Protecting the CCSDS 123.0-B-2 Compression Algorithm Against Single-Event Upsets for Space Applications

Daniel Báscones, Francisco García-Herrero, Óscar Ruano, Carlos González, Daniel Mozos, *Member, IEEE*, and Juan Antonio Maestro, *Senior Member, IEEE*

***Abstract*—Hyperspectral imaging is an excellent tool to remotely analyze the Earth from in-orbit devices. Satellites capture these images containing vast information about the ground pixels. To optimize storage and transmission speeds, compression is often performed onboard the satellite. To that end, algorithms such as the CCSDS 123.0-B-2 are implemented on FPGAs, enabling this process in an efficient and fast manner. Single-Event Upsets (SEU) are commonplace in this scenario, e.g. bit flips in the FPGA's configuration memory which can catastrophically alter the algorithm's output. In this paper, we propose a fault tolerance technique for this specific case. The compression core is checked periodically by running a golden model designed to excite the full internal datapath based on a synthetic image. A failure in this check will trigger a reconfiguration of the compression core. Results show better detection rates than Dual Modular Redundancy (DMR) at a fraction of the resource cost, proving this technique as a viable alternative. Furthermore, other algorithms with similar processing flows might benefit as well from this technique.**

***Index Terms*—Remote sensing, reconfigurable architectures, fault tolerance, image coding, satellite applications.**

I. Introduction

In recent years, the number of satellites orbiting the Earth has grown to almost 10,000 active satellites. Although most of them are dedicated to telecommunications, navigation, and spatial sciences, over 10% are employed for remote sensing. One of the many technologies that is used in this particular field is hyperspectral imaging [1]. These images include, for each spatial pixel, information about hundreds of different points in the electromagnetic spectrum. These *hyperspectral signatures* help scientists track climate change effects [2], monitor crop fields [3], detect targets [4], identify different materials and minerals [5], and many more [6].

Received 20 May 2024; revised 27 September 2024; accepted 2 December 2024. This work was supported in part by the Spanish Ministry of Science and Innovation under Grant PID2023-147059OB-I00 and Grant PID2020-112916GB-I00. Recommended for acceptance by J. Abella. *(Corresponding author: Juan Antonio Maestro.)*

The authors are with the Department of Computer Architecture and Automatics, Computer Science Faculty, Complutense University of Madrid, 28040, Madrid, Spain (e-mail: danibasc@ucm.es; francg18@ucm.es; oruano@ucm.es; carlosgo@ucm.es; mozos@ucm.es; jamaestro@ucm.es).

Digital Object Identifier 10.1109/TC.2024.3512203

Their on-orbit collection poses many challenges, especially considering the great amount of data that can be captured by a single sensor such as AVIRISNG [7]. Data needs to be quickly processed and sent back to Earth in order to avoid memory bottlenecks. To optimize the transmission link capacity, compression algorithms can be used to reduce the image size beforehand.

Many compression techniques have been proposed over the years: generalizations of traditional image compression techniques [8], extensions that take into account the spectral correlation present within pixels [9], and *ad hoc* algorithms targeting the specific characteristics of hyperspectral data. The latter have excelled in both compression ratio and quality, with the CCSDS 123.0-B-2 algorithm [10], [11] being the reference in performance. It is a standard supported by the main spatial agencies, achieving high compression rates in both lossy and lossless compression modes while at the same time designed to be hardware-friendly.

Many implementations have arisen in recent years, with special focus on real-time performance to avoid the aforementioned bottlenecks. FPGAs have been generally the platform of choice [12], [13], [14], [15], [16], traditionally outperforming CPUs and GPUs [17] in energy consumption with similar or better performance. An added benefit of using FPGAs is the fact that there have been radiation-hardened versions available for some time [18], which can be readily used for space missions.

These missions often deal with unpredictable errors due to radiation altering the internal state of electronic components. One of the most common kinds of errors are the so-called Single-Event Upsets (SEU) [19], which induce spontaneous bit flips in the storage elements of the circuits. For reprogrammable devices, this is especially problematic since the errors can occur not only in the data, but in the datapath itself, altering functionality until reprogramming is performed.

In ASIC-based systems, hyperspectral image compression is often considered a non-critical mission operation because an SEU typically affects only a single image, and the error dissipates afterward. This is because, in ASICs, the configuration is static, and any bit flip caused by radiation is isolated to the affected image and does not propagate further. Once the error affects the data of that image, the system continues to function normally, as soft errors do not affect the structure of the circuits but only the data. This characteristic of ASIC systems makes them less vulnerable to the persistent effects of soft errors, as

these are transient and do not impact the ongoing operations permanently.

However, in FPGAs, the situation is more critical. If an SEU affects the configuration memory, it can alter the behavior of the circuit, causing incorrect compression of not just one image, but all subsequent images until the configuration is repaired. Unlike ASICs, FPGAs are reconfigurable, and errors in the configuration memory can result in a cascading effect, where the failure of one component leads to further erroneous outputs until the system is reprogrammed. This situation can significantly impact mission operations, especially in environments like space.

In the context of small satellite missions, such as CubeSats, these challenges are even more pronounced. CubeSats typically have strict limitations on power, size, and shielding capabilities, making the use of heavy-duty fault-tolerant mechanisms less practical. Therefore, it becomes crucial to explore alternative protection strategies, such as the one proposed in this paper, that offer a more efficient and resource-conscious solution while still ensuring reliable error correction for radiation-induced failures.

By design, CCSDS 123.0-B-2 compression and decompression is a process that is very sensitive to errors. A predictive model is used to estimate input data values based on available information. The better the model, the higher the compression ratio. This model is grown from input data statistics and intermediate values generated by the compressor. An error in any of them will propagate forward until the end of the compression process, rendering the information unrecoverable [20].

It is thus of uttermost importance that the compression and decompression process is carried out unaffected by errors. Decompression is done back at the ground stations (where SEUs are not usually a problem), transmission can be protected by Cyclic Redundancy Check (CRC), and memory by generic Error Correction Codes (ECC), but compression must be done error-free at the capture device.

In these cases, techniques based on modular redundancy, such as Dual Modular Redundancy (DMR) and Triple Modular Redundancy (TMR), have been traditionally used [21]. These techniques respectively duplicate or triplicate the circuit logic, in order to detect errors, and even to correct them in the case of TMR. However, the overhead (both in area, power, resources, etc.) is replicated with each copy of the circuit. This makes these techniques very costly, and many times unfeasible. Thus, other approaches are usually preferable.

An alternative is to use reduced precision techniques. Although these techniques are also based on redundancy, the extra copies that are added are not identical to the original circuit, but simplified versions which incur smaller overheads. This usually applies to arithmetic and filtering systems, where functionally-equivalent copies can be obtained by reducing the precision of the calculations [22].

Algorithm-Based Fault Tolerance (ABFT) is another protection strategy that is widely used in circuits that present certain mathematical properties. In this way, some checks can be applied in different points of the circuits to verify that the property is met. If not, that would imply that an error is present in the system [23].

Scrubbing is another approach where the configuration memory of the FPGA is periodically refreshed. Data corruption detection still needs to be addressed since the technique itself does not perform checks on output data, but only on the inner configuration bits. This technique relies on high configuration refresh rates to avoid errors from persisting, with just a small area overhead [24].

Protection needs to address two important factors: First, it needs to detect the error fast, in order not to lose critical information. Second, it needs to be power and resource efficient, since in the context of on-board satellite FPGA processing, both are required to be under strict thresholds.

In this paper, we propose a novel fault tolerance technique for the case of CCSDS 123.0-B-2 compression. Our approach works by periodically running a golden dataset through the core, which is based on a synthetic image [25] and designed to use all of the internal components. If the output does not match the golden model, we can assume that some internal component has a critical error, and thus reprogramming is necessary. This method only requires the golden data to be available to the core under protection, being much cheaper than other approaches as the golden model is between 100 and 2,000 times smaller than non-synthetic images from the sensors.

Results show that the proposed technique incurs less than 10% overhead, when compared to the 100% overhead of DMR. Still, detection capacity remains as high with more than 99.7% of the critical errors detected, proving this method to be effective in the case of error detection.

Specifically, the contributions of this work are:

- The design of two different error detection techniques that protect the CCSDS 123.0-B-2 compressor architecture by computing a synthetic image.
- The hardware implementation of the suggested methods on an FPGA device based on a low-area/low-latency architecture.
- The verification of the performance of the error detection technique by running the architecture on an FPGA device (also simulated in software testing 50,000 image compressions) and testing the behavior with different input data from three sensors.
- The proposal of a procedure to check if the architecture is reliable or not, correct the errors in the architecture and compress, if necessary, the image to meet the timing constraints of three different sensors.

It is important to remark two questions:

- The technique is applied to protect the hardware of the compressor architecture, not to check if the input data from the images have been altered or not. The focus of this contribution is to protect the configuration memory of the FPGA device that contains the architecture of the compressor itself, as other parts (such as the memory buffers) can be easily protected with the error correction codes integrated into the devices. Unlike ASIC architectures, in which the errors may be transient, when FPGA devices suffer an event upset the error is permanent, because it modifies the configuration memory. Therefore, recapturing and recompressing the image cannot be applied directly, as the

hardware of the compressor has been corrupted and it will not recover until a reconfiguration of the memory is performed.

- The suggested techniques to protect this hardware are based on a synthetic image. To show their effectiveness, different input images are tested, but in all cases, the image computed to detect the errors is a synthetic one which is between 100 and 2,000 times smaller than the input images from the different sensors under test. Using this small image that excites all the hardware units by design, ensures constant and low overhead with the independence of the input information.

The paper is organized as follows: First, the context of the problem is given, providing an idea of applicability. Then, the proposed error detection technique is explained, followed by the test platform used to simulate SEUs. Finally, results are presented and compared with existing techniques, followed by the conclusions of this research.

II. Problem Context

Satellites are constantly exposed to radiation, thus suffering multiple errors as SEUs. This is especially critical in small satellites (e.g. CubeSats [26]) due to their hard constraints in terms of how much shielding they can support or how much power they can devote to implement more fault-tolerant systems. Unlike large-scale missions (such as e.g. those involving the ISS) where the instrumentation can afford more robust error-mitigation techniques, the equipment used in small satellites is more aligned with the limitations of these platforms. The focus of this paper is precisely on addressing the specific needs of small satellite missions.

Although it varies depending on the mission details, a CubeSat can usually afford around 1-Watt power budget [27] operating in Low Earth Orbit (LEO). This constrains the operation of the satellite, having to be extremely efficient in data management and transmission. In particular, satellites that perform Earth observation missions are especially sensitive to this situation. CubeSats in LEO orbits have quite limited time in view of the ground station, which is the only time interval in which the downlink transmission can happen. On average, the typical view time of such missions is around 5 - 10 minutes per pass [28]. Since observation sensors have a much higher throughput, this means that satellites should store multiple images onboard until the transmission link to the ground station is enabled. Also, since that transmission window is limited, the amount of information has to be minimized, and that is why images are usually compressed before they are stored and sent to the station. The operation workflow is depicted in Fig. 1.

First, the sensors acquire the images, which are immediately compressed, and the output of this compression is stored onboard. When a transmission window opens, the stored images are downlinked to the station. This procedure poses several risks from the reliability perspective. Since images spend a significant time in the satellite, the probability of being affected by an SEU is relevant. Thus, it is critical to apply protection mechanisms to the information in order to preserve its integrity.

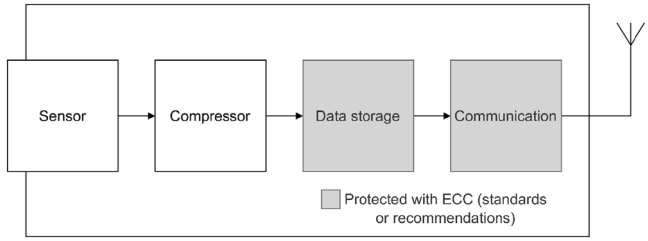

Fig. 1. Block diagram of the system from a fault tolerance perspective.

While the images are stored in the onboard temporary memory, Error Correction Codes (ECC) are usually implemented in order to achieve this protection. A typical memory technology for these applications is three-dimensional memory devices such as HMC and HBM [29], [30], which have been deeply studied against radiation effects and integrate several fault tolerant architectures based on coding techniques like the ones in [31].

After this, in the transmission period, images are usually protected using CRC and similar techniques. For example, the Consultative Committee for Space Data Systems (CCSDS) suggests the use of low-density parity-check codes of different rates with excellent error correction performance (these codes are capacity achieving) [32] or low-complexity schemes to protect the downlink transmission such as Bose–Chaudhuri–Hocquenghem codes (BCH codes) [33]. Even new non-binary LDPC codes are suggested to protect synchronization frames and telecommands, ensuring that the transmission of data from the satellite to the ground stations and vice versa is preserved [34].

But these efforts are worthless if the images have suffered an SEU during the compression process, since that error would be stored and subsequently transmitted no matter how well these processes are protected. The issue with the compression is that, since it is implemented on an FPGA, any configuration memory error would alter the structure (and therefore the functionality) of the algorithm. To prevent this from happening, a technique to detect the error has to be implemented, and once this happens the FPGA has to be reconfigured in order to get rid of the bit flip in the configuration memory. The scope of this paper is to provide an efficient solution to this issue, which will be explained in the next sections.

III. A Novel Error Detection Technique

CCSDS 123.0-B-2 works by creating a predictive model with the input data. New samples are predicted with the existing model, and only the differentials are encoded. The encoder itself has also a model to adapt to the differentials' statistics. This method works extremely well if both the predictor and encoder models can efficiently adapt to the input data. A diagram of the process is shown in Fig. 2.

The decoder must be able to reconstruct the exact same model in order to correctly decode the differentials. If at some point the models are out of sync, the decoder will try to decode data based on a model different from the one it was encoded with, and the decoded data will not be correct. CCSDS 123.0B-2 works by

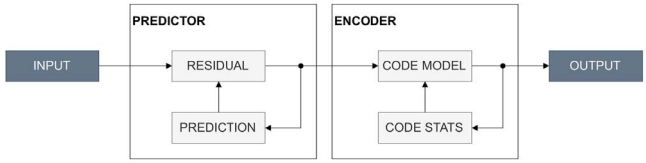

Fig. 2. Simplistic view of the CCSDS compression core, showing both the predictor and encoder feedback loops.

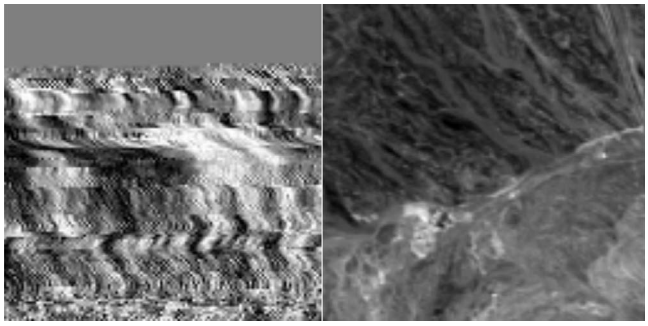

Fig. 3. Original and decompressed images after an SEU in the datapath. The black and white part is erroneously reconstructed, the grey part corresponds to missing bits.

TABLE I
RESULTS FOR 10000 SEU SIMULATION IN DIFFERENT INTERNAL REGISTERS

Register	Corrupted?	Detection	
		Last 8B Check	Length Check
Central difference	10000	10000	9999
Local Sum	10000	10000	10000
Mapped residual	10000	10000	10000
Input sample	10000	10000	10000
Weight vector	10000	10000	9991

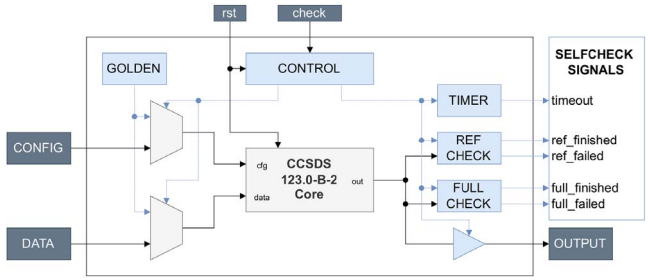

Fig. 4. Proposed protection technique for the CCSDS 123.0-B-2 core. Note that GOLDEN is the memory that stores the synthetic image that is used to detect the errors in the architecture of the compressor.

decoding in reverse order (we have used the hybrid encoder), so if the final model is corrupted, deconstruction will not be possible. An example of this effect can be seen in Fig. 3.

The models can get out of sync when an SEU occurs in the datapath, eventually affecting the internal state and all subsequent outputs, including the final model. This fact alone motivates protecting the circuit against errors, since erroneous outputs cannot be easily recovered. Instead of using traditional protection techniques such as DMR and TMR, we can exploit this cascading error characteristic.

The idea is as follows: If an error was introduced in the FPGA's configuration memory, it would end up affecting the prediction or coding of a sample (either from an error in the compressor data path, or in the internal state). The compressor's output will be erroneous from that point forward. Thus, by detecting a single error in the output, we can assume that an error occurred in its generation, and the circuit is malfunctioning. Furthermore, since the last bytes of the output would always be erroneous under this assumption, only those need to be checked.

We tested this hypothesis by simulating SEUs in software and checking the output results. We chose a register, a random bit within that register and a random time slot within the compression time. After the specified time, that bit was randomly flipped when read. With this approach, we performed a total of 50,000 image compressions, using the Cuprite Valley image as a reference. We compared the raw outputs, the last 8 bytes, and the compressed length. Results are shown in Table I. The output image was always corrupted when an SEU was introduced. By checking the last 8 bytes, corruption was detected 100% of the times. By checking the output length, corruption was detected $>$ 99.91% of the times.

To know what the expected output bits or length are, we would need another compressor producing the same output, going back to a DMR approach. Instead of working over random data, we stored a precomputed result over a test pattern ("TestPattern" in [25]). The test pattern input is a set of values (a $17 \times 36 \times 100$ pixel image) which are designed to excite the full range of internal values of the compressor. The test pattern output is the compressed version of that pattern.

This test pattern is proposed by the CCSDS as a verification mechanism for the compression standard, as indicated in the CCSDS 120.2-G-2 informational report [35]. If a given implementation compresses this test pattern correctly, it is assumed that the implementation properly handles edge cases in the internal counters and registers, and thus will also function correctly under typical excitation values. We theorize that a failure in the compressor circuitry will most likely result in errors when compressing the test pattern image. This is confirmed later in Section V, where a circuit failure that leads to corruption of standard images is detected over 99.7% of the time by running the test pattern through the compressor and verifying its output.

To test the circuit for errors, the test pattern (which is stored in a TMR BRAM with periodic scrubbing to preserve its integrity) is run through the compressor periodically, and the output is compared with the golden output. If they differ, that means that the core is producing erroneous output bits and thus, an internal error has occurred either in the compressor or in the checker. A reconfiguration would be triggered in this case to ensure the circuit is corrected. Two are the main advantages of this method over other solutions: i) It has error-detecting capabilities like DMR (only triggering reconfiguration if needed), while ii) using a small amount of resources, like scrubbing.

Two types of modules are used in this error detection scheme, which can be seen in Fig. 4.

- A timer counts the number of elapsed cycles, triggering a timeout signal when the test pattern is supposed to be compressed (we can know beforehand the number of cycles that the test pattern takes to compress since the circuit is fully deterministic). If the timeout signal triggers, but the other detection modules have not finished, an error has occurred that has resulted in less data output than expected.
- A comparator checks that the output matches the expected result. Two different comparators are introduced: one that checks the full output image, and another one that just checks for output size and final frame (64 bits) to match. The idea is that, given the unpredictability of the output when an error occurs, this second checker will perform just as well as the full checker, while using a fraction of the memory.

As a side note, errors might also impact the protection technique itself, potentially rendering it non-functional. In our implementation, just the critical components (timer, comparators) have been triplicated in order to avoid this problem. Also, whenever an error in these components is detected, reconfiguration is triggered as with the rest of the system errors. The overhead due to the triplication of these components is negligible due to their small relative size in the design.

The proposed solution relies on the previously mentioned pattern in [25], which will indicate the presence of an error when its compressed output does not match the pre-calculated behavior. However, this incurs a time overhead due to the periodic check of this pattern. Therefore, an interesting matter would be to identify the minimal size that such a pattern should have in order to be effective. The smaller the pattern is, the less time it would consume while being checked. But, as mentioned before, it is important that the pattern excites all the parts of the circuit, in order to verify that no error is present. A smaller pattern could eventually lose this coverage and therefore it would not be so effective. Therefore, in order to test this, two alternatives to the proposed method will be considered:

- Method A: To check the full compressed pattern output, as described before.
- Method B: To use only the last compressed frame of the pattern, as the minimal element to be checked.

The novelty of Method B is based on leveraging how the compressing algorithm works, i.e., propagating the calculations (and thus any eventual error) from the beginning to the last bytes. Therefore, these last bytes form a kind of signature that allow us to detect an error in the system just by examining these bytes. This produces savings on both time and power consumption respect to Method A, with a very similar error detection capability.

One important point is that there are other situations, apart from SEUs, in which this proposal would add extra value in terms of reliability. For example, in order to detect Total Ionizing Dose (TID). TID is produced over time (not as the result of a particle strike as in the soft error case), and acts on the device cumulatively. When the accumulated radiation surpasses a certain amount, the device will not work anymore, making its effects permanent. Eventually, scrubbing mechanisms would not work in the device, posing a potential risk to reliability. The presented technique could also be used to detect this situation, in the same way it works with SEUs.

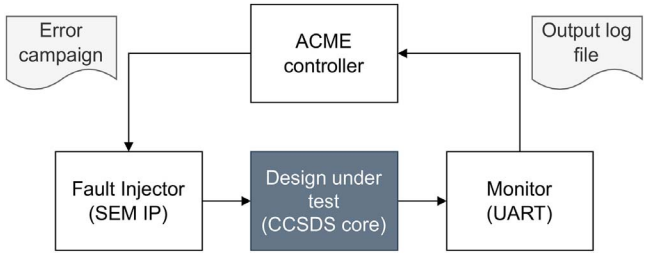

Fig. 5. Schematic diagram of the fault injection environment based on the ACME tool.

IV. Test Platform Based on FPGA Fault Emulation

Nowadays, there are different ways to test circuits implemented in an FPGA to verify their correct behavior against radiation effects. The first one consists of performing physical experiments at ground facilities, trying to replicate the conditions of the space environment. This possibility would be conducted at particle accelerators to replicate the impact of particles in the device. The main issues for this kind of approach are both the cost and the availability of the facilities in addition to the control of the experiments. Due to this, tests based on fault emulation techniques are presented as a more convenient option. This methodology (which is not intended to replace radiation tests, but to perform a first analysis on soft errors that would be eventually refined later with actual physical tests) is based on inducing artificial bit flips using several methods. In this work, the injection procedure is done by flipping bits on a Zynq-7000 SoC configuration memory through the automatic configuration memory error-injection tool (ACME) [36]. In this way, the design can be altered and its behavior can be assessed using different benchmarks and test cases both exhaustively and not.

As a particularity, to inject errors in the configuration memory, ACME makes use of the Xilinx Soft Error Mitigation (SEM) IP Controller [37]. In Fig. 5 the ACME schematic diagram is presented. The modules that make up the injection tool are the following:

- ACME Controller is the core that controls the error injection process, from the error campaign generation to the analysis of the results.
- The fault injector is the module in charge of executing the injection models generated by the ACME controller. For this purpose, SEM IP is the Xilinx core used to write on the Zynq-7000 SoC configuration memory to emulate bit flips.
- Design under test (DUT) is the target circuit to be studied in the presence of bit flips (SEUs). In this case, the protection techniques proposed in the previous section for a real-time compressor based on the CCSDS 123.0-B-2 standard are the target of the experiments.
- Monitor is the message-passing interface to trace all the system interactions via UART protocol.

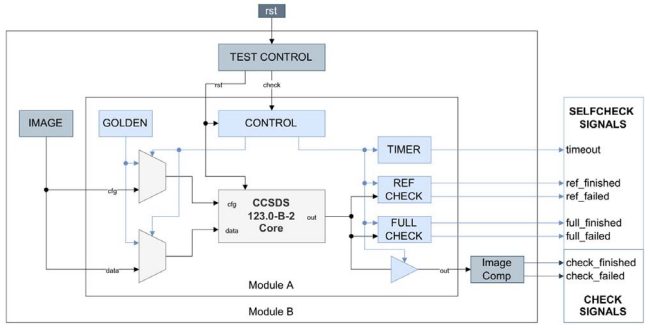

Fig. 6. Experimental setup. Module A is the proposed technique from Fig. 4, Module B encapsulates it for testing.

The adoption of the SEM IP injector engine responds to the need for a solution based on a non-invasive way to read and write into the Xilinx FPGA configuration memory, to manage the fault injection process. Also, although the error campaign has been performed in an exhaustive way in this work, which means 2,935,196 injections for a Zynq-7000 SoC board, ACME was able to improve the performance in some cases, working on reduced sets of the configuration bits defined by Xilinx in [38], [39]. This optimization capacity is offered by the ACME tool [40], which is able to extract the essential bits of the specific design.

V. Experimental Results

A. Error Detection Analysis

In this section we describe the experimental setup that was followed to perform the analysis of faults in the FPGA implementation of the compressor (with different images and degrees of compression) and to validate the proposed solution. In Fig. 6, a block diagram with the different modules of the test can be seen. Module A includes the protected design that consists of the architecture from [16] and the extra control units and memories required to detect failures. As we can see, Module A has five different outputs: *selfcheck-timeout*, *selfcheck-full-failed*, *selfcheck-full-finished*, *selfcheck-ref-failed* and *selfcheck-ref-finished*.

The *selfcheck-full-finished* and *selfcheck-ref-finished* signals are activated during the detection phase, when the expected amount of output bits have been emitted by the compressor. It is important to remember once again that, as we are using a fixed image to test the status of the compressor, the number of clock cycles is known *a priori*. The *full* and *ref* tags of the signals indicate if we are detecting the errors by running and comparing the full reference pattern (method A defined in Section III) or just by comparing the last frame (method B, Section III). As one frame has 64-bit length, we started with the hypothesis that checking only the last frame would provide a large enough error detection coverage.

Signals *selfcheck-full-failed* and *selfcheck-ref-failed* are active when the output of the compressor differs from the output of the pre-compressed pattern, when checking the full-frame or only the last one (method A and B respectively). Module B includes a golden model of a compressed normal image in which no error is injected. Module B has two different outputs that are active when the compressed image from Module A differs from the output in Module B (*check-failed*), or when Module B ends but Module A is still running (*check-finished*). In other words, Module B outputs allow us to control the real deviations of Module A compressor from the ideal behavior when faults are injected.

For the injection, in Module A we use a modified version of the ACME tool described in Section IV, in which we select the area of Module A on which to perform an exhaustive injection campaign, so that all the essential bits involved in the configuration memory of the FPGA are tested. For each of the tests, we performed 2,935,196 injections on a Zynq-7000 SoC running the compressor in real time. Results are summarized in Tables II, III, and IV. It is important to notice that, for all the experiments, module A implements the unprotected version of the compressor and the protected architectures that include methods A and B. Finally, we should remark that the objective of the technique is to detect errors in the hardware that implements the compression algorithm, not in the images. The tests are performed with real images just to show that regardless of the input of the sensor, the detection capability is similar to DMR, but

TABLE II
Results for the Fault Injection Campaign of Image 1: Cuprite Valley, Nevada. Compressed with Max Absolute Error Set to 1024, and Max Relative Error Set to 4096

Total injections	2,935,196	100%
Errors at the output (compressed) image	431,051	14.6856%
Detected errors method A	429,765	99.7017%
Undetected errors method A	1,286	0.2983%
Detected errors method B	429,762	99.7010%
Undetected errors method B	1,289	0.2990%

TABLE III
Results for the Fault Injection Campaign of Image 2: Suwannee Wildlife Area, Florida. Compressed With Max Absolute Error Set to 1, and Max Relative Error Set to 0

Total injections	2,935,196	100%
Errors at the output (compressed) image	2,166,356	73.8062%
Detected errors method A	2,166,172	99.9915%
Undetected errors method A	184	0.0085%
Detected errors method B	2,166,172	99.9915%
Undetected errors method B	184	0.0085%

TABLE IV
Results for the Fault Injection Campaign of Image 3: Cuprite Valley, Nevada. Compressed With Max Absolute Error Set to 16, and Max Relative Error Set to 16

Total injections	2,935,196	100%
Errors at the output (compressed) image	2,507,985	85.4452%
Detected errors method A	2,507,972	99.9995%
Undetected errors method A	13	0.0005%
Detected errors method B	2,507,972	99.9995%
Undetected errors method B	13	0.0005%

with lower complexity. The data generated by the sensor would be stored in memory resources protected by the ECCs included in the FPGA device.

The first important conclusion, which is not related to the protection method, is that although the compressor implemented is the same in the three cases, the number of errors in the compressed images is linked to the input image and the level of compression (loss). It can be seen how the image with a higher degree of compression (image 1, compression of 100 to 1, from Table II) only suffers errors in the compressed image in 14.69% of the cases[1], and for the image with the lowest degree of compression (image 3, compression of 4 to 1, from Table IV) the impact of faults is larger, 85.45% of the injections produce errors at the compressed output. An intermediate case with an image that has a compression of 18 to 1 (image 2 and Table III) produces errors in the compressed images in 73.81% of the injections.

This means that not all the faults in the compressor are going to have a direct effect on the output image. This is something relevant from the protection side, as it means that not all the images would require the same number of reconfigurations of the FPGA if techniques such as DMR are implemented, but at the same time, it means that with these DMR techniques, faults may be unnoticed, and the probability of accumulating errors would increase.

These deviations may come from the fact that, with a higher level of compression, inner functions such as the residual mapping stop being surjective. In fact, the higher the compression, the more input values map to the same outputs. As an example, when generating the quantization intervals, errors as high as half of the quantization bin size might be unnoticed since they will all map to the same bin center.

On the other hand, as the detection always uses the same pattern to verify the correct behavior of the compressor, the difference between images and compression parameters does not have significant changes. For example, for image 1, 2 and 3 there is 99.701%, 99.992% and 99.999% of errors detected, respectively.

In addition, there is a negligible difference between using the whole pattern (method A) or only the last frame for the detection (method B) of error (only 3 injections out of 2,935,196), concluding that the test of the last frame is enough, and the extra area and time required to check the whole compressed pattern is not worth the extra cost. Moreover, the percentage of detection compared to DMR is very similar (99.67% reported in [20]), but with less area as we will discuss in the next subsection. The main difference is that the pattern image also detects the faults in the configuration memory, even when there is no error on the compressed images, preventing the design from accumulating errors due to masking, but with more reconfigurations than DMR.

B. Area and Power Analysis

Area and power estimations have been obtained for the XQRKU060 FPGA [41] which is considered a reference for spaceborne missions [42]. The core is synthesized for a maximum target image size of 4096 frames by 640 pixels per frame by 512 bands per pixel. Table V shows the area and power results for the compression core for the unprotected version, the protected methods A and B and the classical DMR.

TABLE V
AREA AND POWER RESULTS FOR DIFFERENT CORE PROTECTION METHODS

	None	Method A	Method B	DMR
Power(W)	0.531	0.761 (43.31%)	0.660 (24.29%)	1.027 (93.41%)
Max MHz	200.88	187.23 (-6.80%)	197.27 (-1.79%)	197.39 (-1.74%)
LUTs	16,152	16,293 (0.87%)	16,277 (0.77%)	32,310 (100.04%)
REGs	15,337	15,519 (1.19%)	15,477 (0.91%)	30,747 (100.48%)
CLBs	3,810	4,192 (10.03%)	4,043 (6.12%)	7,681 (101.60%)
BRAMs	443	475 (7.22%)	459 (3.61%)	886 (100%)
DSPs	30	30 (0%)	30 (0%)	60 (100%)

Two main observations can be done:

- First, including the unprotected design and full protections with method A and B, the core mainly increases power consumption, though it stays under the budget of 1W. Compared to the DMR version, where the power consumption increases by 93% with respect to the unprotected version, the proposed methods require between 1.34 and 1.55 times less power obtaining and error detection of more than 99.6%. This is due to the fact that the proposed methods process a small synthetic image, which is the golden model, to detect the errors, while DMR needs to process the input image from the sensor twice, increasing the dynamic power consumption between 100 and 2,000 times. For DMR the maximum frequency can be reduced to ensure that it is within the power budget. However, for some sensors, the timing overhead will be non-negligible as we will discuss in the next section. On the other hand, the same reduction in frequency can be applied to methods A and B to decrease power consumption even more.
- Second, the increase in area resources for the methods A and B is around 1% in LUTs and registers, and less than 11% in CLBs and BRAMs. DSP resources for our proposals are the same as the ones implemented in the unprotected version. In contrast, DMR has an increase of area larger than 100% for LUTs, registers and CLBs, and 100% for BRAMs and DSPs, which increases the area of the architecture and hence the cross section of the design and the probability of suffering a fault in the final application.

C. Timing Analysis

The critical path of the protected versions is increased in all cases compared to the unprotected one, lowering the maximum frequency achievable (Table IV).

The largest degradation is introduced in method A, where the maximum frequency is reduced by 6.80%. On the other hand, method B and DMR suffer similar reductions, about 1.7% less. One of the main advantages of DMR is that it does not add any extra latency unlike our proposed solutions, which require extra clock cycles to process the pattern and check if there are errors in the architecture. It is also able to detect both hard and soft

[1]This percentage is computed considering the number of deviations of the compressed image in Module B, by means of the *check-failed* and *check-finished* signals and the total number of fault injections.

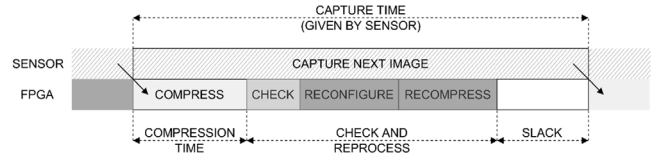

Fig. 7. Timing diagram (dark grey steps are executed only if the check step finds an error).

TABLE VI
TIMING ANALYSIS OF DIFFERENT SENSORS FOR THE PROPOSED SOLUTION. ESTIMATES BASED ON [43]* AND [1]**

SENSOR	Image Size	Speed	Capture
AVIRIS	224 × 680 × 512	1.827 MS/s	42,666.67 ms
AVIRIS-NG	480 × 640 × 512	37.5 MS/s	4,194.304 ms
NACHOS**	350 × 350 × 2000	12.25 MS/s	20,000 ms
CHIME*	420 × 1,024 × 2,048	125 MS/s	7,046.43 ms

SENSOR	Compressor (× 2)	Self-check	Reconfigure	Slack (abs)
AVIRIS	390.20 ms (× 2)	2.2 ms	1,016 ms	40,868.06 ms
AVIRIS-NG	1,045.64 ms (× 2)	2.2 ms	1,016 ms	1,084.82 ms
NACHOS**	1,225 ms (× 2)	2.2 ms	1016 ms	16,531.8 ms
CHIME*	4,404.91 ms (× 2)	2.2 ms	1,016 ms	-2,781.59 ms

errors on the fly when the DMR output does not match. However, the problems with DMR are two: i) it does not detect accumulated faults, and ii) the power consumption exceeds the budget of some small platforms such as CubeSats (it is almost twice the power consumption of methods A and B, see Table IV). For this reason, the proposed solution seems to be more appropriate for integration in low-cost environments, but the latency constraints need to be met. To verify this, we need to ensure that our proposal can: i) compress the image that we want to transmit, ii) run the test pattern (in method A or B), iii) do the shelf check for errors explained in the previous sections, iv) if errors are detected: reconfigure the FPGA, and compress the image stored in the buffer again. This has to be done in the capture time of the sensor (Fig. 7), assuming that: i) the sensor operates continuously and ii) we use a ping-pong buffer to temporarily store images in case they need to be reprocessed. Optionally, the buffers could be removed completely if the image could be recaptured in the event of failure, or if the data loss of one image is not significant.

The unprotected architecture runs at 3.2Gbps at 200MHz in a XQRKU060 FPGA device. The whole detection method takes 2.2ms for method B to test this architecture. In addition, the slowest process to perform reconfiguration is of 100Mbps, and the length of the .BIT file is 12.7MB, which requires about 1s. Table VI shows, for different sensors[2], the image size, sensor speed, and timing results about capture, compression, self-check with method B and configuration times assuming a 200MHz clock. We can see how for the AVIRIS, AVIRIS-NG and NACHOS sensors, we have enough time to perform the whole process, so we can run everything in real-time, meet power constraints and protect from accumulated errors in more than 99.7% of the cases. After the process of Compression, self-check and reconfiguration, there is a gap of 41,258.26ms, 2,130.5ms and 17,756.8ms for the AVIRIS, AVIRIS-NG and NACHOS sensors, and only 390.2ms, 1,045.6ms and 1,225ms are required respectively for compression. In other words, if an error is produced, we have enough time to correct it and perform the compression again (avoiding the loss of the information from the sensor). This can be seen as positive slack in the last column.

The main limitation is found in the CHIME Sensor (dimensions and speed are estimates based on [43]), in which the system is not fast enough, and we only have time to compress a part of the image in the event of reconfiguration. Either real-time is sacrificed, or a certain percentage of data loss is assumed with the current design. Other techniques based on high-speed scrubbing [44] of the configuration memory would take longer time (about 1,800ms more in a worst-case scenario with MBUs), which would not allow a new compression of the image for AVIRIS-NG sensors. Even in the best scenario, with a small number of errors, scrubbing would require at least between 8 and 30ms [45], which is between 3.6 and 13.6 times more latency compared to our proposed solution, also depending on the number of errors, while in method A and B latency is constant and the performance is not limited for MBUs. In addition, another advantage of the suggested techniques is that they are active, so they just perform the reconfiguration/correction when an error is detected during the computation. Therefore, no interaction with the configuration memory is required if no error is detected, while scrubbing needs to check the whole memory to ensure if there is an error or not, increasing power consumption.

Although the suggested technique is robust and can be used as a standalone solution, with less than 0.3% undetected errors, note that the schedule proposed in Fig. 7 can also be combined with scrubbing techniques with longer periods to reduce unnecessary memory access and power consumption and reduce the percentage of undetected errors. For the rest of the sensors in which the proposed method is fast enough and there is extra time, the maximum frequency of the design can be reduced to save power.

Note that the timing analysis for method A is omitted as it requires more clock cycles than method B and the fault detection is almost the same (a difference of less than 0.0007%, see Table I) but with 1.15 times extra power consumption and larger area requirements.

D. Comparison With Scrubbing

In reconfigurable systems, scrubbing is a common and widely used technique to protect against radiation-induced errors that affect the configuration memory of FPGAs. Scrubbing periodically detects and corrects errors by reloading the correct configuration, ensuring that the system continues to operate reliably. Given the widespread use of this method, it is important to compare scrubbing with our proposed protection technique. The following analysis evaluates the efficiency of both methods, showing when our technique is preferable to scrubbing.

Our calculations show two scenarios. For higher SEU arrival rates, scrubbing the system after each image may be more efficient, as it does not require a full board reconfiguration (which is

[2]We include compression and recompression in the same column (Compressor (×2)) following the schedule suggested in Fig. 7.

the most time-consuming operation) after each SEU. However, for lower SEU arrival rates, our proposed method would be preferable since the number of reconfigurations would be fewer, and checking the entire device is faster than with scrubbing techniques.

Let I be the capture time of an image in seconds. Define $N_I = (24 \cdot 3600)/I$ as the number of images captured per day. Let T_c represent the time taken to perform our protection technique, and T_s the time taken to scrub the device (including detection and correction). Let T_r be the reconfiguration time, and T_p be the processing time for a full image. Let f represent the error frequency (in errors/bit-day). Define S_c as the number of critical configuration bits in our protection technique, and S_s as the number of configuration bits in the scrubbing method.

The expected time spent on checking and correcting errors is:

$$T_{proposal} = N_I \cdot T_c + (T_r + T_p) \cdot f \cdot S_c$$

$$T_{scrub} = N_I \cdot T_s + (T_p) \cdot f \cdot S_s$$

The first term in both equations represents the total time spent running the protection techniques on the images. Specifically, in the first equation, it is the number of images multiplied by the time required by our technique, and in the second equation, it is the number of images multiplied by the device scrubbing time. This term depends solely on the number of images captured per day and is independent of the number of errors that occur. The second term, on the other hand, varies with the number of errors, as it represents the time needed to correct the errors that occur with both techniques. Therefore, this term is weighted by the error frequency, f. Note that the scrubbing method does not need to reconfigure the device in the event of an error, as the check itself is self-repairing. Additionally, scrubbing must handle the entire set of configuration bits in the device, whereas our technique is only affected by the subset of critical bits (those which have any effect on the design).

Now let us assume that the AVIRIS-NG sensor is used, with $I = 4.194$, along with the XQRKU060 running the core at 200Mhz. Running times under this configuration are $T_c = 2.2$ms, 8ms $< T_s <$ 30ms, $T_r <$ 1016ms and $T_p = 1046$ms. The maximum number of critical bits found in our experiments was $S_c = 2{,}507{,}972$ out of $S_s = 2{,}935{,}196$. Using these values, we have that $T_{proposal} = T_{scrub} = 339.37$ when $f \approx 5.686 \cdot 10^{-5}$ errors/bit-day. Repeating the calculations for the NACHOS sensor yields $T_{proposal} = T_{scrub} = 79.05$ when $f \approx 1.237 \cdot 10^{-5}$ errors/bit-day.

This means that for SEU arrival rates higher than this threshold, the scrubbing approach would be better, and for SEU arrival rates lower than that, our proposal would be more efficient. A typical on-orbit SEU error rate ranges from 10^{-5} errors/bit-day in commercial devices [46], down to 10^{-7} to 10^{-8} errors/bit-day in rad-tolerant devices [47] (such as the XQRKU060) and with even lower rates at 10^{-10} to 10^{-12} errors/bit-day in rad-hard devices [48]. These figures can go below 10^{-15} errors/bit-day in non-orbital scenarios [49]. Although these typical values are usually refined for specific missions using tools like SPENVIS [50], the calculated values should be within the same order of magnitude.

All the mentioned frequencies are below the threshold that has been calculated. This means that our proposed technique is more favorable, as it requires less total processing time. It is important to note that these calculations assume the best-case scenario for scrubbing and the worst reconfiguration times and number of critical bits for our technique, suggesting that, with more refined values, the proposed method would be even more efficient.

Finally, to verify the consistency of the results on other FPGAs, the previous calculations were repeated for two additional boards (the Zynq-7000 and the Ultrascale+). After these calculations, the obtained break-even error frequencies, f, for the AVIRIS-NG sensor are as follows: $29.34 \cdot 10^{-5}$ for the Zynq-7000 and $19.74 \cdot 10^{-5}$ for the Ultrascale+. And for the NACHOS sensor, the corresponding frequencies are $6.15 \cdot 10^{-5}$ and $4.14 \cdot 10^{-5}$, respectively.

It can be observed that these frequencies are of the same order of magnitude as those obtained for the first FPGA, and therefore the conclusions regarding the advantages of using our technique versus scrubbing remain unchanged.

VI. Conclusion

Compression algorithms are a necessary piece of the hyperspectral puzzle. Their applicability in imaging satellites, implemented on FPGAs, require protection techniques to be put in place, to avoid data corruption.

A novel fault-tolerance technique is proposed in this paper for the CCSDS 123.0-B-2 hyperspectral image compression algorithm. Instead of the traditional approaches of replicating the implementation (or parts of it) to detect errors, our technique is based on output checking.

A golden reference is periodically input to the compressor, while checking the output for errors against a reference vector. If an SEU is present within the FPGA's configuration memory, an error will arise at some point during the computation, cascading down to the output and corrupting the result from that point forward. The checker will detect the difference and issue a reconfiguration process.

Results show this technique to be comparable in detection accuracy to that of DMR, at a fraction of the resources and power cost. In a real-time scenario, the detection speed is enough to issue and perform a reconfiguration before the next image is available for most sensors.

This technique might also be useful in other scenarios where an error in the algorithm propagates indefinitely until processing terminates.

References

[1] S. P. Love et al., "NACHOS, a CubeSat-based high-resolution UV-Visible hyperspectral imager for remote sensing of trace gases: System overview and science objectives," *CubeSats SmallSats Remote Sens. V*, vol. 11832, SPIE, 2021, pp. 85–99.

[2] S. K. Meerdink, D. A. Roberts, K. L. Roth, J. Y. King, P. D. Gader, and A. Koltunov, "Classifying California plant species temporally using airborne hyperspectral imagery," *Remote Sens. Environ.*, vol. 232, 2019, Art. no. 111308.

[3] X. Zhang, Y. Sun, K. Shang, L. Zhang, and S. Wang, "Crop classification based on feature band set construction and object-oriented approach using

hyperspectral images," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 9, pp. 4117–4128, Sep. 2016.
[4] S. Yang and Z. Shi, "Hyperspectral image target detection improvement based on total variation," *IEEE Trans. Image Process.*, vol. 25, no. 5, pp. 2249–2258, May 2016.
[5] M. K. Tripathi and H. Govil, "Evaluation of AVIRIS-NG hyperspectral images for mineral identification and mapping," *Heliyon*, vol. 5, no. 11, 2019, Art. no. e02931.
[6] M. J. Khan, H. S. Khan, A. Yousaf, K. Khurshid, and A. Abbas, "Modern trends in hyperspectral image analysis: A review," *IEEE Access*, vol. 6, pp. 14118–14129, 2018.
[7] J. W. Chapman et al., "Spectral and radiometric calibration of the next generation airborne visible infrared spectrometer (AVIRIS-NG)," *Remote Sens.*, vol. 11, no. 18, 2019, Art. no. 2129.
[8] E. Christophe, C. Mailhes, and P. Duhamel, "Hyperspectral image compression: Adapting SPIHT and EZW to anisotropic 3-D wavelet coding," *IEEE Trans. Image Process.*, vol. 17, no. 12, pp. 2334–2346, Dec. 2008.
[9] Q. Du and J. E. Fowler, "Hyperspectral image compression using JPEG2000 and principal component analysis," *IEEE Geosci. Remote Sens. Lett.*, vol. 4, no. 2, pp. 201–205, Apr. 2007.
[10] "Low Complexity Lossless and Near-Lossless Multispectral & Hyperspectral Image Compression (CCSDS 123.0-B-2)," The Consultative Committee for Space Data Systems (CCSDS), 2019. Accessed: Oct. 3, 2023. [Online]. Available: https://public.ccsds.org/Pubs/123x0b2c3.pdf
[11] M. Hernandez-Cabronero et al., "The CCSDS 123.0-B-2 "Low-complexity lossless and near-lossless multispectral and hyperspectral image compression" standard: A comprehensive review," *IEEE Geosci. Remote Sens. Mag.*, vol. 9, no. 4, pp. 102–119, Dec. 2021.
[12] J. Fjeldtvedt, M. Orlandic, and T. A. Johansen, "An efficient real-time FPGA implementation of the CCSDS-123 compression standard for hyperspectral images," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 10, pp. 3841–3852, Oct. 2018.
[13] M. Orlandic, J. Fjeldtvedt, and T. A. Johansen, "A parallel FPGA implementation of the CCSDS-123 compression algorithm," *Remote Sens.*, vol. 11, no. 6, 2019, Art. no. 673.
[14] Y. Barrios, A. J. Sánchez, L. Santos, and R. Sarmiento, "SHyLoC 2.0: A versatile hardware solution for on-board data and hyperspectral image compression on future space missions," *IEEE Access*, vol. 8, pp. 54269–54287, 2020.
[15] A. Tsigkanos, N. Kranitis, D. Theodoropoulos, and A. Paschalis, "High performance COTS FPGA SoC for parallel hyperspectral image compression with CCSDS-123.0-B-1," *IEEE Trans. Very Large Scale Integr.*, vol. 28, no. 11, pp. 2397–2409, Nov. 2020.
[16] D. Báscones, C. Gonzalez, and D. Mozos, "A real-time FPGA implementation of the CCSDS 123.0-B-2 standard," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, pp. 1–13, 2022.
[17] O. Ferraz, V. Silva and G. Falcao, "Hyperspectral parallel image compression on edge GPUs," *Remote Sens.*, vol. 13, no. 6, 2021, Art. no. 1077.
[18] R. Roosta, "A comparison of radiation-hard and radiation-tolerant FPGAs for space applications," *NASA Electron. Parts and Packag. Program*, vol. 30, 2004.
[19] P. E. Dodd and L. W. Massengill, "Basic mechanisms and modeling of single vent upset in digital microelectronics," *IEEE Trans. Nucl. Sci.*, vol. 50, no. 3, pp. 583–602, Jun. 2003.
[20] L. A. Aranda, A. Sánchez, F. Garcia-Herrero, Y. Barrios, R. Sarmiento, and J. A. Maestro, "Reliability analysis of the SHyLoC CCSDS123 IP core for lossless hyperspectral image compression using COTS FPGAs," *Electronics*, vol. 9, no. 10, 2020, Art. no. 1681.
[21] R. E. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," *IBM J. Res. Dev.*, vol. 6, no. 2, pp. 200–209, 1962.
[22] K. W. Gear, A. Sánchez-Macián, and J. A. Maestro, "Reduced length redundancy adaptive protection for the cascaded integrator-comb interpolation filter on FPGA," *Microelectron. Reliab.*, vol. 118, 2021, Art. no. 114043.
[23] L. A. Aranda, A. Sánchez-Macián, and J. A. Maestro, "An algorithmic-based fault detection technique for the 1-D discrete cosine transform," *IEEE Trans. Very Large Scale Integr.*, vol. 28, no. 5, pp. 1336–1340, May 2020.
[24] K. W. Gear, A. Sánchez-Macián, and J. A. Maestro, "An analysis of FPGA configuration memory SEU accumulation and a preventative scrubbing technique," *Microprocess. Microsyst.*, vol. 90, 2022, Art. no. 104467.
[25] "Corpus of hyperspectral and multispectral images," The Consultative Committee for Space Data Systems. Accessed: May 13, 2022. [Online]. Available: http://cwe.ccsds.org/sls/docs/sls-dc/123.0-B-Info/TestData
[26] S. Liu et al., "A survey on CubeSat missions and their antenna designs," *Electronics*, vol. 11, no. 13, 2022, Art. no. 2021.
[27] A. Tadanki and E. G. Lightsey, "Closing the power budget architecture for a 1U CubeSat framework", Georgia Institute of Technology. Accessed: Nov. 27, 2024. [Online]. Available: https://hdl.handle.net/1853/73553
[28] O. Popescu, "Power budgets for CubeSat radios to support ground communications and inter-satellite links," *IEEE Access*, vol. 5, pp. 12618–12625, 2017.
[29] A. Agnesina et al., "A COTS-based novel 3-D DRAM memory cube architecture for space applications," *IEEE Trans. Very Large Scale Integr.*, vol. 28, no. 9, pp. 2055–2068, Sep. 2020. doi:10.1109/TVLSI.2020.3002211.
[30] R. Hadidi, B. Asgari, B. A. Mudassar, S. Mukhopadhyay, S. Yalamanchili, and H. Kim, "Demystifying the characteristics of 3D-stacked memories: A case study for hybrid memory cube," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, 2017, pp. 66–75, doi:10.1109/IISWC.2017.8167757.
[31] H. Jun et al., "HBM (high bandwidth memory) DRAM technology and architecture," in *Proc. IEEE Int. Memory Workshop (IMW)*, 2017, pp. 1–4, doi:10.1109/IMW.2017.7939084.
[32] CCSDS Standards Development. Accessed: May 5, 2022. [Online]. Available: https://public.ccsds.org/Publications/default.aspx
[33] S. Arunkumar and T. Kalaivani, "FPGA implementation of CCSDS BCH (63, 56) for satellite communication," in *Proc. IEEE Int. Conf. Electron. Des., Syst. Appl. (ICEDSA)*, 2012, pp. 248–253, doi:10.1109/ICEDSA.2012.6507808.
[34] A. Álvarez, V. Fernández, and B. Matuz, "An efficient NB-LDPC decoder architecture for space telecommand links," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 68, no. 4, pp. 1213–1217, Apr. 2021, doi:10.1109/TCSII.2020.3034392.
[35] "Low-complexity lossless and near-lossless multispectral and hyperspectral image compression: Informational report (CCSDS 120.2-G-2)," Consultative Committee for Space Data Systems (CCSDS), 2022. [Online]. Available: https://public.ccsds.org/Pubs/120x2g2.pdf
[36] L. A. Aranda, A. Sánchez-Macián, and J. A. Maestro, "ACME: A tool to improve configuration memory fault injection in SRAM-based FPGAs," *IEEE Access*, vol. 7, pp. 128153–128161, 2019, doi:10.1109/ACCESS.2019.2939858.
[37] *Soft Error Mitigation Controller V4.1*. Xilinx, San Jose, CA, USA, 2018.
[38] "7 Series and SEM IP—How to use the SEM IP error report to look up bit error locations using essential bit data in an EBD file," Xilinx, 2016. Accessed: Nov. 27, 2024. [Online]. Available: https://www.xilinx.com/support/answers/67337.html
[39] UltraScale and SEM IP, "How to use the SEM IP error report to look up bit error locations using essential bit data in the EBD file" Xilinx, 2016. Accessed: Nov. 27, 2024. [Online]. Available: https://www.xilinx.com/support/answers/67086.html
[40] L. A. Aranda, O. Ruano, F. Garcia-Herrero, and J. A. Maestro, "ACME2: Improving the extraction of essential bits in Xilinx SRAM-based FPGAs," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 3, pp. 1577–1581, Mar. 2022, doi:10.1109/TCSII.2021.3105558.
[41] "RT Kintex UltraScale FPGAs for ultra high throughput and high bandwidth applications", Xilinx. Accessed: Oct. 3, 2023. [Online]. Available: https://www.xilinx.com/content/dam/xilinx/support/documents/white_papers/wp523-xqrku060.pdf
[42] P. Chatziantoniou, A. Tsigkanos, and N. Kranitis, "A high-performance RTL implementation of the CCSDS-123.0-B-2 hybrid entropy coder on a space-grade SRAM FPGA," in *Proc. 7th Int. Workshop On-Board Payload Data Compression (OBPDC), Online Event*, 2020, pp. 21–23.
[43] Y. Barrios, P. Rodríguez, A. Sánchez, M. González, L. Berrojo, and R. Sarmiento, "Implementation of cloud detection and processing algorithms and CCSDS-compliant hyperspectral image compression for CHIME mission," in *Proc. 7th Int. Workshop On-Board Payload Data Compression (OBPDC), Online Event*, 2020, pp. 21–23.
[44] T. Fujimori and M. Watanabe, "High-speed scrubbing demonstration using an optically reconfigurable gate array," *Opt. Exp.*, vol. 25, no. 7, pp. 7807–7817, 2017.
[45] A. Stoddard, A. Gruwell, P. Zabriskie, M. Wirthlin, and J. Michael, "A hybrid approach to FPGA configuration scrubbing," *IEEE Trans. Nucl. Sci.*, vol. 64, no. 1, pp. 497–503, Jan. 2017.

[46] BYU ScholarsArchive. Accessed: Nov. 27, 2024. [Online]. Available: https://scholarsarchive.byu.edu/cgi/viewcontent.cgi?article=2306&context=facpub
[47] Xilinx. Accessed: Nov. 27, 2024. [Online]. Available: https://www.xilinx.com/content/dam/xilinx/publications/presentations/rtkintex-press-presentation.pdf
[48] NASA LLIS. Accessed: Nov. 27, 2024. [Online]. Available: https://llis.nasa.gov/lesson/824
[49] Xilinx. Accessed: Nov. 27, 2024. [Online]. Available: https://docs.xilinx.com/r/en-US/ug116/SEU-and-Soft-Error-Rate-Measurements
[50] SPENVIS. Accessed: Nov. 27, 2024. [Online]. Available: https://www.spenvis.oma.be/

Daniel Báscones received the bachelor's degree in both mathematics and computer science and the M.Sc. degree in computer science from the Complutense University of Madrid, Madrid, Spain, in 2016 and 2018, respectively. He was a Research Associate during the time of his M.Sc. with the Department of Computer Architecture and Automatics. His research interests include hyperspectral image compression on field-programmable gate arrays, dealing with fast lossless algorithms that aid with data transmission and more complex lossy algorithms for long-term storage, a field in which he obtained his Ph.D. thesis in 2020.

Francisco García-Herrero received the B.Sc. degree in telecommunication engineering from the Escuela Politecnica Superior de Gandía, Spain, in 2008, and the M.S. and Ph.D. degrees in electrical engineering from the Universitat Politecnica de Valencia, Spain, in 2010 and 2013, respectively. He has worked as a Lecturer and a Researcher at several universities, including the European University Miguel de Cervantes, the Universitat Politecnica de Valencia, and the Universidad Antonio de Nebrija. Currently, he is an Associate Professor and a Researcher with the Universidad Complutense de Madrid. His research interests include hardware and algorithmic optimizations of error-control decoders and fault-tolerance electronics in communication and storage systems.

Óscar Ruano received the M.Sc. and Ph.D. degrees in computer engineering from the Universidad Antonio de Nebrija, in 2005 and 2011, respectively. He has served as a Lecturer and a Researcher with the Universidad Nebrija with a research contract supported by the Regional Government of Madrid. Currently, he is working for the Universidad Complutense de Madrid. He has developed his activity in the Space field, with different research projects on fault tolerance optimization against radiation effects in microelectronic circuits. He is the Author of several technical publications, both in journals and international conferences. Also, he has worked with different multinational companies in the IT consultancy field, as Accenture. His research interests include digital design, fault-tolerance, and reliability.

Carlos González Calvo received the M.S. and Ph.D. degrees in computer engineering from the Complutense University of Madrid, Madrid, Spain, in 2008 and 2011, respectively. Currently, he is an Associate Professor with the Department of Computer Architecture and Automation, Complutense University of Madrid, where he is also a Research Member of the GHADIR Group, where he mainly focuses on applying runtime reconfiguration in aerospace applications. His research interests include remotely sensed hyperspectral imaging, signal and image processing, efficient implementation of large-scale scientific problems on reconfigurable hardware, and the acceleration of artificial intelligence algorithms applied to games. He attended the Design Competition of the 2009 and 2010 IEEE International Conference on Field-Programmable Technology and received the second prize in 2012. He received the Best Paper Award of an Engineer under 35-years old at the 2011 International Conference on Space Technology.

Daniel Mozos (Member, IEEE) received the B.S. degree in physics and the Ph.D. degree in computer science from the Complutense University of Madrid, Madrid, Spain, in 1986 and 1992, respectively. He was the Dean of the Computer Science Faculty, Complutense University of Madrid, from 2010 to 2018, where he is currently a Full Professor with the Department of Computer Architecture and Automation, Complutense University of Madrid, and leads the GHADIR Research Group on dynamically reconfigurable architectures. His research interests include design automation, computer architecture, and reconfigurable computing.

Juan Antonio Maestro (Senior Member, IEEE) received the M.Sc. degree in physics and the Ph.D. degree in computer engineering from the Universidad Complutense de Madrid, Madrid, Spain, in 1994 and 1999, respectively. Currently, he is a Full Professor with the Universidad Complutense de Madrid, in the computer architecture field. His current activities are oriented to the space industry, with several projects on the protection of digital circuits against the effects of radiation, including microprocessors, memories and auxiliary systems. He also collaborates with institutions as the European Space Agency, Stanford University, University College Dublin or the Harbin Institute of Technology, among others. He is the Author of numerous technical publications in journals and international conferences. His research interests include computer architecture, digital design, fault tolerance, reliability, small satellites, and space applications.