A SURVEY ONFAULTS AND MITIGATION TECHNIQUES IN FPGAS

Radwa M. Tawfeek
Benha Faculty of Engineering
Benha University
Benha, Egypt
radwa.tawfeek@bhit.bu.edu.eg.

Mohamed G. Egila
Electronics Research Institute
Cairo, Egypt
mohamed.gamal@eri.sci.eg

Yousra Alkabani and I. M. Hafez
Faculty of Engineering
Ain Shams University
Cairo, Egypt
yousra.alkabani@eng.asu.edu.eg,ismail_hafez@eng.asu.edu.eg

Abstract:**Field Programmable Gate Arrays (FPGAs) are sensitive to upsets that occur in aerospace. Drastic device shrinkage, power supply reduction, and increasing operating speeds significantly reduce noise margins and thus reliability; hence the rate of transient and intermittent faults is increased in modern electronic systems. Improving the fault tolerance of reconfigurable devices is increasingly important in domains ranging from mission critical embedded applications to the use of FPGAs in physically remote environments such as satellites. This paper shows the sources of radiation in the space and their effects on FPGAs. Also, main mitigation techniques used to recover from faults in FPGAs are explored.**

Index Terms: Fault-Tolerance, FPGA, Radiation, Faults, Redundancy, Scrubbing.

1. INTRODUCTION

FPGAs are becoming a popular target for processing and communications in space systems. FPGAs provide good performance for Digital Signal Processing (DSP) and communications applications often used by satellites. The reconfiguration capability of FPGAs also allows the circuits implemented to be changed in flight for later upgrades, bug fixes, and to add extra functionality.

Like all semiconductor devices, anFPGA can be affected by faults at various stages of its lifetime. While most defects appear immediately following fabrication, occasionally, after extended periods of device use, operational faults can affect in-service programmable components[1]. Unfortunately, the harsh space environment makes processing using standard SRAM based FPGAs difficult. Outside the atmosphere of the Earth, there is a large amount of radiation that may interfere with the electronics of a spacecraft. Memory cells are especially susceptible to the effects of this radiation. Since SRAM-based FPGAs are based on large arrays of memory cells, they are particularly susceptible to radiation-induced upsets, called single event upsets (SEUs). FPGA systems exposed to harsh radiation environments, can suffer from faults, either transient and/or permanent. Transient faults are temporary faults usually caused by Single Event Effects (SEEs), and are mitigated via full or partial reconfiguration of the configuration memory. Permanent faults can also occur, due to the Total Ionizing Dose (TID) the device is exposed to, or aging of the device. Permanent faults are mitigated by relocation of the application on the FPGA[2].

2. SOURCES OF RADIATION

Radiation can be defined as the propagation of energy through matter or space. Radiation can be electromagnetic waves, or energetic particles.The radiation environment is composed of different particles generated from sun and stars activities[3]. Energetic particles are particles with energies that range from several *KeV* to *GeV* and beyond. There are main radiation sources and secondary sources as shown in Figure 1 and discussed in the following subsections.

2.1 The PrimarySources

Trapped Radiation: The Earth's magnetic field is responsible for trapping particles. Since the Earth's magnetic field is not symmetric, this leads to local distortions. Energetic particles are trapped magnetically in the Van Allen belts that consist of electrons and protons. When a spacecraft passes this area, it is exposed to an increased level of radiation [4] [5].

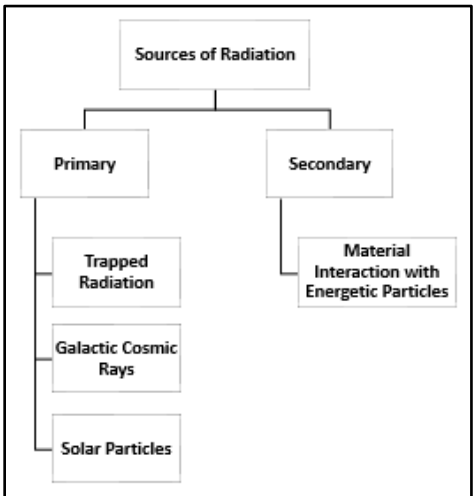


Figure 1: Sources of Radiation.

- Galactic Cosmic Rays (GCR):These are heavy ions that are produced by the explosion of supernovas or collisions among celestial bodies outside the solar system and even the Milky Way. The charged particles enter the solar system from outside, and are composed of protons, electrons and fully ionized nuclei[3] [4] [5].
- Solar Energetic Particles:Sun activity or solar storms due to solar flares, constitute a highly concentrated explosive release of the sun's mass and energy. Particles are heated and accelerated in the solar atmosphere. The particles consist of protons, electrons and heavy ions[4] [5].

2.2 Secondary Radiation

Radiation generated by the interaction of energetic particles with materials. One example is α-particles emitted by radioactive isotopes elements found in the silicon wafer or packaging material. They have a high probability to generate upsets.

3. RADIATION EFFECTS ON ELECTRONIC DEVICES

The massive presence of radiations in the space possibly causes glitches in the system elaboration. Three main radiation effects are observed in the field of electronic components: Total Ionizing Dose (TID), Displacement Damage Dose (DDD) and Single Event Effect (SEE) as shown in Figure 2.

3.1 Total Ionizing Dose (TID)

TID is the dose that is deposited in the electronics through ionization effects only. TID effects have the potential to destroy the device. TID describes a long-term degradation of an electronic component. The degradation is caused by an accumulation of energy, which is deposited in the material over a long period of time. TID may cause threshold voltage shift, static supply leakage and degradation of timing parameters.TID is the measure of how much energy has been absorbed by the semiconductor. It is measured in *rad* (radiation absorbed dose).

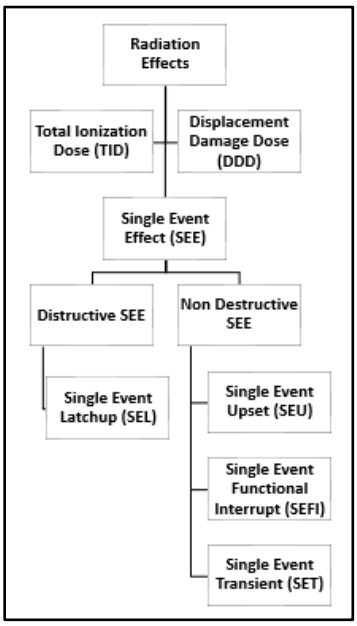


Figure 2: Radiation Effects.

In general, TID effects can be mitigated through proper use of shielding materials. For space-qualified Virtex4QVandVirtex-5QVdevices, the TID is of no concern since the dose is guaranteed to be 300 *Krad* for Virtex-4QV devices, 1 *Mrad* for Virtex-5QV devices[4] [6].

3.2 Displacement Damage Dose (DDD)

The second area of the cumulative effects of radiation is the displacement effects. DDD is caused by highly energetic particles (protons and neutrons). When the high-energy particles strike the atoms, they may penetrate into the crystal lattice of the silicon.In this case, the atomsare "displaced" from their position to various locations. Unless the end location is an exact duplicate of the former position, the regular order of the crystalline lattice is disturbed. The resulting crystal contains empty positions of knocked-out atoms, which are clustered elsewhere in the crystal. These places are sources of problems, as they serve as recombination centers. As in the TID effect, the degradation is long-term and often has similar long-term degradation characteristics, although it is based on different physical mechanism. In general, displacement effects are also mitigated through proper use of shielding materials[5] [6].

3.3 Single Event Effects (SEE)

SEEs are effects caused by a single, energetic particle radiation on an electronic circuit, which causes transient errors

and it can take on many forms. These effects are classified as non-destructive and destructive faults. Non-destructive faults (soft faults) include single event upsets (SEU), and single event transients (SET). Destructive faults include single event latch up (SEL), single event burnout (SEB),single event functional interrupt (SEFI), and single event gate rupture (SEGR). Destructive faults cause permanent damage if no appropriate mitigation technique is used[5] [7].

- SEU is a special case of SEE, where the sudden conductance of the transistor (results from ionization by an energetic particle) may result in bit flip in the configuration bits that control the routing, logic behavior, and other critical aspects of the FPGA designs[8]. The change is not permanent and is easily recovered by writing a new value or by resetting the FPGA. SEU is the most common effect for SRAM-based FPGAs as it may affect the configuration memory as well as memory cells that are used as part of the user logic (flip-flops, embedded RAM) [4].
- SET is a transient effect that can be observed as a current/voltage spike, or a short pulse on the clock or data net. It affects both analog and digital components. The device can continue normal operation thereafter. If this transient effect passes through a memory cell at the same moment that the cell is capturing and storing its input, the result is the same as an SEU.
- SEFI is a type of SEU that may cause the circuit to stop operating, where the SEU affect the control logic. A loss of control over the device is often observed. SEU susceptibility of the control logic, JTAG, and dedicated ports for configuration downloading are the main source of SEFI. The error can be corrected by rewriting the original information which might involve hard reboot (power cycle) or soft reboot (software restart)
- SEL is potentially destructive. SEL results in a high operating current, above device specifications. This state can be released only by power reset. The latch-up can be detected by the increased device current. The current density or the local overheating may destroy the device, especially when the current is not limited. Similar to the TID effect, SELs are of no concern for Virtex-4QV and Virtex-5QV devices since both devices have high guaranteed latch-up immunity [4] [5] [7].

Non-destructive SEEs are recovered by resetting or reconfiguring the FPGA, whereas destructive SEEs have a permanent effect and must be mitigated by relocating the design on a new unused area of the FPGA.

4. AGING EFFECTS

Beside radiation effects, device aging can have major effects especially for long lasting space missions, where maintenance or substitution is very difficult. Aging effects are destructive and are classified as follows[9]:

- Time Dependent Dielectric Breakdown (TDDB): If the leakage current is increased, it causes a breakdown and eventually a short circuit because of the charge trapping within the gate dielectrics.
- Electro-Migration (EM): It is a development of voids in metal lines due to heavy current densities over a period of time. This can cause faults due to the creation of open and/or short circuits.
- Hot-Carrier Effects (HCE): This is an effect that leads to a buildup of trapped charges in the gate-channel interface region. HCE causes increase in threshold voltage and gradual reduction in the channel mobility. This effect makes the switching speed slow, and causes delay faults.
- Negative Bias Temperature Instability (NBTI):This is the degradation dependent on the time a PMOS transistor is stressed in the circuit. NBTI leads to delay faults as in HCE.

The Fault types due to the previous effects can be summarized inFigure 3.

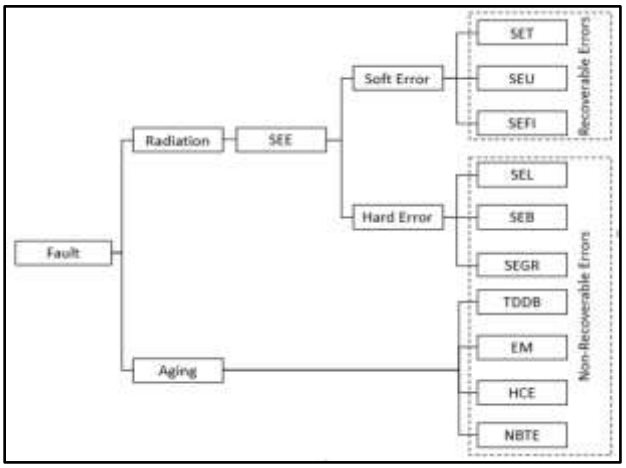


Figure 3: Fault Causes and Types.

5. FAULT MODES IN SRAM-BASED FPGAS

SRAM-based FPGAs encompass a configuration memory layer, which stores the configuration (bitstream) of the FPGA in SRAM memory cells that define the functionality performed by the FPGA, and a user logic layer where the actual circuit design is implemented and the application data is being processed are stored.

The bitstream on the configuration memory controls the resources implemented by the FPGA, including the routing between the resources, the LUTs content (combinational part of the design) and the configuration of the block random access memory (BRAM), configurable logic blocks(CLB), digital signal processing blocks (DSP) and input/output blocks (IOB) blocks.

If a particle strikes the FPGA, it may affect memory resources including the configuration memory and the user logic layer. Upsets are seen as faults that may cause a failure. The system is fortunately recovered from such failures by updating the memory cells with the correct values. Since the configuration bits (bitstream) control nearly everything, the configuration memory is the main concern of most mitigation techniques [4] [10]. The fault modes in FPGA are:

- Fault in Configuration memory: A SEU in the configuration memory can change the logic implemented on the FPGA and hence alters the function and goals of the circuit. In Xilinx terminology configuration memory bits range from unused bits to critical bits as shown inFigure 4. The designer of the system can decide which bits are critical. Critical bits must be guaranteed to be masked or recoverable from failures[11].

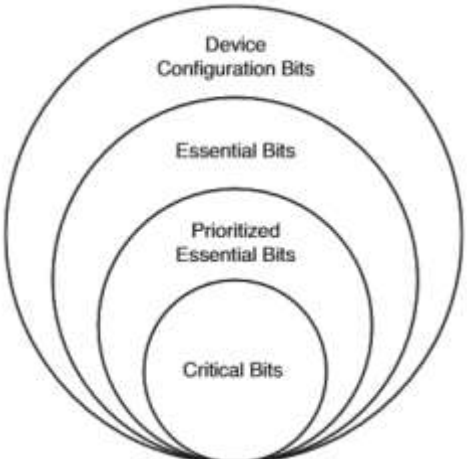


Figure 4: Configuration Bits Classification.

- A fault in a user flip-flop may cause a failure if its value is used by subsequent circuitry. The failure can be measured at the output if it is propagated through the system although it is often transient failure. If the failure is trapped in a feedback loop the logic must be reset to an initial state.
- A fault in a Block RAM cell may cause a failure in the next read access. Often, the memory is not accessed immediately and the failure demonstration is delayed.

6. MITIGATION TECHNIQUES CLASSIFICATION

Fault Management Handbook[12] stated that failures can be mitigated by five different techniques; hence failures can be prevented or tolerated. The main techniques are fault avoidance which is divided into two techniques, and fault tolerance which is classified into three techniques as shown in Figure 5. In the fault avoidance (prevention), actions are taken to prevent failures from occurring either at design time or at run time, whereas in fault tolerance actions are taken to detect and/or correct faults after their occurrence.

6.1 Fault Avoidance

- Design-Time Fault Avoidance: Minimizes the risk of a fault and its resulting failure during design by using, for example, high quality parts, or high QA processes.
- Operational Failure Avoidance: Predicts that a failure may occur in the future during operation and takes action to prevent or delay it from occurring, for example, by maintenance or operational change.

6.2 Fault Tolerance

If faults cannot be avoided, other possibilities of improving the levels of reliability should be searched. Thepossible occurrence of faults is taken into account in the design and implementation of the system so that defects will show only a minimal impact on the system. This type of reaction to the faults is referred to as fault tolerance. A system that reacts in this way is called a fault tolerant system[13].

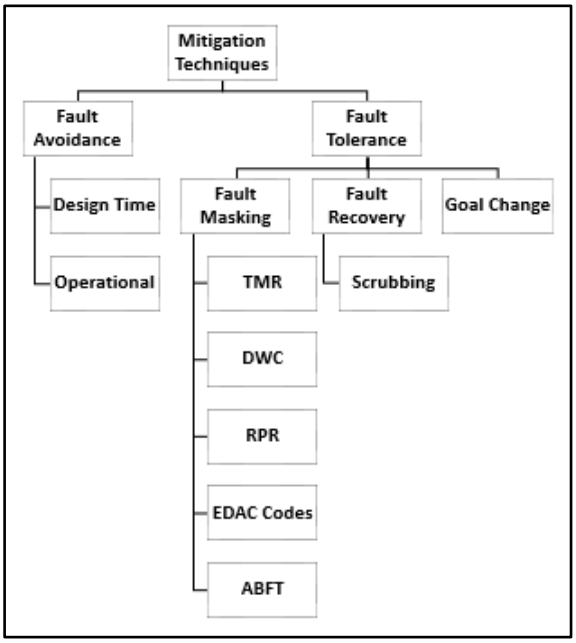


Figure 5: Fault Mitigation Techniques

- Failure Masking Techniques: When a failure occurs, its effect is masked so it doesn't affect the system function.
- Failure Recovery Techniques: A failure may temporarily affect the system function, but a recovery process is performed before the failure affects the system goal.
- Goal Change Strategies: A failure may temporarily affect the system function, and a responseis achieved by changing the system's goals to degraded goals.

7. FAULT TOLERANCE TECHNIQUES IN SRAM-BASED FPGAS

Two famous techniques are used in SRAMFPGAs: fault masking compromising some form of redundancy, and fault recovery by reconfiguration.

7.1 Periodic Reconfiguration (Scrubbing)

Periodic reconfiguration known as scrubbing is a method for mitigating SEUs in the configuration memory by continuously rewriting the original configuration. It can also be used for preventing the accumulation of failures to improve the reliability of the FPGA. There are 4 major scrubbing implementations: Blind vs. Readback Scrubbing, Device vs. Frame-Oriented Scrubbing, Periodic vs. On-Demand Scrubbing, and External vs. Internal Scrubbing[4].

7.1.1 Blind vs. Readback Scrubbing:

The most basic scrubbing technique is the blind scrubbing where the configuration memory is re-written at chosen intervals with a good copy of the original configuration bits. This copy is known as the 'golden copy'. The golden copy is stored in an external, hardened memory. A controller is used to control the downloading of the golden copy via one of the configuration interfaces of the FPGA. Blind scrubbing requires little system overhead since it updates the system whether there is fault or not. For the same reason, it can be classified as operational fault avoidance technique. SEFI detection is done before each re-write access.

In the readback feature the configuration memory is read to detect faults. If faults are detected, the golden copy is re-written. This scrubbing technique can be classified as a failure recovery technique.One detection technique is based on comparison where the configuration memory is readback and compared with the golden copy. Comparison may be done by bit-wise comparison, or by computing a Cyclic Redundancy Check (CRC) checksum during the readback process then it can be compared with the CRC of the golden copy. The second detection technique is based on information redundancy using Error-Correcting Code (ECC). Single Error Correction and Double Error Detection (SEC/DED) code can detect up two faults and correct a single fault. If there are more than two faults, the syndrome is indeterminate and faults can't be detected. The syndrome is computed during the readback process. According to its value, faults are detected and maybe localized in the case of correction[4] [6]. The methodology described in [14] use blind periodic scrubbing, while that in [15] use readback scrubbing technique by comparing the golden copy with the readback copy.

7.1.2 Device vs. Frame-based Scrubbing

The configuration memory can be scrubbed with the full bitstream or scrubbed by frames. Full scrubbing is known as device-based scrubbing. Its implementation is simple by downloading the golden copy from a memory to the configuration interface. The main disadvantage is the possibility of the SEFIs to occur. If this happens during the bitstream download, the whole design can be corrupted.

To reduce the SEFIs effect, frame-based scrubbing is used to isolate it to a single frame. Frame-based scrubbing is more complex. It requires the configuration controller to prepare each frame before download. There are some drawbacks of the frame-based scrubbing. First the scrubbing speed is decreased because a SEFI check must be done for each frame. Second, the overhead is increased because each frame bitstream has its header. The frame-based technique is used in [14].

7.1.3 Periodic vs. On-Demand Scrubbing

When scrubbing is independent of any other mitigation techniques, the configuration memory is periodically scrubbed, and scanned for faults at a fixed rate.In other designs, a mitigation technique is used to detect faults and trigger the scrubbing process. These designs are known as on-demand scrubbing. On-demand scrubbing has the advantage of power saving hence the scrubbing component remains idle until trigger. In many researches on-demand scrubbing is mentioned as dynamic partial reconfiguration[16] [17].

7.1.4 External vs. Internal Scrubbing

The scrubbing circuit may be implemented internally on the user logic layer or externally. If external scrubbing is used, the SelectMAP interface is commonly used because it has high throughput rate[18]. If internal scrubbing is used, ICAP can be used[16]. Although internal scrubbing is low cost and simple implementation since it does not require an external controller or hardened memory to store the golden copy of the bitstream, external scrubbing is more robust and recommended by Xilinx[4].

7.2 Redundancy Techniques

Redundancy-based fault tolerance strategies employ a predetermined additional set of physical resources such as hot or cold spares at fixed granularity[18]. When a fault is detected they supply a prearranged replacement to recover from it[19]. Redundancy can be provided by extra components (hardware redundancy), by an additional execution time or different instants of data sampling (time redundancy), or by additional code segments (software redundancy) [20].

7.2.1 Hardware Redundancy

Modular hardware redundancy is the most popular approach to achieve the required reliability and the timing constraints. In Modular Redundancy multiples of replicas are used for the same module. An N- modular redundancy (NMR)

system replicates a design into N-modules and uses voter to select the correct output. Hence, for NMR it is possible to tolerate the fault as long as it happens in no more than (N/2) modules[21].

- Triple Modular Redundancy (TMR): The circuit is triplicated and a majority vote is used to choose the correct output. Since the voter is susceptible to upsets it is triplicated too as shown inFigure 6 and Figure 7 respectively.

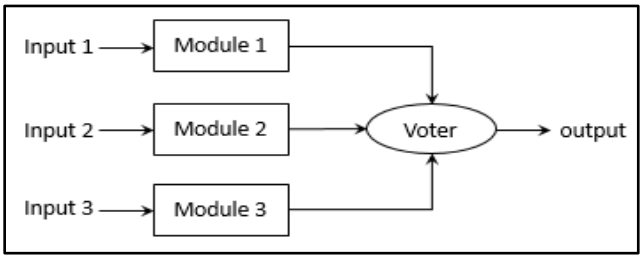


Figure 6: Basic TMR.

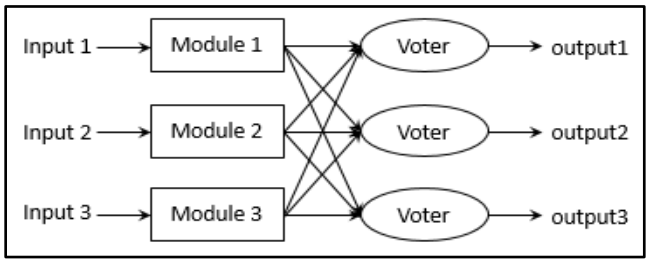
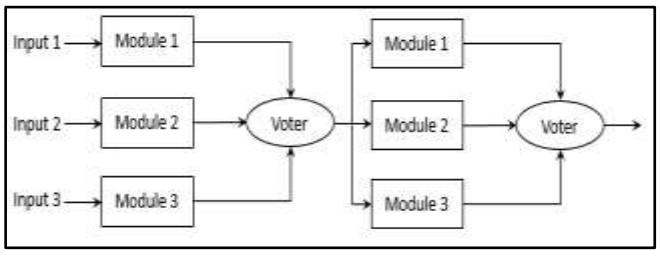


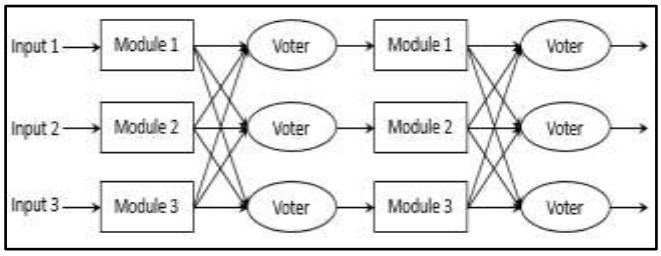
Figure 7:TMR with Repeated Voter.

TMR can detect and correct only single error for the whole circuit. To decrease the area that has upset and avoid the propagation of errors, the circuit can be partitioned to modules and adding additional voters for every module set as shown in Figure 8.

TMR has the disadvantage of high area overhead since the circuit as triplicated. If all the redundant modules are hot the design suffers also from higher power dissipation and increased temperature. TMR technique is used in many researches [21] [22] [23] [24].



(a) BasicTMR for Moduled System.



(b) TMR with Repeated Voter for Moduled system.

Figure 8: TMR for Module System.

- Duplication with Comparison (DWC): DWC is used to reduce the area overhead in the TMR. The circuit is duplicated and the output is compared as shown in Figure 9. DWC can only detect the error without correcting it, so a recovery technique must be used beside it to correct the fault.A common recovery technique used in companion with DWC is the on-demand scrubbing. Work in [25] uses the DWC technique along with dynamic partial reconfiguration to increase the reliability of the design.

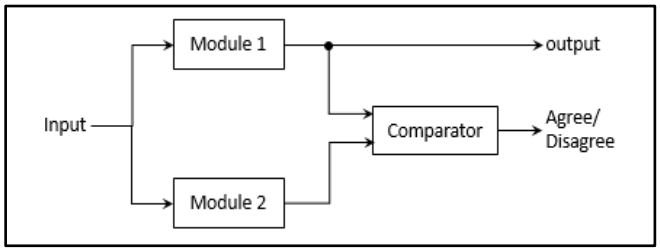


Figure 9: DWC Implementation.

- Standby Sparing: One of the modules is used to provide the output, and the remaining modules work as spares (active or passive). An error detection technique is used to identify faulty modules and a fault-free module is selected to provide the output as shown in Figure 10[26].
- Pair and Spare: This technique combines DWC and standby sparing. Two modules are online and compared as in DWC, and any spare can replace either of the online modules, as shown in Figure 11[27].

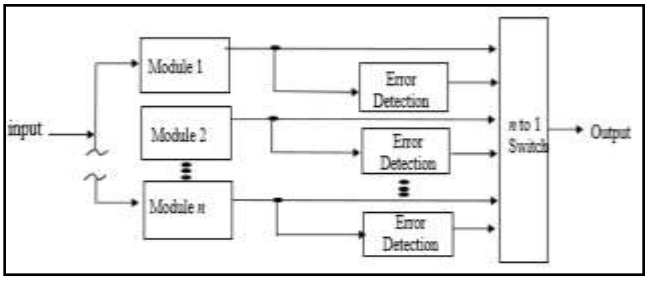


Figure 10: Standby Sparing Implementation.

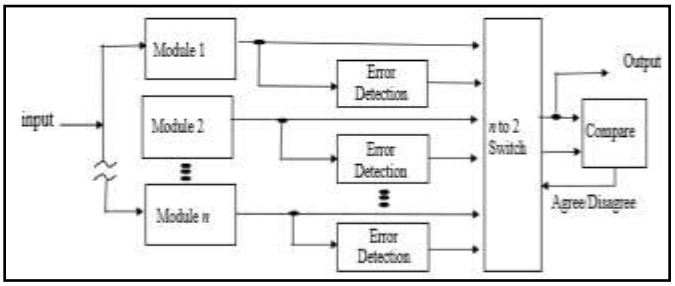


Figure 11: Pair and Spare Scheme.

- Self-Purging: The correct output is compared with the output of each module to identify the faulty modules for NMR with N>2 as shown in Figure 12[27].

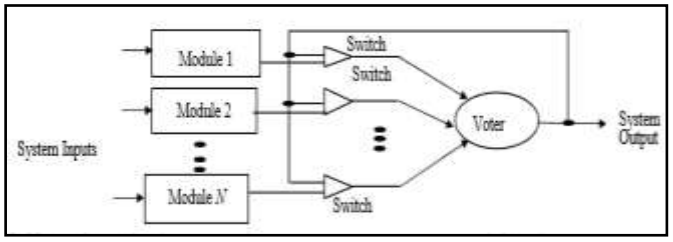


Figure 12: Self-Purging Scheme.

The main drawback of error detection using hardware redundancy is the area overhead. Also, it provides a limited resolution to identify the faulty module unless using extra logic for this purpose[9].

7.2.2 Time Redundancy

Time redundancy techniques rely on the advantage of the transient fault by comparing the output signals at two different moments. The output is latched at two or more different times, where the clock edge of each latch is shifted by time *d*as shown in Figure 13. A comparator or a voter indicates the occurrence of transient fault[20]. Time redundancy along with DWC is used in [28].

The area overhead in time redundancy comes from the added sample latches. Also, the performance is affected. The performance penalty is given by $clk+(N-1)d+tp$, where *d*denotes the expected duration of the transient pulse , N is the number of latched times and tp is the majority voter delay [20].

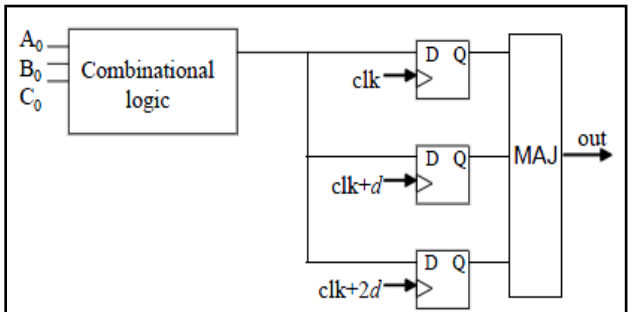


Figure 13: Time Redundancy Implementation.

7.2.3 Information Redundancy

Information redundancy depends on the use of EDAC (Error Detecting and Correcting Codes) by adding extra bits.These bits help in detecting and localizing the errors, and hence, correcting them using unique syndrome calculation. The block diagram of EDAC is shown in Figure 14. Information redundancy is mostly used to tolerate faults in FSM (Finite State Machine) and the data stored in the BRAM (Block RAM). There are many techniques to implement EDAC. The most famous codes are:

- Parity bits: The simplest coding technique used, can be implemented simply using XOR gates. It is rarely used alone because it is weak[27].
- Hamming Codes: This code adds *c* check bits and can detect multiple faults and may correct a smaller number of faults depending on the hamming distance. The most famous type is the single error correction/double detection code[20] [27].
- Rectangular Codes: These codes require more computations than Hamming codes, but it is suitable for data stored as a matrix. Parity bits are generated for both horizontal and vertical data. These codes are used in ABFT (Algorithm-Based Fault Tolerance).
- Reed-Solomon Codes: used to detect/correct multiple faults, but decoding and encoding circuits are very complex compared to Hamming codes[20].
- Cyclic Redundancy Codes (CRC): Blocks of data get a short check value attached to them, based on the remainder of a polynomial division of the contents. On retrieval, these calculation is repeated and, in the case the check values do not match, corrective action can be taken against data corruption. CRC is usually used in rollback scrubbing [29].

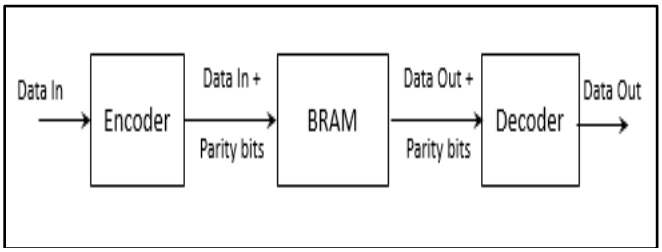


Figure 14: Block Diagram of EDAC.

8. CONCLUSION

This paper has listed the main sources of faults in FPGA systems. Both radiation and aging effects are explained briefly. The presented survey showed different fault-tolerance techniques that can be used to mitigate either permanent or transient faults. There is no single mitigation technique that is significantly better than the others. Thus, mixing between two or more techniques maybe used for highly reliable systems.

REFERENCES

1] S. S. Meshram and U. A. Belorkar, "Design Approach for Fault Tolerance in FPGA Architecture," *International Journal of VLSI design & Communication Systems (VLSICS)*, vol. 2, no. 1, pp. 87-95, 2011.

2] V. Dumitriu, L. Kirischian and V. Kirischian, "Run-Time Recovery Mechanism for Transient and Permanent Hardware Faults Based on Distributed, Self-organized Dynamic Partially Reconfigurable Systems," *IEEE Transactions on Computers*, vol. 65, no. 9, pp. 2835-2847, 08 September 2016.

3] F. Kastensmidt and P. Rech, FPGAs and Parallel Architectures for Aerospace Applications, Switzerland: Springer International Publishing, 2016.

4] F. Siegle, T. Vladimirova, J. Ilstad and O. Emam, "Mitigation of Radiation Effects in SRAM-based FPGAs for Space Applications," *ACM Computing Surveys (CSUR)*, vol. 47, no. 2, pp. 1-34, January 2015.

5] S. Duzellier, "Radiation Effects on Electronic Eevices in Space," *Aerospace Science and Technology*, vol. 9, no. 1, pp. 93-99, January 2005.

6] T. S. Reddy, J. Santosh and J. Prabhakar, "Fine-Grain Redundancy Techniques for HighReliable SRAM FPGA`S in Space Environment: A Brief Survey," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (IJAREEIE)*, vol. 3, no. 12, pp. 14047-14051, December 2014.

7] H. Quinn, "Radiation Effects in Reconfigurable FPGAs," *Semiconductor Science and Technology*, vol. 32, no. 4, 2017.

8] H. Ziade, R. Ayoubi, R. Velazco and T. Idriss, "A New Fault Injection Approach to Study the Impact of Bitflips in the Configuration of SRAM-Based FPGAs," *The International Arab Journal of Information Technology*, vol. 8, no. 2, pp. 155-162, April 2011.

9] E. Stott, P. Sedcole and P. Y. Cheung, "Fault Tolerant Methods for Reliability in FPGAs," in *International Conference on Field Programmable Logic and Applications FPL*, Heidelberg, Germany, 2008.

10] J. Hussein and G. Swift, "Mitigating Single-Event Upsets," Xilinx, 2015.

11] R. Le, "Soft Error Mitigation Using Prioritized Essential Bits," Xilinx, 2012.

12] Fault Management Handbook, Washington: NASA, 2012.

13] B. Dobrucky, P. Sindler, J. Cuntala and A. Kondelova, "Increasing of Reliability of FPGA Implemented Microcontroller Using the Error Self Correcting Techniques," *Journal of Communication and Computer*, vol. 12, pp. 219-227, 2015.

14] S. Wichman, S. Adyha, S. Ahrens, R. Ambli, B. Alcorn, D. Connors and D. Fay, "Partial Reconfiguration Across FPGAs," in *Military and Aerospace Applications of Reconfigurable Logic Devices and Technologies (MAPLD)*, Washington, 2006.

15] I. Herrera-Alzu and M. López-Vallejo, "Design Techniques for Xilinx Virtex FPGA Configuration Memory Scrubbers," *IEEE Transactions on Nuclear Science*, vol. 60, no. 1, pp. 376-385, 2013.

16] Z. Zhao, D. Agiakatsikas, N. T. H. Nguyen, E. Cetin and O. Diessel, "Fine-grained Module-based Error Recovery in FPGA-Based TMR Systems," in *Field-Programmable Technology (FPT)*, Xi'an China, 2016.

17] L. Pereira-Santos, G. L. Nazar and L. Carro, "Exploring Redundancy Granularities to Repair Real-Time FPGA-Based Systems," *Microprocessors and Microsystems*, vol. 51, pp. 264-274, 2017.

18] G. L. Nazar, L. P. Santos and L. Carro, "Scrubbing Unit Repositioning for Fast Error Repair in FPGAs," in *Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, Montreal, Canada, 2013.

19] C. A. Sharma, A. Sarvi, A. Alzahrani and R. F. DeMara, "Self-Healing Reconfigurable Logic Using Autonomous Group Testing," *Microprocessors and Microsystems*, vol. 37, no. 2, pp. 174-184, March 2013.

20] F. Kastensmidt, L. Carro and R. Reis, Fault-Tolerance Techniques for SRAM-Based FPGAS, Dordrecht: Spriger, 2006.

21] S. C. Anjankar, M. T. Kolte, A. Pund, P. Kolte, A. Kumar, P. Mankar and K. Ambhore, "FPGA Based Multiple Fault Tolerant and Recoverable Technique Using Triple Modular Redundancy (FRTMR)," in *7th International Conference on Communication, Computing and Virtualization*, 2016.

22] S. C. Anjankar and M. T. Kolte, "Fault Tolerant and Correction System Using Triple Modular Redundancy," *International Journal of Emerging Engineering Research and Technology*, vol. 2, no. 2, pp. 187-191, 2014.

23] P. Balasubramanian, K. Prasad and N. E. Mastorakis, "A Fault Tolerance Improved Majority Voter for TMR System," *WSEAS Transactions on Circuits and Systems*, vol. 15, pp. 108-122, 2016.

24] D. Shinghal and D. Chandra, "Design and Analysis of a Fault Tolerant Microprocessor Based on Triple Modular Redundancy Using VHDL," *International Journal of Advances in Engineering & Technology (IJAET)*, vol. 1, no. 1, pp. 21-27, March 2011.

25] Q.-Z. Zhou, X. Xie, J.-C. Nan, Y.-L. Xie and S.-Y. Jiang, "Fault Tolerant Reconfigurable System with Dual-Module Redundancy and Dynamic Reconfiguration," *Jouranl of Electronic Science and Technology*, vol. 9, no. 2, pp. 167-173, June 2011.

A. Ejlali, B. M. Al-Hashimi and P. Eles, "Low-Energy Standby-

26] Sparing for Hard," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Ssystems*, vol. 31, no. 3, pp. 329-342, March 2012.

27] I. Koren and C. M. Krishna, Fault Tolerant Systems, London, UK: Elsevier, 2007.

28] V. Tiwari and P. S. Patwal, "Design and Analysis of Software Fault-Tolerant Techniques for Softcore Processors in Reliable SRAM-Based FPGA," *International Journal of Computer Technology and Applications*, vol. 2, no. 6, pp. 1812-1819, December 2011.

29] A. Ebrahim, T. Arslan and X. Iturbe, "On Enhancing the Reliability of Internal Configuration Controllers in FPGAs," in *Adaptive Hardware and Systems (AHS)*, Leicester, UK, 2014.