A Review on Scrubbing Techniques in Static Random Access Memory Based Field Programmable Gate Array

O.E. Haruna[1*], I. Mafiana[2], A.M. Ogunleye[3], A.Y. Ihiabe[4], S.Magaji[5]

[1,2,3,4]*Centre for Satellite Technology Development (National Space Research and Development Agency), Abuja, Nigeria.*

[5]*Computer Engineering Department, Ahmadu Bello University, Nigeria*

Abstract: **Static Random Access Memory (SRAM) based Field Programmable Gate Array (FPGA) are semiconductor devices with large logic resources, programmable interconnects, and other resources making the device re-programmable thus, having broad applicability. SRAM-based FPGAs are sensitive to radiation induced Single Event Upset (SEU) within the configuration memory, whereby a fault as a result of radiation strike from high energetic particles causes the logic state of the SRAM cell to flip. The configuration memory defines the operation of the Configurable Logic Blocks (CLBs), routing resources, Input-Output Blocks (IOBs), and other FPGA resources and upset in the configuration memory can change the operation of the circuit. Therefore configuration memory scrubbing is a solution to mitigate against SEU. In this paper we present a review of existing scrubbing techniques, the parameters considered, results obtained and possible modifications are suggested as well.**

Keywords: **SRAM, FPGA, Configuration memory, Scrubbing, Single event upset.**

I. INTRODUCTION

SRAM-based FPGAs are Complementary Metal Oxide Semiconductor (CMOS) devices having thousands to millions system logic gates with hundreds of millions of configuration bits dominating the SRAM cells and the cells are composed of transistors linked with interconnect wires [11]. Unlike Application Specific Integrated Circuits (ASICs), whose functions cannot be altered after fabrication, SRAM-based FPGAs have the advantage of being reprogrammed providing high computational density and efficiency for many computing applications by allowing circuits to be customized to any application of interest [22].FPGAs are versatile devices that allow a function to be implemented by mapping it into the FPGA's pre-existing logic resources. The mapping is referred to as its Configuration [3], SRAM-based FPGAs are more prone to soft errors in the form of SEU since radiation strike in the configuration memory has a permanent effect on the functionality of the mapped design [17].

The presence of high energy protons, heavy ions, and galactic cosmic rays in space and other radiation environment cause a number of problems for electronics including FPGAs. This radiation can induce a number of negative effects including upsets in the internal state of the device and can cause several Problems in FPGA-based systems. As mentioned earlier, SEUs can corrupt the configuration memory of the device causing the design configured on the devices to operate incorrectly. In SRAM FPGAs, SEUs can affect the configuration and provoke errors that remain until the device is reconfigured [23]. These errors which could be either SEU or Multiple Bit Upset (MBU) within the configuration memory are mitigated through the process called scrubbing where the correct logic bit is written back into the SRAM cell(s) [18].

II. BACKGROUND

A. The FPGA Architecture

FPGA can be divided generally into application layer and configuration layer which is also referred to as the configuration memory layer. The application layer includes the logic and memory elements managed by user's application and configuration layer includes the logic bits and memory elements that allows configuring the logic and routing resources in the application layer. Figure 1 shows a conceptual layers of a Field Programmable Gate Array.

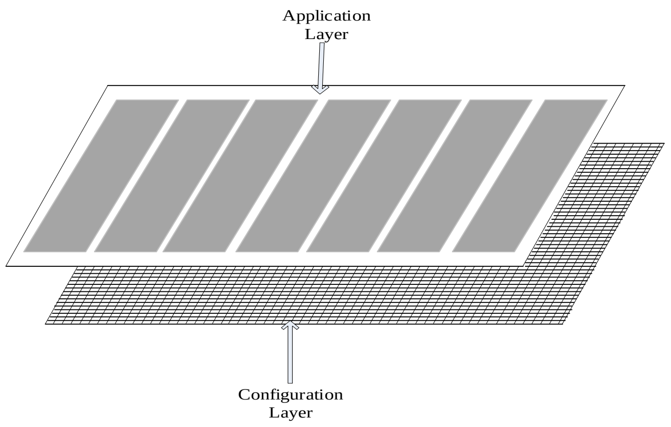

Figure 1: Conceptual Layers of an FPGA

B. Application Layer

The application layer comprises the logic, memory and input/output resources for user application. The layer implements Configurable Logic Blocks (CLBs), which can be configured to implement any user sequential or combinational circuit and provides the basic logic and storage functionality for a target application design. FPGAs are typically made up of highly configurable logic blocks containing Look up Tables (LUTs) that defines logic functions and registers used for sequential logic. Each CLB in turn consists of several slices, and each slicecontains LUTs, registers and carry logic [13].

C. Configuration Layer

The configuration layer comprises the configuration memory otherwise known as bitstream and associated access ports and control logic. The layer consist of access ports for reading and writing from/to configuration memory The configuration port is internally connected to a built-in circuitry that decodes the configuration data providing read or write access to the configuration memory [8]. The functionality of an FPGA is dependent on the contents of its configuration memory [4].

D. The SRAM cell Technology

Fundamental technology of FPGA consists of arrays of logic elements and registers, along with a configurable interconnection matrix. The current shrinking of device size helps FPGAs to be faster and denser, but at the same time also make them less reliable due to shrinking transistor size, the threshold voltage required to switch them, also shrinking makes transistors more prone to radiation induced errors [19]. SRAM based FPGA will lose their configuration when power is removed, therefore, SRAM devices require a non-volatile configuration memory unit to store configuration data. This data must be loaded into the SRAM device after the FPGA is powered-on before use [7].

The SRAM Cell Architecture

The SRAM cell widely used is the full CMOS 6-Transistor (6T) memory cell. The 6T SRAM is a type of semiconductor memory that uses bi-stable latching circuitry to store each bit. The SRAM cell consists of two inverters P1, N1 and P2, N2 alongside with two access Metal Oxide Semiconductor Field Effect Transistor (MOSFET) N3 and N4 as shown in Figure 2.

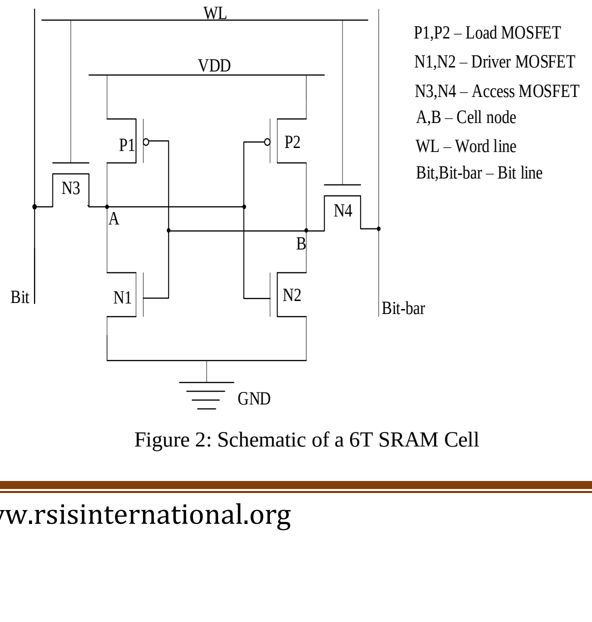

Figure 2: Schematic of a 6T SRAM Cell

The 6T SRAM cell operates in three different modes. The first mode is the standby mode (circuit is idle) the word line is not triggered therefore the word line is 0V, so the pass transistors N3 and N4 which connect 6T cell from bit lines are turned off. It means that cell cannot be accessed. The two cross coupled inverters formed by N1-N2 will continue to feedback each other as long as they are connected to the supply, and data will hold in the latch.

The second mode is the read mode (data has been requested). In this mode, the word line is triggered therefore the word line is “1”, so word line enables both the access transistors which connects the cell from the bit lines. Now values stored in node “A” or “B” are transferred to the bit lines to be accessed, either the value in node “A” or “B” is accessed. Assuming that “1” is stored at node A so bit line bar will discharge through the driver transistor N1 and the bit line will be pull up through the load transistor P1 toward VDD, hence a logic “1” is confirmed. SRAM cell technology ensures read stability therefore data are not disrupted during read operation.

The third mode is the write mode (updating the content). In this mode, if “1” is initially stored in the cell and needs to be written with a “0”, the bit line is lowered to 0V and bit bar is raised to VDD, a cell is selected by raising the word line to VDD, for SRAM to operate in write mode it must have write-ability which is minimum bit line voltage required to flip the state of the cell.

The power consumption of SRAM varies widely depending on how frequently it is accessed, it can be high when used at high frequencies. On the other hand, SRAM used at a somewhat slower pace at lower frequency draws very little power and can have a nearly negligible power consumption when sitting idle [2].

III. SCRUBBING TECHNIQUES

Carmichael & Brinkley Jr., [6] proposed a technique which involved the use of redundancy such as TMR were three identical FPGAs and a voting circuit where implemented inside or outside the FPGA and only sensitive logic resources were triplicated. An error detection scheme whereby during readback process a Cyclic Redundancy Check (CRC) code was generated for a portion of the configuration data containing sensitive resources and a second readback of the same portion of the configuration data was performed to recalculate the CRC code and a comparison between the first and second checksum was performed for error detection. Another error detection technique involved a self readback process of the configuration memory and a comparison is performed on a duplicate copy. The first correction technique involved reconfiguration of the frame only with the upset bit rather than the entire design and the second technique involved a blind scrubbing, where a total reconfiguration of the configuration memory is performed periodically whether an upset is detected or not. However, frequent reconfiguration of the configuration memory from a duplicate copy irrespective of error increased the energy consumption of the scrubbing process.

Jonathan et al., [12] proposed an error detection technique for FPGAs called Duplication with Compare (DWC). In the work FPGA bitstream was duplicated together with the signal nets for a full duplication operation to provide the greatest coverage for error detection and the redundant copy was stored in an external radiation memory, and comparator insertion for external system was deployed since it was placed at the final output of the two modules because it was less complex to be implemented and also saved cost. In an event of upset which was implemented in the work using a fault injection simulator to create SEUs, the bitstream of the original module was compared with the golden copy by the comparator and discrepancy at the output of the two module, the comparator signals the system with an error flag indicated the presences of error. However, this mechanism was only able to detect errors and there was a delay from the time an upset occurs to the time when it was detected.

Heiner et al., [10] proposed a technique that allowed Partial Reconfiguration (PR) to be used together with configuration memory scrubbing from a golden memory that explores external configuration interface. The scrubber utilizes portion a of the FPGA, performs the necessary operations to reconfigure a portion of the design while continuously scrubbing the entire FPGA. The system isolated the design into subsections by dividing it into static reconfigurable region which contains all the design critical bitstream that do not change during PR and dynamic reconfigurable region which contained non-critical component for re-configurability. The PR allowed portions of the design to be modified or upgraded after deployment. Anytime PR is performed, the golden configuration memory bitstream in the external memory no longer matches the logic within the FPGA therefore the external memory must also be updated to reflect the changes in the FPGA else the scrubber will detect many discrepancies between the expected bitstream and the bitstream on the FPGA. However, PR and scrubbing cannot have access to the configuration interface since PR will try to write to the configuration memory while the scrubber tries to read and write to it as well.

Lanuzza et al., [14] proposed a configuration memory scrubbing core used for internal detection and correction of radiation induced configuration single and multiple bit errors without requiring external scrubbing. The proposed approach makes FPGA devices able to self-repair SEU, whereby detection and correction were performed inside the FPGA chip by exploiting the internal readback port for an integrity check of the configuration memory using an error detection and correction (EDAC) Circuit. The work presented a compression scheme to reduce the memory resources, needed to store correction reference codes, thus making the technique easily scalable. The EDAC codes are pre-computed and stored in each sensitive frame capable of correcting up to quadruple bit upset, and error detection and correction process is performed by reading back sensitive frames. The sensitive frames are scanned one by one through internal readback operations and for each frame an EDAC code are internally computed and compared to the corresponding golden reference stored in the EDAC code memory. The check operations are repeated cyclically by incrementing the frame address till the last frame was checked and the process starts all over. However, the EDAC code will be corrupted by SEU since the circuit was embedded in the FPGA and those not have any form of redundancy.

Herrera-Alzu & Vallejo, [9] proposed a self-reference scrubber for FPGA TMR system. An external selectable microprocessor access port interfaces the bidirectional links with the three FPGAs and a three-way external voter was introduce for majority voting which was implemented in a radiation-hardened device. The scrubber periodically detects both single and multiple error and correction was performed by peer frame comparison without needing to access a golden configuration memory. Three identical configuration memory frames are readback from the FPGA and the frame triplets are compared on a byte basis if frames are found identical, readback resumes and new frames are read back. If there was discrepancy in one of the frame, the dirty frame was scrubbed with a correct frame from one of the other two frames. The work did not used a golden memory for correction purposes it was mainly used for power-on configuration and single event functional interrupt when a SEU causes a significant change in the functional operation of the FPGA. However, the use of golden memory in an event of a single event functional interrupt incurs extra power consumption since an external interface was used.

Akagic & Amano [1] presented a study of designing highly adaptable coprocessors using CRC on an FPGA bus. The work introduced an FPGA based architecture for accelerating CRC with 32 and 64 bits Generator Polynomials (GPs) in which a circuit was designed alongside a module for generating content of LUT. In order to reduce the number of logic resources used, the work explored possibility of using partial reconfiguration to minimize the area overhead when CRC standards are changed. The co-processor adapts to new CRC standards through the designed circuit. In the table generator module, in every clock cycle the counter generates input data, these data passes through the input reflection unit or the remainder generator, depending on the CRC standard and the input reflection unit reflects data bits by swapping them around its center and the data was shifted by the left depending on a data-path width and the length of the slice. The remainder generator unit performed long division operation. Number of cycles required to generate one remainder depends on the offset of a byte in an input data. At the end, remainder is reflected or forwarded to output, depending on a CRC standard. However the CRC adaptable co-processor affects the area utilization despite the reduced resources used this is because integrating the table generation module with the CRC module requires additional area in the FPGA floor plan.

Nazar et al., [15] proposed a shifted scrubbing technique whenever an error occurred the scrubber starts scrubbing the associated partition where the critical bit(s) are flipped. A partitioned designed was use for the scrubbing technique and each partition was a Dual Modular Redundancy (DMR) as an error detection mechanism which indicated error, the scrubber starts scrubbing the associated partition after an upset was been detected. A fault injection platform was also implemented. The concept explored in this work was that scrubbing does not need to start at the first configuration memory frame since different regions have different concentrations of critical bits, one can find the optimum starting frame that has more upset in the region with high number of critical bits that also minimizes the time to repair, since the work was partitioned based on the sensitivity of the bits representing the mapped design. The Drawback of the proposed technique was that the scrubber do not have knowledge of the optimum starting frame for each partition in an event of a single event upset thereby increasing error detection time as the scrubber reads back the frames to get this information which will consequently impact negatively to the correction time.

Vera et al., [21] implemented an error detection and correction code core for femto architecture and its implementation does not required the use of a golden bitstream for scrubbing and also the method those not required any modification at the architectural level of the FPGA. The femto was a unit that controlled the scrubber and fetched data or command from an interface which supported a limited set of instructions focused on moving data from and to the memory under scrubbing. The work adopted a methodology of using Bose and ray-ChaudHuri (BCH) codes for error correction which involved addition of parity bits. Each frame is added a random sequence of 1’s and 0’s prior to encoding with the BCH coder to produced n check bits. The BCH encoder/decoder generated 57 check bits per frame and a CRC bit was also calculated and stored in a golden frame together with the BCH code for an uncorrupted frame. During the device operation where the frames were subjected to radiation, the scrubber reads the frame and correct any error using the BCH code, afterwards the CRC code was recalculated for the frame and if there was a mismatch with the reference CRC code in the frame, then this signifies an error in the frame and hence the frame has to be updated from a programmable read only memory which implied a penalty on latency and power consumption. However, implementing BCH encoder/decoder increase error correction latency since these added to the complexity of the scrubber.

Rao et al., [17] proposed a scrubbing scheme which reconstructs erroneous configuration memory frame based on the concept of erasure codes for MBU in the configuration frames of FPGAs. The erasure codes recovered the original frame when some of the bits were flipped. The scrubbing technique was implemented as a soft module alongside with the user design and both were mapped into the FPGA. The work employed a low-cost interleaved two-dimensional (I2D) parity technique to detect MBUs in the configuration memory frames of the FPGA. The interleaving distance was improved based on the actual MBU patterns because each horizontal parity bit is the exclusive-OR operation of the bits in the multiple row, all rows that were separated by a constant distance forms an interleaving group and all the bits within that group are covered by only one horizontal parity bit. A scrubber unit periodically checks the configuration frames to detect possible erroneous frames. Once an error was detected by assuming that the erroneous frame was erased, its contents were reconstructed using an erasure code by computing an exclusive-OR operation of all bits in the temporary block that were initialized with zero bits by the recovery unit, thus, the temporary block was written into the erroneous frame. The encoding of the erasure code was done once offline in advance during when the design was mapped into to the FPGA device. However, the erasure codes are also be subjected to SEU since they can also embedded inside the FPGA.

Jing et al., [11] proposed a Duplication with Recovery (DWR) technique to correct soft error in routing resources, which contributed to the majority of soft errors in FPGAs and are most vulnerable to SEU because they occupy most area and seventy to ninety percent of the configuration bits were attributed to the routing resources. DWR leveraged redundant resources and enables fault recovery. The work required a the fault controlling circuitry; which controlled the fault signal on site in the routing Programmable Interconnect Points (PIPs), fault recovery structure; which recovered the correct signal with the duplication before the sink logic, and net duplication; which offered the opportunity for signal recovery. The duplicated nets were routed closely to its original net for convergence purposes. The work showed that the protection using limited resources can be sufficient for non-critical mission application. However, the technique only focused on mitigating SEUs in the routing resources indicting partial mitigation technique.

Tonfat et al., [20] proposed a novel scrubbing technique using internal frame redundancy called FLR-scrubbing. It was based on having each redundant TMR domain with the same configuration frame information to allow vote out the frames copies to restore themselves. The scrubber reads the first frame of each region then executes a bit level majority voting on the three frames, the scrubbing technique enabled the self-correction of faulty frames using the information of the other two copies. In the methodology adopted in the work, the readback and scrubbing process were complementary because the readback process generated the correct frame that the scrubbing process needed for a possible write operation. Implementation results demonstrated minimum area and energy consumption overhead when compared to blind scrubbing technique, the scrubbing used the Internal Configuration Access Port (ICAP). However, the voting mechanism for correction was on a bit level basis which incurred longer correction time which led to increase energy consumption as the whole configuration memory frame has to be readback on bit level attempting to performing voting without any error detection mechanism.

Wirthlin & Harding, [23] presented a novel hybrid configuration scrubbing for the Xilinx 7-Series FPGAs by exploiting the on-chip frame Error Correction Code (ECC). A dedicated non-configurable logic was built into the FPGA to compute a check word for each frame during configuration readback. The scrubber continuously reads configuration frames and computes a syndrome for the readback frame. The syndrome was compared with the internal ECC word of the frame determined whether the frame was error free, contains a single error or multi-bit error. This process continues through all frames of the configuration bitstream. After scanning the all block of frames, the process was repeated with the first frame in the configuration. If a single-bit error was found, the internal ECC circuitry computes the location of the error and corrects the upset. The internal scan unit then writes the corrected frame back into the configuration memory. Some multi-bit errors (three bits or more) within a frame may not be detected by the ECC. To detect this condition, a global CRC was provided for the entire set of frames. This CRC was recomputed during each full scan of the configuration memory and compared against an internal global CRC. If a multi-bit error occurred that was not detected by the individual frame ECC, the recomputed CRC will differ from the global CRC signifying that an undetected error exists somewhere in the configuration memory, therefore, the scrubbing system does not know the location of upset frame and the scrubber scrubs every frame. However, when multi-bit error was continuously detected this resulted to increase energy consumption because continuous device configuration will be required.

Cannon et al., [5] presented strategies to mitigate against the effect of Single Effect Upsets (SEUs), three strategies was presented: increased routing, incremental placement and striping. The presence of single CRAM (Configuration Memory) bits in the mitigation designs limits the effectiveness of TMR by causing TMR failures. This place a limit in the maximum achievable improvement TMR with repair can provide. The paper identified the underlying architectural causes for these bits that causes Common Mode Failures (CMF) as well as introduced mitigation techniques to address their problems. An incremental routing strategy provides fine-grain control over the process and is implemented after the routing stage. However it is difficult to create a router that would have the same quality as the one provided by the vendor, which is a draw back for the incremental routing strategy. The incremental placement strategy is performed in between the placement step and routing step, incremental placement is capable preserving most of the optimization from placement and routing is also performed using vendors tools unlike incremental routing. Striping is different from the incremental techniques discuss, the technique requires no vendors tool and separate flow. The strategies where tested using both fault-injection and a wide spectrum neutron beam with the best technique offering a 400x reduction to the designs sensitive neutron cross-section. Results showed that no single bits caused failure and multi-cell upsets were the main cause of failure for these strategies.

Zhang et al., [25] proposed a scrubbing method based on Triple Modular Redundancy (TMR) for Static Random Access Memory (SRAM) based Field Programmable Gate Arrays (FPBGA). The method improves the reliability of TMR design by reducing the possibility of Single Effect Upset (SUE) accumulation. A fault tolerant technique was developed to improve the reliability of SRAM-Based FPGA in space applications, the technique implemented a method other than the conventional TMR and scrubbing method, but a hybrid technique that combine this two method to ensure effective and fast repairs of the SEU in TMR design. The method combined detecting scrubbing and Error Correction Code (ECC), the syndrome value in one frame including the ECC bits is calculated. This gives a confirmation whether an SEU has occurred by reading the frame in the configuration memory. As a result additional space is not required because the original correct frames need not to be stored. The majority voter which is a TMR method and scrubber which is also a scrubbing method are put together to form the hybrid scheme. The voter is optimized that not only can filter the output of error module but also it can identify which module is wrong. The proposed method was validated through fault injection, and experimental results showed that the proposed scrubbing method can improve the reliability of TMR in SRAM FPGA. However the authors has not put into consideration the processing time and speed of SEUs repairs, additional process in the hybrid scheme cam increase the computational time of repair.

Zheng et al., [26] proposed a rapid scrubbing technique that ensures early scrubbing when SEU occurs, the technique applies position-aware Duplication with Compare (DWC) on the critical circuit that reduces the redundancy cost, and links the application circuit with configuration frames that enables the error locating in a greatly reduced number of configurations frames. The rapid scrubbing technique was achieved by first identifying the modules unit with high sensitivity in the implemented circuit. Secondly, sensitive modules are duplicated and positions of the modules are obtained based on their configuration frames, creating a link between the application circuit and configuration frames. The information of this link is kept in the block RAM for scrubbing use. Thirdly, frame generator hardware is added to translate the output to a configuration memory address. Fault injection based evaluation on a Xilinx Kintex-7 FPGA was used to evaluate the performance of the technique, results showed that the technique can deliver an average of 45% Mean Time To Detect (MTTD) and 16% Mean Time To Failure (MTTF) improvement with little cost when compared to conventional scrubbing techniques

IV. CONCLUSION

The previous section has described a couple of scrubbing techniques in SRAM-based FPGA. The parameters and methodology considered were highlighted and the results obtained. From the foregoing, improvements can be made by combining different scrubbing methodologies and scrubbing architecture by location. Different scrubbing techniques imposes an overhead either in terms of large area space, power consumption, scrubbing time, complexity, reliability and cost. In other to obtain an optimized result, choosing the right methodology for an application typically implies a trade off in terms of these parameters mentioned.

REFERENCES

[1]. Akagic, A., & Amano, H. (2012). *A study of adaptable co-processors for cyclic redundancy check on an FPGA.* Paper presented at the 2012 International Conference on Field-Programmable Technology (FPT), 119-124.

[2]. Aishwarya, S., & Mahendran, G. (2016). *Multiple Bit Upset correction in SRAM based FPGA using self repairable Erasure codes.* Paper presented at the International Conference on Emerging Engineering Trends and science (ICEETS-2016), 356-362.

[3]. Berg, M., Poivey, C., Petrick, D., Espinosa, D., Lesea, A., LaBel, K., .Phan, A. (2008). Effectiveness of internal vs. external SEU scrubbing mitigation strategies in a Xilinx FPGA: Design, test, and analysis. 1-8.

[4]. Bolchini, C., Miele, A., & Sandionigi, C. (2011). A novel design methodology for implementing reliability-aware systems on SRAM-based FPGAs. *IEEE Transactions on Computers, 60*(12), 1744-1758.

[5]. Cannon, M. J., Keller, A. M., Rowberry, H. C., Thurlow, C. A., Pérez-Celis, A., & Wirthlin, M. J. (2018). Strategies for removing common mode failures from TMR designs deployed on SRAM FPGAs. *IEEE Transactions on Nuclear Science*, *66*(1), 207-215.

[6]. Carmichael, C. H., & Brinkley Jr, P. E. (2006). Techniques for mitigating, detecting and correcting single event upset effects in systems using SRAM-based field programmable gate arrays (pp. 1-26): Google Patents.

[7]. Codinachs, D. M., & Weigand, R. (2009). Overview of FPGA activities in the European Space Agency. *III National School, 23rd April*, 1-55.

[8]. Herrera-Alzu, I., & Lopez, V. M. (2013). Design techniques for Xilinx Virtex FPGA configuration memory scrubbers. *IEEE Transactions on Nuclear Science, 60*(1), 376-385.

[9]. Herrera-Alzu, I., & Vallejo, M. L. (2011). Self-reference scrubber for TMR systems based on xilinx virtex FPGAs. *Integrated Circuit and System Design. Power and Timing Modeling, Optimization, and Simulation*, 133-142.

[10]. Heiner, J., Sellers, B., Wirthlin, M., & Kalb, J. (2009). *FPGA partial reconfiguration via configuration scrubbing.* Paper presented at the International Conference on Field Programmable Logic and Applications, FPL2009. , 99-104.

[11]. Jing, N., Zhou, J., Jiang, J., Chen, X., He, W., & Mao, Z. (2015). *Redundancy based Interconnect Duplication to Mitigate Soft Errors in SRAM-based FPGAs.* Paper presented at the Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, 764-769.

[12]. Jonathan, J., Howes, W., Wirthlin, M., McMurtrey, D. L., Caffrey, M., Graham, P., & Morgan, K. (2008). *Using duplication with compare for on-line error detection in FPGA-based designs.* Paper presented at the Aerospace Conference, 2008 IEEE, 1-11.

[13]. Johnson, J., & Wirthlin, M. (2009). Voter Insertion Techniques for Fault Tolerant FPGA Design. *I/UCRC Program of the National Science Foundation under Grant No. 0801876*, 1-8.

[14]. Lanuzza, M., Zicari, P., Frustaci, F., Perri, S., & Corsonello, P. (2010). Exploiting self-reconfiguration capability to improve SRAM-based FPGA robustness in space and avionics applications. *ACM Transactions on Reconfigurable Technology and Systems (TRETS), 4*(1), 1-22.

[15]. Nazar, G. L., Santos, L. P., & Carro, L. (2013). *Accelerated FPGA repair through shifted scrubbing.* Paper presented at the 2013 23rd International Conference on Field Programmable Logic and Applications (FPL), 1-6.

[16]. Nigam, A. K., Singh, S., & Tiwari, A. (2015). 6T SRAM Cell: Design And Analysis. *Intl J Engg Sci Adv Research, 1*(3), 125-127.

[17]. Rao, P., Ebrahimi, M., Seyyedi, R., & Tahoori, M. B. (2014). *Protecting SRAM-based FPGAs against multiple bit upsets using erasure codes.* Paper presented at the IEEE Design Automation Conference (DAC), 2014 51st ACM/EDAC, 1-6.

[18]. Sanchez, C. A., Entrena, L., & Garcia-Valderas, M. (2015). *Partial TMR in FPGAs Using Approximate Logic Circuits.* Paper presented at the 2015 15th European Conference on Radiation and Its Effects on Components and Systems (RADECS), Spain, 1-4.

[19]. Tiwari, A., & Tomko, K. A. (2005). Enhanced Reliability of Finite-State Machines in FPGA Through Efficient Fault Detection and Correction. *IEEE Transactions on Reliability, 54*(3), 1-9.

[20]. Tonfat, J., Fenanda, L. K., Paolo, R., & Ricardo, R. (2015). *Energy efficient frame-level redundancy scrubbing technique for SRAM-based FPGA*s. *IEEE Transactions on Nuclear Science, 62*(6), 3080-3087.

[21]. Vera, G. A., Ardalan, S., Yao, X., & Avery, K. (2013). Fast Local Scrubbing for FPGA's Configuration Memory. *Aerospace Research Central*, 1-21.

[22]. Wirthlin, M. (2015). High-Reliability FPGA-Based Systems: Space, High-Energy Physics, and Beyond. *Proceedings of the IEEE, 103*(3), 379-389.

[23]. Wirthlin, M., & Harding, A. (2016). Hybrid Configuration Scrubbing for Xilinx 7-Series FPGAs. *FPGAs and Parallel Architectures for Aerospace Applications* (pp. 91-101): Springer.

[24]. Wirthlin, M. J., Keller, A. M., McCloskey, C., Ridd, P., Lee, D., & Draper, J. (2016). *SEU Mitigation and Validation of the LEON3 Soft Processor Using Triple Modular Redundancy for Space Processing.* Paper presented at the Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 205-214.

[25]. Zhang, R. S., Xiao, L. Y., Cao, X. B., Li, J., Li, J. Q., & Li, L. Z. A Fast Scrubbing Method Based on Triple Modular Redundancy for SRAM-Based FPGAs. In *2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)* (pp. 1-3). IEEE.

[26]. Zheng, S., You, H., He, G., Wang, Q., Si, T., Jiang, J., ... & Jing, N. (2019, May). A Rapid Scrubbing Technique for SEU Mitigation on SRAM-Based FPGAs. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*(pp. 1-5). IEEE.