Soft-Errors in Embedded Applications

Kajeepan Umaibalan
Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
Kajeepan.umaibalan@stud.hshl.de

***Abstract*—The rapid rise in transistor counts, coupled with higher clock speeds and voltages and reduced load capacitance, is intensifying the risk of soft errors in embedded systems. Today's transistors are constructed with components separated by small distances, making them susceptible to state alterations from even minor voltage fluctuations. As the space between the transistors continue to shrink, the implications of these intricate connections between chip components are increasingly distressing, directly impacting the reliability of embedded systems and rendering them highly vulnerable to soft errors.**
This paper explores soft errors in embedded applications, aiming to thoroughly examine their origins, causes of soft errors, and solutions for effective resolution. Through an in-depth analysis of soft errors, the study seeks to offer valuable insights into understanding the root causes of errors and proposing practical solutions to minimize and manage them effectively.[6]

I. Introduction

Soft errors are one of the main obstacles in the domain of embedded applications, where maintaining system dependability is of utmost importance. Given the continuous advancement of semiconductor technology, the increasing complexity of embedded systems, and the persistent need for enhanced performance, the susceptibility to soft errors has emerged as a critical issue. These errors arise from temporary disturbances in the status of electronic devices and can result in operational glitches, data distortion, and potentially severe repercussions. Hence, understanding and addressing these issues is vital for ensuring the uninterrupted functioning of embedded applications.[6]
Contemporary embedded systems are susceptible to soft errors, mainly attributed to ongoing manufacturing trends, including the reduction in size and supply voltage, along with amplified radiation and other environmental factors. Numerous strategies, both hardware and software-based, have been demonstrated in this paper to identify and rectify soft errors. Several hardware solutions rely on redundancy, such as duplicating entire systems or incorporating additional redundant bits into data structures to enable the detection of data corruption.[2]

II. Embedded Systems

Embedded systems are a fundamental component of contemporary technology, seamlessly integrating computational abilities into different devices and applications for the execution of particular tasks. These systems are defined by their specialized functionality, frequently operating within restricted environments that have limited resources. Embedded systems have wide-ranging applications across various sectors, including consumer electronics, automotive systems, medical devices, and industrial automation. They are engineered to execute predetermined functions with efficiency, often in real-time, and are typically optimized for compact size and low power consumption.[1]

III. Type of errors in embedded applications

Embedded applications can be affected by two distinct types of errors: hard errors and soft errors.

A. Hard-errors

In the realm of embedded applications, hard errors present persistent and typically irreversible malfunctions or breakdowns within the embedded system. These errors often result from physical imperfections and permanent harm to hardware components that impede the system from operating as intended. In contrast to soft errors, which are usually temporary and recoverable, hard errors typically necessitate physical repair or replacement of the impacted components. Hard errors can trigger total system failure, data loss, or severe operational disruptions, presenting substantial hurdles in upholding the dependability and effectiveness of embedded systems. Employing effective design methodologies, rigorous testing protocols, and robust quality assurance measures is vital in mitigating the risks associated with hard errors in embedded applications.

B. Soft-errors

Within the domain of embedded applications, soft errors manifest as momentary and typically reversible disruptions or malfunctions occurring in the embedded system. These disturbances occur from external factors, such as cosmic radiation, electrical interference's, or fluctuations in the surrounding environment, causing momentary disturbances in the system's regular operations. These consequences may involve data inconsistencies, operational challenges, or reckless modifications to the state of the embedded system. Developing a comprehensive understanding of the origins and implications of soft errors is key in devising effective strategies for identifying and rectifying these errors. Therefore, ensuring the consistent and uninterrupted performance of embedded systems, particularly in critical applications where system dependability is paramount.[1]

IV. Causes of Soft-errors

From the many factors that cause Soft-Errors, the three elements that stand out the most are alpha particles, cosmic radiation and electro magnetic waves.

A. Alpha Particles

Emission of alpha particles in Uranium and Thorium found in packaging materials are the main reason why soft errors happen in electronic deviecs such as DRAM devices, which store data in it's capacitors, leading to inaccurate values. Impurities in these two elements emit alpha particles with discrete energy levels of 4 to 9 MeV. To address these issues, manufacturers utilize various methods, such as error-correcting codes and protective shielding, to bolster the reliability of electronic systems.[9]

Alpha particles, being relatively large and positively charged, can interact with the semiconductor materials in electronic devices, generating electrical charges in the process. If an alpha particle hits a memory cell in a way that disturbs the electrical charge, it can potentially cause a flip in the stored bit, leading to errors in the data. This unintended alteration is a cause of soft error.[9]

The presence of uranium and thorium in the ceramic packaging of DRAM chips posed a significant challenge due to their tendency to emit alpha particles. These energetic particles were capable of ionizing the surrounding silicon, creating electron-hole pairs. If an alpha particle struck a strategic location within the silicon, it could generate a substantial amount of free charge carriers, causing electrons to accumulate within the memory well. This, in turn, could inadvertently flip a stored bit "1" to a "0," leading to data corruption and system malfunctions.[20] Fig 1 shows how alpha particles changes bit flip.

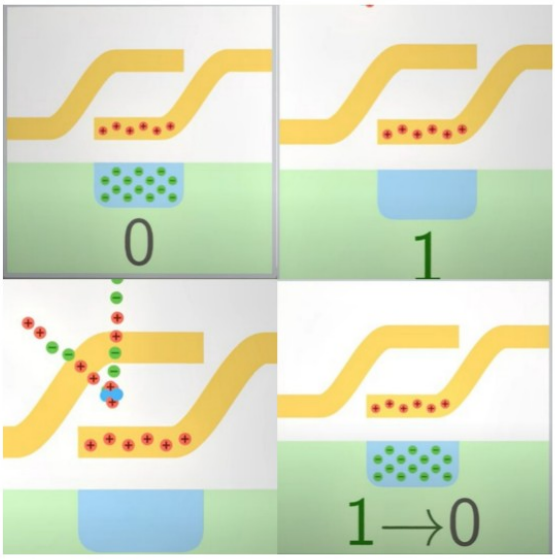

Fig. 1. Effect of alpha particles [20]

B. Cosmic Radiation

The second cause of soft errors are cosmic rays from outer-space. cosmic rays do not enter earths atmosphere instead they react with the air molecules in the Exosphere and produce complex secondary particles. As these particles delve deeper into the Earth's atmosphere they collide and decay into tertiary particles producing less than one percentage of primary flux. After reaching sea level the final state of primary flux particles are composed of muons, protons, neutrons, and pions.[21] Fig 2 shows the particles formed from cosmic rays that enters earths atmosphere.

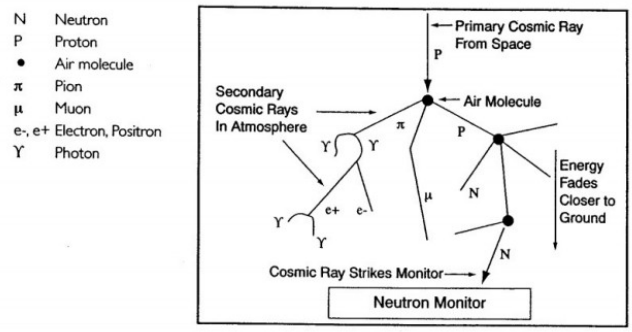

Fig. 2. Cosmic radiation particles [21]

The vast neutron flux at higher altitudes pose a challenge for electronic systems operating in these environments. Neutrons are one of the higher flux components and their reactions have higher linear energy transfer. Neutrons can interact with silicon inthe semiconductor materials. Figure 3 shows the reaction products of the silicon and neutrons.[9]

Reaction Products and Threshold Energies for n + ^{28}Si Reactions

Reaction Product	Threshold Energy (megaelectron volt)
^{25}Mg + α	2.75
^{28}Al + p	4.00
^{27}Al + d	9.70
^{24}Mg + n + α	10.34
^{27}Al + n + p	12.00
^{26}Mg + ^{3}He	12.58
^{21}Ne + 2α	12.99

Fig. 3. Neutron Reaction with Si [9]

Cause of neutron reaction with semiconductor material soft errors or data corruption occurs. To mitigate these effects, designers of electronic systems that are operating at high altitudes employ various techniques, such as shielding and error-correcting codes, to ensure the reliability of their devices.[9] Fig. 4. defines how many neutrons over the given energy range are incident on a device at sea level

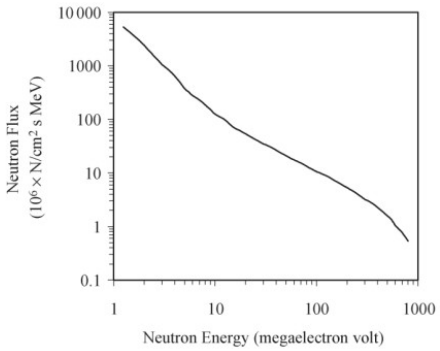

Fig. 4. Cosmic radiation particles [21]

C. High Frequency Electro-magnetic Waves

Electromagnetic induction is a phenomenon where changing magnetic fields induce voltage in conductors, it does not directly cause soft errors in semiconductor memory devices. Instead, soft errors in semiconductors are primarily caused by ionizing radiation from external sources.

- Electromagnetic Interference (EMI)
  High-frequency electromagnetic waves can induce unwanted electrical signals in nearby conductive materials. These induced signals may interfere with the normal operation of electronic components, including memory cells. If the induced signals are strong enough, they can lead to transient voltage changes in the memory cells, causing soft errors.
- Voltage Spikes and Noise
  Rapid changes in electromagnetic fields can induce voltage spikes and noise in the power supply lines or data lines of embedded systems. These voltage fluctuations may disturb the stability of memory cells, potentially leading to soft errors.
- Cross-Talk
  High-frequency waves can also cause cross-talk between adjacent conductive traces on a circuit board. Cross-talk refers to the unintentional coupling of signals between different paths. This interference can introduce disturbances in the signals traveling through memory circuits, leading to soft errors.
- Resonance Effects Certain frequencies may resonate with the natural frequencies of components in the embedded system, amplifying the impact of electromagnetic waves. This resonance can induce unexpected behaviors in electronic components, potentially causing soft errors

V. Impact of Soft-errors in Embedded application

Soft errors in embedded applications can have significant implications for the reliability and functionality of the embedded systems. Soft errors refer to transient faults that occur in a system, causing temporary errors or data corruption. These errors are typically caused by external factors such as radiation, cosmic rays, electromagnetic interference, and other environmental disturbances[1]. Here are some of the potential impacts of soft errors in embedded applications:

- Data Corruption
- System Malfunction
- Unpredictable Behavior
- System Crashes

VI. Mitigation Techniques

Soft errors, also called transient errors or single event upsets (SEUs), occur when semiconductor devices are hit by high-energy particles like cosmic rays or alpha particles, temporarily changing the state of a memory bit. This can result in corrupted data or malfunctions within the system.[3]
There are several techniques that can be used to mitigate soft errors

A. Error Correcting Codes (ECC)

ECC is a method of adding redundant bits to data to allow for the detection and correction of errors. When a soft error occurs, the ECC can be used to identify the corrupted bit and restore the correct value. ECC is an effective mitigation technique for soft errors, but it can also introduce some overhead in terms of memory usage and processing time.[10]

- Hamming codes
  Hamming codes enhance data integrity by incorporating additional bits to the original message. These supplementary bits, though not part of the original data, are intricately linked to it, creating redundancy. This redundancy enables the detection and correction of errors in the data.[18]
  The number of redundant bits added to the data correlates to its length. For instance, a [7,4] Hamming code embeds three redundant bits within a four-bit data word. Fig 3 shows how parity bit and the data is transmitted.[18]
  These redundant bits play a crucial role in generating parity checks. A parity check involves summing a specific group of bits. The parity of a group of bits can be either even or odd.[18]
  Error detection and correction are facilitated by these parity checks. When a single error occurs in the data, the parity checks become erroneous. This deviation pinpoints the exact location of the error[18]

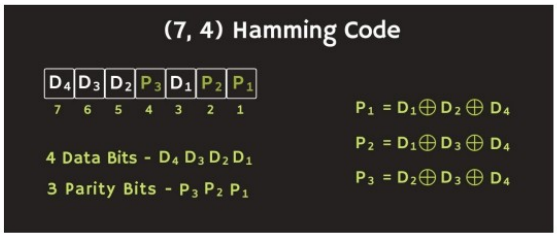

Fig. 5. Hamming code [18]

- Reed solomom code
  Reed-Solomon codes work by adding redundant symbols to the data. These redundant symbols are not part of the

original data, but they are related to the data in a specific way. The redundancy can be used to detect and correct errors in the data. Fig 8 shows how Reed Solomon code system functions.[12]
The number of redundant symbols that are added to the data depends on the length of the data and the desired error-correction capability. For example, a Reed-Solomon code that can correct up to t errors in a block of n symbols will add n - k symbols to the data, where k is the number of information symbols that are being transmitted.Fig 6 below shows how data and parity is being added to data[12]

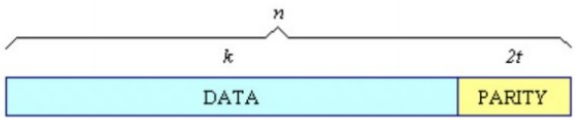

Fig. 6. Reed-Solomon codeword[12]

The redundant symbols are generated using a mathematical function called a generator polynomial. This polynomial is chosen so that it satisfies certain properties that allow it to be used for error correction.[12] Fig 7 shows the decoder of the reed solomom code and the plolynomials

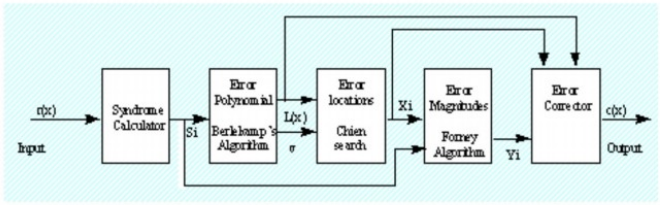

Fig. 7. Reed-Solomon decoder[12]

To detect errors in the data, the receiver can calculate a set of syndromes. These syndromes are a function of the received data and the generator polynomial. If the received data contains any errors, then the syndromes will be incorrect. The receiver can then use the syndromes to identify the location of the errors. Fig 6 shows how the decoder works to detect errors.[12]

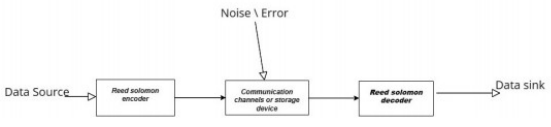

Fig. 8. Reed-Solomon system[12]

To correct errors in the data, the receiver can use the redundant symbols to determine the exact values of the missing data symbols. This is done by solving a system of linear equations.[12]

B. Redundancy

Redundancy involves replicating critical data or circuitry to provide backup in case of a soft error. For example, double modular redundancy (DMR) and triple modular redundancy (TMR) are techniques that use two or three copies of a circuit to detect and correct errors. Redundancy can provide very high levels of reliability, but it can also be expensive and complex to implement.

- Double modular redundancy
  DMR is a fault tolerance technique that uses two different implementations of a module to detect and correct soft errors. The two implementations, called M1 and M2, are designed to produce distinct error patterns when a soft error occurs. This allows the system to determine which module is faulty and use the output of the other module as the correct output.[11] Fig 9 shows how DMR is used to check redundency

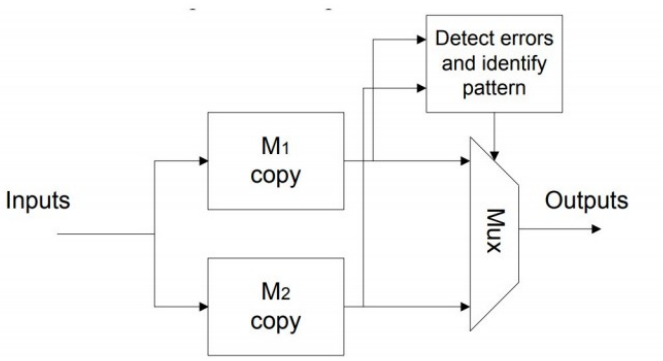

Fig. 9. Double Modular Redundancy approch[11]

In some cases, the difference in error patterns can be designed to appear over time. For example, one module might corrupt the output for two clock cycles, while the other module only corrupts it for one clock cycle. This makes it possible to differentiate between the two modules even if the error occurs during a clock cycle.[11] The error detection and correction process is as follows:

- The outputs of the two modules are compared.
- If the outputs are different, then a soft error is detected.
- The difference in the outputs is analyzed to determine which module is faulty.

The output from the other module is used as the final, corrected output. DMR is an effective way to protect against soft errors, but it is not without its limitations. One limitation is that it requires two implementations of the module, which can increase the cost of the system. Another limitation is that it is not always possible to design two modules that produce distinct error patterns.[11] Despite these limitations, DMR is a valuable technique for protecting critical systems from soft errors. It is particularly useful in applications where the cost of a

soft error is high, such as in aerospace and medical devices.[11]

- Triple modular redundancy
  TMR is a fault tolerance technique that uses three identical implementations of a module to detect and correct soft errors. This means that there are three copies of the same module, and the output of the module is determined by the majority of the three outputs.[14]

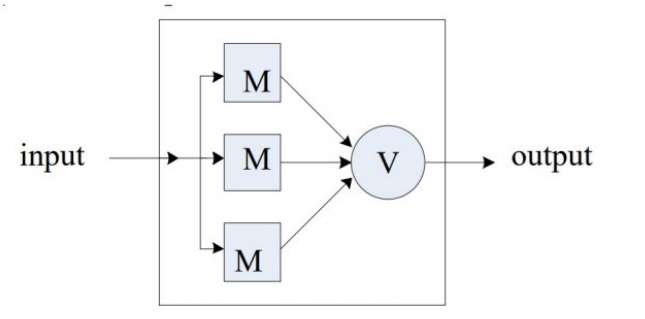

Fig. 10. Tripple Modular Redundancy approch[11]

The input to the module is applied to all three replicas. The outputs of the three replicas are compared to each other. If all three outputs are the same, then no error has occurred. If any two of the outputs are different, then a soft error has been detected. The module with the erroneous output is identified. The correct output is determined by the majority of the remaining two outputs. Fig 10 shows how TMR input and output is configured[13]
TMR is an effective technique for protecting against soft errors because it can detect and correct a wide variety of single-event transients (SETs) and single-event upsets (SEUs). SETs are caused by high-energy particles that pass through the circuit and temporarily disrupt the operation of transistors. SEUs are caused by high-energy particles that directly change the state of a transistor.[14]
When a soft error occurs in one of the replicas, the majority of the outputs will still be correct. The voter block will then select the correct output as the final output. This will prevent the soft error from propagating through the system and causing a system malfunction.[14]

C. Process Hardening

Process hardening is a set of techniques that are used to make semiconductor devices more resistant to soft errors. This can involve using higher-quality materials, reducing the sensitivity of transistors to charge, and controlling the manufacturing process more tightly. Process hardening can be very effective at reducing the overall rate of soft errors, but it can also be expensive and time-consuming to implement.[9]

D. Shielding

Shielding is a mitigation technique that can be used to protect against soft errors in electronic systems. It involves physically protecting the system from the sources of soft errors, such as cosmic rays and alpha particles. This can be done by using materials that are less susceptible to radiation damage, or by placing the system in a shielded enclosure.[15]
Shielding is a highly effective technique for protecting against soft errors. It can reduce the soft error rate by a factor of 10,000 or more. However, it can also be expensive and difficult to implement.[16]
Here are some specific examples of how shielding can be used to mitigate soft errors:

- Using materials that are less susceptible to radiation damage: GaAs and SiC are two materials that are less susceptible to radiation damage than silicon. This can be used to build radiation-hardened electronics that are less likely to be affected by soft errors.
- Placing the system in a shielded enclosure: This can be done using lead, steel, or other materials that are good at absorbing radiation. Shielding can be used to protect individual modules, or the entire system.
- Using active shielding: This is a more sophisticated technique that involves using electrical fields or magnetic fields to deflect radiation away from the system. Active shielding can be even more effective than passive shielding, but it is also more expensive.

Shielding is a critical technique for protecting against soft errors in critical electronic systems, such as those used in aerospace, medical devices, and high-performance computing.[8]

E. Software Mitigation

In addition to hardware-based mitigation techniques, there are also some software-based mitigation techniques that can be used to reduce the impact of soft errors

VII. Testing and Fault Injection

A. Testing

Testing is the process of verifying the functionality and performance of an electronic system under normal operating conditions. It helps to identify and fix any defects that may cause errors during normal use. However, testing alone cannot guarantee that a system will be completely immune to soft errors. This is because soft errors can occur sporadically and may not always be reproducible under controlled test conditions.
There are two main types of testing

- Structural testing: This type of testing focuses on the structure of the software or hardware, such as the code or circuit layout. It involves checking that all the components are connected correctly and that there are no logical errors in the code.
- Structural testing: This type of testing focuses on the structure of the software or hardware, such as the code or circuit layout. It involves checking that all the components are connected correctly and that there are no logical errors in the code.

Both structural and functional testing can be used to detect soft errors. However, structural testing is more likely to detect

errors that occur in the hardware, while functional testing is more likely to detect errors that occur in the software.

B. Fault Injection

Fault injection is a technique that deliberately introduces faults into a system to test its resilience against unexpected errors. It involves simulating the effects of soft errors by injecting them into the system's hardware or software. This allows engineers to observe how the system responds to faults and identify any potential vulnerabilities that may lead to system failures.[17]

- Hardware fault injection: This involves physically modifying the hardware of the system to simulate the effects of a soft error. For example, an electron beam can be used to disrupt the operation of a transistor, or a pulse of high voltage can be applied to a circuit to cause a transient error.
- Software fault injection: This involves introducing errors into the software of the system. This can be done by modifying the program code, or by injecting errors into the input data

Fault injection can be a valuable tool for improving the reliability of electronic systems against soft errors. By systematically injecting faults and observing the system's response, engineers can identify and fix potential vulnerabilities before they lead to real-world failures.[17]

VIII. Future Directions and Research Opportunities

The research on soft error mitigation is an active area of research, and there are many potential future directions. Some of the key areas of focus include:

- Developing new materials that are less susceptible to radiation damage. This could involve the use of new materials with better radiation tolerance, such as gallium arsenide (GaAs) or silicon carbide (SiC).
- Improving error correction codes (EDCCs). This could involve developing new EDCCs that are more efficient and effective at detecting and correcting multiple-bit errors.
- Improving error correction codes (EDCCs). This could involve developing new EDCCs that are more efficient and effective at detecting and correcting multiple-bit errors.
- Designing new architectural techniques that can help to reduce the impact of soft errors. This could involve the use of techniques such as replication, error detection and correction, and error handling.
- Developing new fault injection techniques. This could involve developing new fault injection techniques that are more precise and efficient.
- Investigating the use of machine learning and artificial intelligence (AI) to detect and mitigate soft errors. This could involve the use of AI to analyze data from electronic systems and identify patterns that are indicative of soft errors.

By pursuing these and other research directions, we can continue to develop more reliable and secure electronic systems that are less susceptible to soft errors.

IX. Conclusion

Soft errors, transient faults that can disrupt the normal operation of embedded applications, pose a significant threat to their reliability and security. These elusive errors, often caused by external factors such as cosmic rays and electromagnetic radiation, can silently corrupt data, lead to performance degradation, and even cause system malfunctions. As embedded systems become more complex and operate in increasingly demanding environments, the prevalence of soft errors is expected to rise, demanding proactive mitigation strategies to safeguard the integrity and resilience of embedded applications.[1]

A range of mitigation techniques, each with its own strengths and limitations, has been proposed to address this critical challenge. Hardware redundancy, exemplified by triple modular redundancy (TMR), provides enhanced reliability by replicating critical components, ensuring continued operation even if one component succumbs to a soft error. Error detection and correction codes (EDCCs), such as Hamming code, Reed-Solomon code, and Turbo code, offer another layer of protection by detecting multiple-bit errors and enabling their correction. Shielding, both passive and active, physically protects electronic devices from radiation sources, minimizing their exposure to soft error-inducing particles. Lastly, software mitigation techniques, such as parity checking, data scrambling, and error handling, complement hardware-based approaches, offering additional resistance to soft errors.[6]

References

[1] Azimi, Sarah De Sio, Corrado Portaluri, Andrea Rizzieri, Daniele Vacca, Eleonora Sterpone, Luca Codinachs, David. (2022). Exploring the Impact of Soft Errors on the Reliability of Real-Time Embedded Operating Systems. Electronics. 12. 169. 10.3390/electronics12010169.

[2] J. Vankeirsbilck, H. Hallez and J. Boydens, "Soft Error Protection in Safety Critical Embedded Applications: An Overview," 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), Krakow, Poland, 2015, pp. 605-610, doi: 10.1109/3PGCIC.2015.89.

[3] L. Sterpone, S. Azimi, C. De Sio and F. Parisi, "Analysis and Mitigation of Soft-Errors on High Performance Embedded GPUs," 2022 21st International Symposium on Parallel and Distributed Computing (ISPDC), Basel, Switzerland, 2022, pp. 91-98, doi: 10.1109/ISPDC55340.2022.00022.

[4] D. Alexandrescu, "A comprehensive soft error analysis methodology for SoCs/ASICs memory instances," 2011 IEEE 17th International On-Line Testing Symposium, Athens, Greece, 2011, pp. 175-176, doi: 10.1109/IOLTS.2011.5993833.

[5] I. Polian, S. M. Reddy, I. Pomeranz, X. Tang and B. Becker, "On Reducing Circuit Malfunctions Caused by Soft Errors," 2008 IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems, Cambridge, MA, USA, 2008, pp. 245-253, doi: 10.1109/DFT.2008.20.

[6] Sadi, M.S., Ahmed, W. Jürjens, J. Towards Tolerating Soft Errors for Embedded Systems. SN COMPUT. SCI. 2, 101 (2021). https://doi.org/10.1007/s42979-021-00497-9

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[8] Baumann, R.. (1991). Investigation of the Effectiveness of Polyimide Films for the Stopping of Alpha Particles in Megabit Memory Devices. 10.13140/RG.2.1.4428.7840.

[9] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," in IEEE Transactions on Device and Materials Reliability, vol. 5, no. 3, pp. 305-316, Sept. 2005, doi: 10.1109/TDMR.2005.853449.
[10] Li, Sida Hu, Xiaochang Huang, Zhiping Zhou, Jing. (2022). ECC-BERT: Classification of error correcting codes using the improved bidirectional encoder representation from transformers. IET Communications. 16. 10.1049/cmu2.12357.
[11] Reviriego, Pedro Bleakley, Chris Maestro, Juan Antonio. (2013). Diverse Double Modular Redundancy: A New Direction for Soft-Error Detection and Correction. Design Test, IEEE. 30. 87-95. 10.1109/MDT.2012.2232964.
[12] Riley, M. and Richardson, I. (1998) An introduction to Reed-Solomon codes: principles, architecture and implementation, Reed-solomon codes.
[13] Nowee, S. (2022) Reliability analysis of Triple Modular Redundancy. tech.
[14] Shernta, Samira Tamtum, Ali. (2018). Using Triple Modular Redundant (TMR) Technique in Critical Systems Operation. 52-60. 10.21467/proceedings.2.7.
[15] Jr, C. Rodbell, Kenneth Gordon, Michael. (2007). Alpha particle mitigation strategies to reduce chip soft error upsets. Journal of Applied Physics - J APPL PHYS. 101. 10.1063/1.2404482.
[16] J. A. Pellish et al., "Impact of Spacecraft Shielding on Direct Ionization Soft Error Rates for Sub-130 nm Technologies," in IEEE Transactions on Nuclear Science, vol. 57, no. 6, pp. 3183-3189, Dec. 2010, doi: 10.1109/TNS.2010.2084595.
[17] Daveau, J.-M Blampey, Alexandre Gasiot, Gilles Bulone, Joseph Roche, P.. (2009). An Industrial Fault Injection Platform for Soft-Error Dependability Analysis and Hardening of Complex System-On-a-Chip. IEEE International Reliability Physics Symposium Proceedings. 212 - 220. 10.1109/IRPS.2009.5173253.
[18] Toghuj, Wael. (2020). Modifying Hamming code and using the replication method to protect memory against triple soft errors. TELKOMNIKA (Telecommunication Computing Electronics and Control). 18. 2533. 10.12928/telkomnika.v18i5.13345.
[19] Reviriego, Pedro Bleakley, Chris Maestro, Juan Antonio. (2013). Diverse Double Modular Redundancy: A New Direction for Soft-Error Detection and Correction. Design Test, IEEE. 30. 87-95. 10.1109/MDT.2012.2232964.
[20] T. C. May and M. H. Woods, "A New Physical Mechanism for Soft Errors in Dynamic Memories," 16th International Reliability Physics Symposium, San Diego, CA, USA, 1978, pp. 33-40, doi: 10.1109/IRPS.1978.362815.
[21] Houseman , J. and Fehr, A. (no date) Listening for Cosmic Rays! The Inuvik Neutron Monitor. Aurora College.
[22] Azimi, Sarah De Sio, Corrado Portaluri, Andrea Rizzieri, Daniele Vacca, Eleonora Sterpone, Luca Codinachs, David. (2022). Exploring the Impact of Soft Errors on the Reliability of Real-Time Embedded Operating Systems. Electronics. 12. 169. 10.3390/electronics12010169.

Unterschrift
Signature

X. Affidavit

I hereby confirm that I have written this paper independently and have not used any sources or aids other than those indicated. All statements taken from other sources in wording or sense are clearly marked. Furthermore, I assure that this paper has not been part of a course or examination in the same or a similar version.

Umaibalan Kajeepan
Name,Vorname
Last Name, First Name

Lippstadt 23/12/2023
Ort,Datum
Location, Date