Shaik Jani Begum, 2026, 14:3
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Simulation and Implementation of Hybrid Error Correction Codes for Space Applications

Shaik Jani Begum[1], J Ravisankar[2]

[1]M. Tech Student, Department of VLSI and Embedded Systems, MAM Women's Engineering College, Kesanpalli, Narasaraopet, Palnadu District, Andhra Pradesh.

[2]Associate Professor, Department of Electronics and communication engineering, MAM Women's Engineering College, Kesanpalli, Narasaraopet, Palnadu District, Andhra Pradesh

Abstract- This research aims to develop a new EDAC system that can detect and repair mistakes in space-grade onchip memory produced by cosmic radiation and multi-cell disturbances in extreme environments. Traditional XOR-based two-dimensional codes have a data width that may vary between 32 and 64 bits, limiting them to single-bit or restricted multi-bit protection. In contrast, the proposed approach provides far wider fault coverage. The verification procedure of block-wise XOR redundancy and cyclic redundancy check (CRC) efficiently addresses issues like burst faults by leveraging the diagonals, parity, and creation of the check-bit. The decoding process employs the encrypted 64-bit input to recover the original 32-bit data, allowing for full data recovery with high accuracy. After we develop and synthesize the design in Xilinx Vivado to test efficiency, we will go into power consumption, LUT utilisation, flip-flop deployment, and I/O efficiency in depth. The experimental findings suggest that the approach provides high resilience at minimal redundancy, together with lightweight protection for memory in aircraft systems, and outstanding dependability. In addition, we have introduced a hybrid CRC-XOR coding and signaling scheme that strikes a balance between error resilience and minimal hardware overhead utilisation, we have ensured that the recovery still returns 32 bits when scaling to 32-bit inputs, and we have explicitly supported burst errors through error correction.

Keywords: Burst error, Xilinx Vivado, power analysis, UT utilisation, aerospace memory systems, 64-bit encoded, 32-bit decoded, error detection and correction (EDAC), and Xilinx.

I. INTRODUCTION

Because systems like spacecraft and satellites are exposed to cosmic radiation and variable temperatures, the reliability of on-chip memory is critical in space engineering [1]. As a result of data corruption and decreased system function, these hostile elements may produce both single-event upsets (SEUs) and multiple cells upset (MCUs). As space missions rely more and more on sophisticated and data-intensive processing, there is an immediate need for powerful Error Detection and Correction (EDAC) algorithms that provide great fault coverage without power/area losses. Conventional error correcting codes, such Hamming codes or systems based on single bit or modest multi-bit parity; do not provide many safeguards [2-4].

Burst errors, which are defects that genuinely occur between consecutive memories cells, are a common symptom of the more complicated error behaviour seen in high radiation environments. Traditional methods for space borne systems have a number of problems, the most notable of which being the rise in hardware complexity and power consumption [5]. To address these issues, this research introduces an improved 2D XOR-based ECC method that multiplies a 16-bit input by 32 to produce a 64-bit sendable or storable word. The strategy makes use of conventional wisdom: By performing a block-wise XOR on 32-bit input data, we may compute the diagonal, parity, and check bits, which provide a multi-directional representation of the input that is resistant to burst and random errors. The encoding method includes this step.

A Cyclic Redundancy Check (CRC) layer, when included, makes burst error detection more reliable [6]. • Decoding Process: The receiver uses syndrome and region selection scheme calculation techniques to pinpoint where the errors are. While keeping the original 32-bit data output, the method may rectify inaccurate symbols by removing unneeded bits [7]. Clustered error detection, correction efficiency, and

resilience in hostile space circumstances are all improved by integrating CRC with classical XOR-based redundancy in a hybrid ECC framework [8]. Through careful planning and synthesis in Xilinx Vivado, we were able to provide accurate predictions of power consumption, LUT count and use, flip-flop count and use, and delay overall performance [9–11]. Despite its low power consumption and area overhead, the findings show that the 32-bit long-extension strategy considerably increases error coverage when compared to traditional methods. Because of this, it is highly recommended for aeronautical applications that deal with limited resources [12–13].

II. PROPOSED METHOD

The encoder and the decoder are the two primary building parts of the proposed architecture for error detection and repair. While the decoder corrects errors and returns the original data, the encoder strengthens the sent data by encoding it into a coded signal. In order to ensure that the original information is intact as the codeword grows from 32 bits to 64 bits, the encoder painstakingly includes bits for parity, diagonal, and check. At the other end of the wire is the decoder, whose job it is to take the encoded data, use syndromes to find any differences, and then rectify it so it sends back the original message.

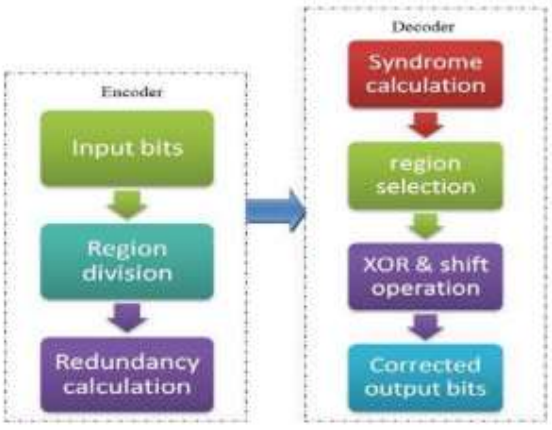

Fig. 1. The suggested error correcting technique for encoders and decoders, shown as a block diagram
When assembled, these components guarantee accurate data integrity in jobs that are vulnerable to radiation. The data flow, module interactions, encoder and decoder stages, and general structure of the proposed EDAC system are shown in Figure 1.

In order to extend a 32-bit input word to 64 bits, the encoder employs structured XOR operations to generate redundancy. It increases the resistance to burst errors of radiations among neighboring bits by creating diagonal, parity, and check bits. Figure 2 shows this redundancy in action; thus, it is critical to guarantee reliable recovery of the original data during transmission and storage.

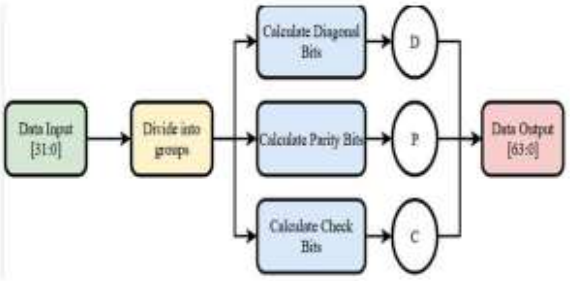

Fig. 2. ECC Encoder Architecture for 32-bit Data

Using XOR logic, the decoder delivers the newly computed redundancy bits after receiving the 64-bit encoded data. These, together with the preserved redundancy, may be used to compute the syndrome values, which will help pinpoint the problem's type and location. By using its correction algorithms to identify and rectify the introduced faults, the decoder successfully restores the original 32-bit data without compromising any information, as seen in Figure 3.

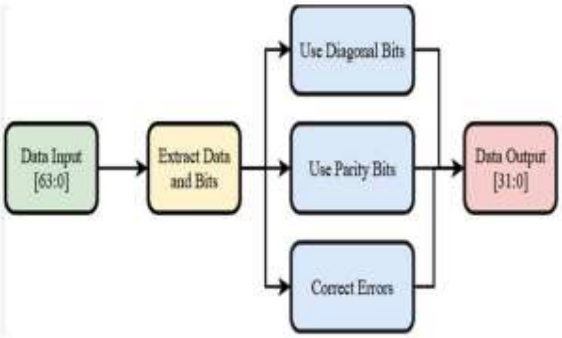

Fig. 3. ECC Decoder Architecture for 32-bit Data

The given decoding method is perfect for usage in environments prone to faults, such as space, since it is resilient against burst and single bit errors. Lastly, we will compare it to the present methods: While traditional ECC-based designs, such as the 16-to-32-bit designs, should be enough for basic bit flips, they often aren't able to prevent dense burst mistakes that are comparable to harsh instances. By decentralizing XOR operations in larger bit fields and completely extending the data blocks, this 32-to 64-bit technique considerably simplifies the process of discovering and repairing pack issues. A safety net that doesn't increase hardware needs or power consumption is achieved by combining diagonal redundancy with standard parity.

A high-speed, scalable encoder-decoder system capable of converting 64-bit dumps to 32-bit strings is the end result. It is suitable for use with aircraft and other mission-critical machinery that must avoid data breaches at all costs. A First Look at the Coding Process Figure 4 shows the result of a structured, step-by-step encoding process that successfully identifies and fixes faults from the 32-bit input. The methodical use of this encoding ensures robust data integrity throughout transmission. This is of utmost importance in cases involving radiation or space.

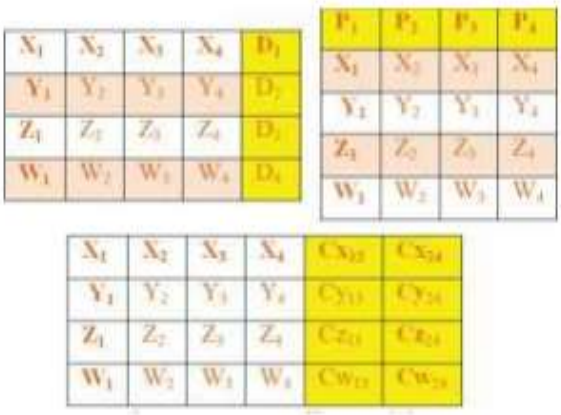

Fig. 4. Encoding scheme 1)

At first, the 32-bit input is broken down into eight groups: Matrix Formations, Data Grouping Xi, Yi, Zi, Wi, Ai, Bi, Ci, and Di. Once they are grouped in this manner, they are positioned into a 4 × 8 matrix that is intentionally designed to make redundancy generation easier. The basic building block for further XOR calculations is here in this matrix form.

The second set of procedures includes the first check bit, parity, and diagonal XOR. To improve the data's encryption, a series of carefully chosen XOR operations on the combined sets are executed: "Diagonal Bits" (D) were created by XORing data diagonals (e.g., Xi with Ai, Zi with Ci) in order to identify two-dimensional diagonal errors. For each group to be covered, the computation of parity bits (P) requires XORing along the row and columns. You can detect burst and multi-bit errors with these bits. • Check Bits (C): These bits enable precise data localization and correction during decoding; they are raised following a little data XOR operation.

Thirdly, the data that has been 64-bit encoded the last step of the encoding process is to increase the size of the original 32-bit message to 64 bits. Here the newly created Diagonal, Parity, and Check bits are appended to the initial data. Both the length and the systematic redundancy increase the likelihood of error. 4) Inflexibility in the face of duplication and errors the inclusion of diagonal, parity, and check bits strengthens the data structure against catastrophic mistakes. Redundancies like this allow fault-sensitive applications to reliably transmit data by detecting faults of many types, including single-bit, multi-bit, diagonal, and burst. In order to locate and correct the errors, the decoder uses this data. Section B: The Encoding Process The purpose of the decoding process is to extract the 64-bit codeword from the 32-bit source data.

The decoder extracts, decodes, and employs redundancy (diagonal parity, and check bits) inside the codeword to detect and correct errors. In the end, we want to be able to retrieve the original 32-bit data without any loss of quality, even in a scenario where cell disruptions are common. The First Step: Syndrome Calculation The decoder begins by recreating the redundant bits of the 64-bit codeword. Table I displays the XORed values that result from the redundancy bits that are an intrinsic part of the encoding process:

TABLE I. TABLE I REGION SELECTION CRITERIA

Region	Selection criteria
Region 1	SD1+SD2+SP1+SP2 > SD3+SD4+SP3+SP4
Region 2	SD1+SD2+SP1+SP2 < SD3+SD4+SP3+SP4
Region 3	SD1+SD2+SP1+SP2 = SD3+SD4+SP3+SP4

Here,

$S = R_{redundant} \oplus C_{redundant}$

The generated syndrome bits document the kind and location of data errors and function as unique markers. 2) Examining Syndromes and Finding Defects The value range of the syndrome has been thoroughly analysed and debugged, and potential causes and errors have been identified: Use the diagonal syndrome bits to find and classify the cross-group diagonal faults. • Bits with parity syndrome indicate that the error has happened exactly once in both sets of bits. When a 64-bit codeword has a damaged bit, the syndrome bits check may pinpoint it to the exact bit.

In order to identify whether the error is one bit, several bits, or cross-regional, the decoder employs all three types of cross-correlation. What is the location of the region where I may correct its mistakes? With the use of XOR, the decoder corrects the data from the syndrome analysis (Table I) to remedy particular incorrect locations: • the diagonals of data blocks are redirected for correction in diagonal syndromes. • Parity syndromes deal with problems that occur in the row or in the bits. If you're experiencing gastrointestinal issues, try concentrating on one dish at a time. By substituting the proper value for each bit, the process of flipping these bits eliminates the effect of the wrong bits on the output data. 4) Error Correction and Target Area Selection Syndrome analysis provides syntactic corruption data, which the recoding unit then employs in an XOR fashion to repair particular issue places (Fig. 3).

Diagonal syndromes are used to control the correction across data block diagonals. The parity syndromes deal with data mistakes that occur in rows and bits. Focus order to correct problems more accurately, syndromes zero focus on particular regions of inaccuracy. By inverting all the incorrect bits, we restore normalcy to the 64-bit codeword. 5. Getting the Original Data Back the process of rectification involves the methodical deletion of the diagonal bit, as well as the check and parity bits. Only the recovered pre-encoding data that matches the current state of the data makes up the remaining 32 bits. Thus, information will be preserved regardless of radiation exposure that results in single-event upsets (SEUs) or multiple-cell upsets (MCUs). C. Combinatorial and Boolean Algebra Logic Circuits The 64-bit decoder functions as a combinational circuit that produces outputs while operating in real-time mode; it does not need any memory components whatsoever. • Boolean Logic: XOR gates are used for redundancy, syndrome computation, and error correction in Boolean Logic, whereas AND/OR gates are used for decision logic. The scalability of the 32-to-64-bit variation is superior to that of the 16-to-32-bit model due to its greater redundancy capacity and increased resilience to multiple failures.

Redundancy makes it more durable to radiation-triggered multi-bit upsets (MCUs), making it adaptable to space applications. Since the decoder can safely restore the original 32-bit data from corrupted 64-bit codewords, it is valuable for protecting memory in aircraft systems and other systems that need high dependability.

III. RESULTS AND DISCUSSION

The RTL (Register-Transfer Level) diagram shown in Figure 5 is the encoder's schematic and depicts the logic components used in the encoding process graphically. By use of a succession of combinational circuits and logical gates, it illustrates the transformation or processing of 32-bit input data into 64-bit coded output. The picture depicts the two-stage grouping of input data into a matrix, and the final determination of the diagonal bits, check bits, and parity bits by XOR operations.

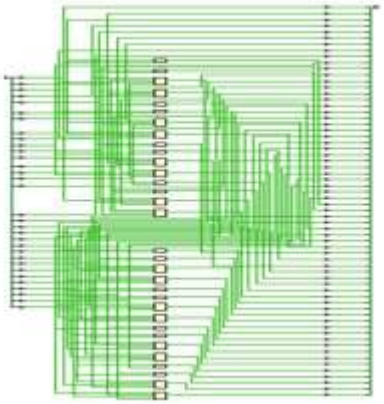

Fig. 5. RTL Schematic of Encoder

The RTL architecture of the decoder, seen in Figure 6, employs reverse encoding to recreate the 32-bit data. Here we may see the procedure for calculating the syndrome, identifying errors, and recovering from them. The graphic illustrates the interconnection of logic devices such as XOR gates that can identify and correct multibit errors, enabling the effective recovery of the original data.

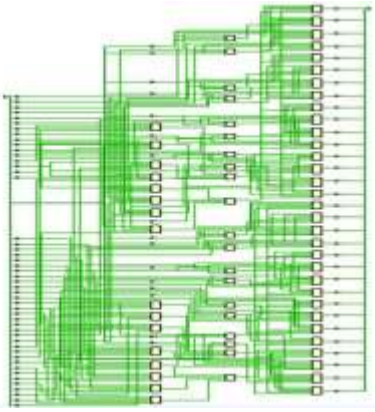

Fig. 6. RTL Schematic of Decoder

Encoder development is the main emphasis here (Fig. 7). In expanding upon the RTL schematic, it offers a more thorough picture of the parts and their connections. In order to construct the encoding process, this figure further demonstrates the precise implementation of the combinational logics utilising certain gates and flip-flops. It delves deeply into the steps used to convert 32-bit input data into 64-bit coded output.

Fig. 7. Elaboration of Encoder

The decoder is described in full in Figure 8. Explains how the decoder works in great detail. This breakdown of the decoder shows how the various logic gates and circuits work together to compute syndromes, discover errors, and fix them. The original data is available in 32 bits, but the decoder can rebuild it using the 64-bit encoded input. You may find the method outlined here.

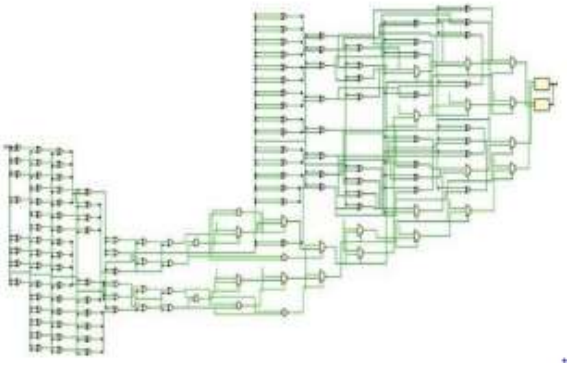

Fig. 8. Elaboration of Decoder

The simulated waveform of the encoder, seen in Figure 9, reveals the timing and order of the encoding procedure. You can see how the input bits go through XOR-based procedures to get the final encoded data by looking at the 32-bit input data (abcdef12) and the 64-bit encoded output (81b21c12abcdef12). This example verifies the latency, propagation delays, and functional correctness of the encoding process in real-time by showing the bit-by-bit progression through the encoder.

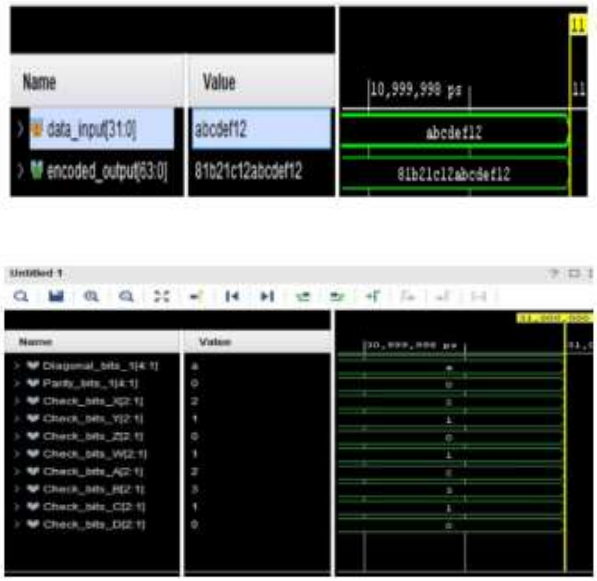

Fig. 9. Simulation Waveforms of Encoder

Figure 10 shows the incorrect input word and the decoder's response (81b21c12ab89ef56) and corrected output (abcdef12). In order to illustrate how the original 32-bit data is recovered from the 64-bit encoded input, the waveform displays the many stages of decoding, beginning with the creation of the syndrome and ending with the correction. Both the decoder's high fault-tolerance capability and the recommended error-correction approach are shown by this depiction.

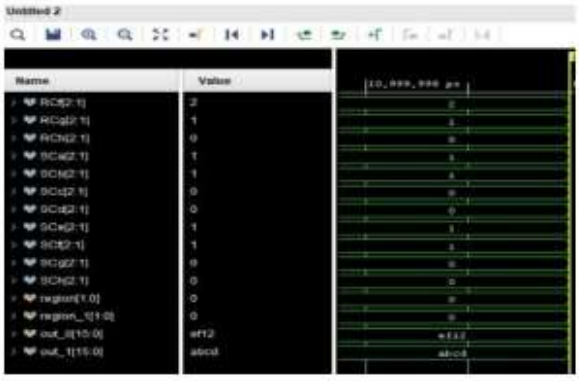

Fig. 10. Simulation Waveforms of Decoder

C. Comparison Table

TABLE II. ENCODER DELAY

S.N.	Type of Delay	End Points
1	Maximum Delay	256
2	Minimum Delay	256

As shown in Table II, the encoder delay allows one to manage the encryption speed, which is dependent on the total number of endpoints and the throughput between the input and output. The encoding time is defined by the Max Delay, the maximum number of endpoints on the logic gates, in the worst-case scenario. The number is 256. • Consistent performance across all input circumstances is guaranteed by a minimum delay of 256.

TABLE III. DECODER DELAY

S.N.	Type of Delay	End Points
1	Maximum Delay	128
2	Minimum Delay	128

According to Table III, the decoder delay provides the necessary information to recover the original data from the encoded input. There is a maximum delay of 128 endpoints on the longest route that involves error identification and correction. With a minimum delay of 128 endpoints, operations may move more swiftly and error-free.

TABLE IV. ENCODER RESOURCE UTILIZATION

S.NO	Type	Resource	Utilization	Available
1	LUT	17	134600	0.01
2	IO	96	400	24.00

Table IV shows the resource use of the encoder: Only 17 (or 0.01%) of the 134,600 possible lookup tables (LUTs) are currently in use. The implementation of logical functions makes use of 96 out of 400 available pins, or 24% of the total. It is the job of the latter to transfer information to and from the FPGA.

TABLE V. DECODER RESOURCE UTILIZATION

S.NO	Type	Resource	Utilization	Available
1	LUT	52	134600	0.03
2	IO	96	400	33.00

Of the 134,600 look-up tables, 52 (or 0.03%) were LUTs, as shown in Table V, which demonstrates the resource utilisation of the decoder architecture. Compared to the encoder, the decoder is more sophisticated since it uses more logic to detect and resolve errors. Only 96 out of 400 input/output pins are really in use, which is a 33 percent utilisation rate. Since the encoder processes a greater number of external signals, it is much smaller.

IV. CONCLUSION AND FUTURE SCOPE

A comparison of 32-bit and 16-bit ECC methods reveals that the 32-bit architecture provides more balanced and efficient data transport. Even though they need more hardware resources (in terms of LUTs and I/Os) than the 16-bit system, the 32-bit encoder and decoder are perfect for radiation-prone and fault-sensitive situations, such as aerospace and space applications. Nonetheless, enhanced mistake detection and repair capabilities directly transfer this burden. A 32-bit architecture consumes almost twice as much power as a 16-bit system, with an encoder power consumption of 26.234 W and a decoder power consumption of 18.14 W, respectively, compared to 13.126 W and 9.116 W, respectively. The scalability and superior error resistance of the 32-bit implementation, however, more than compensate for the increase in price. Additional timing study reveals that the delay for both systems is almost the same, standing at 5.8 ns for decoders and 4.6 ns for encoders, indicating that performance efficiency remains same despite the increase in bit-width. When compared to its 16-bit predecessor, the 32-bit ECC architecture is more durable, reliable, and fault-tolerant without significantly sacrificing temporal speed. This is why 32-bit ECC algorithms are better suited to real-world, mission-critical applications. In the future, the design might be enhanced to support higher bit width architectures, such a 64-bit or 128-bit input system, so larger data words can be efficiently handled. Radiation fault-injection testing may be performed in real-time when connected with FPGA-based devices. Design upgrades that minimise latency and hardware resource costs, as well as code improvements like hybrid ECC methods, might broaden its potential uses in the realms of satellite communications, aerospace, and fault tolerant computer networks.

REFERENCES

1. F. Lumpe and M. Seidl, “Benefits of using functional safety in commercial space applications,” J. Space Saf. Eng., Jan. 2025.
2. M. Yan, J. C. Chavez, K. El-Sankary, L. Chen, and X. Lu, “A 10-bit 50-MS/s radiation tolerant split coarse/fine SAR ADC in 65-nm CMOS,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 1, pp. 1–11, Jan. 2025.
3. K. Velagapudi, S. Wu, and N. Wang, “Low-power CRC-BCH error correction integrated with LFSR and clock gating: A comparative analysis,” in Proc. 2025 IEEE 15th Annu. Comput. Commun. Workshop Conf. (CCWC), Jan. 2025, pp. 00164–00170.
4. A. Smith and B. Lee, “Improved ECC with 15-bit redundant code for 32-bit data,” IEEE Trans. Comput., 2025.
5. L. Alex, M. E. Kreutz, and S. M. Dias, “A parity-based dual modular redundancy approach for the reliability of data transmission in nanosatellite’s onboard processing,” IEEE Access, vol. 12, pp. 90815– 90828, Jan. 2024.

6. J. J. S. and K. Rasadurai, "Enhanced on-chip memory reliability for space engineering using a novel 2-dimensional error detection and correction scheme," in Proc. 2024 Int. Conf. Syst., Comput., Autom. Netw. (ICSCAN), Dec. 2024, pp. 1–9.
7. N. A. Koca, C. H. Chang, A. T. Do, and V. P. Nambiar, "Exploring error correction circuits on RISC-V based systems for space applications," May 19, 2024.
8. G. Achala, P. Srihari, and U. S. Acharya, "On the synthesis of channel codes for NAND flash devices in space application," in Proc. 2021 IEEE Int. Conf. Electron., Comput. Commun. Technol. (CONECCT), Jul. 2024, pp. 1–6.
9. K. S. Nakul, K. Chaitanya, P. Abhiram, and M. Vinodhini, "Row-wise Hamming code for memory applications," Aug. 7, 2024, pp. 30–34.
10. 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 Proc. 2023 IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT), Oct. 2023, pp. 1–6.
11. J. Guo, H. Zhou, and Y. Sun, "Decimal matrix code for enhanced reliability against MCUs," IEEE Trans. Device Mater. Rel., 2023.
12. S. Tripathi, J. Jana, and J. Bhaumik, "Fast and power-efficient SECDAEC and SEC-DAEC-TAEC codecs on FPGA," IEEE Trans. Circuits Syst. I, Reg. Papers, 2023.
13. S. Gamini and A. Yelamanchili, "Random error detection and correction codes for multiple bits," in Proc. 2022 IEEE 3rd Global Conf. Adv. Technol. (GCAT), Oct. 2022, pp. 1–5.