IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

FPGA Implementation of MBU Detection in Signal Processing Applications

[1]Dr.L.Malathi, [2] P.G Aswathi, [3]B.Kavinmalya, [4] S.Nivetha,[5] V.S.Miruthu bashini

[1]Assistant Professor, [2345] UG Scholar,

[1] Department of Electronics and Communication Engineering,

[1]Sri Ramakrishna Institute of Technology, Coimbatore, Tamilnadu, India

Abstract: SRAM-based FPGAs are vulnerable to radiation-induced errors, especially Multiple-Bit Upsets (MBUs), which can affect their reliability in space. Existing error correction methods often require too many resources or fail to handle burst errors effectively. Readback scrubbing is thought to be a useful technique for fixing mistakes in Field Programmable Gate Arrays (FPGAs) based on Static-RAM (SRAM). Nevertheless, the error correction %per unit area overhead of the existing systems is modest. In order to attain a high error correction %per unit area overhead, proposed work suggests two novel error detection/correction techniques that combines frame readback scrubbing with multi directionally applied Error Correction Codes (ECCs). According to experiments, the suggested approaches use up to 59.37% less area overhead than other state-of-the-art methods while having an excellent error correction percentage (above 99%), particularly for multi-bit upsets.

1.INTRODUCTION

Modern Field Programmable Gate Arrays (FPGAs) based on Static-RAM (SRAM) are becoming increasingly important in space applications because of their performance and operational capacity. Furthermore, after launch, these devices can be redesigned to accommodate different functional needs and modifications to the device environment [1]. However, as a result of technological advancements that have produced denser circuits, these gadgets are now susceptible to radiation effects known as Single Event Upsets (SEUs), which are frequent in space conditions.

SEUs have the potential to unintentionally alter the SRAM bit configuration, which could alter the implemented circuit & functionality [2]. A Single-Bit Upset (SBU) or single mistake occurs when SEUs only impact one bit. On the other hand, a multiple-bit upset occurs when multiple bits are impacted in succession.

To reduce SEUs in SRAM-based FPGAs, a number of strategies have been put forth. Hardware/spatial redundancy is the most popular solution [3]. Triple Modular Redundancy (TMR) [4] [5] replicates three times the hardware module to be protected and votes on their outputs, identifying the right results and a possible faulty module. However, this approach imposes a great overhead in terms of area and power consumption. Moreover, this method does not avoid error propagation if more than one component produces erroneous output. Duplication With Compare (DWC) [6] is an alternative approach to reduce the TMR overhead. It compares the output results of duplicated modules in order to identify the errors .Though it can start the appropriate processes, such a complete re-execution, it cannot fix them.

1.1 OVERVIEW OF MBU

A Multiple Bit Upset (MBU) happens when a single radiation event affects multiple nearby bits at the same time. This is more challenging to handle than SEUs because conventional error correction methods, which are typically designed for single-bit errors, may fail to detect or correct them. As electronic devices become smaller and more densely packed, MBUs are becoming increasingly common, making advanced error correction techniques crucial for ensuring reliability in environments like space, avionics, and nuclear applications.

1.2 PROBLEM STATEMENT

The increasing susceptibility of SRAM-based FPGAs to SEUs, especially MBUs, presents significant challenges in ensuring reliable operation in radiation-prone environments like space. Current error correction methodologies either impose excessive area and power overheads or fail to effectively address burst errors. There is a critical need for a solution that achieves a high error correction rate while minimizing resource consumption. The proposed H3 and P2H schemes aim to address these challenges with enhanced correction efficiency and reduced overhead..

2.LITERATURE SURVEY

Researchers have found that readback scrubbing is a useful technique for error correction in SRAM-based FPGAs [10]-[14]. Three types of readback.

The literature contains references to scrubbers. By directly comparing read frames with the golden copy, the first category makes defect detection possible [13]. By replacing the defective frame with the appropriate golden duplicate frame, the error is fixed. An extensive comparison with the golden copy is not used in the second category [14]. By comparing the online computed error detection codes (EDCs) with the initial ones that were calculated and externally stored for every frame, it finds the errors. The golden copy of the frame is used for the fault recovery, much like in the preceding category. The third category makes fault detection and computation possible.

Lanuzza et al. [12], hamming codes are applied to a data word that is obtained from frame bit interleaving in order to rectify burst errors in SRAM-based FPGAs. The interleaving technique improves rectification efficiency by decreasing the likelihood of several bit-faults occurring in a single data word. However, if a very high error correction efficiency is needed, this method might not be appropriate because it is restricted by the quantity of bit interleaving.

Argyrides et al. [10] provide the Matrix Code (MC) scheme, which combines parity and hamming codes to allow for the detection and correction of numerous mistakes in a frame for an FPGA design. A matrix of subwords is created from a frame word. Single Error Correction Double Error Detection (SECDED), parity codes for each column, and hamming codes for each row are used to repair errors. Because the ECC code is unable to detect errors that occur in more than two consecutive rows, this technique is ineffective at managing MBUs. Park and colleagues [11] suggest an integrated 2-D Hamming Product Code (2-D HPC) approach. Hammering codes created by organizing the the 2-D array& FPGA configuration frame. Thus, for every row and column of the 2-D array, hamming codes are calculated. This approach is ineffective at handling burst faults because, like the earlier work, it cannot identify more than two errors that occur in the same row or column.

3.EXSISTING METHODOLOGY

Error mitigation in SRAM-based FPGAs has relied on several established techniques to address the susceptibility of these devices to Single Event Upsets (SEUs), caused by radiation in space environments. These methodologies include:

Triple Modular Redundancy (TMR): This techniqueinvolves replicating the hardware module threefold and voting on their outputs. While effective in correcting single faults, it incurs a high area and power overhead. TMR fails to prevent error propagation when multiple faults occur in the replicated modules.

Duplication with Comparison (DWC): DWC duplicates modules and compares their outputs to detect faults. Although it reduces the area overhead compared to TMR, it cannot correct errors independently and instead triggers additional operations for error rectification.

Blind Scrubbing: This technique uses an externally saved golden copy to rewrite configuration frames on a regular basis. Although it successfully stops faults from spreading, it is unable to identify faults and depends on precise SEU frequency estimations to maximize scrubbing intervals. Error Correction Codes (ECCs), including parity and Hamming codes, are applied across rows and columns of configuration data using matrix and 2D Hamming Product Codes. They perform well for double error detection (DED) and single-bit error correction (SEC), but they have trouble with multi-bit upsets (MBUs), which impact nearby bits in a burst pattern. Especially in settings with high SEU rates, these approaches either have large resource overhead or fall short in addressing burst errors.

4.PROPOSED WORK DESCRIPTION

4.1 BITSTREAM INPUT

The process begins with the bitstream, which contains the necessary configuration data to program the FPGA. This data determines how the logic blocks, interconnections, and switches are set up to perform the intended functions.

4.2 ROW AND COLUMN PARITY GENERATION

Before storing the bitstream into the configuration memory, parity bits are generated for both rows and columns of the data. These parity bits introduce redundancy, which helps in detecting and correcting potential errors that may occur in memory during FPGA operation.

4.3 CONFIGURATION MEMORY

Before storing the bitstream into the configuration memory, parity bits are generated for both rows and columns of the data. These parity bits introduce redundancy, which helps in detecting and correcting potential errors that may occur in memory during FPGA operation.

4.4 FPGA TILE STRUCTURE

The FPGA consists of multiple tiles, each containing essential components for executing logic operations.
These include:

- Configuration Logic Block (CLB): Responsible for implementing the required logic functions.
- Switch Block: Manages the routing between different logic elements.
- Connection Block: Facilitates interconnections between routing resources and logic components to ensure smooth communication.

4.5 MULTIBIT ERROR DETECTION

To ensure reliable operation, a multi-bit error detection mechanism continuously monitors the configuration memory and logic tiles. Using the stored parity bits, this system can identify errors that affect multiple bits in the configuration data, which could otherwise compromise FPGA functionality.

4.6 MULTIBIT ERROR CORRECTION

Once errors are detected, an error correction mechanism is activated. This system applies Error Correction Codes (ECC) or utilizes the stored parity information to correct faults. By rectifying errors, the FPGA can maintain its configuration and continue functioning as intended.

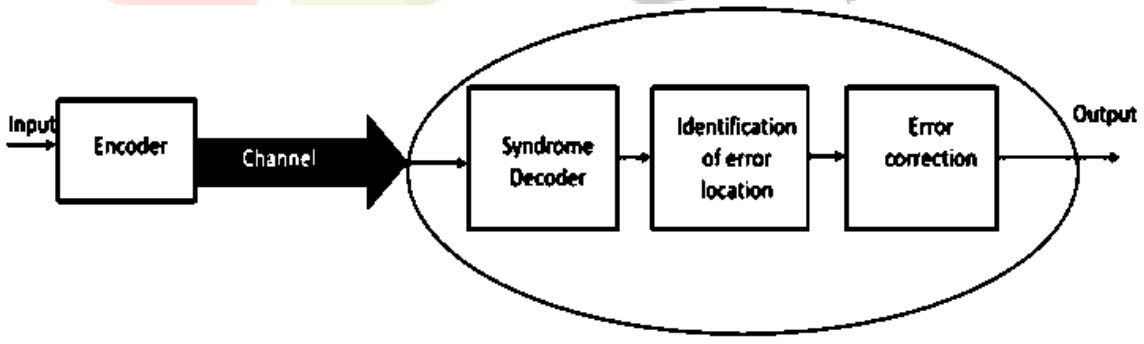

Fig. 1 Basic of MBU Detection

In fig 1, the input is given to encoder that encodes then its output is transferred over a channel to the receiver. In between prior to the receiver it ensures the error free message that is being transferred further for processing. That error is properly detected and it must be corrected to receive the original message. Hence the proposed work is analyze the proper methodologies for error-free messages. It is the major and very essential task for the exact communication in between transmitter and receiver.

4.7 PROPOSED WORK FLOW

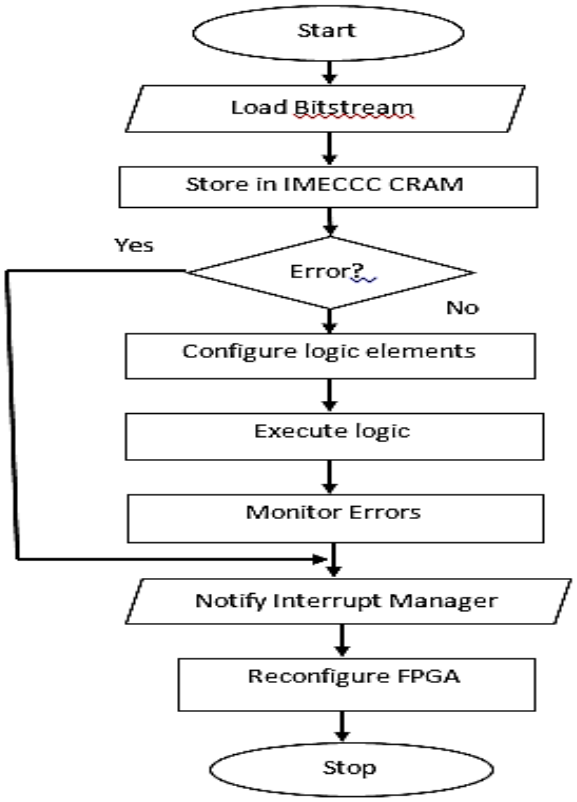

Fig. 2 Flow chart of MBU Detection

4.8 BLOCK DIAGRAM

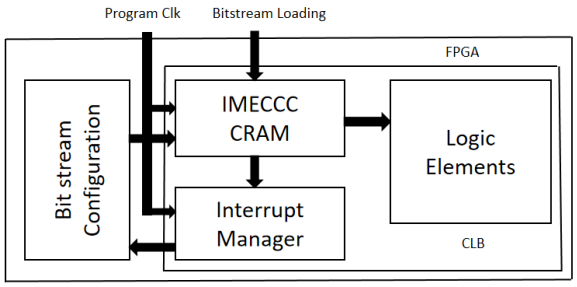

Fig. 3 Block Diagram of MBU Detection

Bitstream Configuration: This block handles the configuration data for the FPGA. Bitstream configuration is the process of loading a specific configuration (design or logic functions) into the FPGA. The bitstream data is essentially a binary file that configures the internal resources of the FPGA to implement the desired logic.

Program Clk (Program Clock): The program clock is a dedicated clock signal that synchronizes the bitstream loading process. This clock ensures the correct timing during configuration, enabling smooth and error-free loading of the bitstream data into the FPGA.

Bitstream Loading: This line indicates the flow of the configuration bitstream data into the FPGA. It is directed towards the IMECCC CRAM, where the bitstream is stored.

IMECCC CRAM: IMECCC stands for "Internal Memory Error Correction Code Circuitry". The CRAM (Configuration Random Access Memory) stores the configuration data of the FPGA. The IMECCC CRAM includes error correction features to detect and correct potential errors in the configuration memory, enhancing the reliability of the FPGA.

Interrupt Manager: This module handles interrupts that may occur during FPGA operation. If an error is detected by the IMECCC CRAM, the Interrupt Manager can trigger an interrupt signal. This interrupt can alert the system to take corrective action, such as reconfiguring a portion of the FPGA to restore proper functionality.

Logic Elements: This part represents the programmable logic elements within the FPGA, where actual user-defined logic functions are implemented. These logic elements execute the computations or functions defined by the bitstream configuration

5.SOFTWARE DESCRIPTION

Xilinx offers a range of software tools designed for FPGA and SoC development, each serving different aspects of the design process. Vivado Design Suite is the primary tool for FPGA and SoC design. It provides everything needed for synthesis, place and route, simulation, and debugging. It includes High-Level Synthesis (HLS) for converting C/C++ code into RTL, as well as an IP integrator for block-based design. Timing analysis and optimization are also integral features, making it a comprehensive solution for hardware development. Vitis Unified Software Platform is aimed at software development on Xilinx hardware, including FPGAs, SoCs, and ACAP devices. It allows developers to write software applications, optimize performance, and accelerate workloads using pre-built libraries.

Vitis also integrates with Peta Linux for embedded Linux applications, making it suitable for embedded system development. Peta Linux Tools are specifically designed for creating and managing Linux-based embedded systems on Xilinx SoCs. Developers can configure the Linux kernel, customize the root filesystem, and generate boot images. This tool is essential for deploying Linux-based applications on Zynq and Versal platforms.

Xilinx Model Composer is a model-based design tool integrated with MATLAB/Simulink. It is widely used for developing DSP and AI algorithms, enabling system simulation and validation directly within the Simulink environment. Previously, Xilinx provided the SDx toolchain, which included SDAccel, SDSoC, and SDSynth for high-level synthesis and software acceleration. However, these tools have been replaced by Vitis, which combines their functionalities into a single platform.

6.SIMULATION RESULTS

6.1 SCHEMATIC VIEW

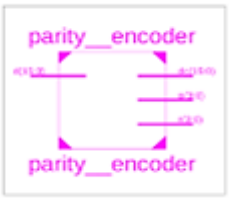

Fig. 4 Parity Decoder

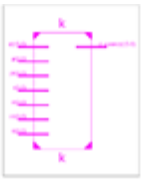

Fig. 5 Parity Encoder

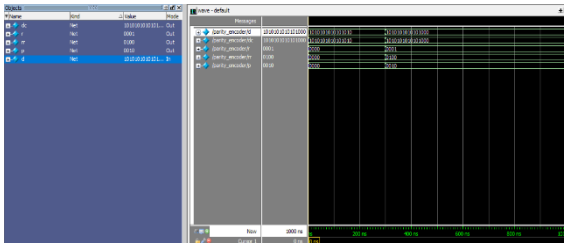

Fig. 6 Parity Encoder

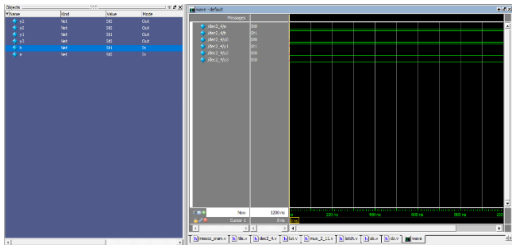

Fig. 7 Decoder

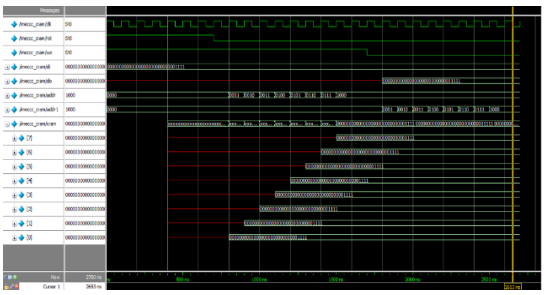

Fig. 8 IMECC CRAM

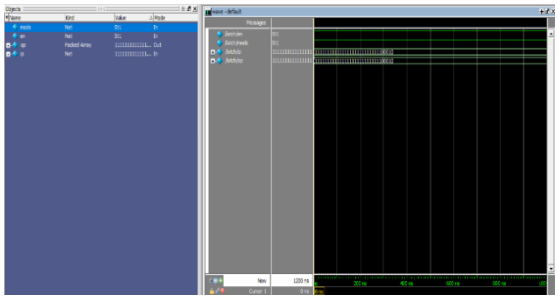

Fig. 9 Multiplexer

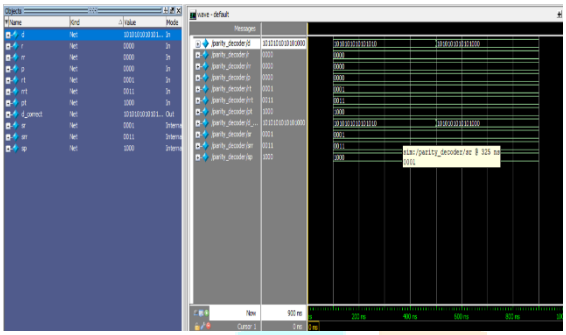

Fig. 10 Parity Decoder

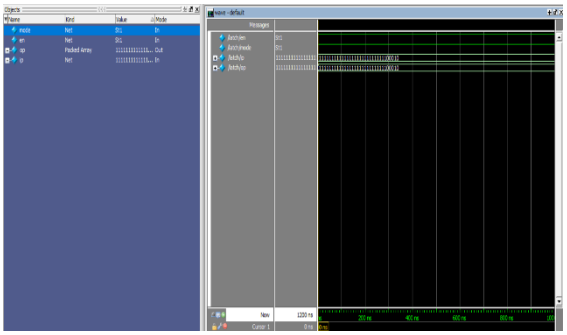

Fig. 11 Latch

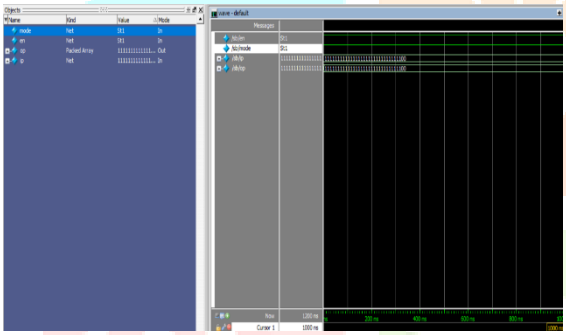

Fig. 12 Switch Block

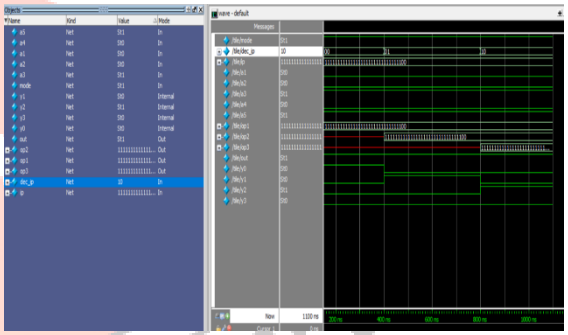

Fig. 13 Tile

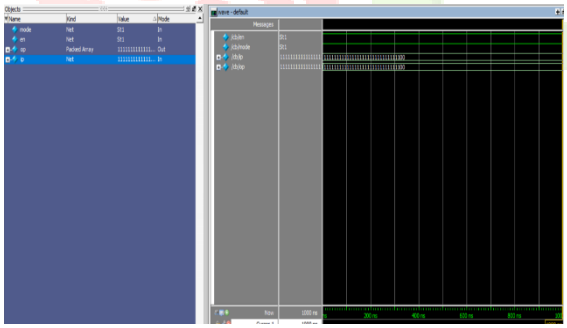

Fig. 14 Connection Block

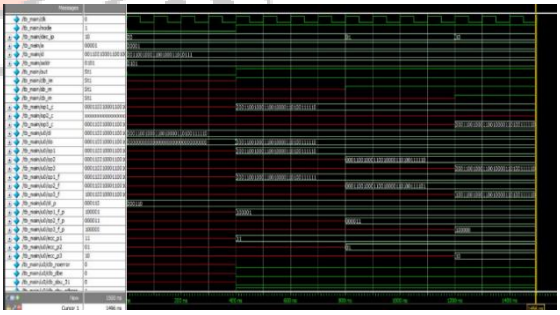

Fig. 15 Error Corrected Result

6.2 Report Analysis

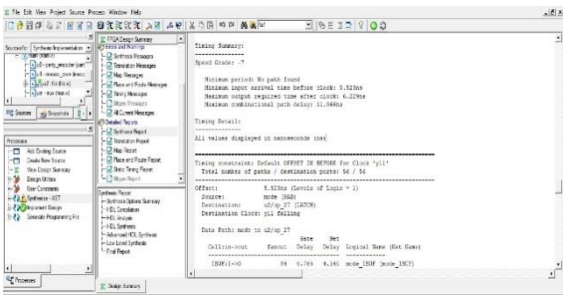

Fig. 16 Delay of output

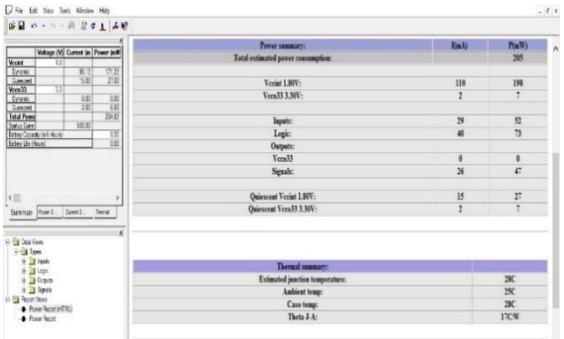

Fig. 17 Power of output

7. COMPARISON

The comparison table highlights the key differences between the work of S. P. Park et al. and the proposed approach in terms of error correction capabilities and error types. While S. P. Park's method primarily addresses Single Event Upsets (SEUs), the proposed work extends the correction capability to include Multi-Bit Upsets (MBUs), offering a more robust solution. Additionally, the error type coverage is broader in the proposed method, indicating an advancement in handling complex fault scenarios compared to the earlier technique.

Table 1.1 Comparison of Bit Detection

Parameters	S. P. Park et.al	Proposed work
Correction Capability	SEU	MBU
Error type	SEU	MBU

8. CONCLUSION

The two novel error detection/correction techniques are proposed that as follows, SBUs and MBUs in SRAM-based FPGAs can be mitigated by combining these techniques with frame readback scrubbing. Additionally, a suitable architecture that makes use of COTS FPGAs has been described for the implementation of the suggested schemes. These two systems are assessed and contrasted with the previous relevant studies in terms of their execution time ratio, error correction percentage, and ECC overhead. Based on the findings, we suggest applying the H3 method in settings where the error correction percentage is crucial and the SEU conditions are higher. However, because the P2H method has a modest error correction capability with a lower ECC overhead, it will work better in situations with a moderate SEU.

9. DISCUSSION

Proposed work can be used in developing adaptive error correction techniques. By analyzing error patterns in real-time, the system could predict and prevent potential failures before they occur. Additionally, integrating more efficient low-power error correction mechanisms would enhance FPGA performance in power-sensitive applications. Exploring advanced fault-tolerant architectures can further strengthen reliability, making FPGA systems more robust for aerospace, automotive, and industrial automation applications.

10. ACKNOWLEDGEMENT

Our sincere Thanks to Sri Ramakrishna Institute of Technology, Coimbatore, India for providing us the environment and support to complete our project in successful manner.

11.BIOGRAPHY

Dr. L. Malathi, She received her Ph.D. in the field of VLSI Design under Faculty of Information and Communication Engineering (ICE) from Anna University, Chennai in the year 2023. Currently she is working as Assistant Professor (Sl. Gr.) in Department of ECE at Sri Ramakrishna Institute of Technology, Coimbatore, Tamilnadu, India. She is having 16 years of teaching experience. She has presented and published papers in various National and International Conferences, Journals and patents. Her field of interest is VLSI, Digital Signal Processing and Embedded Systems. She is a lifetime member of professional bodies such as ISTE, IAENG and ACM.

P.G.Aswathi, B.Kavinmalya, V.S.Miruthubashini and S.Nivetha are UG scholars in Department of Electronics and Communication Engineering at Sri Ramakrishna Institute of Technology, Coimbatore, Tamilnadu, India.

12.REFERENCES

1. Sáánchez-Maciáán, P. Reviriego, and J. A. Maestro, "Combined SEU and SEFI protection for memories using orthogonal latin square codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 11, pp. 1933–1943, Nov. 2016.
2. Qi, L. Xiao, T. Wang, J. Li, and L. Li, "A highly reliable memory cell design combined with layout-level approach to tolerant single-event upsets," *IEEE Trans. Device Mater. Rel.*, vol. 16, no. 3, pp. 388–395, Sep. 2016.
3. H. B. Wang *et al.*, "An area efficient stacked latch design tolerant to SEU in 28 nm FDSOI technology," *IEEE Trans. Nucl. Sci.*, vol. 63, no. 6, pp. 3003–3009, Dec. 2016.
4. H. Ueno and K. Namba, "Construction of a soft error (SEU) hardened latch with high critical charge," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst.*, Sep. 2016, pp. 27–30.
5. S. Kumar, S. Chellappa, and L. T. Clark, "Temporal pulse-clocked multibit flip-flop mitigating SET and SEU," in *Proc. Int. Symp. Circuits Syst.*, 2015, pp. 814–817.
6. K. Katsarou and Y. Tsiatouhas, "Soft error immune latch under SEU related double-node charge collection," in *Proc. 21st Int. On-Line Test. Symp.*, 2015, pp. 46–49.
7. White and Xilinx Corporation, *Considerations Surrounding Single Event Effects in FPGAs, ASICs, and Processors*, ser. White Paper WP402, 2012.
8. Argyrides, D. Pradhan, and T. Kocak, "Matrix codes for reliable and cost efficient memory chips," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2011.
9. M. Lanuzza, P. Zicari, F. Frustaci, S. Perri, and P. Corsonello, "A selfhosting configuration management system to mitigate the impact of Radiation-Induced Multi-Bit Upsets in SRAM-based FPGAs," in *IEEE International Symposium on Industrial Electronics (ISIE'10)*, 2010.
10. J. Heiner, B. Sellers, M. Wirthlin, and J. Kalb,
11. "FPGA partial reconfiguration via configuration scrubbing," in *Field Programmable Logic and Applications (FPL'09)*, 2009.
12. Sarkar, F. Mueller, H. Ramaprasad, and S. Mohan, "Push-assisted migration of real-time tasks in multi-core processors," *SIGPLAN Not.*, 2009.
13. Bolchini, A. Miele, and M. D. Santambrogio, "TMR and Partial Dynamic Reconfiguration to mitigate SEU faults in FPGAs," in *Defect and Fault-Tolerance in VLSI Systems (DFT'07)*, 2007.
14. M. Berg, "The nasa goddard space flight center radiation effects and analysis group virtex 4 scrubber," *Xilinx Radiation Test Consortium*
15. *(XRTC) Meeting*, 2007.
16. Koren and C. Krishna, *Fault-tolerant systems*. Morgan Kaufmann, 2007.
17. Pratt, M. Caffrey, P. Graham, K. Morgan, and M. Wirthlin, "Improving FPGA Design Robustness with Partial TMR," in *44th Annual IEEE International Reliability Physics Symposium Proceedings*, 2006.
18. Xilinx Corporation, *Virtex FPGA Series Configuration and Readback*, ser. Application Note XAPP138, 2005.
19. Huang, L. Xu, and S. Member, "An efficient coding scheme for correcting triple storage node failures," in *4th Usenix Conference on File and Storage Technologies (FAST)*, 2005.
20. H. Quinn, P. Graham, J. Krone, M. Caffrey, and S. Rezgui, "Radiation induced multi-bit upsets in SRAM-based FPGAs," *IEEE Transactions on Nuclear Science*, 2005.
21. Ratter, *FPGAs on Mars*, ser. Journal n50. Xilin Technical Report, xCell Jornal, 2004.
22. C.Carmichael, M. Caffrey, and A. Salazar, "Correcting single-event upsets through virtex partial configuration," Xilinx, Tech. Rep., 2000.