Reliability Analysis of a Low-Cost CCSDS 123 Hyperspectral Image Compressor

Wesley Grignani, Douglas A Santos, Luigi Dilillo, Douglas R Melo

Title: Reliability Analysis of a Low-Cost CCSDS 123 Hyperspectral Image Compressor

Author(s): W. Grignani, D. A. Santos, L. Dilillo, D. R. Melo

Document version: Pre-print version (Final draft)

Please cite the original version:
W. Grignani, D. A. Santos, L. Dilillo, D. R. Melo, "Reliability Analysis of a Low-Cost CCSDS 123 Hyperspectral Image Compressor, "2024 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2024.

Reliability Analysis of a Low-Cost CCSDS 123 Hyperspectral Image Compressor

Wesley Grignani*, Douglas A. Santos†, Luigi Dilillo†, and Douglas R. Melo*
*LEDS, University of Vale do Itajaí, Itajaí, Brazil
†IES, University of Montpellier, CNRS, Montpellier, France
wesley.grignani@edu.univali.br, {douglas.santos, luigi.dilillo}@umontpellier.fr, drm@univali.br

Abstract

Remote sensing techniques in space applications use images that provide a large volume of information to collect data about the Earth. Hyperspectral images (HSIs) present a high volume of data and affect restrictions on the storage capacity and processing in space systems, highlighting the demand for compression. Systems that operate in space are susceptible to faults due to adverse conditions and require the implementation of protection techniques to mitigate these faults and ensure correct operation. This work implements a fault-tolerant CCSDS 123 HSI compressor using Triple Modular Redundancy (TMR) and Hamming Error Correcting Code (ECC) techniques to protect from Single Event Upsets (SEUs). We present the compressor in a standard version, followed by partially-protected and fully-protected versions, by applying hardening in different circuit components. We performed a reliability analysis through a fault injection campaign considering all implementations. The standard version showed 97.9% of executions with errors, which was reduced in the versions partially hardened, reaching no error propagation in the fully hardened version. In addition, all implementations accelerated the application, achieving a performance increase of up to 24× compared to the software solution.

Index Terms

Systems-on-Chip, Hardware Accelerator, Hyperspectral Images, CCSDS 123.0-B-2, Fault Tolerance.

I. Introduction

Remote sensing in space applications is a crucial technology that allows data about the Earth to be collected from satellites and space probes. Among these techniques is hyperspectral imaging. Hyperspectral images (HSIs) capture information in hundreds of spectral bands, allowing the detection of very specific characteristics of the materials observed. This proves valuable for applications such as Earth image acquisition, climate analysis, and forest environment surveillance. Due to their capability of capturing several spectral bands, HSIs present a large volume of data [1].

The high number of bands in HSIs makes it a data-intensive processing structure for space systems, impacting ground station storage, processing, and transmission capabilities. To handle the high data volume of HSIs and enhance transmission and processing efficiency, the Consultative Committee for Space Data Systems (CCSDS) developed the CCSDS 123 compression algorithm [2]. This algorithm is designed for low-complexity hardware implementation, offering lossless and near-lossless compression in its latest release (B-2).

Some works implemented hardware accelerators for HSI compression in FPGA (Field Programmable Gate Array). The work [3] presents a low-cost solution for the prediction step of the CCSDS standard. A high-performance solution is presented in [4], while the works [5] and [6] focused on fault-tolerant solutions using SG (Space Grade) FPGAs. The work [7] focused on a reliability analysis duplicating and triplicating the accelerator developed in [6], using EDAC (Error Detection and Correction) in some internal memories.

In this work, we extend our previous project [8], focusing on implementing a fault-tolerant CCSDS 123 hardware accelerator version. We designed the accelerator in different configurations by combining spatial and information redundancies to improve its reliability. We present the accelerator in a standard version, followed by partially-protected and fully-protected versions. We apply techniques such as TMR (Triple Modular Redundancy) to protect the control units and Hamming ECC (Error Correcting Code) to protect memory elements.

This work was supported in part by the Foundation for Support of Research and Innovation, Santa Catarina (FAPESC-2021TR001907), the Brazilian National Council for Scientific and Technological Development (CNPq - process 50794/2023-5), the Brazilian National Coordination of Superior Level Staff Improvement (CAPES/PROSUC), the EU project RADNEXT - Horizon 2020 (grant 101008126), and Project HARV in the framework of the action ”Accelerateur d’innovation” from the University of Montpellier.

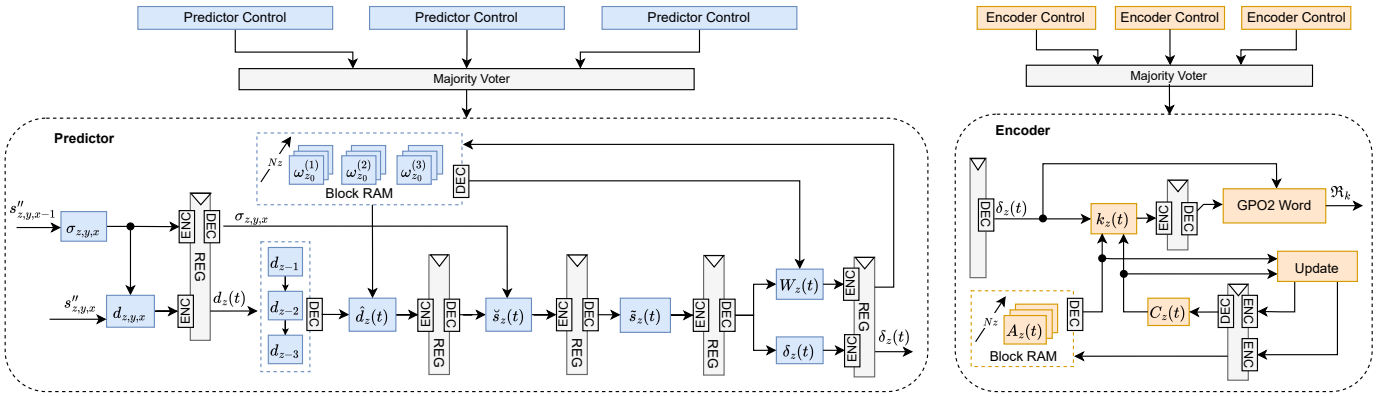

Fig. 1. Hardened compressor design overview.

II. CCSDS 123 Compressor

In our previous project [8], we implemented the compressor in HDL (Hardware Description Language) and HLS (High-Level Synthesis) using the AXI4-Lite communication bus in a SoC (System-on-Chip) integration. In this work, we designed a fault-tolerant version of the accelerator, presented in Fig. 1. The architecture was improved by implementing a sample buffer inside the accelerator and the SoC communication architecture to use an AXI4-Stream bus with DMA (Direct Memory Access). These changes were made to eliminate the need for samples to be buffered outside the processor and to reduce the communication bottleneck in the previous system.

The fault-tolerant version was implemented to protect from SEUs (Single Event Upsets) affecting a single bit. This technique was applied considering the results in [9], which observed that most of the events in the experiments consisted of single-bit upsets in the memory elements. We designed the accelerator in different hardened configurations, where TMR and Hamming ECC were applied to different components. These configurations were compared in terms of cost, performance, and reliability using different HSI configurations.

The accelerator was designed to minimize resource usage. This includes the local sum $\sigma_{z,y,x}$ in column-oriented mode with BIP (Band Interleaved by Pixel) processing orders, allowing the vector size of local differences $U_z(t)$ to be based on the number of previous bands P_z, reducing resources compared to other processing orders. The size of weights vectors $W_z(t)$ is based on the number of Nz bands in the image.

The sample buffer is implemented considering the local sum mode and the processing order. As the local sum mode is column-oriented, the only value that needs to be stored is above the current sample. In addition, the BIP processing order goes through all the samples in that band and then moves on to the next band. Thus, the size of the sample buffering is based on the dimensions Nx and Nz of the image.

A. Hardware Implementation

The components were designed to work sequentially, and some steps have been parallelized to reduce the cycles needed to process a sample. As the encoder only needs the output of the predictor, it works in a pipeline with the predictor.

The first steps of the predictor are local sum and local difference, which are set to execute simultaneously. Next, the central prediction multiplies the values of the weights and local difference vector. The high-resolution prediction uses the values calculated in the central prediction and local sum step. The double-resolution prediction is next performed, and the mapped quantized index and weights update step are executed simultaneously at the final step.

The encoder starts by registering the information from the predictor. Next, the index $k_z(t)$ is calculated, defining the number of bits to encode the mapped residual. Next, is calculated the GPO2 $\Re_k(\delta_z(t))$ word, with its output formatted in the proper form of the standard. In the end, the accumulator $A_z(t)$ and counter $C_z(t)$ memories are updated.

We created the local difference vector using registers to allow reading all values in a single cycle. The weight vectors were created using block RAMs. As one sample of each band is processed at a time, the z index drives the weight vector selection. In the encoder, simultaneous access to all accumulator values is unnecessary, allowing the block RAMs to implement these vectors and reduce resource usage.

We used a Zedboard Zynq-7000 with an ARM Cortex-A9 to prototype the developed compressor. The DMA transfers data between the memory and the compressor and receives it back to store in memory. The AXI interconnect interfaces between master and slave signals, with one interconnect used for AXI4-Lite and the other for the AXI4-Stream Interface.

B. Fault Tolerance

We created 6 different configurations to harden different compressor components:

- **STD**: the accelerator without fault tolerance techniques.
- **TMR_CTRL**: TMR applied in the predictor and encoder controllers.
- **HAM_BUF**: protects the buffers with Hamming ECC. The buffers in this circuit represent the Weight, Accumulator, and BIP memories.
- **HAM_REG**: protects the registers with Hamming ECC. The registers store intermediate calculations in the predictor and the encoder.
- **HAM_BUF_REG**: both buffers and registers are protected using Hamming ECC.
- **FULL_FT**: represents the compressor protecting the registers and buffers with Hamming ECC and triplicating the predictor and encoder controllers.

The controllers were hardened using TMR with a simple bit-wise majority voter system. The Hamming ECC was used solely for error correction.

III. Reliability Evaluation

We created a simulation setup using an AVIRIS [10] image cropped in 4 different sizes. The size of the weight and accumulator vector are based on the Nz bands of the image, while the size of the BIP memory is based on the Nx and Nz size. Thus, for a reliability evaluation, we can analyze the impact of these memories changing the spectral resolution (Nz), the spatial resolution (Nx, Ny), and both spectral and spatial resolution.

This work used a simulation-based injection method to perform the fault injection campaign. The solution from [11] performs SEU injections into the registers and was customized to operate on the designed compressor.

The fault injection involves initially simulating without fault injections to obtain a golden run. The next stage involves listing all the registers in the circuit and randomly choosing one to perform the fault injection at a random time. After that, the simulation continues up to the total simulation time.

If the output of any external port differed from the golden run, it was assumed that the fault resulted in an error in the compressed image. For each of the image configurations, 1000 simulations were performed.

IV. Results

We used the Xilinx 2020.1 tool to collect the synthesis and performance results, targeting the Zynq-7020 FPGA. The fault injection campaign was performed by Modelsim simulations executed on a computer running a Linux operating system with an Intel Core i7-12700 processor and 16 GB of RAM.

A. Compressor Synthesis

Table I presents the results in terms of LUTs (Look-up-Tables), FFs (Flip-Flops), BRAMs (Block RAMs), and DSPs (Digital Signal Processors), along with the maximum operating frequency and estimated power dissipated for all compressor versions. The synthesis results considered the image ($128\times128\times40$) configuration.

TABLE I
Compressor Synthesis results for the 128x128x40 image.

Configuration	LUTs	FFs	DSPs	BRAMs (32Kb)	Fmax (MHz)	Power (mW)
STD	1,172	443	4	6.0	102.78	119
TMR_CTRL	1,201	469	4	6.0	102.32	123
HAM_BUF	1,294	443	4	8.0	80.14	138
HAM_REG	2,196	588	4	6.5	72.17	143
HAM_BUF_REG	2,370	588	4	8.0	67.98	151
FULL_FT	2,563	643	4	8.0	67.40	159

The STD version has the lowest resource utilization and highest frequency because it does not implement any redundancy. In the other configurations, we observed that resource usage increased as different protection configurations were applied, as well as a reduction in the maximum frequency. In the full hardened configuration (FULL_FT), the resources increased 118% in LUTs and 45% in FFs, and the maximum frequency decreased by 34% compared to the non-hardened configuration (STD).

B. Compressor Performance

Table II presents the performance results of each compressor version. The throughput consider the number of cycles to process a sample at the maximum operating frequency of the circuit. The difference in throughput is because the maximum frequency of the circuit decreases as the fault tolerance techniques are applied. Energy consumption also changes, reaching an increase of up to 2× compared to the STD version.

TABLE II
Performance results of compressor configurations.

Configuration	Throughput (MSa/s)	Energy[1] (mJ)
STD	20.55	5.80
TMR_CTRL	20.47	6.00
HAM_BUF	16.02	8.60
HAM_REG	14.44	9.90
HAM_BUF_REG	13.59	11.10
FULL_FT	13.48	11.79

[1] Estimated energy consumption to process 1MSa.

C. Compressor SoC

Table III presents the SoC synthesis results, where we observed that LUTs and FFs increase by an average of 2.6× and 7.6×, and the BRAMs increases by 1.33× compared to the isolated compressor. Also, the maximum frequency of the SoC system slightly decreases in all compressor configurations.

TABLE III
SoC Synthesis results for the 128x128x40 image.

Configuration	LUTs	FFs	DSPs	BRAMs (32Kb)	Fmax (MHz)	Power (W)
STD	3,763	3,890	4	8.0	100.58	1.701
TMR_CTRL	3,825	3,940	4	8.0	100.07	1.702
HAM_BUF	3,879	3,879	4	10.0	78.93	1.704
HAM_REG	4,767	4,035	4	8.5	71.11	1.711
HAM_BUF_REG	4,943	4,035	4	10.0	66.74	1.713
FULL_FT	5,072	4,130	4	10.0	66.21	1.717

The resource increase is explained by introducing the DMA in the system, which also adds interconnection components for the AXI4-Lite and AXI4-Stream interfaces, which communicate with the ARM processor.

To evaluate the acceleration of the system, we obtained the execution time of all versions of the compressor and compared it with the software solution running on the ARM processor at a frequency of 667 MHz. The results obtained showed an acceleration of 24× for the STD configuration. We observed an increase in the processing time of the other configurations, which leads to a 16× acceleration in the FULL_FT version.

D. Reliability Analysis

We performed 1000 simulations for each configuration of the compressor and image sizes, totaling 300 hours. Table IV presents the simulation results.

TABLE IV
Reliability results for different image sizes.

Nx,Ny,Nz	20,20,5	20,20,20	128,128,20	128,128,40
STD	705	884	970	979
TMR_CTRL	700	859	967	974
HAM_BUF	34	13	3	1
HAM_REG	499	788	928	965
HAM_BUF_REG	7	2	0	0
FULL_FT	0	0	0	0

The STD configuration showed 705 errors in the first image configuration, increasing to 974 in the last image configuration. We observed that most errors occurred in the BIP, weight, and accumulator memories. As the size

of the image increases, the size of these memories also increases, concentrating most of the fault injections in these elements.

A fault in the weight memory impacts compression directly since it is multiplied by the local difference vector. In BIP memory, a fault affects compression for a sample, as the local sum step utilizes the incorrect value. However, errors don't propagate to subsequent compressions since this memory position is overwritten with a new sample. This explains the elevated error count in the STD configuration.

In the TMR_CTRL version, the number of errors starts at 700 and increases to 974 in the final image configuration. Here, we see slightly fewer errors compared to the STD version due to the TMR applied to the controllers. As the controllers use few memory elements compared to the rest of the circuit, applying TMR only to the controllers is not sufficiently effective in reducing errors.

We observed a significant reduction in the number of errors in the HAM_BUF version. In the first image configuration, the number of errors is 34, representing a reduction of 20$\times$ compared to the STD version. The number of errors decreases in the other configurations, reaching only 1 error in the last image configuration. The HAM_REG version presented an error reduction of about 28% in the first image configuration compared to both STD and TMR_CTRL versions, but the errors increase as the image size increases. This reinforces the argument that these memories are subject to a higher incidence of faults as the image size increases.

The combined HAM_BUF_REG configuration shows some errors in the first image configuration, reducing to no errors in the last one. As this configuration protects both buffers and registers, the errors observed occurred in the controllers. They were reduced to 0 due to increased memories, reducing the incidence of faults in the controllers. The FULL_FT configuration was the only one that presented no errors in every image configuration since all the memorization components of the compressor are protected in this version.

E. Discussion

The techniques applied showed different results in the compressor configurations. It was also possible to notice that most faults occurred in the weight, accumulator, and BIP memories due to the high use of memorization elements compared to the rest of the circuit.

The different image configurations allowed for verifying the effectiveness of each protection configuration. In addition to the FULL_FT version, the partially hardened configurations, such as HAM_BUF and HAM_BUF_REG, proved to be effective as the image size increased and can be considered alternatives to a fully protected version.

Compared to the FULL_FT configuration, the HAM_BUF_REG shows a reduction of 7% in LUTs and 8% in FFs with almost the same throughput. On the other hand, the HAM_BUF configuration uses 49% less LUTs and 32% less FFs with higher throughput than the FULL_FT.

Among these results, the HAM_BUF version reduced errors as the image size increased despite not fully protecting the circuit. It could be an intermediate solution as a trade-off to maintain low resource utilization and high throughput.

V. Conclusion

In this work, we compared the performance and resource utilization of the different configurations. We also used different image sizes to perform a fault-injection campaign for a reliability analysis. The results show good resilience against SEUs, and all configurations accelerated the compression application compared to the software solution.

For future work, we plan to extend the fault injection campaign with different images and perform an experimental analysis in particle accelerators. We also plan to improve the performance of the compressor by implementing a pipelined architecture to increase the throughput and integrate it into a future multi-core system.

References

[1] F. Viel, R. C. Maciel, L. O. Seman, C. A. Zeferino, E. A. Bezerra, and V. R. Q. Leithardt, "Hyperspectral image classification: An analysis employing CNN, LSTM, Transformer, and Attention Mechanism," *IEEE Access*, vol. 11, pp. 24 835–24 850, 2023.

[2] CCSDS, "Low-complexity lossless and near-lossless multispectral and hyperspectral image compression," Available at: https://public.ccsds.org/Pubs/123x0b2c3.pdf. Accessed: February 06, 2024, p. 102, 2021, cor.3.

[3] L. M. Pereira, D. A. Santos, C. A. Zeferino, and D. R. Melo, "A low-cost hardware accelerator for CCSDS 123 predictor in FPGA," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2019, pp. 1–5.

[4] J. Fjeldtvedt, M. Orlandić, and T. A. Johansen, "An efficient real-time FPGA implementation of the CCSDS-123 compression standard for hyperspectral images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 10, pp. 3841–3852, 2018.

[5] A. Tsigkanos, N. Kranitis, G. Theodorou, and A. Paschalis, "A 3.3 Gbps CCSDS 123.0-B-1 multispectral & hyperspectral image compression hardware accelerator on a space-grade SRAM FPGA," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 1, pp. 90–103, 2018.

[6] 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. 54 269–54 287, 2020.

[7] 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, p. 1681, 2020.

[8] W. Grignani, D. A. dos Santos, L. Dilillo, F. Viel, and D. R. de Melo, “A low-cost hardware accelerator for CCSDS 123 lossless hyperspectral image compression,” in *DFT 2023-36th IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, 2023.
[9] D. A. Santos, A. M. P. Mattos, D. R. Melo, and L. Dilillo, “Characterization of a fault-tolerant RISC-V system-on-chip for space environments,” in *2023 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2023, pp. 1–6.
[10] NASA, “Airborne Visible InfraRed Imaging Spectrometer,” Available at: https://aviris.jpl.nasa.gov/. Accessed: February 9, 2024.
[11] D. R. Melo, C. A. Zeferino, L. Dilillo, and E. A. Bezerra, “Maximizing the inner resilience of a network-on-chip through router controllers design,” *Sensors*, vol. 19, no. 24, p. 5416, 2019.