Survey of Lockstep based Mitigation Techniques for Soft Errors in Embedded Systems

Eduardo Weber Wächter, Server Kasap, Xiaojun Zhai, Shoaib Ehsan and Klaus McDonald-Maier
School of Computer Science and Electronic Engineering
University of Essex
Colchester, UK
{eduardo.wachter, server.kasap, xzhai, sehsan, kdm}@essex.ac.uk

***Abstract*—Soft errors are one of the significant design technology challenges at smaller technology nodes and especially in radiation enviro nments. This paper presents a particular class of approaches to provide reliability against radiation-induced soft errors. The paper provides a review of the lockstep mechanism across different levels of design abstraction: processor design, architectural level, and the software level. This work explores techniques providing modifications in the processor pipeline, techniques allied with FPGA dynamic reconfiguration strategies and different types of spatial redundancy.**

***Index Terms*—Lockstep, Reliability, Fault Tolerance, soft error mitigation, radiation effects**

I. Introduction

One of the largest and most complex environmental remediation tasks in the whole of Europe is the cleaning up the legacy nuclear waste, which projected costs more than £115bn and perhaps as high as £220bn, over the next 120 years [1]. Much of this must be done by robots because the materials are too hazardous for humans. But many of the necessary robotic solutions remain to be developed since electronic circuits effects in extreme nuclear radiation environments have not been thoroughly studied. One example is the robots that keep failing when entering the Fukushima nuclear site.

Additionally, robotics will play an increasingly important role in in-service maintenance and inspection of the current nuclear fleet to support plant life extension (PLEX). Finally, robotic systems will soon also become an essential design element of new-build reactors, as well as helping to build them in the first place, thereby reducing risk and cost while concurrently improving safety.

Soft errors are induced by energetic particles (e.g. alpha particles, heavy ions, neutrons) striking the semiconductor substrate of transistors [2]. These errors have a transient behaviour that does not permanently damage digital circuits.

Historically, soft errors have been primarily concerned with the design of reliable systems or systems which are deployed in environments hostile to electronics, such as outer space. As devices scale, the soft error rate for each device due to radiation is projected to remain approximately constant or expected to decrease (depending on the specific device). However, the exponential increase in the integration of the chip (i.e., the computing system) leads to correspondingly dramatic decreases in reliability. Today, soft errors are one of the significant design technology challenges at and beyond the 22 nm technology nodes [3].

This survey introduces soft error mitigation from the perspective of the lockstep mechanism and its variations. It also provides a survey of the existing soft error mitigation methods across different levels of design abstraction: processor design, the architectural level, and the program level.

II. Motivation

There is a strong motivation to use robots, and consequently, electronic devices to enter radiation facilities, e.g. nuclear power plants, nuclear waste disposal sites. When using electronic devices, human beings are spared from entering harsh environments. There are two examples where electronic equipment can be employed and then spare human lives. The first one is to manage (move, dispose of) nuclear waste generated by nuclear plants. The second is in case of a nuclear disaster. In both cases using robots instead of human beings shows a more sensible solution.

One possible path to lead the use of digital circuits in these harsh environments is the adoption of fault mitigation techniques. There are multiple approaches proposed in the literature [3], [4]. The lockstep technique is a well-know technique that has been employed for years with many different implementations. For some cases it shows better area-error correction tradeoffs. Therefore a survey of this techniques is required.

III. Mitigation Techniques at Software Level

Basic mitigation techniques can be classified into two broad categories: (i) detection and (ii) correction techniques. Detection techniques are responsible to identify an error in a given hardware module while correction techniques should correct it (or mask using redundancy modules).

A. Checkpoint Recovery

The goal of checkpoint recovery is to roll back the state of the processor as soon as the occurrence of transient faults is detected. The idea consists of saving the correct state of the system regularly (Checkpoints). If an error occurs, the latest safe state stored is then restored (Recovery). Fig 1 present one example approach used in the State-of-the-Art.

A few changes to the original approach propose to use the roll forward method. In this method, when the system detects that it has made an error, roll-forward recovery takes the system state at that time and corrects it, to be able to continue the execution without errors.

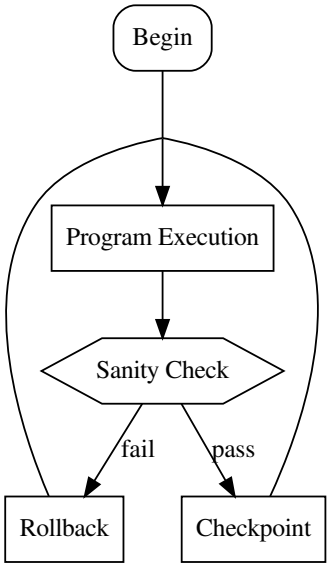

Fig. 1: Flowchart of rollback recovery using checkpoint. [5]

IV. Mitigation Techniques at Hardware Level

A. Dual Modular Redundancy

Dual Modular Redundancy (DMR) is a type of spatial redundancy-based mitigation method. Fig 2 presents an example of possible implementation. These methods rely on the replication of modules to increase the reliability of the system. They do not require multiple repetitions of the computation, but fully or partially duplicate the hardware to implement redundant computation. DMR mainly duplicates the module. This way it can be used for error detection: if the result given by the two modules mismatches and error is detected. Unfortunately, using a comparison module does not point out which one of the modules has been affected by a fault. Expanding the spatial redundancy to three modules deliver the Triple Modular Redundancy (TMR) which then, can be used for error detection and recovery [6], [7].

The main reason most of the works adopt DMR is that the area overhead is lower when compared to TMR. Roughly, the DMR imposes a two times area overhead while TMR introduces three times. Also, the overheads apply to power consumption and critical path (usually there is a comparator at the end).

V. Lockstep Techniques

The first proposals in the literature proposing spatial modular redundancy techniques (e.g. TMR, DMR) date to the early 1960s, i.e. during the start of space programs, since reliability provisions were required to avoid human lives being jeopardised by computer failure [9]. The initial examples

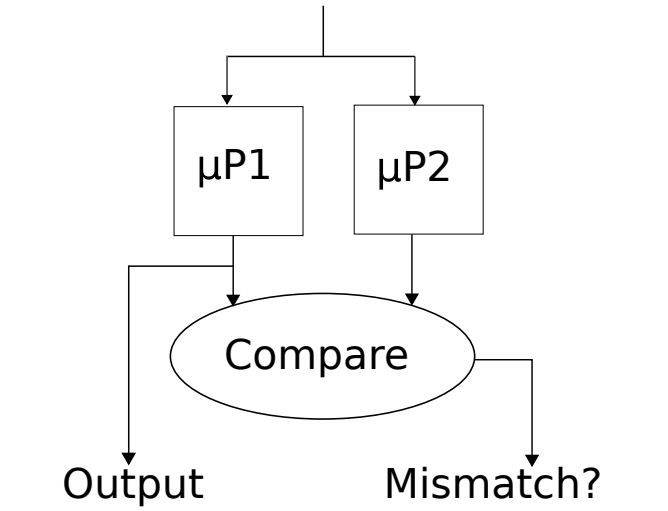

Fig. 2: Duplication with comparison basic lockstep scheme.[8]

proposing a lockstep mechanism started showing in the early 2000s [10]. Table I presents a comprehensive comparison of the further published works.

Hua Ng [11] presents an implementation of a Lockstep mechanism using the EDA (Electronic Design Automation) tools from Xilinx. The implementation employs two PowerPC 405 processors in a Virtex II Pro FPGA, executing a MontaVista Linux operating system. The Author also explain how to manually stop one of the processors to force a mismatch and observe the comparator module compute it. There are no results showing area overheads or fault injection.

In 2008 Abate et al. [5] present a standard implementation of lockstep mechanism on a Xilinx FPGA containing two PowerPC processors. Soon after they extend [12] the approach trying to reduce the saving context overheads. The novelty is a hardware module named Write History Table (WHT) that reduces the checkpoint overhead temporarily storing the addresses and values that have been written by the application during one execution cycle. In this way, the checkpoint is only executed for the modified addresses and values. The Authors do not present any results of the recovery overheads neither apply fault injection or actual radiation tests.

In [13] the Authors present a lockstep mechanism with LEON3 processors with two FPGA boards. The second board is responsible for rebooting the other FPGA when two consequent rollbacks are executed. The target is a space application that runs a periodic task. Later, the same group extended this work [14] and proposed an automated design environment capable of generating designs and necessary libraries for some families of SPARC and PowerPC processors. Both works evaluated the designs under radiation-induced heavy ions in two scenarios to estimate the availability of the system. Results have shown that the area overhead, as expected, is less than a TMR implementation: roughly 2.10 times for the lockstep while the TMR is 3.50 times the baseline processor. The Authors also reveal the finding of persistent errors in 1% of the injected faults. In those cases a rollback is going to use

TABLE I: Comparison of State-of-the-Art implementations of Lockstep mechanism.

Author	Year	Processor	Approach	Area Overhead	Saving Overhead	Recovery Overhead	Actual Radiation Test	Fault Injection Simulation	Persistent Errors
Xilinx [11]	2007	PowerPC	Lockstep with Linux	N/A	N/A	N/A	no	yes	N/A
Abate [5]	2008	PowerPC	Classic Lockstep	5869 LUTs 16x 16KB BRAM	N/A	N/A	no	yes	N/A
Abate [12]	2009	PowerPC	reduce checkpoint with HW module	6 991 slices and 48x16kB BRAM	10x app exec time	N/A	no	yes	N/A
Reorda [13]	2009	Leon 3	2 FPGAs: 1 for reconfiguring	30% of slices 16% of memory of Xilinx xc2v6000	N/A	N/A	no	no	every 14 s to 1 hour
Violante [14]	2011	Leon	Classic Lockstep	2.10x compared to baseline	17% to 53% app time	250ms	no	yes	every 14 s to 1 hour
DARA [15]	2012	SH-2 ISA	double pipeline checkpoint treated as a branch misprediction	5.47 mm^2 for 0.18 μm technology	not necessary	negligible	yes	yes	N/A
Pham [8]	2013	Microblaze	roll-forward	2,97x slices 112KB BRAM	2.6us (325 cc)	494us	no	yes	2.3%/0.4%
Oliveira [16]	2018	ARM A9	interruption checks registers values. If mismatch, interruption to recover	275% LUTs 244% FF compared to baseline	1.26x to 6.48x app time	N/A	yes	yes	N/A

a faulty context, forcing the processor to fail again. To solve this issue the Authors adopt a configuration memory scrubbing [17] and a processor reset to correct it.

[15] presents an implementation of a dual SH-2 instruction set architecture to provide fault tolerance for SEU caused by radiation. The actual prototype shows a DMR implementation with a SH-2 instruction set architecture. The idea is that every time a pipeline stage does not match in the two implementations, the last valid instruction is roll-back as it would occur in a normal branch misprediction. As the recovery is executed at the pipeline level, the recovery overhead is negligible and there is no saving overhead. The authors also prototyped the chip in a 0.18 μm technology and executed a radiation test to evaluate the behaviour under fault injection. Results observed a 0.58 bit per second error injection rate on the actual chip while having 0.3 to 0.4 recoveries per second.

In [8] the Authors propose a lockstep mechanism using two MicroBlaze processors allied with a fault-tolerant Configuration Engine (CE) using a TMR PicoBlaze core. The approach relies on the partial reconfiguration technique present on Xilinx FPGAs, allowing an affected area by SEU to heal once the device is reconfigured so that the design can return to normal operation. The CE has a scan motor module that continuously reads the configuration bits of the FPGA to check for errors. In paralell the two MicroBlaze outputs are compared. Upon a mismatch detection, the system is stopped, the CE then scans the comparator, and then the MicroBlaze cores. If a different configuration bit is found, the PicoBlaze core communicates with the interface to start the partial reconfiguration of the selected area. The procedure uses a roll-forward error recovery that reconfigures only the faulty MicroBlaze processor (or the comparator). The saving overhead is small (325 clock cycles or 2.6 μs) while the recovery is costly due to the partial reconfiguration (494 μs). The Authors also report persistent errors — 2.3% for MicroBlaze and and 0.4 % for the comparator.

In [16] a proposal for rollback/recovery is proposed for FPGAs containing ARM A9 processors. The proposal is a dual-core lockstep (DCLS) fault-tolerance technique to mitigate radiation-induced faults in embedded processors. The DCLS system is composed of a dual-core ARM (CPU0 and CPU1), two BRAM memories, an external DDR memory, and a checker module. Each ARM CPU is connected to its own private dual-port 64-KB BRAM memory protected by error detection and correction (EDAC), where all the data and context are stored. The lockstep works by executing the same application in both cores simultaneously. The application is divided into blocks, and a verification point (VP) is added between each one. When the execution reaches a VP, the processor status is saved, and its execution paused, then a hardware module compares it to the last safe stored context. If no difference is detected, the system is considered to be in a safe state, and it may execute the next block. If there is any mismatch in the results, an interruption is generated to recover the processors using the rollback mechanism. Unfortunately, this approach is not capable of correcting all faults. If a bit-flip affects one of the specific registers, generating illegal data or instruction, the system may lead the ARM processor to stop handling interruptions.

Finally, [18] shows a fault analysis for state-of-the-art lockstep techniques. The approach employs an FMEA (Fault Mode and Effects Analysis) tool to execute a fault-tree analysis. Based on this, a proposal is presented, but not implemented, on how to achieve higher reliability of these techniques.

VI. Discussion

A few proposals that avoid the context saving overhead seems promising [15] and [8]. The first one focuses on the internal modifications that do not require to pay the overhead to save the context since it only requires the same action as a branch misprediction in the pipeline. The drawback is that this approach requires a one-time costly implementation for each different architecture since the designer should know the details of the hardware implementation to apply it. The second works come with a different approach that does not need to modify the internal hardware of the processors. The drawback is the requirement of a dynamic partial reconfiguration unit and additional modules to determine the location to be reconfigured.

A good idea to avoid internal hardware modifications is to use COTS (Commercial Off-The-Shelf) processors which some works propose [5], [12], [13], [14], [16]. The drawback seems the limitation of flexibility this choice imposes. Some works [16] cite problems with handling interruptions that could be avoided with little modifications in the hardware platform.

The opposite approach is presented in [15] where a low recovery overhead is necessary. The authors employed a thorough modification of the pipeline to allow to handle a mismatch of the outputs as an instruction branch misprediction. This is an example that with the flexibility of internal hardware modifications could generate substantial savings, in this case, context saving and recovery overhead.

Some works explicitly avoid using caches [8], [16] since if they are enabled, the lockstep process becomes more complicated. This is necessary for the context saving step, and a cache flush is also required during the rollback, which may lead to a higher impact on performance overhead. Furthermore, the cache would need protection from errors, like error-correcting code (ECC).

Only a couple of works [15], [16] carry out actual radiation tests. In the first case, the proposed circuit was capable of recovery from injected faults. In the second case, if the fault affects come control-flow registers, a hard reset is necessary.

One could divide the different approaches into two categories: intrusive and non-intrusive. The non-intrusive approaches employ COTS processors to implement the lockstep method while the intrusive ones need internal modification of the processor. Non-intrusive approaches seem to lack in excellent features that could force the adoption of this approach. In the other side, intrusive works applying architectural modifications shows good results such as very low recovery overhead and roll-forward techniques.

No current work adopts a state of the art full open source (hardware code and ISA) hardware processors like RISC-V [19] which has become widely adopted in FPGAs development and for the open source community.

VII. Conclusion

This paper presents a comprehensive survey of lockstep approaches in the state-of-the-art of the literature. Different state-of-the-art approaches are presented and compared. The works are compared concerning area overhead, context saving and restore overheads.

Acknowledgment

This work is partially supported by the UK Engineering and Physical Sciences Research Council through grants EP/R02572X/1 and EP/P017487/1.

References

[1] "NDA," https://www.gov.uk/government/publications/nuclear-provision-explaining-the-cost-of-cleaning-up-britains-nuclear-legacy/nuclear-provision-explaining-the-cost-of-cleaning-up-britains-nuclear-legacy [Accessed: 2019-03-29].

[2] R. Baumann, "Soft errors in advanced computer systems," *IEEE Design Test of Computers*, vol. 22, no. 3, pp. 258–266, May 2005.

[3] T. Li, J. A. Ambrose, R. Ragel, and S. Parameswaran, "Processor design for soft errors: Challenges and state of the art," *ACM Comput. Surv.*, vol. 49, no. 3, pp. 57:1–57:44, Nov. 2016.

[4] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secur. Comput.*, vol. 1, no. 1, pp. 11–33, Jan. 2004.

[5] F. Abate, L. Sterpone, and M. Violante, "A new mitigation approach for soft errors in embedded processors," *IEEE Transactions on Nuclear Science*, vol. 55, no. 4, pp. 2063–2069, Aug 2008.

[6] F. Lima, C. Carmichael, J. Fabula, R. Padovani, and R. Reis, "A fault injection analysis of virtex fpga tmr design methodology," in *6th European Conference on Radiation and Its Effects on Components and Systems*, Sep. 2001, pp. 275–282.

[7] L. Sterpone and M. Violante, "A new partial reconfiguration-based fault-injection system to evaluate seu effects in sram-based fpgas," *IEEE Transactions on Nuclear Science*, vol. 54, no. 4, pp. 965–970, Aug 2007.

[8] H. Pham, S. Pillement, and S. J. Piestrak, "Low-overhead fault-tolerance technique for a dynamically reconfigurable softcore processor," *IEEE Transactions on Computers*, vol. 62, no. 6, pp. 1179–1192, June 2013.

[9] A. Aviziens, "Fault-tolerant systems," *IEEE Transactions on Computers*, vol. 25, no. 12, pp. 1304–1312, dec 1976.

[10] J. Klecka, W. Bruckert, and R. Jardine, "Error self-checking and recovery using lock-step processor pair architecture," patent US6 393 582B1.

[11] H. H. Ng, "Ppc405 lockstep system on ml310," Xilinx, Tech. Rep. Xilinx Application Note 564, 2007.

[12] F. Abate, L. Sterpone, C. A. Lisboa, L. Carro, and M. Violante, "New techniques for improving the performance of the lockstep architecture for sees mitigation in fpga embedded processors," *IEEE Transactions on Nuclear Science*, vol. 56, no. 4, pp. 1992–2000, Aug 2009.

[13] M. S. Reorda, M. Violante, C. Meinhardt, and R. Reis, "An on-board data-handling computer for deep-space exploration built using commercial-off-the-shelf sram-based fpgas," in *2009 24th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, Oct 2009, pp. 254–262.

[14] M. Violante, C. Meinhardt, R. Reis, and M. Sonza Reorda, "A low-cost solution for deploying processor cores in harsh environments," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 7, 2011.

[15] J. Yao, S. Okada, M. Masuda, K. Kobayashi, and Y. Nakashima, "Dara: A low-cost reliable architecture based on unhardened devices and its case study of radiation stress test," *IEEE Transactions on Nuclear Science*, vol. 59, no. 6, pp. 2852–2858, Dec 2012.

[16] B. de Oliveira, G. S. Rodrigues, F. L. Kastensmidt, N. Added, E. L. A. Macchione, V. A. P. Aguiar, N. H. Medina, and M. A. G. Silveira, "Lockstep dual-core arm a9: Implementation and resilience analysis under heavy ion-induced soft errors," *IEEE Transactions on Nuclear Science*, vol. 65, no. 8, pp. 1783–1790, Aug 2018.

[17] C. C. Yui, G. M. Swift, C. Carmichael, R. Koga, and J. S. George, "Seu mitigation testing of xilinx virtex ii fpgas," in *2003 IEEE Radiation Effects Data Workshop*, July 2003, pp. 92–97.

[18] J. Arm, Z. Bradac, and R. Stohl, "Increasing safety and reliability of roll-back and roll-forward lockstep technique for use in real-time systems," vol. 49, no. 25, pp. 413 – 418, 2016, 14th IFAC Conference on Programmable Devices and Embedded Systems PDES 2016.

[19] K. Asanović and D. A. Patterson, "Instruction sets should be free: The case for risc-v," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146, Aug 2014.