IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2320-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 10, Issue 7, July 2021

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.569

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 7, July 2021 ||

| DOI:10.15680/IJIRSET.2021.1007003 |

Embedded Systems based on Soft Cores for Use in Critical Aircraft Structures

Chiranjeevi Aradhya

Sr Software Engineer, Collins Aerospace, United States of America

ABSTRACT: Commercial Off the Shelf (COTS) devices are increasingly desired in the aerospace sector to decrease system costs. New trends are shown in the engineering of innovative microsatellites and nanosatellites. However, higher susceptibility to radiation-induced transient defects of these sub-micron devices has led to the use of this technique in lower-risk applications. This article is a new approach to building reliable and reduced-cost critical systems, using the features of soft-core microprocessor-based embedded systems with reconfigurable devices. To do this, we must fine-tune the protection approach by implementing both hardware and software-based fault mitigation strategies that are chosen carefully. By combining design restrictions and dependability needs, as well as taking advantage of avoidance of unnecessary protection methods, the resulting system both fulfils the criteria of design and of dependability, as well as providing an excellent level of protection. In this case study, a wide range of trade-offs are examined between performance, reliability, memory size, and hardware cost when hardware and software protection methods are used in a reliable subsystem.

KEYWORDS: Aerospace, Embedded Systems, Soft Core based system

I.INTRODUCTION

The availability of cost-effective access to space broadens the variety of business possibilities and markets that open up. It may be argued that using commercial off-the-shelf (COTS) electronics is a viable low-cost option for creating micro and nanosatellites (1,2). It may be argued that although more performance and power have been achieved via advances in technology, this results in devices being less dependable and prone to transitory failures.

Because transients' defects are triggered by external events, such as exposure to radiation, they are known as intermittent faults. These imperfections may not lead to long-term harm, but they might affect signal transfers or stored values, thereby causing erroneous circuit behaviour.

In the past, radiation-induced disturbances were limited to aircraft applications. The electrical components' susceptibility to transient errors increases every day, though. Temporary defects have emerged as a critical cause of electronic system failures even at the surface [1].

In other words, transient faults (i.e., defects that last only briefly) are becoming more common in application areas, such as novel applications in the fields of embedded systems in which automobiles, industrial systems, avionics, networking, and defence are used. For example, reliable, low-cost subsystems for satellites [2], spacecraft attitude control systems [3], reconfigurable computers for space systems [4], and more are some of the latest applications being developed for the aerospace application area.

On a related note, many technical bodies across the globe have also recently issued papers detailing specific qualification criteria that electronic components must satisfy before they may be used. Examples of these include: For avionics, IEC/TS 62396 (Process Management for Avionics: Atmospheric Radiation Effects) [6]; For aerospace applications, ESA PSS-01-609 (The Radiation Design Handbook) [5]; For military applications, MIL-HSBK-817 (System Development Radiation Hardness Assurance) [7].

When one compares the current industrial fields, which are seeing increased usage of embedded microcontrollers/microprocessors, to the industrial fields of the past, one discovers a really fascinating new application backdrop for embedded systems. The various degrees of freedom that we may conceive of lead us to think about fault mitigation in embedded systems as a novel research case.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 7, July 2021 ||

| DOI:10.15680/IJIRSET.2021.1007003 |

When you have various levels of memory (scratchpad, cache, RAM, flash, ROM, etc.) connected to the same system, it is usually the case that they will co-operate and be attached. Different tactics are required for every situation. One would also often find fault finding/correcting modules, such as watchdog drivers, throughout the computer. As a rule, fault mitigation hardware and software solutions are focused on the programme while it is operating, as opposed to situations when no application is running.

Also, we may use several deployment technologies, such as ASIC, SRAM-based FPGA, and non-volatile FPGA.

An entire hardware/software co-design approach to fault mitigation is possible because of embedded systems with soft-core soft- microsystems. This increased plasticity provides us with a significant advantage: It allows us to thoroughly explore the design space in order to minimise errors, and also provides us with meaningful measurements for comparing outcomes to embedded systems that have less constraints. This article describes a novel way to harden systems by implementing embedded hardware and software on a family subset of those based on soft-core processors. It is hard to imagine the app without the embedded systems and flexibility and co-design capabilities it uses.

While optimising the design, it is important to make sure that resources, like software or hardware, are not unnecessarily utilised to avoid excessive or unnecessary use of protection mechanisms.

This method uses fine-tuning of the protection strategy by integrating several fault mitigation approaches in order to adjust the defence mechanisms. It is possible to get the optimum trade-offs among performance, reliability, programme size, and hardware cost while also having software- and hardware-based security as design goals. Our method is shown via a case study, which shows how a reliable embedded system is built.

We would want to use flash-based FPGAs [8] in the various deployment scenarios. On the other hand, this technique eliminates the need of complicated scrubbing methods, which have to be employed in SRAM-based FPGAs, to ensure the integrity of the configuration memory. In contrast, the use of hybrid mitigation methods along with the protection provided by the flash technology results in a substantial reduction in economic cost when compared to rad-hard FPGA families [9].

II. RELATED WORK

Single Event Upset (SEU) is the subject of this article. This is due to ionisation, which is the result of a charged particle colliding with an electrical component. The bit-flip fault model, in which just one bit-flip of a storage cell happens while executing the circuit, is used to represent it. This fault model is utilised extensively by the scientific community, yet despite its simplicity, it really simulates faults well, as shown in [10].

Additional to that, in this part there are certain crucial words that should be introduced. To assess the system's dependability, a fault injection approach was used. Then, the system behaviour was monitored. Similar to Mukherjee et al. [11], the injected defects are now classified according to their impact on the anticipated behaviour.

Unnecessary for Architecturally Correct Execution: The bit which was flipped, and therefore the fault itself, are labelled as unnecessary in the event the system completes its execution and delivers the anticipated result.

Silent data corruption, or SDC, a defect that makes the system finish its execution, but does not provide the intended result. This is termed as a Hang if the failure causes the system behave malfunction or the programme iterates in an infinite loop.Because SDC and Hang are both undesired outcomes, pay attention to them (categorized together as ACE faults). Redundancy must be introduced in the system when fault mitigation methods are employed. This results in two significant problems that must be taken into consideration. First, applying these methods to the software will extend the runtime of the programme, thus exposing the vulnerable areas of the design for a longer period of time. In addition, redundancy will result in a greater number of bits in the system, increasing the likelihood of errors occurring.

Redundant hardware is typically included inside the system to help minimise the radiation-induced impacts. In order to manipulate more complex parts, for example, functional units, co-processors, or duplex systems on multiprocessor/multithreaded architectures, begin with lower-level building blocks, such as error-correcting code (ECC), parity bits, and triple modular redundancy (TMR) or single error correction (SEC) Hamming code [12]. Go from low level structures to more complex ones by means of using techniques like: Error-Correcting Code (ECC), parity bits, and Triple Modular Redundancy (TMR). From lower-level structures, like structures utilising error-correcting code (ECC), parity bits, and triple modular redundancy (TMR) or single error correction (SEC) Hamming code, use higher level newer methods advocate the use of selective hardening [13], where just the susceptible portions

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 7, July 2021 ||

| DOI:10.15680/IJIRSET.2021.1007003 |

are protected. Most hardware-based mitigation techniques are successful in dealing with transitory errors. Unfortunately, the methods used to achieve these goals also result in a penalty in power, die size, design time, and economic cost, which is why they are impractical in many instances.

The use of redundant software has allowed many ideas that use redundant software to provide both software detection and software corrective features. This set of activities reflects the need for low-cost, reliable solutions [14]. Redundancy is used in software-based techniques, both in order to identify problems and to minimise source code overhead and performance deterioration. While a couple of these approaches have been used to restore the system, there are no further methods to be found. They are less costly, but are not as reliable as hardware-based methods, since they must execute extra instructions. This approach has previously been utilised in space systems, satellites, and missions in actual space.

Even though hardware-only and software-only fault tolerance solutions abound, in many instances, the most appropriate solution is one that straddles the divide between these two opposing approaches. So, it is important to use software and hardware techniques to build the system. An embedded system's design should focus on aspects like cost and performance, with dependability coming in as a close second. Therefore, as a result, there is an increased need for fault-tolerant co-design with appropriate development tools. This means that some recent works have showed promise, with results. Additionally, some methods have focused on providing enough energy while allowing for flaws. Despite these being detailed methods, they are lacking in the capacity to make the optimal design trade-offs.

III.HYBRID FAULT MITIGATION METHODS

A complicated design process has many criteria that must be met all at once. The criteria fall into two categories: ones that rely on various factors such as dependability, design limitations, and costs, and others that don't. Because soft-core embedded systems provide plasticity and flexibility on both sides, we recommend they be used for this article. To build a system that satisfies both design restrictions and dependability needs while avoiding excessive usage of protective measures, adoption of a soft-core embedded system is feasible. As a solution, it is advised to use an unique fault mitigation method. This approach takes into consideration the different hardware and software protection approaches, and explores the design space between them.

Software fault mitigation methods, when used in combination, may provide an appropriate solution in certain instances, but it can also result in an overly redundant design with various limitations such as significant additional costs and increased penalties in performance. In other words, we work hard to expand our search into the design space by thoroughly using our options for protecting the software on both sides, while utilising special approaches to safeguard the hardware. This system controls its selectiveness by only installing safeguards on the areas of the system most susceptible to compromise. Moreover, because this determination influences the system differently, it is possible to choose whether protection is applied through software or hardware. This design method enables the designer to individually create a personalised fault mitigation strategy.

Fig. 1 summarises the suggested tailored fault mitigation co-design.

Our method starts with the system requirements being specified from the embedded application's perspective. As a design restriction, these criteria may directly affect the overall appearance of the product. However, to meet a certain reliability characteristic, these requirements may affect a specific application. With regard to design limitations, which we refer to as silicon area, performance, power consumption, and hardware cost, dependencies refer to fault coverage, reliability, availability, safety, security, and recovery time. In the co-design flow, the application specifications dictate the design choices. These design options are weighed considering dependability criteria and design limitations.

In determining the set of acceptable software implementations for the system, adopting various software fault mitigation methods will guide the selection. The methods may be used in whole or selectively.

Each resultant software revision is then analysed to determine the degree of software overhead, which is assessed by calculating the total code size and execution time for the hardened and non-hardened programmes. Furthermore, it is possible to do preliminary dependability assessments using a simulation-based reliability evaluation tool. The greatest applicants based on the aforementioned assessments and in accordance with the microprocessor implementation specifications are chosen to be tested on the actual microprocessor implementation. Reliability can only be obtained by

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 7, July 2021 ||

| DOI:10.15680/IJIRSET.2021.1007003 |

evaluating each hardware and software con- figuration, which requires rigorous testing. Once the designer discovers these trade-offs, he or she may evaluate many software parameters, such as size, performance, and dependability.

If no configuration that fulfils all the criteria is found, hardware-based methods must be used to implement the protective strategy. Since this, the designer has to choose a strategy to secure the hardware, prioritising which elements of the design need protection and using redundancy elsewhere. As a result, the hardware cost of a new parameter should be included into the trade-off analysis.

The solution consists of using the best candidates on the software side, and the protected versions of the hardware, and then executing the hybrid fault mitigation method.

Innovation doesn't only happen in one location. In the design space, you have many possible points that may have equal value. Some of these locations will be found to satisfy design limitations and reliability requirements, while others may not. The designer does, however, have enough knowledge to make an informed choice on the optimal system configuration based on the exploratory trade-offs.

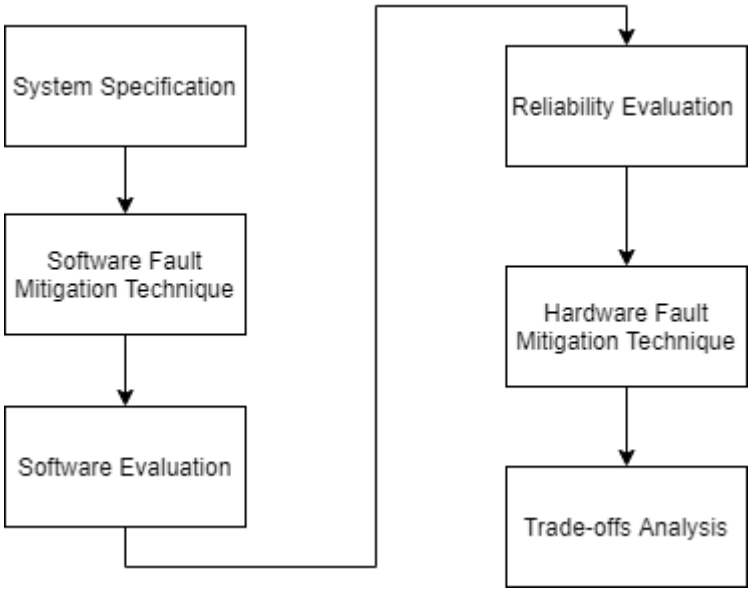

Figure 1: Fault Mitigation

IV.TOOLS USED

Because the suggested method promotes software and hardware fault mitigation approaches, proper design and evaluation tools must be used. Why it is decided to utilise the following tools in this article:

A compiler-based hardening environment for application and software-based mitigation methods to design and apply automatically. The FTUnshades[15] is a fault injection based reliability assessor.

A compiler-based hardening environment is established for fault-tolerant software development, which is a comprehensive tool. Automated fault mitigation can be designed and implemented alongside software-based methods, but software-based techniques may also be used to automatically apply the fault mitigation. The environment is made up of the following components: a multi-target compiler (Hardener), an instruction set simulator (ISS), and several compiler front-ends and back-ends. This tool has the following benefits:

- It is built on a Microprocessor Generic Architecture (MGA) that allows re-targeting the output as required (i.e. to process a code written for any supported microprocessor and generate protected code targeting to the same original architecture or to another one by means of the various back-ends).
- Solves the software fault mitigation design, implementation, and evaluation issues in a platform-independent manner.
- Automatically transforms code according to rules (assembler).

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 7, July 2021 ||

| DOI:10.15680/IJIRSET.2021.1007003 |

- Facilitates the use of software-based methods in a targeted manner. Sphere of Replication (SoR) is the logic domain of duplicate execution, which in turn dictates protective coverage.
- Assists the designer in the decision making since it helps identify software and hardware design elements that need to be secured.

The environment has two distinct purposes, which are to harden the compiler back-end with the compiler front-end, and to simulate the compiler back-end with the compiler front-end. Using the command-line interface, both of these may be controlled. Each tool has to provide source code for simulating and hardening a programme followed by a set of parameters. A description of each tool's major features is shown below.

Since the designer has access to all of the available options, they can select from these options to help them in making a decision: the microprocessor architecture that will be targeted, the hardening technique to apply to the programme, the replication level (duplication, triplication) that will be utilised, and the preferred recovery technique (i.e. among different implementations of majority voters, in case of recovery techniques, or routines to be applied when a fault is detected). Additionally, the Hardener provides the capability to apply a selective hardening approach using software-based methods. In this case, the tool notices that there aren't enough resources to carry on with redundancy, such as more registers, and it may keep doing so until such resources are no longer needed. The designer is able to manage the resources to be hardened, allowing the selective hardening approach to be regulated. In the end, the final produced output of the compiler front-end is the compiled programme with all the chosen choices, and the hardened source code is the original programme with all those options selected.

Information is generated regarding the status of the system's architectural resources throughout and after the simulation process as is typical for the ISS instruction set simulators. Moreover, it allows for the program's functionality to be verified by comparing it to non-hardened functionality. A hardening report after the simulation includes important information, such as: programme size overhead, performance loss, and how many CPU registers each programme accesses.

Additionally, the ISS is able to do a bit-flip test on one bit of the architectural resources to predict SEU errors. While the ISS has access to many key architecture resources, it does not have access to microarchitectural resources such as those registers placed in the pipeline. Because of this, the ISS fault coverage findings should be seen as estimates that should be checked by another tool that has complete access to all micro-architectural resources (FTUnshades).

Also, the FTUnshades is an FPGA-based platform for studying digital circuit dependability, transient defects, and reliability issues under stress. Simulated SEUs by way of dynamic partial reconfiguration trigger bit-flips across the FPGA user registers. This design system is comprised of an FPGA emulation board and a suite of software tools that are used for designing the system, as well as for testing and evaluating outcomes.

Predictive injection is used by FTUnshades, which allows users to understand the effect that hierarchical injection has on each CPU submodule. This means the tool may search for the best candidates to be protected among the interior components.

FTUnshades is a non-intrusive design approach, enabling the full use of all the design's physical resources, but leaving them unchanged. Using the available resources in the FPGA configuration layer enables this to be done. Because the final deployment technology will be a flash-based FPGA, defects introduced into the FPGA user registers will be less likely to affect the user's system than if defects were injected into the SRAM-based FPGA's configuration memory.

For this article, FTUnshades is utilised to examine the whole system physical implementation approach applied to the entire hardware/software mitigation strategy in the system. It is feasible to test the designs even at the micro-architectural level by mimicking SEUs. While mimicking faults in the ISS is accurate, replicating them in the FTUnshades is inaccurate, since faults can only be injected into the microprocessor registers using FTUnshades. In addition, only using FTUnshades can the hardware that has been hardened be tested.

V.CONCLUSION

A soft-core embedded system has been used to create a low-cost and reliable system in this study.In order to refine its risk-reduction approach, the design team focuses on integrating several fault-mitigation methods across both the hardware and software layers. Additionally, the soft-core based system's flexibility provides an easy path to exploration

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 7, July 2021 ||

| DOI:10.15680/IJIRSET.2021.1007003 |

of the design space. It is useful since it makes the assessments of many factors such as performance, code size, and reliability easier. Designers can find a hardware and software system configuration that satisfies both design restrictions and dependability criteria, since this makes things much easier.

This concept motivates the design of fault mitigation to meet application needs as well as to prevent expensive protection measures from being deployed unnecessarily. This helps designers who are creating systems such as spaceship attitude control systems and onboard computing subsystems, since it acts as a guide for their design decisions. One method that may be extended to many application fields, such as the automobile industry, the military industry, and medicine, to name a few.

REFERENCES

[1]. R. Edwards, C. Dyer, E. Normand, Technical standard for atmospheric radiation single event effects (SEE) on avionics electronics, in: IEEE Radiation Effects Data Workshop (REDW), IEEE, 2004, pp. 1–5.
[2]. D. Del Corso, C. Passerone, L.M. Reyneri, C. Sansoè, M. Borri, S. Speretta, M. Tranchero, Architecture of a small low-cost satellite, in: DSD 2007: 10TH EUROMICRO Conf. Digital System Design Architectures, Methods and Tools, Drager, 2007, pp. 428–431.
[3]. T. Arif, Spacecraft Attitude Control, Aerospace Technologies Advancements, Intech, Olajnica 19/2, 32000 Vukovar, Croatia, 2010 (Chapter 4).
[4]. G. Donohoe, D. Buehler, K. Hass, W. Walker, P.S. Yeh, Field Programmable Processor Array: Reconfigurable Computing for Space, in: IEEE Aerospace Conference, Big Sky, MT, 2007.
[5]. ESA, The Radiation Design Handbook ESA PSS-01-609, Tech. rep., European Space Agency, 1993.
[6]. IEC, IEC/TS 62396-1, Tech. rep., International Electrotechnical Commission, March 2006.
[7]. DoD, MIL-HDBK-817, Military Handbook System Develop Radiation Hardness Assurance, Tech. rep., Department of Defense, USA, 1994.
[8]. S. Rezgui, New Reprogrammable and Non-Volatile Radiation-Tolerant FPGA: RT ProASIC3, Aerospace Technologies Advancements, Intech, Olajnica 19/2, 32000 Vukovar, Croatia, 2010 (Chapter 6).
[9]. Xilinx, Radiation-Hardened, Space-Grade Virtex-5QV FPGA Data Sheet: DC and Switching Characteristics, Data Sheet DS692 (v1.0.1), Xilinx Inc., August 2010.
[10]. M. Rebaudengo, M.S. Reorda, M. Violante, M. Torchiano, A source-to-source compiler for generating dependable software, in: Proceedings of the First IEEE International Workshop on Source Code Analysis and Manipulation, 2001, pp. 33–42.
[11]. S. Mukherjee, C. Weaver, J. Emer, S. Reinhardt, T. Austin, A systematic methodology to compute the architectural vulnerability factors for a high- performance microprocessor, in: Proceedings of the 36th International Symposium on Microarchitecture, San Diego, CA, December 03–05, 2003, pp. 29–40.
[12]. J. Von-Neumann, Probabilistic logics and synthesis of reliable organisms from unreliable components, in: C.E. Shannon, E.J. McCarthy (Eds.), Automata Studies, Princeton Univ., Princeton, NJ, 1956, pp. 43–98.
[13]. P. Samudrala, J. Ramos, S. Katkoori, Selective triple modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs, IEEE Transactions on Nuclear Science 51 (5, Part 4) (2004) 2957–2969, doi:10.1109/ TNS.2004.834955.
[14]. N. Oh, S. Mitra, E.J. McCluskey, (EDI)-I-4: error detection by diverse data and duplicated instructions, IEEE Transactions on Computers 51 (2) (2002) 180– 199.
[15]. M.A. Aguirre, J.N. Tombs, F. Munoz, V. Baena, H. Guzman, J. Napoles, A. Fernandez-Leon, F. Tortosa-Lopez, D. Merodio, Selective protection analysis using a SEU emulator: testing protocol and case study over the LEON2 processor, IEEE Transactions on Nuclear Science 54 (4, Part 2) (2007) 951–956.

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor:
7.569

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details