Evaluating Compiler-Based Reliability with Radiation Fault Injection

Davide Baroffio*, Tomás Antonio López†, Federico Reghenzani‡, William Fornaciari§
Department of Electronics, Information and Bioengineering
Politecnico di Milano, Milan, Italy
davide.baroffio@polimi.it*, tomasantonio.lopez@polimi.it†, federico.reghenzani@polimi.it‡, william.fornaciari@polimi.it§

***Abstract*—Compiler-based fault tolerance is a cost-effective and flexible family of solutions that transparently improves software reliability. This paper evaluates a compiler tool for fault detection via laser injection and α-particle exposure. A novel memory allocation strategy is proposed to mitigate the effects of multi-bit upsets. We integrated the detection mechanism with a recovery solution based on mixed-criticality scheduling. The results demonstrate the error detection and recovery capabilities in realistic scenarios: reducing undetected errors, enhancing system reliability, and advancing software-implemented fault tolerance.**

***Index Terms*—SIHFT, compilers, fault tolerance**

I. Introduction

Safety-critical systems face increasing reliability challenges from soft errors caused by environmental radiation [1]. Software-Implemented Hardware Fault Tolerance (SIHFT) techniques address these challenges by moving the fault tolerance mechanisms into software. Compared to hardware solutions, SIHFT enables gains in terms of cost reduction and flexibility, as it enables the adoption of Commercial-Off-The-Shelf (COTS) components in critical systems [2].

ASPIS (Automatic Software-based Protection and Integrity Suite)[1] is an LLVM-based SIHFT solution that applies data and Control-Flow Graph (CFG) protection by implementing – and enhancing [3] – three state-of-the-art algorithms: EDDI [4], CFCSS [5] and RASM [6]. While prior studies primarily focused on evaluating ASPIS through software-based fault injection [3], this work does so by using physical radiation testing. The experiments were conducted via laser-based testing and exposure to californium isotopes (^{252}Cf) at the Materials & Electrical Components Laboratory, part of the ESA ESTEC facility, in The Netherlands. New reliability challenges surfaced throughout the experiments due to Multi-Cell Upsets (MCUs). This work evaluates a new memory allocation strategy to mitigate these issues and evaluates ASPIS's integration with FreeRTOS for task recovery in real-time systems. With the introduction of SIHFT, fault modelling began considering more fine-grained scenarios where a (hardware) fault may or may not cause a software error, which in turn may or may not cause a system failure. Compared to previous evaluations that performed software-level bit-flip injection into registers and SRAM cells, the present experiments expand the fault model to multi-bit and multi-cell upsets via laser and radiation testing. Laser-based fault injection provides a controllable experimental setup as it allows experimenters to target exact locations on the silicon die and regulate the energy and duration of the pulse [7]. Radiation testing via ^{252}Cf, on the other hand, is traditionally used to test against alpha radiations that are typically emitted by impurities in the packaging materials of COTS devices [8], and heavy ions [9], [10].

This work is partially supported by the National Resilience and Recovery Plan (PNRR) through the National Center for HPC, Big Data and Quantum Computing and the European Space Agency (OSIP no. 4000133770/21/NL/MH/hm).

[1]https://github.com/HEAPLab/ASPIS.

II. ASPIS for detection and recovery

ASPIS is a set of LLVM passes that provides data and CFG fault detection at the intermediate-representation (IR) level.

We recommend reading the original paper on ASPIS for further details [3].

A. Protecting against MBUs and MCUs

The assumption of ASPIS that the hardware suffers from a single-bit upset at a time did not hold in the experiments we performed, as both sources caused multi-bit and multi-cell upsets. Originally, ASPIS interleaved original and duplicate data (AlternatedMemMap). However, this negatively impacted the system reliability against MCUs due to the geometrical topology of the SRAM. In fact, through laser experiments, we observed that targeting an SRAM cell induced bit-flips in the same bit across multiple cells. This behaviour compromises system reliability, as a fault could corrupt both the original data and its duplicate, making the fault undetectable. To tackle this, we implemented a new memory allocation strategy (SequentialMemMap) that allocates all original variables before all duplicates in global and local stack variables. This way, we minimized the possibility of having Silent Data Corruptions (SDCs) due to contiguously allocated original-duplicate pairs.

B. ASPIS and recovery on FreeRTOS

At the time of writing, the literature shows that ASPIS was tested only as a detection mechanism and not for recovery purposes. Despite not providing any built-in recovery mechanism, we integrated ASPIS with a task re-execution mixed-criticality recovery approach by modifying the FreeRTOS scheduler. After ASPIS detects a fault during task execution, it triggers a recovery routine that modifies the list of tasks scheduled by FreeRTOS. In particular, the criticality of the system is increased causing only high-criticality tasks to be executed, and the broken task is re-instantiated and re-executed to guarantee

978-3-9826741-0-0/DATE25/© 2025 EDAA

TABLE I
TESTING OUTCOME RATIOS WITH RESPECT TO THE NUMBER OF EXECUTED JOBS ON MM AND FREERTOS. EACH BENCHMARK WAS TESTED UNPROTECTED, WITH ASPIS USING ALTERNATEDMEMMAP, AND VIA LASER AND CALIFORNIUM WITH ASPIS USING SEQUENTIALMEMMAP.

		SDC (%)	Detected (%)	Hard fault (%)	Timeout (%)	No Effect (%)	#jobs	ε ($\pm$)	SDC/bit-flip ($\cdot 10^{-2}$)
MM	Unprot.	4.418	-	0.080	0.200	95.396	27910	0.771	1.4875
	Alt.	0.025	16.707	0.108	0.141	82.977	12019	1.175	0.0047
	Seq.	0.034	18.389	0.119	0.124	81.330	23449	0.841	0.0065
	^{252}Cf Seq.	0.076	35.078	0.824	0.090	63.932	21110	0.89	0.0181
FreeRTOS	Unprot.	0.635	-	0.948	0.627	97.790	66976	0.498	0.0987
	Alt.	0.060	2.440	2.808	0.480	94.212	160435	0.322	0.0053
	Seq. Recovery	0.016	2.098	1.174	0.260	96.453	48486	0.585	0.0024
	^{252}Cf Seq.	0.105	5.423	1.241	0.004	93.226	45677	0.60	0.0061

output correctness. The mixed-criticality real-time scheduling mechanism guarantees timing correctness and determinism.

III. EXPERIMENTAL EVALUATION

The experimental campaign was conducted using a specifically designed HW/SW stack, and the device under test was an STM32 microcontroller (STM32F427VIT6TR). To get an experimental setup that resembled an actual irradiated environment, we performed laser fault injection on the SRAM of the microcontroller and whole-die α-testing with ^{252}Cf.

During the experiments, the DUT was running different software configurations hardened with different memory allocation strategies. In particular, we tested different bare metal applications and a full-fledged real-time operating system (FreeRTOS). At the end of each job execution[2], the result is compared with the output of a golden run to detect Silent Data Corruptions (SDCs). The output of each test can be either: no effect, SDC (an undetected corruption), detected by ASPIS, timeout, or hard fault. The most critical type of error is SDC, as an incorrect output would go unnoticed by the system. Hence, we focus on reducing this category of errors.

Table I shows the results of the experimental campaigns under both laser fault injection and α particle irradiation. The results are reported with respect to the total amount of jobs executed. Moreover, we report the statistical error ε with confidence of 99% and the SDC rate normalized per bit-flip. This last value is computed as follows. Let δ_{E_t} be the SDC percentage per job for an experiment E and a benchmark t, as reported by Table I. We call D_{E_t} the average amount of bit-flips per job, computed experimentally. We obtain the probability that a bit-flip causes an SDC as the ratio $\frac{\delta_{E_t}}{D_{E_t}}$. Compared to other compiler-based solutions, like COAST [11], ASPIS provides more than double the improvement in MWTF [12] on the MM benchmark ($14.91\times$ vs. $7.1\times$ of COAST), confirming its efficacy under real irradiation conditions.

A. Impact of memory allocation

From the experimental campaigns, we concluded that SequentialMemMap is particularly effective if the program data is highly heterogeneous. However, if the program data primarily consists of large data structures (e.g., MM) or is not memory-intensive, the effectiveness of the allocation strategy becomes code-dependent. This is because the order of data allocation can unpredictably influence the compiler backend's register spilling behaviour. We intend to explore this further in future research.

B. Recovery capabilities

Recovery experiments with FreeRTOS show that 40% of detections were successfully recovered using the mechanism described in Section II-B, highlighting ASPIS utility in real-time systems. For all the other cases, the system detected the error but crashed or went into a timeout before producing any result. This is because recovery was only possible during task execution; OS-level recovery is yet to be explored.

[2]In the case of FreeRTOS, *job* refers to the execution of a whole hyperperiod.

REFERENCES

[1] J. L. Autran and D. Munteanu, *Soft Errors: From Particles to Circuits (1st ed.)*. CRC Press, 2015.

[2] O. Goloubeva, M. Rebaudengo, M. Reorda, and M. Violante, *Software-Implemented Hardware Fault Tolerance*. Springer US, 2006. [Online]. Available: https://books.google.it/books?id=qX9GAAAAQBAJ

[3] D. Baroffio, F. Reghenzani, and W. Fornaciari, "Enhanced compiler technology for software-based hardware fault detection," *ACM Trans. Des. Autom. Electron. Syst.*, apr 2024. [Online]. Available: https://doi.org/10.1145/3660524

[4] N. Oh, P. Shirvani, and E. McCluskey, "Error detection by duplicated instructions in super-scalar processors," *IEEE Transactions on Reliability*, vol. 51, no. 1, pp. 63–75, 2002.

[5] N. Oh, P. Shirvani, and E. McCluskey, "Control-flow checking by software signatures," *IEEE Transactions on Reliability*, vol. 51, no. 1, pp. 111–122, 2002.

[6] J. Vankeirsbilck, N. Penneman, H. Hallez, and J. Boydens, "Random additive signature monitoring for control flow error detection," *IEEE Transactions on Reliability*, vol. 66, no. 4, pp. 1178–1192, 2017.

[7] A. K. Richter and I. Arimura, "Simulation of heavy charged particle tracks using focused laser beams," *IEEE Transactions on Nuclear Science*, vol. 34, no. 6, pp. 1234–1239, 1987.

[8] *Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices*, JEDEC Std. JESD89B, sep 2021.

[9] J. Karlsson, U. Gunneflo, and J. Torin, *Use of Heavy-Ion Radiation from 252Californium for Fault Injection Experiments*. Vienna: Springer Vienna, 1991, pp. 197–212. [Online]. Available: https://doi.org/10.1007/978-3-7091-9123-1_9

[10] J. H. Stephen, T. K. Sanderson, D. Mapper, J. Farren, R. Harboe-Sorensen, and L. Adams, "Cosmic ray simulation experiments for the study of single event upsets and latch-up in cmos memories," *IEEE Transactions on Nuclear Science*, vol. 30, no. 6, pp. 4464–4469, 1983.

[11] M. Bohman, B. James, M. J. Wirthlin, H. Quinn, and J. Goeders, "Microcontroller compiler-assisted software fault tolerance," *IEEE Transactions on Nuclear Science*, vol. 66, no. 1, pp. 223–232, 2019.

[12] G. Reis, J. Chang, N. Vachharajani, S. Mukherjee, R. Rangan, and D. August, "Design and evaluation of hybrid fault-detection systems," in *32nd International Symposium on Computer Architecture (ISCA'05)*, 2005, pp. 148–159.