International Journal of Gender, Science and Technology
ISSN: 2040-0748
UGC Care Group I Journal.
Vol-13 Issue-02 Dec 2024
https://ijgst.com.2024.v13.i2.pp986-993

Design and Implementation of Reed-Solomon Erasure Code (RS-EC) Decoders with FPGA Implementation that Detects and Locates Faults in User Memory

Dr.A.Ranganayakulu[(1)] Mr.D.Satyanarayana[(2)] Kolli Geetha Priyanka[(3)] Shaik Ruksana Banu[(4)] Kandula Rajeswari [(5)] Gundla Kavitha[(6)]
[1,2] *Krishna Chaithanya Institute Of Technology & Sciences, Ece Department, Markapur, Andhra Pradesh.*
[3,4,5,6,] *Krishna Chaithanya Institute Of Technology & Sciences, UG Student-ECE, Markapur, Andhra Pradesh.*

Abstract. In this paper to recover erasures, Reed-Solomon erasure codes, or RS-ECs, are frequently utilized in packet communication and storage systems. The RS-EC decoder will experience single-event upsets (SEUs) on a field-programmable gate array (FPGA) in a space platform, which may lead to malfunctions. This project first examines the dependability of an FPGA-implemented RS-EC decoder in the presence of user memory faults. Next, a partial re-encoding-based fault identification and localization strategy for the RS-EC decoder's user memory faults is provided. To further enhance the fault location performance, check bits are added to the generating matrix. Theoretical research demonstrates that the approach might identify many errors with low chances of false positives and misses and this project results can be achieved by using Xilinx Vivado Verilog Hdls

.**Keywords:** RS – EC coders and decoders , 5G Communications, FPGA, Verilog HDL, Faults Memories and Hazards.

I.INTRODUCTION

Reed-Solomon Erasure Codes (RS-EC) are widely used in digital communications and storage systems for error correction. These codes are particularly useful in scenarios where data packets may be lost due to errors or failures, such as in satellite communication, optical fiber communication, and storage systems like RAID (Redundant Array of Independent Disks).

The RS-EC works by adding redundant symbols to the original data before transmission. These redundant symbols allow the decoder to reconstruct the original data even if some of the symbols are lost or corrupted during transmission. The number of redundant symbols added determines the code's ability to correct errors and erasures.

Decoders for RS-EC typically operate by performing polynomial interpolation using the received symbols, including both the original data symbols and any redundant symbols. By analyzing the received symbols, the decoder attempts to reconstruct the original message.

The complexity of RS-EC decoders depends on various factors such as the code parameters (e.g., code length, number of parity symbols), implementation optimizations, and the hardware or software platform on which they are executed. Generally, RS-EC decoders are designed to efficiently

handle erasures and correct errors with low computational complexity.

Efficient RS-EC decoder implementations are crucial for real-time applications where latency and computational resources are limited. Therefore, researchers and engineers continuously work on improving decoding algorithms and implementations to achieve better performance and scalability. These improvements may involve techniques such as parallelization, optimization for specific hardware architectures (e.g., SIMD instructions, FPGA, or GPU acceleration), and algorithmic enhancements.

In summary, RS-EC decoders are essential components in systems requiring robust error correction capabilities. They perform polynomial interpolation to reconstruct original data from received symbols, and their complexity depends on various factors including code parameters and implementation optimizations. Continuous research and development aim to improve their efficiency and performance for various applications.

(RS) codes are efficient error correction codes that are widely used in communications and storage systems [1]–[3]. For applications on which some data blocks are lost and their positions are known, a variant known as RS Erasure Codes (RS-ECs) can be used for data recovery [2], [4]. For example, in IP-based communication networks, a message is packed into a sequence of packets and transmitted through the network, and some of them may not arrive at the destination due to network congestion. In this case, an RS-EC code could be used to introduce redundant packets at the transmitter and recover the lost packets at the receiver based on the constraints among received packets [5], [6]. For large-scale storage systems, data are distributed in multiple storage units (disks or servers), and an RS-EC code is usually used to introduce redundant data segments. When some of the storage units fail, the constraints among the available units could be used to recover the original data [7]– [9].

In recent years, space-based Internet has become a hot topic due to its seamless global coverage and strong robustness to failures of terrestrial infrastructures [10]–[12]. In space based Internet, data would be transmitted as IP packets, and large amounts of data will be stored on the satellite platform [11], [12], so RS-EC could be widely used for reliable data transmission and storage. However, different from the application on terrestrial systems, cosmic radiation may cause the failure of the onboard digital electronic systems [13], [14], so the reliability of the RS encoder and decoder becomes an important problem, especially for the decoder whose complexity is much higher than the encoder. On the other hand, SRAM-based field-programmable gate arrays (FPGAs) (SRAM-FPGAs) are a popular option for onboard digital processing due to their rich logic resources and good reconfigurability [15], [16].sensitive to cosmic radiation, and single-event upsets (SEUs) are the most frequent events [16], [17]. SRAM-FPGAs can suffer two types of SEUs: errors on the configuration memory and errors on the user memory. The errors on the configuration memory may change the design, and they are permanent unless the FPGA is reconfigured [17]. For user memory, e.g.,

registers and BRAMs, the SEU may modify the parameters and intermediate variables and corrupt the results. Therefore, if RS codes are implemented in an SRAM-FPGA on a space platform, the reliability of the decoder would be an important issue [18], and the protection of RS decoders against errors induced by SEUs becomes important.

There are several works on the protection of the decoder for RS error-correcting codes against SEUs. In [19], the traditional triple modular redundancy (TMR) method is applied to protect the flip-flops and the clock and reset signals in the encoder and decoder. Cardarelli et al. [20] proposed to detect faults in the RS decoder by checking whether the output is a valid codeword. Instead, the work presented in [21] discussed the same self-checking property of the RS decoder and proposed to reduce the complexity of the fault detection logic with a small loss of the fault tolerance capability. However, as far as the authors know, there is no research aiming at SEU detection and location for the decoder of RS-EC codes.

In this project, an efficient fault detection and location scheme is proposed to detect the faults induced by SEUs and to locate their position in the user memory of the RS-EC decoder. The SEU detection and location information could be used by the fault management logic of the system for efficient recovery.

II.EXISTING SYSTEM

1.Encoder and Decoder for RS-EC

A (k, m) RS-EC is composed of k data symbols (d = [d1, d2,…, dk]T) and m parity symbols (p = [p1, p2,…,pm]T). Each symbol is expressed as a w-bit Word, and the combination of the k + m symbols is defined as a codeword c= [dT pT] T on the Galois field GF(2w). Any combination of k symbols (data or parity) could be used to recover the original k data symbols [5]. In other words, the maximum number of erasures is m.

2. Evaluation of The Proposed Fault-Tolerant Rs-EC Decoder

This project presents the fault injection experiments and evaluates the proposed fault detection and location scheme for SEUs on the user memory in the RS-EC decoder.

2.1. Resource and Time Overhead

The decoder for (6, 3) RS-EC with the proposed fault detection and location scheme was implemented using Verilog and mapped on the same target device, a Xilinx Zynq 7000 SoC (xc7z030ffg 676-1). The theoretical complexity of the original RS-EC decoder and the protected one are compared in Table V in terms of table lookups and additions. As we can see, the complexity of the protected decoder is about 3.8 times that of the unprotected one for single erasure, and the overhead decreases to around 2.3 times for the case of three erasures.

To improve the resource efficiency of the implementation, the partial reencoding (G p × d) reuses the module for G × xde in the decoder, and the two copies of the newly generated parity symbols (px r1 and px r2) are stored in de considering that k is usually larger than 2m. Then, the usages of LUTs, registers, and BRAMs of the original and protected RS-EC decoder are listed in figure. As we can see, the proposed RS-EC decoder consumes an additional 60 LUTs and 73 registers (for partial reencoding and parity check for generator matrix G); thus, the resource overhead is about 1.05 times

that of the original decoder. The time overhead of the proposed solution is about 1.16 and 2.17 times that of the original decoder for the cases without and with fault, respectively.

A. Platform for SEUs Injection Experiments

This section provides a the reliability of the user memory of the RS-EC decoder implemented on FPGAs is first studied by theoretical analysis and fault injection experiments, and the results show that the decoder itself has strong fault tolerance against SEUs. Then, a fault detection scheme based on partial re-encoding and a fault location scheme based on re-decoding are proposed to protect the RS-EC decoder. To further enhance the performance of the basic fault detection and location scheme, the generator matrix is protected separately by parity check bits, and the probability of MD and false alarm of the final scheme is analyzed. Fault injection experiments show that: 1) the fault detection scheme could detect all the faults if the number of erasures is less than m, and MDs appear for m erasures with a very small probability; 2) false alarms exist with a small probability, but they only introduce decoding overhead and are also beneficial to avoid faults accumulation; and 3) the fault location scheme could locate all the detected faults on intermediate variables, generator matrix, and two lookup tables, with an accuracy of 100%.

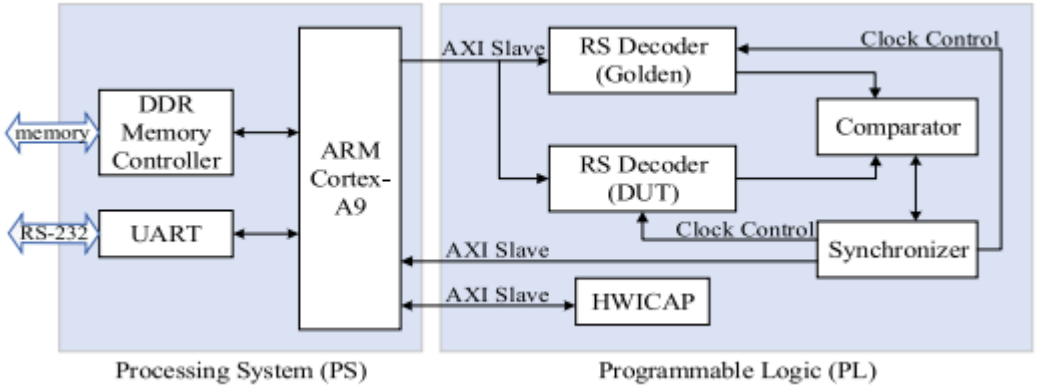

Figure.1 Fault injection platform of the RS-EC decoder.

To assess the effectiveness of the proposed scheme to detect and locate faults in the user memory of the decoder, SEUs have been injected using an adapted version of the fault injection tool in Xilinx and implemented on a Zedboard. The experimental setup for injection on the unprotected and proposed RS-EC decoder with fault detection and location is shown in Fig. 4. The injection platform consists of the processing system (PS) and the programmable logic (PL). The PS consists of the ARM Cortex-A9 processor and dedicated controllers for different peripherals, e.g., DDR memory controller, SD controller, and UART controller. The DDR controller is responsible for storing the list of bits in the registers and the BRAMs. The list is generated during the compile time in Vivado and is used by the injection algorithm for reliability evaluation of the original decoder and the performance

evaluation of the fault detection and location scheme. The UART module in the PS is responsible for logging results to the PC. The PL part consists of two copies of the RS-EC decoder, i.e., golden and design under test (DUT). Moreover, a comparator and a synchronizer block are also present in the PL part. These modules are responsible for concurrent error detection in the golden and DUT RS-EC decoders, and the mismatches are recorded by the processor. The synchronizer module is responsible for controlling the clock to DUT and golden copies of the RS-EC decoder. Furthermore, the RS-EC decoder receives inputs from the processor in the PS part, so the test patterns are software-controlled. Another important module housed by the PL part is the Internal Configuration Access Port (ICAP) module, which allows the processor to access the user memory related to the DUT decoder in run time for error injection. The modules in the PL region are connected to the PS region through AXI buses.

The ARM processor runs the software that controls the fault injection process. The fault injection starts by freezing the clock to the RS-EC decoder in the PL part (through the synchronizer module). This is followed by reading back the target bit from a register/BRAM through the ICAP port. The address of the user memory bits for fault injection is extracted from the fault list stored in DDR memory. The readback values are corrupted by inserting a bit flip for SEU emulation and written back. This is followed by resuming the design's clock. For reliability evaluation of the original RS-EC decoder, the mismatches between the golden and DUT decoders are collected by the ARM processor and communicated to the PC through the UART interface. The fault detection and location results are also collected for the evaluation of the proposed scheme.

2.3. Performance Evaluation of Proposed Fault Detection and Location Scheme for RS-EC Decoder

The MDP and FAP of the proposed fault detection scheme are tested based on the fault injection experiments for each part of the user memory, and the accuracy of the fault location scheme is also evaluated correspondingly.

Fig2. Structure of fault detection and location scheme.

Fig3. Logic for fault detection and location.

algorithm for Reed-Solomon error correction decoding. Reed-Solomon codes are a type of error-correcting code that can detect and correct errors in data transmissions. The decoding algorithm for Reed-Solomon codes typically involves the use of mathematical operations such as polynomial division and interpolation.

Here's a high-level overview of the decoding algorithm for Reed-Solomon codes:

1. Syndrome Calculation: Compute syndromes from the received codeword. Syndromes are evaluations of the received polynomial at certain points.
2. Error Location Polynomial: Using the syndromes calculated in step 1, find the error locator polynomial. This polynomial helps in locating the positions of errors in the received codeword.
3. Error Evaluation: Evaluate the error evaluator polynomial, which helps in determining the magnitudes of errors at each error location.
4. Error Correction: Using the information obtained from the error locator

International Journal of Gender, Science and Technology
ISSN: 2040-0748
UGC Care Group I Journal.
Vol-13 Issue-02 Dec 2024

https://ijgst.com.2024.v13.i2.pp986-993

polynomial and error evaluator polynomial, correct errors in the received codeword.

III. RESULTS AND ANALYSIS DISCUSSION

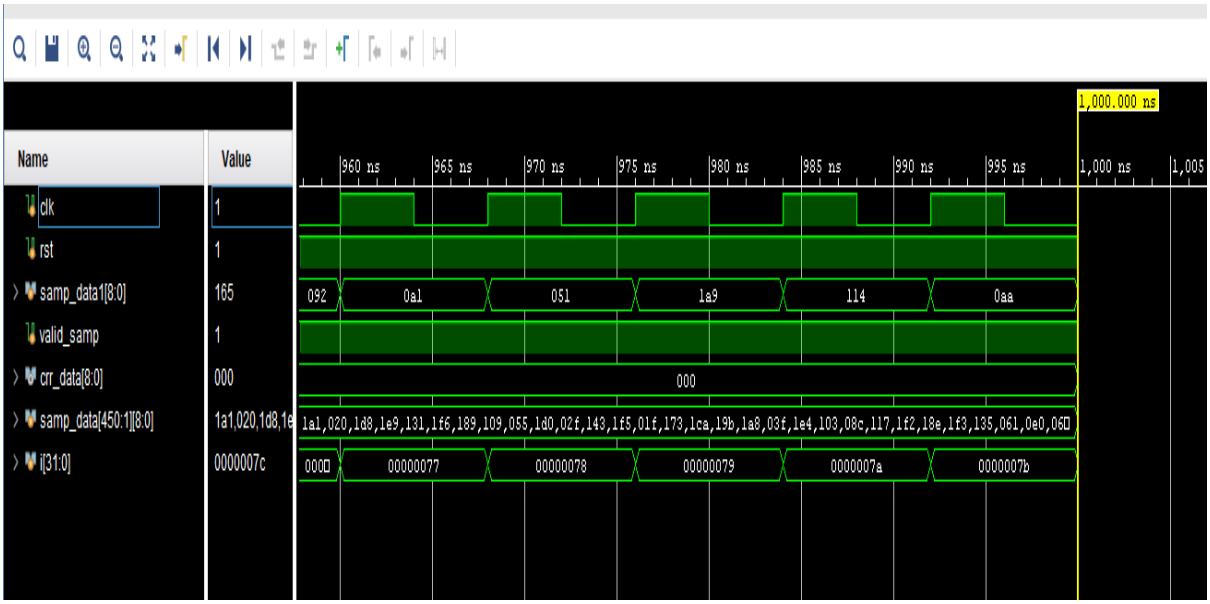

Fig.1.RS-EC Decoder simulation output.

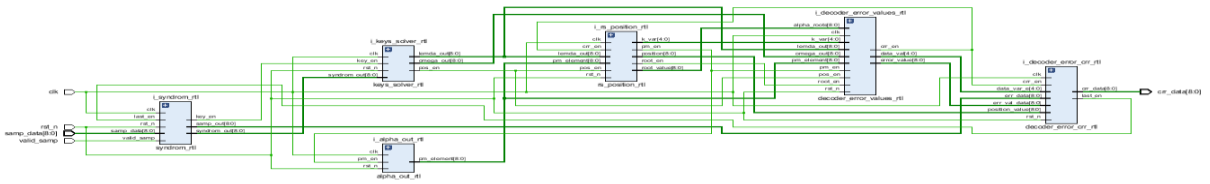

Fig3.RTL schematic

CONCLUSION

In this article, the reliability of the user memory of the RS-EC decoder implemented on FPGAs is first studied by theoretical analysis and fault injection experiments, and the results show that the decoder itself has strong fault tolerance against SEUs. Then, a fault detection scheme based on partial reencoding and a fault location scheme based on redecoding are proposed to protect the RS-EC decoder. To further enhance the performance of the basic fault detection and location scheme, the generator matrix is protected separately by parity check bits, and the probability of MD and false alarm of the final scheme is analyzed. Fault injection experiments show that: 1) the fault detection scheme could detect all the faults if the number of erasures is less than m, and MDs appear for m erasures with a very small probability; 2) false alarms exist with a

small probability, but they only introduce decoding overhead and are also beneficial to avoid faults accumulation; and 3) the fault location scheme could locate all the detected faults on intermediate variables, generator matrix, and two lookup tables, with an accuracy of 100%.

REFERENCES

[1] R. E. Blahut, Theory and Practice of Error Control Codes. Reading, MA, USA: Addison-Wesley, 1984.
[2] T. Zhang and K. K. Parhi, "On the high-speed VLSI implementation of errors-and-erasures correcting Reed–Solomon decoders," in Proc. 12th ACM Great Lakes Symp. (VLSI), New York, NY, USA, 2002, pp. 89–93.
[3] F. Hernando, K. Marshall, and M. E. O'Sullivan, "The dimension of subcode-subfields of shortened generalized Reed–Solomon codes," Des., Codes Cryptogr., vol. 69, no. 1, pp. 131–142, Oct. 2013.
[4] J. Li, "The efficient implementation of Reed–Solomon high rate erasure resilient codes," in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP), Mar. 2005, pp. 1097–1100. [5] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," ACM SIGCOMM Comput. Commun. Rev., vol. 27, no. 2, pp. 24–36, Apr. 1997.
[6] A. A. Al-Shaikhi and J. Ilow, "Packet loss recovery codes based on Vandermonde matrices and shift operators," in Proc. IEEE Int. Symp. Inf. Theory, Jul. 2008, pp. 1058–1062.
[7] G. C. Cardarilli et al., "Design of a fault tolerant solid state mass memory," IEEE Trans. Rel., vol. 52, no. 4, pp. 476–491, Dec. 2003.
[8] J. S. Plank, "A tutorial on Reed–Solomon coding for fault-tolerance in RAID-like systems," Softw., Pract. Exper., vol. 27, no. 9, pp. 995–1012, Sep. 1997.
[9] S. Muralidhar et al., "F4: Facebook's warm BLOB storage system," in Proc. 11th ACM/USENIX Symp. Oper. Syst. Design Implement. (OSDI), 2014, pp. 383–398.
[10] J. Foust, "SpaceX's space-Internet woes," IEEE Spectr., vol. 56, no. 1, pp. 50–51, Jan. 2019.
[11] D. Li, X. Shen, N. Chen, and Z. Xiao, "Space-based information service in Internet plus era," Sci. China Inf. Sci., vol. 60, no. 10, Oct. 2017.
[12] C. Fei, B. Zhao, W. Yu, and C. Wu, "Towards efficient data collection in space-based Internet of Things," Sensors, vol. 19, no. 24, p. 5523, Dec. 2019.
[13] E. G. Stassinopoulos and J. P. Raymond, "The space radiation environment for electronics," Proc. IEEE, vol. 76, no. 11, pp. 1423–1442, Nov. 1988.
[14] M. Tali et al., "High-energy electron induced SEUs and Jovian environment impact," IEEE Trans. Nucl. Sci., vol. 64, no. 8, pp. 2016–2022, Aug. 2017.
[15] S. López, T. Vladimirova, C. Gonzalez, J. Resano, D. Mozos, and A. Plaza, "The promise of reconfigurable computing for hyperspectral imaging onboard systems: A review and trends," Proc. IEEE, vol. 101, no. 3, pp. 698–722, Mar. 2013. [16] F. Siegle, T. Vladimirova, J. Ilstad, and O. Emam, "Availability analysis for satellite data processing systems based on SRAM FPGAs," IEEE Trans. Aerosp. Electron. Syst., vol. 52, no. 3, pp. 977–989, Jun. 2016.
[17] F. L. Kastensmidt, L. Carro, and R. Reis, Fault-Tolerance Techniques for SRAM-Based FPGAs. New Haven, CT, USA: Springer 2006.
[18] Z. Gao, L. Yan, J. Zhu, R. Han, U. Anees, and R. Pedro, "Radiation tolerant Viterbi decoders for on-board processing (OBP) in satellite communications," China

Commun., vol. 17, no. 1, pp. 140–150, Jan. 2020.
[19] M. K. Jaswal, D. Mallik, and M. Kaur, "Radiation hardened SEU tolerant Reed–Solomon encoder and decoder," in Proc. 3rd Int. Conf. Signal Process. Integr. Netw. (SPIN), Feb. 2016, pp. 417–420.
[20] G. C. Cardarilli, S. Pontarelli, M. Re, and A. Salsano, "Concurrent error detection in Reed–Solomon encoders and decoders," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 15, no. 7, pp. 842–846, Jul. 2007.
[21] S. Pontarelli et al., "Self checking circuit optimization by means of fault injection analysis: A case study on Reed–Solomon decoders," in Proc. 13th IEEE Int. On-Line Test. Symp. (IOLTS), Jul. 2007, pp. 194–196.
[22] J. K. Omura and J. L. Massey, "Computational method and apparatus for finite field arithmetic," U.S. Patent 4 587 627, May 6, 1986.
[23] J. Lacan and J. Fimes, "Systematic MDS erasure codes based on Vandermonde matrices," IEEE Commun. Lett., vol. 8, no. 9, pp. 570–572, Sep. 2004.
[24] M. S. Feali, A. Ahmadi, A. Hamidi, and M. Ahmadi, "Fixed-point arithmetic error analysis of sparse LU decomposition on FPGAs," in Proc. Int. Symp. Signals, Circuits Syst. (ISSCS), Jul. 2017, pp. 1–4.
[25] G. Wu, Y. Dou, J. Sun, and G. D. Peterson, "A high performance and memory efficient LU decomposer on FPGAs," IEEE Trans. Comput., vol. 61, no. 3, pp. 366–378, Mar. 2012.
[26] M. K. Jaiswal and N. Chandrachoodan, "FPGA-based high-performance and scalable block LU decomposition architecture," IEEE Trans. Comput., vol. 61, no. 1, pp. 60–72, Jan. 2012.
[27] H. Portillo Oquendo and P. Sánez Pacheco, "Bounds for the 1-norm of the inverses of some triangular matrices," Linear Algebra Appl., vol. 495, pp. 163–173, Apr. 2016.
[28] A. Ullah, P. Reviriego, and J. A. Maestro, "An efficient methodology for on-chip SEU injection in flip-flops for Xilinx FPGAs," IEEE Trans. Nucl. Sci., vol. 65, no. 4, pp. 989–996, Apr. 2018.