ISSN:0975-9646

A Review on Fault Tolerant Analysis for Hard Real Time Safety Critical Embedded System

Nisha O. S., Dr. K. Siva Sankar

IT Department, Nooral Islam University,
Nagercoil, India

Abstract— **Safety-Critical Applications have to function correctly and deliver high level of quality-of-service even in the presence of faults. But radiation problem in embedded system is a serious threat. There is an increasing concern about the mitigation of radiation effects in embedded systems. The protection of processor-based systems to mitigate the harmful effect of transient faults (soft errors) is gaining importance as technology shrinks. This review paper presents various methodologies for facilitating the design of fault-tolerant embedded systems and is supported by an infrastructure that permits to easily combine hardware/software soft errors mitigation techniques. The proposed system combines hardware and a software mitigation technique, which facilitates the design space exploration, developed to support the fault tolerance co-design approach and is added to the most vulnerable parts. A wide industrial consensus about the necessity of a set of safety definitions leads to the introduction of several functional safety standards. To achieve an embedded systems comply with these requirements, thorough testing is needed during early design stages of the integrated device.**

Keywords—Soft error, Hardening Infrastructure, Software Hardening Environment, SEU, SET.

1. INTRODUCTION

Safety is a property of a system, which will not endanger human life or environment. Now microprocessors became the heart of most of the digital systems because of their capabilities in programming, cost effectiveness and performance. Technology improves a lot which results in the use of microprocessors in various application areas. Several security systems are used and most of them are very powerful also. But the problem faced by most of the Embedded System is the radiation problem. This is due to the miniaturization of electronic components.

One of the important advantages in the life of a microcontroller is the miniaturization of electronic components and it results in the improvement of performance and reduction of silicon area. As technology shrinking occurs automatically the voltage. Source level and noise margins are reduced making electronic devices become less reliable and CMOS circuits are more susceptible to transient faults induced by radiation. This radiation problem doesn't make any damage or harm to the system but results in incorreccircuit behaviour and signal alteration named soft errors. Many safety-critical applications have also strict time and cost constrains, which means that not only faults have to be tolerated but also the constraints should be satisfied.

This paper reviews several design optimization strategies and scheduling techniques that take fault tolerance into account. Finally, quality-of-service aspects have been addressed in the thesis for fault-tolerant embedded systems with soft and hard timing constraints. This review paper present several key challenges and some solutions to the design and optimization of such system. In particular it will answer some questions of if hardening should be done and how much hardening should be implemented.

A co-design methodology [1] is used to mitigate soft errors. A design space with software and hardware is used to achieve customized fault tolerant system to meet requirements of application. A hardening infrastructure is used to generate different versions of the design using several combinations of both hardware and software.

Most efforts are devoted in the mitigation of errors in memory that is Single Event Upset (SEU). With nanometric technologies transient faults in combinational logic circuits are also present known as Single Event Transients (SET) which happens when the charge collected from an ionization event discharges in the form of spur ion signal travelling through the circuit.

When the complexity of the system moves from simple to complex several safety definitions [2] are introduced to support it. A flexible fault injection and power estimation platform to enable through examinations of complex system is needed which determine the fault resistance of embedded system.

This paper presents an approach to system-level optimization of error detection implementation in the context of fault-tolerant real time distributed embedded systems used for safety-critical applications. Main focus in this paper is on the efficient implementation of the error detection mechanisms

This paper is organized as follows: section 2 describes about soft error; Section 3 describes the hardening infrastructure; Section 4 describes the fault injection techniques; Section 5 describes a rapid prototyping platform; Section 6 summarizes some concluding remarks.

II. SOFT ERROR

Soft errors may be caused due to the mistake in design or construction of a component. After observing a soft error, there is no implication that the system is any less reliable than before. In a computer's memory system, a soft error changes an instruction in a program or a data value. A soft error will not damage a system's hardware; the only damage is to the data that is being processed.

- Chip-Level Soft Error
  These errors occur when the radioactive atoms in the chip's material decay and release alpha particles into

the chip. Because an alpha particle contains a positive charge and kinetic energy, the particle can hit a memory cell and cause the cell to change state to a different value. The atomic reaction is so tiny that it does not damage the actual structure of the chip.

- System-Level Soft Error
  These errors occur when the data being processed is hit with a noise phenomenon; typically the data is on a data bus. The computer tries to interpret the noise as a data bit, which can cause errors in addressing or processing program code. The bad data bit can even be saved in memory and cause problems at a later time.

If a soft error is detected, it may be corrected by rewriting the correct data in the place of erroneous data. It is difficult to discover a soft error in the starting stage itself. Before the correction can occur, the system may have crashed. Soft errors involve changes to data storage circuit, but not changes to the physical circuit itself. If the data is rewritten, the circuit will work perfectly again.

One technique that can be used to reduce the soft error rate in digital circuits is called radiation hardening. The reduction of structure, sizes in microcontrollers, an environmental conditions results in increase the susceptibility of embedded systems to soft errors. As a result of this fault detection and tolerance became a mandatory task. In the paper named "A JVM for Soft-Error-Prone Embedded system" [3] describes an automated application of fault detection and tolerance measures based on the type of the system, programming language etc and facilitates an easy evaluation of the protection characteristics and costs.

III. HARDENING INFRASTRUCTURE

In computing, hardening is usually the process of securing a system by reducing its surface of vulnerability; in principle a single-function system is more secure than a multipurpose one.

The purpose of system hardening is to eliminate as many security risks as possible. This is typically done by removing all non-essential software programs and utilities from the computer. While these programs may offer useful features to the user, if they provide "back-door" access to the system, they must be removed during system hardening.

In the paper named "Application-driven co-design of fault-tolerant industrial systems" [4], a hardening infrastructure is used to support fault tolerance co design approach. For the hybrid error mitigation in embedded system a hardware /software co-design technique is used. Applying the protection to the most vulnerable parts of both software and hardware results in fault tolerant system. To achieve this first step is the specification of system requirements. Requirements may be either design constraints or dependability parameters associated with specific application. In the paper "A Co-Design Approach for SET Mitigation in Embedded Systems" [1] describes a co-design flow which is motivated by taking both design constraints and dependability parameters. The protection strategy has to be completed only after the application of hardware based techniques. Therefore the designer has to decide suitable strategies to protect the hardware, for protecting the most vulnerable parts of the hardware.

There are several methods present to mitigate soft errors such as SEU and SET effects in embedded system. The methodologies used are Software Hardening Environment (SHE), SEU Emulation Tools and SET Emulation Tools.

A. Software Hardening Environment

The main goal of the SHE environment is to allow the user to design and implement software –only fault tolerance techniques, and also apply automatically them in programs. It is made up of a hardener and a generic Instruction Set Simulator (ISS) jointly with several compiler front ends and back ends to deal with different microprocessor targets fig 1.

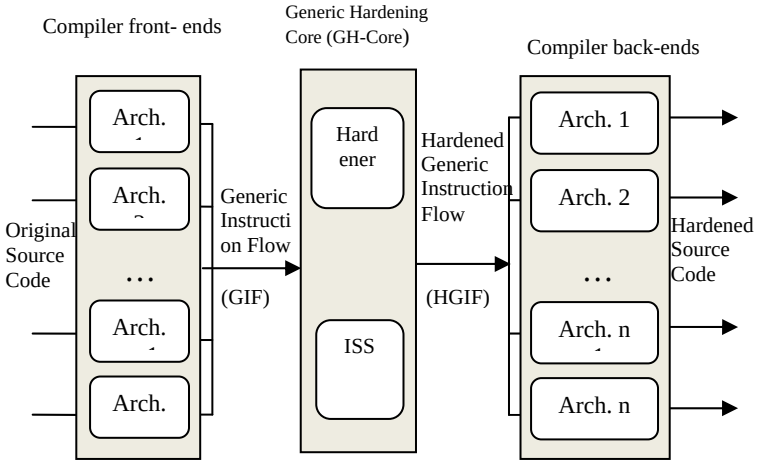

Fig 1: Software Hardening Environment

The ISS offers valuable information about the code overheads of the final result and also has a characterization to simulate program reporting about the resource utilization and register life time.

The given compiler takes the original source code from the supported target performs analysis and finally generates a Generic Instruction Flow (GIF) as output. The output is an intermediate abstraction of a program. Hardening is performed within the Generic Hardening Core (GHC). Finally Hardener produces a Hardened-GIF which is then re-targeted to a custom supported microprocessor.

Considering a hardening process and the hardener manages in a special way that all the instructions are to be cross the boarders of SoR or Sphere of Replication[1]. It's the logical boundary of redundant execution within a system and trade-off between information, time and space redundancy. Here in this paper we can see that ISS performs different analysis on the original and hardened code to verify the correctness of the transformation. Moreover it is also able to simulate SEU and SET faults and to carry out fault injection techniques.

B. SEU Emulation Tools.

SEU's are state changes of memory or register bits caused by a single ion interacting with the chip. They do not cause much damage to the device but may cause lasting damage to the system which cannot recover from such an

error. There are several methodologies are present in the removal of SEU or Single Event Upset. One such method is the compiler level implementation [5] which is present on the paper "Compiler-level implementation of single Event Upset errors mitigation algorithms". Based on the modifications of the source code of the C language compiler protection methods are applied automatically during source code processing at intermediate representation of the compiled program.

Another method is the TMR or Triple Modular Redundancy [9], describes how TMR is implemented to mitigate SEU errors. Here Triple Modular Redundancy is partitioned and is inserted to reduce the SEUs in the FPGA logic paths. In this it creates three identical copies of a module and feeding their outputs into a majority vote, which simply outputs the most popular of the three outputs.

An SRAM based [6] technique is used in which the flipper fault injection platform to allow testing the efficiency of the SEU mitigation scheme is used. One of the main component of the hardening infrastructure is FTUnShades [7]. It is an FPGA based platform for the study of digital circuit reliability against radiation induced soft errors. It permits to inject faults in a selective way on the analysis in microprocessors.

Protection is to be provided in the impact on the reliability of each one of microprocessor sub modules due to hierarchical injection. So it can able to find the best candidates from the internal parts to be protected also helps to permit complete access to all physical resources in the design. We can inject faults in all memory elements of the microprocessor including embedded memories and internal flip flops and not only in microprocessor registers.

C. SET- Emulation System.

Single Event Transients, which are very difficult to model, simulate and analysis than the closely related Single Event Upsets [1]. Single Event Transients are happens when the charge collected from an ionization event discharges in the form of spurious signal travelling through the circuit. Mitigation of SET effects is more costly than mitigate SEU effects. Since it affects any logic node in the circuit also the propagation of SET can produce a multiple error at memory elements or latches.

We can eliminate transient faults by injecting faults into it from an VLIW processors [15] by analyzing the cross domain failures affecting redundant mitigation techniques implemented on a statistically scheduled data path. It describes a fault injection analysis of transient faults affecting the r-VEX VLIW processor implemented on an FPGA platform.

A compiler based [8] methodology is used in SET mitigation. This methodology is supported by an infrastructure that permits to easily combine hardware/software soft error mitigation techniques satisfy both usual design constraints and dependability requirements.

Another technique used is the Invariant Checkers [9]; it is a low cost technique. It uses the software invariants to detect transient errors affecting a system at run time. The technique is based on the use of a publically available tool to automate the invariant detection process and the decomposition of complex algorithms into simpler ones which are checked through verification of their invariants during the execution of the program.

The effectiveness of the co-design relies on the capability to evaluate SET sensitivity in an accurate and fast manner. This goal is achieved using AMUSE system. It is an emulation based system that supports SEU and SET fault injection for any ASIC technology. For SETs pulses of the selected duration can be injected across many clock cycles. AMUSE uses a quantization approach that accurately models dynamic delay effects, including electrical masking effects . It has been also used to evaluate Soft Error Rate due to SET.

IV. FAULT INJECTION TECHNIQUES

Fault injection means the introduction of faults into a system for the simulation and emulation of errors. Researchers have created many novel methods to inject faults, which can be implemented in both hardware and software. Such operation disruptions can be caused by external influences like radiation, attacks, or internal reasons like degradation. Depending on the level of abstraction, several methods have been adopted [2].

A. Hardware Level

Here manufactured devices are available and existing automated test equipment can be reused. The possible points of fault injection are very limited

B. Software Level

The basic principle behind software level fault injection is same as hardware level. The main difference lies in the manipulation target, variables and other system elements are changed directly while the hardware works are originally designed.

There are so many types of errors are seen on embedded system especially in distributed environment. It can be permanent, intermittent or transient. Transient and intermittent faults appear for a short time. The effects of transient and intermittent faults are very high. They may corrupt data or lead to logic miscalculations, which can result in the failure or dramatic quality-of service deterioration. Transient and intermittent faults appear at a rate much higher than the rate of permanent faults and, thus, are very common in modern electronic systems.

Transient and intermittent faults can be seen on hardware devices and can be reduced with hardening techniques, i.e., improving the hardware technology and architecture to reduce the fault rate, or in software. We consider hardware-based hardening techniques and several software-based fault tolerance techniques, including re execution, software replication, and rollback recovery with check pointing.

Engineers most often use low-cost, simulation based fault injection to evaluate the dependability of a system that is in the conceptual and design phases. At this point, the system under study is only a series of high-level abstractions; implementation details have yet to be determined. Thus the system is simulated on the basis of simplified assumptions. Simulation-based fault injection, which assumes that errors or failures occur according to pre

determined distribution, is useful for evaluating the effectiveness of fault-tolerant mechanisms and a system's dependability; it does provide timely feedback to system engineers. However, it requires accurate input parameters, which are difficult to supply:

Design and technology changes often complicate the use of past measurements. Testing a prototype, on the other hand, allows us to evaluate the system without any assumptions about system design, which yields more accurate results. In prototype-based fault injection, we inject faults into the system to [10]

a. identify dependability bottlenecks,
b. study system behaviour in the presence of faults,
c. determine the coverage of error detection and recovery mechanisms, and
d. Evaluate the effectiveness of fault tolerance mechanisms and performance loss.

Fault as a deviation in a hardware or software component from its intended function can arise during all stages in a computer system design process.

Specification, design, development, manufacturing, assembly, and installation throughout its operational life. Most faults that occur before full system deployment are discovered and eliminated through testing. Faults that are not removed can reduce a system's dependability when it is embedded into the system [11].The complex interactions between errors, failures, and fault handling mechanisms can be studied via injection experiments [12].

One method of injecting fault is a multilevel FPGA emulation based fault injection approach for the evaluation of SET effects called AMUSES [13]. This approach integrates Gate level and Register Transfer level models of the circuit under test in a FPGA and is able to switch to the appropriate model as needed during emulation. We can use the fault injection mechanisms for testing of Network-on-Chips [14]. It is an innovative test architecture based on a dual processor system which is able to extensively test mesh based NoCs. It improves methods based on NoCs physical implementation which allows investigating the effects induced by several kinds of faults within the entire network interface and router resources during NoC run time operations.

V. RAPID PROTOTYPING PLATFORM

The proposed infrastructure establishes a complete software hardening development environment allowing the design and implementation of software based techniques to be automatically applied into programs. The infrastructure is made up of a multi-target compiler supporting several common hardening routines, an instruction set simulator and several compiler front ends and back ends. Fig:1.

The identification of the control flow graph and the insertion of instructions into the source code during compilation time are the keys for software based technique [15].

Regarding the memory management, the Hardener is able to identify the memory map, exact memory sections and perform modifications over them and is described on the paper "Rapid prototyping of Radiation Tolerant Embedded Systems in FPGA" , [16]. Three possibilities are supported to keep updated the memory map, dilation, displacement and relocation.

- *Dilation*: When one or more instructions are inserted during compilation time into a memory section, this section grows and some of the instructions addresses inside this memory section should be reassigned.
- *Displacement*: If dilation provokes that two or more memory sections share some addresses, which is an illegal situation, then the section must be completely moved and all its instructions addresses updated.
- *Reallocation*: If there is a memory overflow caused by previous instructions insertions, then it is needed to perform a complete reallocation of the complete memory map. During this process, free memory space among memory sections is fully used. This situation may happen because of the typical reduced memory size in embedded systems.

VI. CONCLUSION

The literature survey revealed that a lot of researches have gone in this area for a radiation tolerant embedded system. This paper presents various methods for embedded system hardening includes software and hardware hardening for the mitigation of soft errors. Another peculiarity of this paper is the introduction of various fault injection techniques to evaluate the fault resistance of safety critical system on chip implementation. Most recent research areas trying to incorporate all the techniques described in this paper to make a perfect radiation tolerant embedded system for real time applications.

ACKNOWLEDGEMENTS

I would like to thank my supervisor and guide Prof. Dr. K. Siva Sankar who advised and encouraged me to research on this area. I would also like to thank all the authors of the journals which helped me in developing this literature survey Finally, I express my special thanks to my family and close friends for their great support throughout my study on this.

REFERENCES

[1] Almudena Lindoso, Luis Entrena, Enrique San Millán, Sergio Cuenca-Asensi, Antonio Martínez- Álvarez, and Felipe Restrepo-Calle, "A Co-Design Approach for SET Mitigation in Embedded Systems" in *IEEE Transactions on Nuclear Science*, Vol. 59, N0. 4, pp. 1034-1039, August 2012.

[2] Armin Krieg,Christopher Preschern, Johannes Grinschgl, , Christian Steger, , Christian Kreiner,, Reinhold Weiss, Holger Bock, and Josef Haid, "Power And Fault Emulation for Software Verification and System Stability Testing in Safety Critical Environments" *in IEEE Transactions on Industrial Informatics*, Vol. 9, No. 2, pp. 1199-1206, May 2013.

[3] Mehdi Modarressi, Hani Javanhemmat , Seyyed Ghasem Miremadi, Shaahin Hessabi, Morteza Najafvand, Maziar Goudarzi, and Naser Mohamadzadeh," A Fault-Tolerant Approach to Embedded-System Design Using Software Standby Sparing", *in Computer Engineering Department, Sharif University of Technology, Tehran*, Iran.

[4] Restrepo-Calle, F., Guzmán-Miranday, H., Palomoy, F.R., Cuenca-Asensi, S., "Application-driven co-design of fault-tolerant industrial systems", in Industrial Electronics (ISIE), 2010 IEEE International Symposium on 4-7 July 2010, pp.2005-2010.

[5] Piotrowski, A. ; Tarnowski, S., "Compiler-level implementation of single Event Upset errors mitigation algorithms" in Mixed Design of

Integrated Circuits & Systems, 2009. MIXDES'09. MIXDES-16th International Conference 25-27 June 2009, pp. 89-92.

[6] Alderighi M,.Casini, F. ; d'Angelo, S. ; Mancini, M. ; Pastore, S. ; Sechi, G.R.," Evaluation of Single Event Upset Mitigation Schemes for SRAM based FPGAs using the FLIPPER Fault Injection Platform, in Defect and Fault-Tolerance in VLSI Systems, 2007. DFT '07. 22nd IEEE International Symposium on , pp.105-113, Sept. 2007.

[7] J. Napoles, H. Guzman, M. Aguirre, J. Tombs, F. Munoz, V. Baena, A. Torralba, and L. Franquelo, "Radiation environment emulation forVLSI designs A low cost platform based on xilinx FPGAs," in *Proc. IEEE Int. Symp. Industrial Electronics*,4-7 July 2007 pp. 3334-3338.

[8] Martinez-Alvarez, A. ;Comput. Technol. Dept., Univ. of Alicante, Alicante, Spain ; Cuenca-Asensi, S. ; Restrepo-Calle, F. ; Pinto, F.R.P, "Compiler-Directed Soft Error Mitigation for Embedded Systems" on Dependable and Secure Computing, IEEE Transactions on March-April 2012. Vol. 9, No. 2, pp.159-172

[9] Grando, C.N. ; Inst. de Inf., Univ. Fed. do Rio Grande do Sul, Brazil ; Lisboa, C.A. ; Moreira, A.F. ; Carro, L, "Invariant checkers: An efficient low cost technique for run-time transient errors detection", in On-Line Testing Symposium, 2009. IOLTS 2009.15th IEEE International conference on 24-26 June 2009, pp. 35 – 40

[10] Mei-Chen Hsueh, Timothy K. Tsai, and Ravishankar K. Iyer University of Illinois at Urbana Champaig, Theme Feature on, "Fault Injection Techniques and Tools" on April 1997 pp. 75-82.

[11] HaissamZiade, RaficAyoubi, and Raoul Velazco, "A Survey on Fault Injection Techniques", in The International Arab Journal of Information Technology, Vol. 1, No. 2, pp.171-186, July 2004.

[12] Christmansson, J. ; Carlstedt Res. & Technol. AB, Goteborg, Sweden ; Hiller, M. ; Rimen, M., "An experimental comparison of fault and error injection" , on Software Reliability Engineering, 1998. ProceedingsThe Ninth International Symposium on**:** 4-7 Nov 1998, pp. 369 – 378.

[13] Entrena, L. ; Dept. of Electron. Technol., Univ. Carlos III of Madrid, Leganés, Spain ; Garcia-Valderas,M. ; Fernandez-Cardenal,R. ; Lindoso.A. in ,"Soft Error Sensitivity Evaluation of Microprocessors by Multilevel Emulation-Based Fault Injection", on Computers, IEEE Transactions onon March 2012 Vol. 61, No. 3, pp. 313-322.

[14] Sterpone, L. ; Dipt. diAutom. e Inf., CAD Group, Italy ; Sabena, D. ; Reorda, M.S, "A New Fault Injection Approach for Testing Network-on-Chips" in Parallel, Distributed and Network-Based Processing (PDP), 2012, 20th Euromicro International Conference on 15-17 Feb. 2012, pp. 530 – 535.

[15] N. oh, P. Shirvani and E.J. McCluskey, "Control flow checking by software signatures", *IEEE Transactions on Reliability*, vol.51, pp.1, 2002.

[16] F. Restrepo-Calle_, A. Mart´ınez-A´ lvarez_, F.R. Palomoy, H. Guzma´n-Miranday, M.A. Aguirrey and S. Cuenca-Asensi, "Rapid Prototyping of Radiation-Tolerant Embedded Systems on FPGA" , *in International Conference on Field Programmable Logic and Applications*,2010,pp.326-331

.