., Mahlke, S.: Shoestring: probabilistic soft error reliability on the cheap. In: Proceedings of the Fifteenth International Conference on Architectural Support for Programming Languages and Operating Systems. p. 385–396. ASPLOS XV, Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1736020.1736063, https://doi.org/10.1145/1736020.1736063

[13] Geier, J., Auer, L., Mueller-Gritschneder, D., Sharif, U., Schlichtmann, U.: Compasec: A compiler-assisted security countermeasure to address instruction skip fault attacks on risc-v. In: Proceedings of the 28th Asia and South Pacific Design Automation Conference. p. 676–682. ASPDAC '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3566097.3567925, https://doi.org/10.1145/3566097.3567925

[14] Goloubeva, O., Rebaudengo, M., Sonza Reorda, M., Violante, M.: Software-Implemented Hardware Fault Tolerance. Springer (2006). https://doi.org/10.1007/0-387-32937-4

[15] Hwang, I., Kim, S., Kim, Y., Seah, C.E.: A survey of fault detection, isolation, and reconfiguration methods. IEEE Transactions on Control Systems Technology **18**(3), 636–653 (2010). https://doi.org/10.1109/TCST.2009.2026285

[16] Jens Vankeirsbilck, Niels Penneman, H.H.J.B.: Computer Safety, Reliability, and Security, chap. Random Additive Control Flow Error Detection. Springer International Publishing (2018), https://www.springerprofessional.de/en/computer-safety-reliability-and-security/16091624

[17] Khudia, D.S., Wright, G., Mahlke, S.: Efficient soft error protection for commodity embedded microprocessors using profile information. In: Proceedings of the 13th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, Tools and Theory for Embedded Systems. p. 99–108. LCTES '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2248418.2248433, https://doi.org/10.1145/2248418.2248433

[18] Dual core Leon3-FT processor Datasheet, https://www.gaisler.com/doc/gr712rc-datasheet.pdf

[19] Quad Core LEON4 SPARC V8 Processor Datasheet and User Manual, https://www.gaisler.com/doc/gr740/GR740-UM-DS.pdf

[20] NG-ULTRA NX2H540TSC Family Datasheet, https://files.nanoxplore.com/f/5d0ab167586d48588a49/?dl=1

[21] NG-Ultra 300 Datasheet, https://files.nanoxplore.com/f/c67b45df77ca40e5b93b/?dl=1

[22] Oh, N., Shirvani, P., McCluskey, E.: Control-flow checking by software signatures. IEEE Transactions on Reliability **51**(1), 111–122 (2002). https://doi.org/10.1109/24.994926

[23] OSIP project "Mixed-Criticality Fault-Tolerant Systems for the Future Generation of Spacecraft Computers, https://activities.esa.int/index.php/4000133770

[24] OSIP project "Mixed Software/Hardware-based Fault-tolerance Techniques for Complex COTS System-on-Chip in Radiation Environments, https://ideas.esa.int/servlet/hype/IMT?documentTableId=45087150962229154&userAction=Browse&searchTerm=Q09UUw&templateName=&documentId=1e143109eccaf1d5419966386049bc79&searchContextId=2d64d5a2e192f57b3745cb2da643912a

[25] PolarFire® SoC Datasheet, https://www.microchip.com/en-us/products/fpgas-and-plds/system-on-chip-fpgas/polarfire-soc-fpgas

[26] Radiation-Tolerant ProASIC3, https://ww1.microchip.com/downloads/aemdocuments/documents/fpga/ProductDocuments/DataSheets/rtpa3_ds.pdf

[27] Rad-Hard 32-bit Arm® Cortex®-M7 Microcontroller Datasheet, https://www.microchip.com/product-change-notifications/#/15191/SYST-05MORJ125

[28] Reghenzani, F.: Enabling software technologies for critical cots-based spacecraft systems. In: Proceedings of the 20th ACM International Conference on Computing Frontiers. p. 236–242. CF '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3587135.3592765, https://doi.org/10.1145/3587135.3592765

[29] Reghenzani, F., Baroffio, D., Lopez, T.A., Fornaciari, W., Borel, T., Krimmel, F.: Laser fault injection methodology for software reliability testing. IEEE Reliability Magazine **2**(1), 52–63 (2025). https://doi.org/10.1109/MRL.2024.3521879

[30] Reghenzani, F., Guo, Z., Fornaciari, W.: Software fault tolerance in real-time systems: Identifying the future research questions. ACM Comput. Surv. **55**(14s) (Jul 2023). https://doi.org/10.1145/3589950, https://doi.org/10.1145/3589950

[31] Reghenzani, F., Guo, Z., Santinelli, L., Fornaciari, W.: A mixed-criticality approach to fault tolerance: Integrating schedulability and failure requirements. In: 2022 IEEE 28th Real-Time and Embedded Technology and Applications Symposium (RTAS). pp. 27–39 (2022). https://doi.org/10.1109/RTAS54340.2022.00011

[32] Reis, G., Chang, J., Vachharajani, N., Rangan, R., August, D.: Swift: software implemented fault tolerance. In: International Symposium on Code Generation and Optimization. pp. 243–254 (2005). https://doi.org/10.1109/CGO.2005.34

[33] Microchip RTG4 Datasheet, https://ww1.microchip.com/downloads/aemDocuments/documents/FPGA/ProductDocuments/DataSheets/RTG4_FPGA_Datasheet.pdf

[34] Sharif, U., Mueller-Gritschneder, D., Schlichtmann, U.: Compas: Compiler-assisted software-implemented hardware fault tolerance for risc-v. In: 2022 11th Mediterranean Conference on Embedded Computing (MECO). pp. 1–4 (2022). https://doi.org/10.1109/MECO55406.2022.9797144

[35] Thati, V.B., Vankeirsbilck, J., Boydens, J., Pissort, D.: Selective duplication and selective comparison for data flow error detection. In: 2019 4th International Conference on System Reliability and Safety (ICSRS). pp. 10–15 (2019). https://doi.org/10.1109/ICSRS48664.2019.8987731

[36] Vankeirsbilck, J., Penneman, N., Hallez, H., Boydens, J.: Random additive signature monitoring for control flow error detection. IEEE Transactions on Reliability **66**(4), 1178–1192 (2017). https://doi.org/10.1109/TR.2017.2754548

[37] Vemu, R., Abraham, J.: Ceda: Control-flow error detection using assertions. IEEE Transactions on Computers **60**(9), 1233–1245 (2011). https://doi.org/10.1109/TC.2011.101

[38] Zhang, Z., Park, S., Mahlke, S.: Path sensitive signatures for control flow error detection. In: The 21st ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems. p. 62–73. LCTES '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3372799.3394360, https://doi.org/10.1145/3372799.3394360