Exploring potentialities of energy-connected buildings: Performance assessment of an innovative low-exergy design concept for a building heating supply system

Soft Error Effects on Arm Microprocessors: Early Estimations vs. Chip Measurements

Pablo R. Bodmann[*] George Papadimitriou[†] Rubens L. Rech Junior[*] Dimitris Gizopoulos[†] Paolo Rech[‡]
[*]*PPGC, Institute of Informatics, Federal University of Rio Grande do Sul,* Porto Alegre, Brasil
[†]*University of Athens,* Greece
[‡]*Dept. of Control and Computer Engineering, Politecnico di Torino,* Turin, Italy

Abstract—Extensive research efforts are being carried out to evaluate and improve the reliability of computing devices either through beam experiments or simulation-based fault injection. Unfortunately, it is still largely unclear to which extend fault injection can provide an accurate error rate estimation at early stages and if beam experiments can be used to identify the weakest resources in a device. The importance and challenges associated with a timely, but yet realistic reliability evaluation grow with the increase of complexity in both the hardware domain, with the integration of different types of cores in an SoC (System-on-Chip), and the software domain, with the OS (operating system) required to take full advantage of the available resources.
In this paper, we combine and analyze data gathered with extensive beam experiments (on the *final* physical CPU hardware) and microarchitectural fault injections (on *early* microarchitectural CPU models). We target a standalone Arm Cortex-A5 CPU and an Arm Cortex-A9 CPU integrated into an SoC and evaluate their reliability in bare-metal and Linux-based configurations. Combining experimental data that covers more than 18 million years of device time with the result of more than 176,000 injections we find that both the SoC integration and the presence of the OS increase the system DUEs (Detected Unrecoverable Errors) rate (for different reasons) but do not significantly impact the SDCs (Silent Data Corruptions) rate which is solely attributed to the CPU core. Our reliability analysis demonstrates that even considering SoC integration and OS inclusion, early, pre-silicon microarchitecture-level fault injection delivers accurate SDC rates estimations and lower bounds for the DUE rates.

Index Terms—CPU reliability, soft errors, failures in time, neutron beam, microarchitecture-level fault injection, performance models

1 Introduction

Reliability is today one of the main constraints for computing devices employed in several domains, from High Performance Computing (HPC) to safety-critical markets [1], [2]. High device or application error rates have been proved to lower scientific productivity of large scale HPC infrastructures, resulting in significant monetary loss [3]. When the computing device is integrated into cyber-physical systems such as cars, airplanes, or Unmanned Aerial Vehicles (UAVs), high reliability becomes paramount as human lives are at risk.

Arm CPU architectures, thanks to their efficiency and flexibility, have been widely adopted in portable user devices such as smartphones, tablets, and laptops. Additionally, today's fastest supercomputer, Fugaku (RIKEN, Japan) [4] and the next clusters of the US Department of Energy's labs at Sandia and Los Alamos, USA [5] are powered by Arm CPUs. Additionally, Arm is working on a dedicated line of products for autonomous vehicles. As Arm moved from consumer applications to HPC and safety-critical applications, their architecture reliability has become a crucial concern for the company and its partners ecosystem [2]. The ability to implement architectural modifications on Arm cores can be extremely beneficial, as selective fault tolerance solutions can be added at the microarchitectural level, once the weakest hardware resources have been identified.

Microprocessors' soft error reliability can be estimated pre-silicon using early design models (called performance or microarchitectural models) and post-silicon by accelerated beam testing on manufactured chips. Despite the popularity of the two approaches, there are three major unknowns. (1) It is still largely unclear if the CPU reliability estimation based on early simulation models (through fault injection or analytical models) provides realistic soft error rates. Without strong evidence that a microarchitectural reliability evaluation is accurate, it is impossible to guarantee that hardening solutions based on weak resources identification will be effective, once implemented [6]. (2) While the CPU early model does not include any other computing core or peripherals, the final product is normally integrated into heterogeneous SoCs (System-On-Chips). Although the reliability of standalone CPUs, FPGAs, and GPUs has already been extensively studied [7], [8], [9], [10], [11], it is still unknown if the CPU reliability estimation remains accurate when the CPU core is integrated into an SoC. (3) Although previous works have evaluated the effects of the OS on codes reliability and even designed a radiation hardened OS [12], [13], there is no concrete indication about the influence of the operating system (OS) on the accuracy of CPU reliability evaluation based on early models. The scope of our work is to investigate and quantify these unknowns.

In this paper, we consider Arm Cortex-A5 and Cortex-A9 that ideally support our investigations: unlike other ISAs

and CPU designs, they are available both in hardware platforms and as microarchitecture-level models (in the widely used gem5 simulator). Through this dual setup, we can combine the realistic physical measurement of the FIT (Failures-in-Time, i.e., failures per 10^9 hours of operation) rates obtained through extensive beam experiments on actual chips (equivalent to 18 million years of natural exposure) with the fine-grain, hardware structure based analysis of faults propagation obtained with microarchitecture fault injection in early stage CPU models (based on 176,000 injections).

To understand the accuracy of microarchitecture fault injection we compare the predicted FIT rate on a gem5 model with the FIT rate measured with beam experiments on a *stand alone* Cortex A5. To consider the design complexity that derives from cores integration, we consider a Cortex A9 *embedded in an SoC*. To better understand the OS influence on the system reliability, we perform the dual experiments both on the A5 and A9 executing codes *bare metal and on top of the Linux* OS.

Figure 1 shows the motivation of this work and an abstract view of the beam experiments versus fault injection FIT rate evaluations. Even on a stand-alone CPU running code bare-metal (leftmost part of Figure 1), assuming fault injection and beam experiments are performed on exactly the same hardware and software setups, there are still several reasons for beam and fault injection FIT rates not to be identical and for neither of them to be perfectly accurate if compared to the true device FIT rate. As some device structures cannot be modeled (e.g. combinational logic and random sequential), fault injection is likely to underestimate the device FIT rate. Additionally, a simplified fault model (typically a single bit flip) is normally used for injections, which adds additional uncertainty to the predicted error rates. On the other hand, when the real hardware is exposed to accelerated neutrons, the whole chip is irradiated and much more realistic behavior is modeled. However, some resources/interfaces in the test board which are not part of the evaluated CPU are exposed to the beam and can be corrupted causing unresponsiveness. Such cases can lead to an overestimation of the device FIT rate. In addition, the particles counts in the irradiation facility are not as precise as fault injection, adding uncertainty to beam results.

The middle part of Figure 1 shows the case of a CPU core embedded in an SoC (without an OS). The integration of the SoC can potentially modify the system FIT rates for better or worse. Finally, the rightmost part of the Figure 1 shows a system with an OS on top of which the user codes are executed.

The main contributions of this paper are:

- An evaluation of the impact of SoC integration and OS deployment in Arm CPUs reliability, based on beam experiments.
- A fine-grained microarchitectural component-based fault injection analysis of the vulnerability of codes executed on top of Linux and bare-metal in Arm CPUs.
- Hints and guidelines on how to estimate, pre-silicon, the reliability of a CPU as stand-alone or integrated into an SoC and when executing programs on top of an OS.

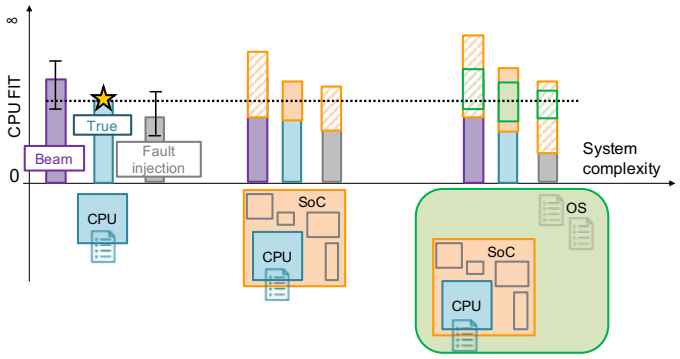


Fig. 1. Abstract view of the FIT rates evaluation uncertainties and possible impact of SoC integration and operating system on the CPU error rate.

We show that employing an early-stage model of the CPU microarchitecture and the manufacturing technology in which the CPU will be implemented can provide an accurate first-order estimation of the expected SDC (Silent Data Corruption) rate, even if the CPU is integrated into an SoC or OS is employed. On the contrary, the DUE (Detected Unrecoverable Errors) can be significantly underestimated if only microarchitecture fault injection is considered. The comprehensive comparison of microarchitecture level fault injection with beam experiments is a step forward in the quest for accurate early predictions of the FIT rates; such predictions allow decision-making during initial design phases to improve the product reliability.

The rest of the paper is organized as follows. Section 2 summarizes background material and reports related work to highlight our contributions. Section 3 presents a description of the adopted evaluation methodologies (architectures, codes, and experimental setups). In Section 4 we combine and in Section 5 we compare the results of beam and microarchitectural fault injection, evaluating the SoC integration and the OS impact on the Arm CPUs reliability. Finally, Section 6 draws the conclusions of our research.

2 Background, Motivation, and Related Work

In this section, we present the background, the motivation of our work, and related work on devices reliability and its assessment methodologies.

2.1 Radiation Effects in Electronic Devices

The interaction of a galactic cosmic ray with the terrestrial atmosphere triggers a flux of particles (mainly neutrons). About 13 $neutrons/((cm^2) \times h)$ reach the earth's surface [14]. A neutron strike may perturb a transistor's state, generating bit-flips in storage elements, or producing current spikes in logic circuits that, if latched, lead to an error [15]. A transient error may have no effect on the program output (i.e., the fault is masked or benign) or may be propagated through the stack of system layers and produce a Silent Data Corruption (SDC - undetected wrong output), or Detected Unrecoverable Errors (DUEs), such as a program crash (application hang) or a device turning not responsive (system crash). The error rate of a software code executed

TABLE 1
Reliability evaluation methodologies characteristics.

Evaluation Method	Time Needed	Cost	Accessible Resources	Fault Source	Availability	Observability
Field, Lifetime data	months/years	very high	all	natural	final product	limited
Beam testing	**hours**	**high**	**all**	**natural**	**final product**	**limited**
Software fault injection	hours	low	limited	synthetic	early/final product	medium
Arch. fault injection	days	low	limited	synthetic	early	medium
Microarch. fault injection	**days/weeks**	**low**	**most**	**synthetic**	**early**	**very high**
RTL fault injection	years	low	all	synthetic	late	very high

on a microprocessor depends on both the raw sensitivity of the memory and logic elements [16] and the probabilities for the fault to be propagated through the hardware design (the microarchitecture) and the program [6], [17], [18], [19].

2.2 Reliability Evaluation Methodologies

The most common ways to evaluate the reliability of computing devices, as listed in Table 1, are: lifetime or field test, accelerated beam experiments, and statistical fault injection at different levels of abstractions (software, architecture, microarchitecture, or Register-Transfer Level - RTL). We include the time and cost required to complete the study, how many of the available resources can be accessed (or are modeled), if the faults are induced by processes that are natural (i.e., realistic error rates) or synthetic (i.e., models chosen by the user), if the study can be performed in the early stages of the project or only on the final product, and how much information can be gathered on faults generation and propagation (observability). On one hand, high observability is essential to identify the most critical resources of a device. On the other hand, to guarantee that the analysis is accurate and valid when implemented in the field, the evaluation must be proved to be realistic.

Field failure studies have to be based on statistically significant amounts of data, and thus require a huge number of devices and, obviously, are very time-consuming (because the natural error rate is very low) [20], [21], [22]. Accelerated particles beams reduce the cost and time of field tests taking advantage of a high particles flux intensity [16], [23], [24].

However, beam experiments have two main limitations: (1) the effects of faults can be observed only when they have already compromised the system functionality, making it very challenging to identify the most vulnerable hardware structures of the system (observability is low). (2) Experiments can be performed at the end of the manufacturing process, when a silicon prototype or the final chip has been produced. Any modification to the design, including the ones necessary to improve the device reliability to soft errors, would be, therefore, extremely expensive.

Fault injection is a complementary approach to beam experiments. Instead of exposing the manufactured chip to radiation, fault injection is based on models of the system and artificially injects faults through simulation (at different levels of abstraction): from RTL to architecture, microarchitecture and software. The probability for a fault to propagate to the output of an application is measured by injecting faults in the accessible resources of each level's model (gates, registers, hardware arrays, variables, instructions, etc.). Fault-injection provides complete observability of the abstraction layer details but has two main limitations: (1) the fault model and fault injection probabilities are synthetic (i.e., defined/modeled by the user and/or the simulator), thus the obtained results may not correspond to the physical phenomena, and (2) faults can be injected only in that subset of available resources that are accessible at each abstraction layer. Software fault injection can be performed on the final device (with the same limitations of beam experiments) but also on instruction set (functional) simulators. Microarchitecture and RTL descriptions, on the contrary, are available in the early stages (RTL much later than the microarchitecture level, but still pre-silicon), where modifications to the project are still possible (fast and cheap at the microarchitecture level; extremely time consuming and expensive at the RTL).

2.3 Motivation and Contribution

Designs that need to comply with certain dependability constraints require decisions to improve the reliability of the system but without adding unnecessary overhead. It is critical to make these decisions as early as possible since any additional re-design iteration can lead to unacceptably high costs. Experiments on the silicon chip (lifetime failure statistics, beam tests, software fault injection) are less than ideal, as they can be performed only on the final product. They can report the reliability of the product but at a stage when improvements are too costly or not impossible. Early-reliability assessments (architecture level, microarchitecture level, and RTL) are often performed in models that exist prior to silicon prototypes. These vary in the level of detail, with the most abstract and less detailed (architecture level) being available earlier in the design flow while the most detailed (RTL) being available at the very late stages before design sign-off. Performance (microarchitectural) models are much more representative of the hardware details (although certain RTL details are missing) but are also very fast to execute. Early decisions for protection against soft errors that are based on fault injection can be useful only with strong evidence that fault injection evaluation is sufficiently close to the actual silicon chip. Without such evidence, which we intend to provide, any mitigation solution risks being ineffective.

While architecture level description is behavioral and not necessarily close to the actual implementation [25], [26], RTL is highly detailed but a fault simulation in this level of abstraction is extremely slow (typically 3 orders of magnitude slower than the microarchitecture level). Thus, we focus on microarchitecture-level as it includes an accurate description of the most important hardware structures of the microprocessor (registers, caches, buffers, queues, etc.).

By showing that microarchitecture level fault injection on Arm CPUs can predict the final product error rate, we move a big step forward in the validation of fault injection evaluation and in the quest of early FIT rates estimation. Additionally, by providing both beam and microarchitectural fault injection results we ensure our analysis to be both accurate and with high observability.

2.4 Related Work

Arm processors have been exposed to accelerated particles beams and have been subject to fault injection in previous studies [27], [28], [29], [30]. [25] and [26] present results on architecture-level and [31] on microarchitecture-level fault injection on Cortex-A9 core. [32] presents a comparative reliability evaluation between microarchitecture and RTL fault injection, for bare-metal workloads running on Cortex-A9, while [33], [34] also includes results of RTL fault injection on Arm CPU cores.

Some work has been done to evaluate the influence of an OS on the reliability of code executions [12], [13], [27], [35], showing that the OS can be beneficial in the presence of cache conflicts.

However, these previous works do not correlate the results of beam testing to fault injection, as we do in this paper to cross validate the two approaches.

Some preliminary studies have proposed a comparison or combination of different reliability evaluation techniques [9], [24], [30], [36], [37], [38]. In [8], a first attempt to compare only the reliability evaluation of Cortex-A9 over Linux using beam experiment and microarchitectural fault injection was made. According to the authors, for SDCs the comparison can be very close, for DUEs the difference is significant. In this paper, we extend the analysis proposed in [8] by considering both an in-order stand alone Cortex-A5 and an out-of-order Cortex-A9 integrated in a SoC. For both devices, we run the codes bare metal and on top of Linux. We can, then, address the impact of both SoC integration and OS on the reliability of a device. This is the first paper that uses both beam experiments and microarchitectural fault injection to better understand how the OS and the SoC integration affect the reliability of a processor.

3 Methodology

In this section we describe the Arm CPUs we analyze in this paper, the OS and codes we use, and the two experimental setups we employ. We perform our evaluation and beam versus microarchitecture fault injection comparison first on a stand-alone CPU (Cortex A5) running the codes bare to the metal. Then, we consider the Cortex A9 CPU, which, in its silicon implementation, is integrated into an SoC. Finally, we study the impact of the OS in both the A5 and the A9.

3.1 Benchmarks, Devices

Our study is performed on an Arm®Cortex™-A5 implemented in a 65nm CMOS technology in the Microchip SAMA5D2 XPLAINED ULTRA board and on the Arm®Cortex™-A9 that is embedded, together with other cores, in a Xilinx Zynq™-7000 SoC implemented in a 28nm CMOS technology. Being low-cost embedded devices, neither the A5 nor the A9 feature any hardware protection technique such as ECC. The ECC has been shown to reduce by about 1 order of magnitude the SDC rate of modern computing devices [9]. The two Arm CPUs have significantly different microarchitectures. The A5 is a simple in-order CPU while the A9 is a more complex out-of-order superscalar CPU with speculative execution. The silicon chips also differ: the Microchip features a stand-alone A5 while the Xilinx features an A9 integrated into an SoC. The SAMA5D2 device has a single A5 core operating at a maximum frequency of 500MHz and the Zynq SoC has two A9 cores operating at 667 MHz and an FPGA (not used in our tests). Each core has a 32 KB 4-way set-associative instruction and data caches and a unified 8-way set-associative Level 2 cache which is 128kB in the A5 and 4x larger, i.e., 512kB, in the A9. We have diligently configured the gem5 model to resemble the physically available A5 and A9 CPUs and in the Zynq SoC, we disabled the second A9 core to make the two evaluation setups as close as possible (details in Section 3.4). Table 2 presents the main characteristics of the two setups.

TABLE 2
Summary of setup attributes.

Property	Cortex-A5		Cortex-A9	
Setup	Beam	Gem5	Beam	Gem5
Platform	SAMA5D2	VExpress	Zynq 7000	VExpress
Technology	65 nm	N/A	28 nm	N/A
CPU cores	1	1	1*	1
L1 (4-way)	32 KB	32 KB	32 KB	32 KB
L2 (8-way)	128 KB	128 KB	512 KB	512 KB
Kernel	4.14	3.13	3.14	3.13

We have selected Arm chips and in particular, the A5 and A9 CPUs for our work because they are the only Arm CPU cores that are publicly available both as stand-alone and as SoC-integrated silicon devices and can both be modeled in gem5. We aim at providing a methodology and a beam *vs* fault injection comparison that can be applied to other Arm CPUs, as long as their gem5 model is available. The fact that the methodology holds for two significantly different devices provides a good indication that it can be extended to other CPU cores' ISAs. We have chosen codes with different computational characteristics from the *mibench* [39] benchmarks suite which has been extensively employed for reliability and other studies. We have selected the codes based on the recommendations for reliability evaluation provided in [40]. Using the selected codes helps in correlating the observed reliability behaviors with the computational characteristics. The benchmarks are:

- CRC32 **(CPU intensive)**, that calculates the corresponding 32-bit Cyclic Redundancy Check (CRC).
- FFT **(memory intensive)**, that performs the Fast Fourier Transform (FFT) on a wave on a floating-point array.
- MatMul **(memory intensive)**, that multiplies two square float matrices.
- Qsort **(memory and control flow intensive)**, that sorts an array using the quick-sort algorithm implemented in the GNU C standard library.

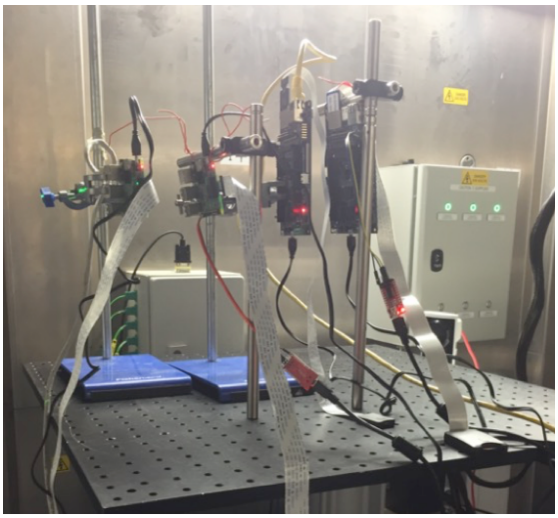


Fig. 2. Part of the beam test setup at ChipIR.

We use exactly the same input values and size for each benchmark in both the beam experiments and the fault injection campaigns (details in Section 3.5).

3.2 Bare-metal vs Linux Setup

The reliability of the two devices running the selected benchmarks was evaluated in both bare-metal (without an OS) and on the top of the Linux OS. The Linux kernel version that was used on the beam setup is 3.14 for the A9 and 4.14 for the A5 while in gem5 we used the 3.13 version. These were the closest kernel versions that have been ported on the two platforms.

All executables were generated with the same compiler and were statically linked. This means that all the OS functions used during computation are exactly the same between the A5 and A9, between bare-metal and Linux execution.

The **Silent Data Corruption (SDC)** detection, identical for bare-metal and Linux setup, is performed comparing the output of the experiment (for either beam or fault injection) with the expected golden output (calculated in a fault-free execution). Any mismatch triggers SDC detection. **Detected Unrecoverable Errors (DUEs)** manifestation is different for bare-metal and Linux. In a bare-metal execution, a DUE simply hangs the application and, subsequently, the device. We call this event **Crash**. When the application runs on top of Linux, a fault can hang the application but not the OS (Linux is still responsive and a new instance of the application can be launched) or can hang the entire OS, requiring a hard reboot. We call the former an **AppCrash**, and the latter a **SysCrash**.

3.3 Neutron Beam Experiments

Our radiation experiments were performed at the ChipIR facility of the Rutherford Appleton Laboratory (RAL) in Didcot, UK. ChipIR delivers a neutron beam suitable to mimic the atmospheric neutron effects in electronic devices [41], allowing to measure the Failures In Time (FIT) rate of the device executing a code.

Figure 2 shows part of our setup at ChipIR. We irradiate two Xilinx Zedboards and three Microchip boards with a 3×3 cm beam spot, which is sufficient to irradiate the chip uniformly without affecting the main memory or other onboard peripherals (data in the DDR is not exposed to the beam).

The available neutron flux was about $3.5 \times 10^6 n/(cm^2/s)$, i.e. about 8 orders of magnitude higher than the terrestrial flux ($13 n/(cm^2 \times h)$ at sea level [14]). Since the terrestrial neutron flux is low, in a realistic application it is highly unlikely to observe more than a single corruption during the program execution. We have carefully designed the experiments to maintain this property (observed error rates were lower than 1 error per 1,000 executions). Experimental data, then, can be scaled to the natural radioactive environment without introducing artifacts. Each of the 16 configurations (4 codes per device, bare-metal and with Linux) were tested for at least 100 effective hours (i.e., not considering setup, initialization, and recovery from crash times). The 1,600 hours of testing, when scaled to the natural exposure, account for more than 18 million years.

3.4 Microarchitectural Fault Injection

The microarchitecture-level reliability assessment was performed on top of gem5 simulator [42], using the GeFIN fault injection framework [43]. GeFIN is employed in our work to quantify the Architectural Vulnerability Factor (AVF) [17] of each modeled hardware component of the system, which expresses the probability for a fault leading to a failure. Gem5 has been demonstrated to accurately resemble Arm Cortex microarchitectural configurations [44].

For both the A5 and A9 cores gem5 includes a detailed model of the CPU pipeline (in-order and out-of-order, respectively) along with cache memories, predictors arrays, and TLBs. Compared to the RTL, microarchitecture-level simulation has three orders of magnitude higher throughput, allowing simulation of long workloads, both in bare-metal and on top of an OS.

We have configured GeFIN to inject single-event transient faults during system simulation in the following components, which cover the vast majority (>90%) of SRAM cells inside the CPU core: L2 Cache, L1 Data and Instruction Caches (both the data arrays and the tags arrays of all caches), Physical Register File, and the Data and Instruction Translation Lookaside Buffers (DTLB, ITLB). To achieve a statistical sample of at most 4% error margin and 99% confidence level, we have injected at least 1,000 single bit transient faults on each of the target components [45]. In total, 176,000 fault injections have been performed.

3.5 Fault Injection and Beam Experiment Comparison

To avoid any difference not related to the reliability evaluation methodology that can bias our results, we used exactly the same source code, compiler, compiler options, and input vector (size and values) for both the GeFIN fault injections and the neutron beam experiments. Still, the setups used for beam experiments and fault injection are intrinsically different as we are comparing the execution on actual hardware vs. gem5 simulation. We have tuned as much as possible the two setups to avoid major differences in the execution of the same code; to validate this

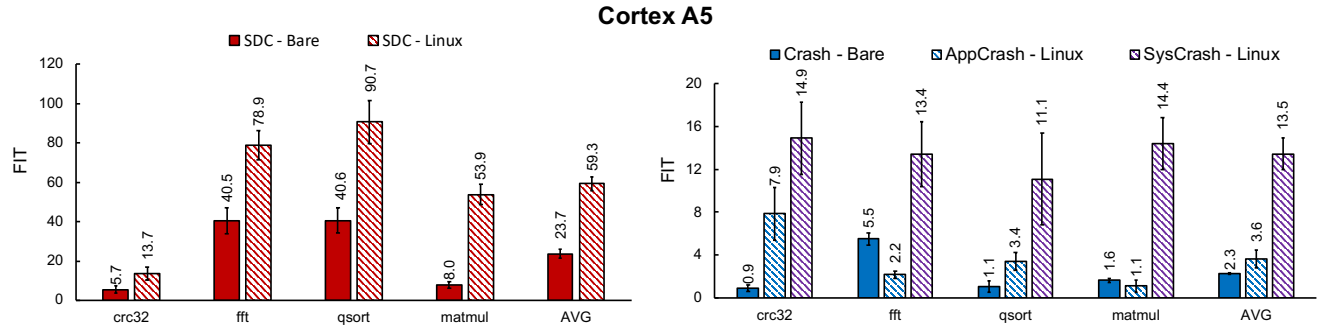


Fig. 3. Cortex A5 Bare-metal and Linux beam FIT rates for SDCs (left) and Crashes, Application Crashes, and System Crashes (right).

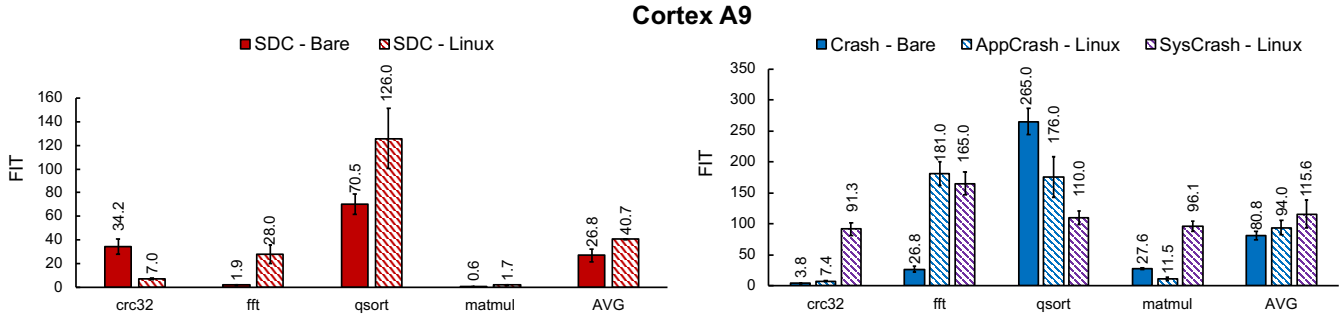


Fig. 4. Cortex A9 Bare-metal Linux beam FIT rates for SDCs (left), and Crashes, Application Crashes, and System Crashes (right).

synchronization effort we measured the values in 7 different hardware performance counters on both the chips and the simulator: CPU cycles, branch miss-predictions, L1 data cache accesses, L1 data cache misses, L1 data TLB misses, L1 instruction cache misses, and L1 Instruction TLB misses and has observed only small differences. Still, as literature has already identified, certain design differences exist in the implementations of gem5 and Arm Cortex microarchitectures [44]. Our analysis contributes to understanding if microarchitectural simulations can provide accurate insights on hardware reliability.

To estimate the FIT rate of a code using fault injection and compare it with the FIT rate measured with beam experiments, it is necessary to know the raw (intrinsic) fault rate per bit along with the probability of each fault becoming a failure. In principle, multiplying the per bit raw failure rate of each microarchitectural component to its AVF and its size (in bits) provides the failure rate (FIT) of the code executed on the device:

$$FIT = \sum_{component(i)} AVF(i) \times size(i) \times FIT_{bit}(i)$$

However, while the fab can provide the components FIT rate, measuring the FIT rate for each resource on the actual physical chips would require too much time and, when dealing with an off-the-shelf component, could be unfeasible due to visibility limitations. We decided to use the experimentally measured L1 cache FIT/bit rate, as a reference fault rate for the technology in which the microprocessor is implemented:

$$FIT \simeq FIT_{bit}(tech.) \times \sum_{component(i)} AVF(i) \times size(i)$$

This simplification is justified by the fact that caches are normally the most vulnerable resource in a microprocessor and the targeted components in gem5 are all implemented, in silicon, with the same technology as the L1 cache.

4 Fault Injection and Beam Testing Data

In this section, we first compare and combine the experimentally measured FIT rates for the A5 CPU and on the A9-based SoC. Then, we evaluate how close the FIT prediction based on microarchitectural fault injection is to the experimentally measured FIT rates on the actual chips. The main insights are highlighted (using the italic type) in the text.

4.1 Beam Experiments

Technology Sensitivity: Since the Cortex-A5 and Cortex-A9 are implemented in two different technologies, the raw probability for a neutron to generate a fault is different in the two devices. To measure the technology sensitivity we load a known pattern in the CPU, filling the L1 data cache, expose the device to the beam for seconds, and read back the memory content, counting the eventual bit flips. We tuned the exposure time with the beam intensity, keeping the probability for two different neutrons to generate faults between two reads negligible.

The experimentally measured FIT rates for a single memory bit in the L1 cache is 2.37×10^{-4} for the Cortex-A5 and 2.59×10^{-5} for the Cortex-A9. These values are in line with similar technologies, as shown with life tests in [20]. A neutron is, then, about 9.1x more likely to generate a

fault in the 65nm Cortex-A5 than in the 28nm Cortex-A9. The difference between the FIT rates should not surprise as transistor dimensions, layout, manufacturing process, voltage, etc. impact significantly the device sensitivity [16], [46]. It is also not surprising that the *per-bit* FIT rate is higher for the 65nm CMOS than for the 28nm CMOS device. In fact, while a smaller transistor is likely to have a smaller critical charge (i.e., particles with lower energy can generate a fault), it also has a smaller area (i.e., it is harder for a particle to hit the transistor's sensitive area). This has been previously documented in [47]. Additionally, as the two devices have been manufactured in two different facilities with possibly completely different processes the critical charge might indeed be very different. We also observed that more than 95% of observed errors are single bit-flips, which is expected for 65nm and 28nm planar CMOS technologies [16], [46].

It is worth noting that the FIT rate of a device executing an application does not depend only on the technology sensitivity but also on the amount of resources involved in the computation (the A9 is bigger than the A5) and on the probability for a fault to propagate through computation (i.e., the AVF). We quantify and qualify both aspects next.

A5 and A9 FIT rates: Figures 3 and 4 show on the left the SDCs and on the right the DUEs (Crash for bare-metal, AppCrash, and SysCrash for Linux, as detailed in Section 3.2) FIT rates measured with beam experiments for the A5 and A9, respectively. In each Figure, we show the results obtained when executing the codes bare-metal and on top of Linux. Data is reported with 95% confidence intervals.

As a first quantitative comparison, we can observe that *the average SDC rates are very similar for the two devices* (the average SDC rate difference between the A5 and A9 is 12% for bare metal and 45% for Linux). Interestingly, *the average DUE rate (Crash, AppCrash, SysCrash) is at least one order of magnitude higher on the A9* (Figures 3 and 4 are on different scales). Correlating the technology sensitivity data (the A5 has a 9.1x higher sensitivity than the A9) with Figures 3 and 4 (A5 and A9 have similar SDC rate and the A9 has a 10x higher DUE rate) suggests that the A9 architecture is more vulnerable than the A5, mostly for DUEs. Additionally, as the A9 is bigger (out-of-order and with a four times larger L2 cache) than the A5, it is likely to have a higher probability of being hit by a neutron.

The microarchitectural analysis proposed in Section 4.2 will investigate the A5 and A9 vulnerabilities in finer, per-component, granularity. We anticipate that the higher DUE rates of the A9 are due to the SoC high integration.

From the beam experiment data we can observe that the FIT rate is significantly dependent on the device, program executed, and system stack configuration (Linux vs. bare-metal).

The SDC FIT variation among the different codes is of about 1 order of magnitude for both the A5 and the A9. Qsort is the code with the highest FIT rate in both devices, because of the way it accesses the input data stored in caches and memory: as the core is waiting for new data to be fetched, the elements in the caches are exposed and contribute to the SDCs rate.

Impact of OS: Comparing beam results shown in Figures 3 and 4 obtained running the codes bare-metal (filled bars) and on top of Linux (pattern-filled bars), we can have a first evaluation of how the OS impacts the systems reliability. It is worth noting that we can only identify faults while the OS is executed that impact the application output (SDC) or the system execution (DUE). Possible faults during the OS code execution that do not influence the correct system functionality are not detectable. *The OS presence only slightly increases the code SDC FIT rate*, of about 2.4x for the A5 and 1.9x for the A9, on average. While, at a first glance, the OS impact seems high, it is much lower than the differences between SDC FIT rates of different codes in the same device (and of the impact on FIT rates of other factors, as shown in [16], [30]).

For DUEs, as discussed in Section 3.2, the bare-metal execution can only lead to a Crash (i.e. the application and, then, the device are not responding), while the execution on top of Linux can lead to an AppCrash (i.e., the application does not respond but Linux is still up and running) or to a SysCrash (i.e., Linux not responding). As events are stochastic and uncorrelated, the DUE FIT rate for the Linux execution is to be considered as the sum of AppCrash and SysCrash. When an OS is deployed the overall DUE rate increases by 8.5x for the A5 and 2.6x for the A9, on average.

Crashes in bare-metal are mainly caused by exceptions raised by the CPU when it is hit by a fault, such as undefined instruction, invalid opcode, prefetch abort, a fetch from an illegal address and data abort, data load from or store to an illegal address. Any exception raised in interfaces or communication protocols not used by the bare-metal code will be ignored and will not result in a Crash. The Linux AppCrash results from the kernel terminating the application, while in bare-metal the CPU itself triggers the Crash. The reason for the kernel terminating the application may be the result of a CPU exception (i.e., errors in the control flow, bad memory accesses, etc. as for the bare-metal Crash) handled by the kernel, but also from a signal sent by the kernel to the application. From Figures 3 and 4, we can observe that *for both the A5 and A9 the average AppCrash is about 50% higher than the bare-metal crash*. The SysCrash, on the contrary, is triggered by the corruption of kernel code or data, interfaces, etc. An error in the application being executed can hardly affect the system since it operates in an unprivileged space and can only access kernel space via system call services. The reported experimental data shows that, while the SDC rate and the Crash/AppCrash rates vary significantly across benchmarks, *the SysCrash rate is almost constant for both the A5 and on the A9* (the variation between codes is, on average, 30%). This observation confirms previous studies that demonstrate that the System Crash has a stronger component that depends on the hardware and is almost independent of the executed code [30].

4.2 Fault Injection Results

By employing the GeFIN microarchitectural fault injector, we can analyze the trends observed with beam experiments with a fine granularity (utilizing the full observability on the CPU model that gem5 provides) and further investigate the impact of SoC integration and of Linux.

AVF analysis of A5 and A9: The main outcome of the GeFIN fault injection is the components' AVF, i.e. the

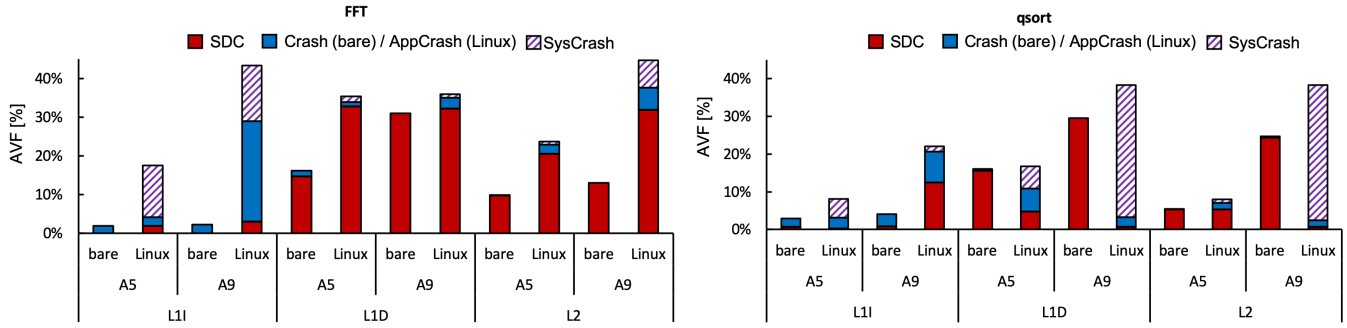


Fig. 5. AVF for FFT (on the left) and qsort (on the right) executed bare-metal or on top of Linux on the A5 and A9.

probability for a transient fault inside a component to propagate through the microarchitecture and the software stack and affect the program execution. AVF is the technology-independent part of the FIT rate as a combination of the hardware design and the software execution on top of it. It is worth noting that, while the silicon A9 is embedded in an SoC, the microarchitectural model of the A9 just considers the stand-alone CPU. This is typical, as microarchitectural reliability studies are performed on the stand-alone CPUs. In Section 5 we will evaluate if considering just the stand-alone A9 still provides accurate reliability estimations.

While the main scope of our paper is to understand at which level fault injection can be used to predict the CPU FIT rates, to highlight the insights GeFIN can provide we discuss in detail the AVFs of some CPU components for FFT and qsort (the findings of the same analysis for the other benchmarks are very similar). Figure 5 shows the AVF of L1 instruction cache (L1I), L1 data cache (L1D), and L2 unified cache for both A5 and A9 microprocessor chips. These hardware structures contribute the most to the overall FIT rate of the CPU (because these structures are the largest components of the CPU; see Section 3.5) when FFT (left) and qsort (right) are executed bare-metal and on top of Linux.

The AVF results in Figure 5 are shown in stacked bars broken down in the AVF results for the different fault effects: SDCs, Crashes/AppCrashes, and SysCrashes.

From Figure 5 we can derive that *the AVF of the three major cache blocks are significantly larger for the A9 core than the A5 core*. In both the bare-metal and the Linux configuration the L1I cache AVF is from 4x to 10x larger in the A9 than the A5 (and the majority of effects are Crashes as expected in an instruction-storing structure). In the L1D cache, the trend is the same: A9 has higher AVF than the A5 for both benchmarks, but the differences are smaller (up to 2x). The L1D cache fault effects are mainly SDCs, again a very natural result in a data-storing structure. Note that the L1 caches in both CPUs have the same size.

The L2 caches (which store both instructions and data) have a mixed fault effects behavior (SDCs and Crashes) but the A9 AVF results are, again, significantly larger (up to about 4x) than the A5. An important aspect is that in the A9 the L2 cache is 4x larger than the A5 (512K vs. 128K). We recall that higher AVF means that the probability for a fault to propagate to the output of the program is higher, and a larger area also increases the probability for the fault to occur.

While not shown in Figure 5, we have seen that faults in the data/instruction translation lookaside buffers (TLBs) are highly likely to lead to a DUE, possibly because they cause illegal accesses to memory or accesses to unmapped memory locations (the Crash AVF is 50% and the Hang AVF is 10%, on average). Only in a few cases, a fault in the data TLB leads to SDCs (SDC AVF is lower than 1%). This happens when the corrupted memory address is still valid and the program considers its data for computation. Finally, an interesting result we obtained is that the out-of-order A9 microprocessor chip has an average AVF for the Physical Register File lower than 3% for SDCs (much lower than the $\sim$30% of L1D or $\sim$15% of L2) and lower than 9% for DUEs.

In summary, the significantly larger AVF results of A9 for the three major structures along with the largest L2 cache size, explain the fact that A5 and A9 total FIT rates observed in the beam experiments are very close to each other in almost all cases despite the much higher technology sensitivity of the A5. The combined effect of the high A9 AVF results and the A9 L2 cache size compensates for the lower raw $FIT_{bit}(tech.)$ value of the A5.

AVF analysis of OS Impact:

Figure 6 shows the distribution of kernel and user mode code execution during the execution of each program in gem5. The OS kernel code execution can be as small as less than 3% (for FFT) and as large as about 67% (for matmul). Kernel code is invoked whenever a program requires to perform I/O operations, communicate with peripherals, or, periodically, serve system tasks. Depending on how much time is spent on kernel mode, a fault has a higher probability to cause a DUE, in all hardware components.

Qsort and FFT show a remarkable difference in the Linux kernel vs user code execution time: the time dedicated to

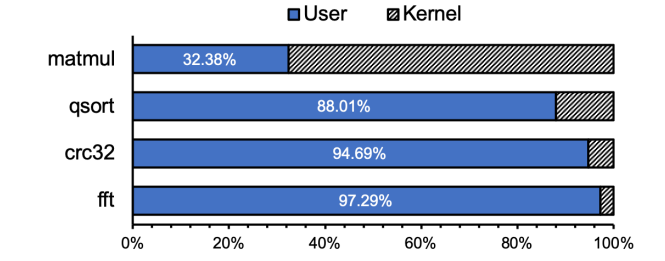


Fig. 6. Distribution of kernel and user mode relative to execution time.

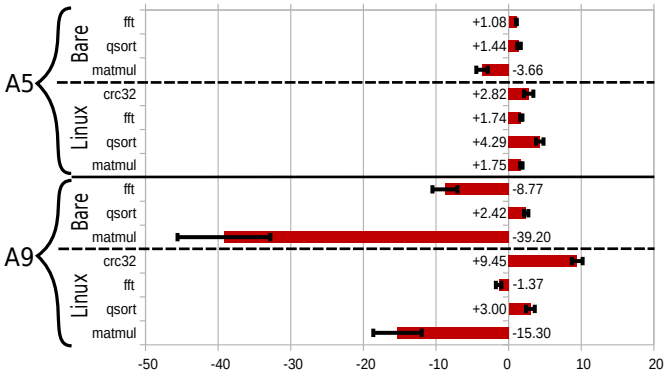


Fig. 7. Beam and fault injection SDC FIT rates comparison

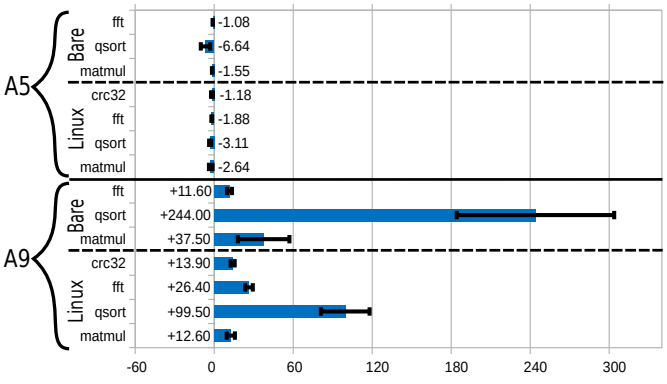


Fig. 8. Beam and fault injection Crash/AppCrash FIT rates comparison

kernel execution in qsort is 12% and in FFT is less than 3% of the total execution time. A longer Linux kernel time is likely to increase the probability for a fault to affect the operating system execution, leading to a crash. This is directly reflected in the AVF breakdown shown in Figure 5. For FFT, which has very short kernel execution cycles, the presence of the OS adds extra DUE AVF (that can be attributed to either the user of the kernel code) in both the A5 and A9 and all three components (L1I, L1D, L2) while the SDC AVF remain the same or are slightly increased. In the qsort benchmark, which has significantly larger kernel operations, we observe significant changes in the SDC/DUE AVF distribution. Particularly interesting is the change in the L2 cache AVF stack for qsort: the vast majority of SDC events observed in the bare-metal setup are manifesting in the Linux setup as DUEs. This is very likely because the fault effect becomes more severe in the Linux setup and the operating system does not allow the code to terminate the execution with an SDC but rather the execution (Application or System) crashes earlier.

5 Fault Injection and Beam Comparison

As discussed in Section 3.5, we can predict the codes' FIT rates with microarchitectural fault-injection multiplying the AVF by the number of available bits and the technology sensitivity (2.37×10^{-4} FIT/bit for the A5 and 2.59×10^{-5} FIT/bit for the A9). By comparing the predicted FIT rate with the beam FIT rate (which we assume is the ground truth) we can evaluate the accuracy of microarchitectural

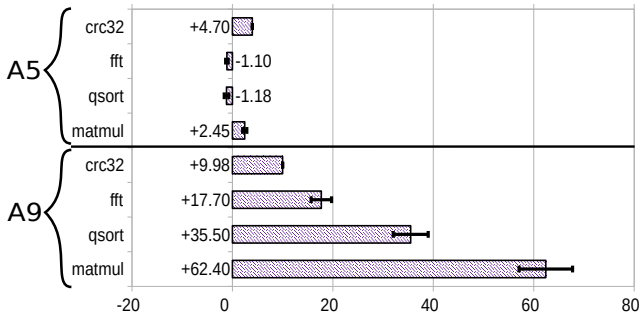


Fig. 9. Beam and fault injection SysCrash FIT rates comparison

fault injection in estimating the FIT rate of codes executed in silicon devices. We recall that the gem5 model includes only the CPU and not the entire SoC. Thus, the comparison between GeFIN and beam data is particularly interesting for the A9, as it shows if microarchitectural fault injection is accurate in predicting the FIT rate of a CPU even when embedded in an SoC.

Figure 7, 8, and 9 show the comparison between FIT rates measured with beam experiments and predicted with GeFIN for SDC, Crashes/AppCrashes, and SysCrashes, respectively. To better visualize the comparison, for each code we divide the highest FIT rate between the one calculated with beam and the one predicted using fault injection by the lowest FIT rate between the two. Whenever the FIT rate obtained with beam experiments is higher than the fault injection the value is represented as positive; negative otherwise. That is, in Figure 7, for matmul executed in bare-metal on the A5, the SDC FIT rate predicted with fault injection is 3.66x higher than the one measured with beam experiments. For the AppCrash FIT rates of qsort executed with Linux on the A9, shown in Figure 8, beam experiments provide a FIT rate that is about 2 orders of magnitude higher than GeFIN prediction.

5.1 Accuracy of SDC prediction

A result of great impact, shown in Figure 7, is that , despite the different setup and intrinsic differences between a Silicon device and a microarchitectural model, for the majority of cases (10 out of 14), the GeFIN SDC FIT rate prediction is very similar to the one measured with beam experiments. Differences for all codes, devices (A5 and A9), and configurations (with or without Linux) but matmul and FFT on the A9, are well smaller than 5x. FFT and, mostly, matmul on the A9, have a higher difference, probably because of the lower statistical significance of the data gathered with beam experiments. This result is of extreme importance as it attests that *microarchitectural fault injection can be used to predict the SDC FIT rate* of the correspondent real hardware of two completely different devices, with and without Linux. As shown in Figures 3 and 4 the OS increases the SDC rate of 2.4x and 1.9x for the A5 and A9, respectively. The comparison in Figure 7 attests that *early pre-silicon measurements on GeFIN are highly accurate in catching the OS impact on the application SDC FIT rate* and the AVF analysis of OS impact in Section 4.2 is directly applicable to explain the OS impact in the SDC error rate even in the silicon device. Additionally, even if not the entire SoC is modeled in GeFIN

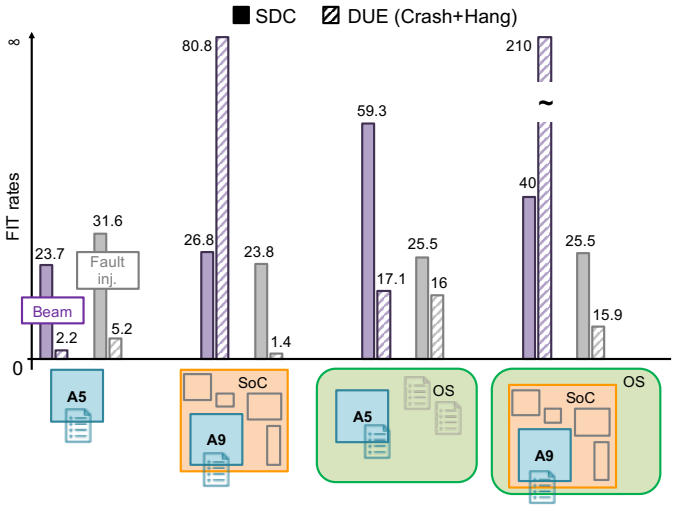


Fig. 10. Abstract view of the difference between beam and microarchitecture level fault injection SDC and DUE (Crash+Hangs) FIT rates. For most configuration, fault injection and beam provide sufficiently close FIT rates.

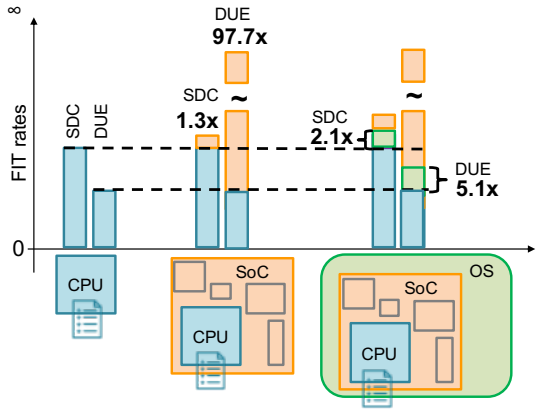


Fig. 11. Overall impact of SoC integration and OS on the CPU error rate as resulting from our data (vertical axis is in log scale). On the average, the cores integration barely increases the SDC rates (about 1.3x) but significantly increases the DUE rate (97.7x). The OS has a smaller impact on SDCs (about 2.1x) and increases DUE by about 5.1x.

(the A9 model in gem5 does not include the cores available in the Zynq board), the predicted SDC FIT rate for the A9 is still sufficiently close to the ground truth. This intrinsically means that *SoC integration does not significantly impact codes' SDC rates which can be safely attributed to the CPU only*.

5.2 Accuracy of DUE prediction

When estimating Crashes and AppCrashes, as can be seen from Figure 8, for the A5, GeFIN reports FIT rates very similar to the ones measured with beam experiments, for both bare-metal and Linux configurations. As shown in Figure 9, even for SysCrashes the FIT rates for the A5 are very similar between fault injection and beam experiments. As observed for SDC, then, *for stand-alone CPUs GeFIN accurately models DUEs (Crashes, AppCrashes, and SysCrashes), considering also the impact of the OS*.

For the A9 the differences diverge, with the beam providing a Crash and AppCrash FIT rate from 1 to 2 orders of magnitude higher than GeFIN, for both Linux and bare-metal. It is interesting to notice that, for DUEs on the A9, which are the only values for which GeFIN could not provide a good prediction for the FIT rate, beam experiment data is always (significantly) higher than GeFIN. We can derive that *a great portion of DUEs in the embedded A9 are generated by faults in resources that are not modeled in gem5*. For the single-core, stand-alone A5, the model used on gem5 and the real hardware are very similar. This is not the case with the board with the A9 which is integrated with an FPGA (that is not utilized) and it is a dual-core (with one core disabled). The integration in an SoC requires additional interfaces, logical or control components that are not modeled in gem5. In practice, this translates to a highly complex interconnect inside the chip whose corruption is likely to lead to a DUE in the physical chip.

5.3 Discussion

A major contribution of our study is that we demonstrate that, under several different system setups, microarchitectural fault injection provides an accurate estimation of the SDC FIT rate (that is mainly attributed to the CPU cores) *before* implementing the device in silicon. It is worth noting that while we have experimentally measured the L1 FIT/bit, it is not necessary to implement the device in silicon to know its technology sensitivity as it can be predicted with simulations knowing the technology node, layout, etc. The fab where the device will be implemented normally provides this value, eventually allowing the customer to choose between different technology qualities (more or less susceptible to faults and more or less expensive).

Figure 10 shows the average FIT rates for SDCs and DUEs (intended as the sum of Crashes and Hangs) measured with beam experiments and predicted with GeFIN on the A5 and A9, executing the codes bare-metal or on the top of Linux. As mentioned in the introduction, both beam and fault injection FIT rates have different reasons not to provide a perfectly accurate FIT rate. However, when beam and fault injection FIT rates are similar we can be reasonably sure that we can give a bound to the *true* device FIT rate. The abstract view of Figure 10 shows that for most configurations, GeFIN and beam SDC FIT rates are similar. Only for DUEs in the SoC, the difference diverges, as discussed in Section 5.2.

A large portion of DUEs, then, is generated by faults *outside* the CPU core. Microarchitectural fault injection can, in this case, only provide an optimistic lower bound for the DUE FIT rate; the actual DUE FIT rate can be expected to be significantly larger. Whenever the FIT rate estimate is too high for a design project's requirements, it is possible to include additional hardware reliability solutions (such as ECC) taking advantage of Arm flexibility in modifying the microarchitecture. As we have shown, GeFIN provides also details about the possible sources of failures (hardware components with higher AVF/FIT rates). The microarchitectural hardening can then be applied only to the resources that are found to be responsible for the majority of failures (for example the caches), avoiding the introduction of unnecessary overhead. According to our results, these CPU architectural hardening solutions may not be effective in reducing the DUE rate if the CPU is integrated into an SoC,

as most DUE are caused by faults in external resources. A re-design of the cores interfaces, including some specific software procedures to handle synchronizations or hangs problems, could be necessary to significantly reduce DUEs.

The combination of fault injection along with beam results quantifies how either the SoC integration or the OS deployment influences the overall FIT rate. Figure 11 shows a high-level summary of the results that were observed with beam experiment and validated using GeFIN (vertical axis is assumed in log scale). It shows the average factors of change in the SDC and DUE FIT rates introduced by the integration of the CPU core in an SoC (average difference between bare-metal FIT rates obtained with beam and fault injection on the A9 embedded in the Zynq board) and the inclusion of the OS in the software stack (average difference between A5 and A9, bare-metal vs Linux). We clearly show that, in both devices, the SDC FIT rate is minimally affected by the SoC integration or the presence of the OS while the DUE FIT rates are affected to a larger extend by both parameters (severely in the case of the SoC integration).

SoC integration, which was present only in the silicon A9 chip exposed to the beam, has a strong impact on the DUE FIT rate (between one and two orders of magnitude). As the A9 microarchitecture was quantified to be 5x more vulnerable using fault injection (Section 4.2) and the different fabrication technology makes the fault rate to be 9.1x higher on A5 (Section 4.1), we should expect a difference of roughly 2-3x between A5 and A9 FIT rates. Higher differences are to be attributed to factors other than the technology or the architecture, i.e., SoC integration. Moreover, the average difference between beam (that includes integration) and GeFIN (that models only the stand-alone CPU) shows an increase of just 1.3x for SDCs but of 97.7x for DUEs. These differences corroborate with the statement that the *SoC integration has a massive impact on the failure rate of the system but not on the codes SDC rate*.

The OS also affects the DUE FIT rate. We have observed a difference of 5.1x (Linux being higher) on the averages DUE FIT rates between A5/A9 bare-metal and A5/A9 Linux on beam experiments, while the difference is limited to 3.5x on fault injection. The DUEs increase is due to the fact that, besides the benchmarks, several other system processes responsible for the OS management are running and susceptible to the radiation.

SDCs FIT rates are only slightly affected by the OS, since the output is only being processed under the actual workload and any OS interference does not participate nor relate to the generated output unless it is serviced by a system call. This is consistently observed in both beam experiments and fault injection campaigns.

6 Conclusions

In this work, we have performed an extensive reliability evaluation of two widely used Arm microprocessors, Cortex-A5 and Cortex-A9, with the primary objective of identifying the differences and similarities between the reported soft error rates from post-silicon neutron beam testing and pre-silicon microarchitecture level fault injection. We extensively analyze the impact that the SoC integration and the OS deployment have on the reported FIT rates from the two setups. The two CPU cores are fabricated using different technologies and different SoC organizations; A5 is an individual CPU core whereas the A9 is integrated into an SoC. Our analysis demystifies the contribution of the different aspects: the bare CPU alone, the SoC integration, and the OS deployment.

The major finding of our analysis is that the SDC FIT rate is practically unaffected by the SoC integration and the presence of the OS. On the other hand, DUEs (both application and system crashes) are significantly increased (up to 2 orders of magnitude) due to SoC integration, showcasing how this attribute can really influence the total system FIT rate. The OS influence on the overall setup is also evaluated and an average DUE FIT rate increase of 5.1x is reported.

Our analysis confirms on two different Arm CPU cores that the majority of the SDCs can be safely attributed to the CPU core itself while executing the user codes. On the other hand, the System and Application Crash parts of the overall system failure rates are significantly affected by the complexity of the SoC integration as well as the inclusion of the OS. The comparison between beam testing and microarchitecture fault injection we provide is a step forward in the quest of early error rate estimation. This is of extreme interest mainly for flexible architectures like Arm, that can be tuned by the customer, adding hardening solutions if necessary, before being implemented in silicon.

7 Acknowledgments

This research has been supported in part by the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 886202, the H2020 UniServer Project (Grant Agreement 688540), the FP7 CLERECO Project (Grant Agreement 611404), and from the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finance Code 001. Neutron beam time was provided by ChipIR (DOI: 10.5286/ISIS.E.RB2000161) thanks to C. Cazzaniga and C. Frost and by LANSCE thanks to Steve Wender and Gus Sinnis.

References

[1] R. Lucas, "Top ten exascale research challenges," in *DOE ASCAC Subcommittee Report*, 2014.

[2] A. Cohen, X. Shen, J. Torrellas, J. Tuck, Y. Zhou, S. Adve, I. Akturk, S. Bagchi, R. Balasubramonian, R. Barik, M. Beck, R. Bodik, A. Butt, L. Ceze, H. Chen, Y. Chen, T. Chilimbi, M. Christodorescu, J. Criswell, C. Ding, Y. Ding, S. Dwarkadas, E. Elmroth, P. Gibbons, X. Guo, R. Gupta, G. Heiser, H. Hoffman, J. Huang, H. Hunter, J. Kim, S. King, J. Larus, C. Liu, S. Lu, B. Lucia, S. Maleki, S. Mazumdar, I. Neamtiu, K. Pingali, P. Rech, M. Scott, Y. Solihin, D. Song, J. Szefer, D. Tsafrir, B. Urgaonkar, M. Wolf, Y. Xie, J. Zhao, L. Zhong, and Y. Zhu, "Inter-disciplinary research challenges in computer systems for the 2020s," tech. rep., National Science Foundation, USA, 2018.

[3] M. Snir, R. W. Wisniewski, J. A. Abraham, S. V. Adve, S. Bagchi, P. Balaji, J. Belak, P. Bose, F. Cappello, B. Carlson, *et al.*, "Addressing failures in exascale computing," *International Journal of High Performance Computing Applications*, pp. 1–45, 2014.

[4] Top500.org, "Japan Captures TOP500 Crown with Arm-Powered Supercomputer." https://www.top500.org/news/japan-captures-top500-crown-arm-powered-supercomputer/, 2020. [Online; accessed 6-July-2020].

[5] N. Hemsoth, "Arm it the nnsa's new secret weapon," 2018.

[6] G. Papadimitriou and D. Gizopoulos, "Demystifying the system vulnerability stack: Transient fault effects across the layers," in *48th International Symposium on Computer Architecture (ISCA 2021), June 14-19, 2021, Worldwide Online Event*, pp. 902–915, IEEE, 2021.

[7] G. Wang, S. Liu, and J. Sun, "A dynamic partial reconfigurable system with combined task allocation method to improve the reliability of fpga," *Microelectronics Reliability*, vol. 83, pp. 14 – 24, 2018.

[8] A. Chatzidimitriou, P. Bodmann, G. Papadimitriou, D. Gizopoulos, and P. Rech, "Demystifying soft error assessment strategies on arm cpus: Microarchitectural fault injection vs. neutron beam experiments," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 26–38, June 2019.

[9] F. F. d. Santos, P. F. Pimenta, C. Lunardi, L. Draghetti, L. Carro, D. Kaeli, and P. Rech, "Analyzing and increasing the reliability of convolutional neural networks on gpus," *IEEE Transactions on Reliability*, vol. 68, pp. 663–677, June 2019.

[10] D. Oliveira, L. Pilla, M. Hanzich, V. Fratin, F. Fernandes, C. Lunardi, J. Cela, P. Navaux, L. Carro, and P. Rech, "Radiation-Induced Error Criticality in Modern HPC Parallel Accelerators," in *Proceedings of 21st IEEE Symp. on High Performance Computer Architecture (HPCA)*, ACM, 2017.

[11] A. Chatzidimitriou, G. Papadimitriou, C. Gavanas, G. Katsoridas, and D. Gizopoulos, "Multi-bit upsets vulnerability analysis of modern microprocessors," in *2019 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 119–130, 2019.

[12] R. K. Iyer and D. J. Rossetti, "Effect of system workload on operating system reliability: A study on ibm 3081," *IEEE Transactions on Software Engineering*, vol. SE-11, pp. 1438–1448, Dec 1985.

[13] T. Santini, P. Rech, L. Carro, and F. R. Wagner, "Exploiting cache conflicts to reduce radiation sensitivity of operating systems on embedded systems," in *2015 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, pp. 49–58, Oct 2015.

[14] JEDEC, "Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices," Tech. Rep. JESD89A, JEDEC Standard, 2006.

[15] N. Mahatme, T. Jagannathan, L. Massengill, B. Bhuva, S.-J. Wen, and R. Wong, "Comparison of Combinational and Sequential Error Rates for a Deep Submicron Process," *Nuclear Science, IEEE Transactions on*, vol. 58, no. 6, pp. 2719–2725, 2011.

[16] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *Device and Materials Reliability, IEEE Transactions on*, vol. 5, pp. 305–316, Sept 2005.

[17] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, "A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor," in *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, (Washington, DC, USA), pp. 29–, IEEE Computer Society, 2003.

[18] V. Sridharan and D. R. Kaeli, "Using hardware vulnerability factors to enhance avf analysis," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ISCA '10, (New York, NY, USA), pp. 461–472, ACM, 2010.

[19] G. Papadimitriou and D. Gizopoulos, "Characterizing soft error vulnerability of cpus across compiler optimizations and microarchitectures," in *IEEE International Symposium on Workload Characterization (IISWC), November 7-9, 2021, Virtual Online Event*, pp. 113–124, IEEE, 2021.

[20] A. Lesea, S. Drimer, J. J. Fabula, C. Carmichael, and P. Alfke, "The rosetta experiment: atmospheric soft error rate testing in differing technology fpgas," *IEEE Transactions on Device and Materials Reliability*, vol. 5, pp. 317–328, Sept 2005.

[21] D. Tiwari, S. Gupta, J. Rogers, D. Maxwell, P. Rech, S. Vazhkudai, D. Oliveira, D. Londo, N. Debardeleben, P. Navaux, L. Carro, and A. B. Bland, "Understanding GPU Errors on Large-scale HPC Systems and the Implications for System Design and Operation," in *Proceedings of 21st IEEE Symp. on High Performance Computer Architecture (HPCA)*, ACM, 2015.

[22] V. Sridharan, N. DeBardeleben, S. Blanchard, K. B. Ferreira, J. Stearley, J. Shalf, and S. Gurumurthi, "Memory errors in modern systems: The good, the bad, and the ugly," in *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '15, (New York, NY, USA), p. 297–310, Association for Computing Machinery, 2015.

[23] N. Seifert, X. Zhu, and L. W. Massengill, "Impact of scaling on soft-error rates in commercial microprocessors," *Nuclear Science, IEEE Transactions on*, vol. 49, no. 6, pp. 3100–3106, 2002.

[24] D. Oliveira, L. Pilla, N. DeBardeleben, S. Blanchard, H. Quinn, I. Koren, P. Navaux, and P. Rech, "Experimental and analytical study of xeon phi reliability," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '17, (New York, NY, USA), pp. 28:1–28:12, ACM, 2017.

[25] G. S. Rodrigues and F. L. Kastensmidt, "Soft error analysis at sequential and parallel applications in ARM cortex-a9 dual-core," in *2016 17th Latin-American Test Symposium (LATS)*, IEEE, Apr 2016.

[26] F. Rosa, F. Kastensmidt, R. Reis, and L. Ost, "A fast and scalable fault injection framework to evaluate multi/many-core soft error reliability," in *2015 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, IEEE, Oct 2015.

[27] T. Santini, L. Carro, F. R. Wagner, and P. Rech, "Reliability analysis of operating systems and software stack for embedded systems," *IEEE Transactions on Nuclear Science*, vol. 63, pp. 2225–2232, Aug 2016.

[28] A. B. de Oliveira, G. S. Rodrigues, and F. L. Kastensmidt, "Analyzing lockstep dual-core arm cortex-a9 soft error mitigation in freertos applications," in *Proceedings of the 30th Symposium on Integrated Circuits and Systems Design: Chip on the Sands*, SBCCI '17, (New York, NY, USA), pp. 84–89, ACM, 2017.

[29] A. Martínez-Álvarez, F. Restrepo-Calle, S. Cuenca-Asensi, L. M. Reyneri, A. Lindoso, and L. Entrena, "A hardware-software approach for on-line soft error mitigation in interrupt-driven applications," *IEEE Trans. Dependable Sec. Comput.*, vol. 13, no. 4, pp. 502–508, 2016.

[30] V. Fratin, D. Oliveira, C. Lunardi, F. Santos, G. Rodrigues, and P. Rech, "Code-dependent and architecture-dependent reliability behaviors," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 13–26, June 2018.

[31] A. Chatzidimitriou, M. Kaliorakis, S. Tselonis, and D. Gizopoulos, "Performance-aware reliability assessment of heterogeneous chips," in *2017 IEEE 35th VLSI Test Symposium (VTS)*, IEEE, Apr 2017.

[32] A. Chatzidimitriou, M. Kaliorakis, D. Gizopoulos, M. Iacaruso, M. Pipponzi, R. Mariani, and S. D. Carlo, "RT level vs. microarchitecture-level reliability assessment: Case study on ARM(r) cortex(r)-a9 CPU," in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, IEEE, Jun 2017.

[33] X. Iturbe, B. Venu, and E. Ozer, "Soft error vulnerability assessment of the real-time safety-related ARM cortex-r5 CPU," in *2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, IEEE, Sep 2016.

[34] J. Blome, S. Mahlke, D. Bradley, and K. Flautner, "A microarchitectural analysis of soft error propagation in a production-level embedded microprocessor," in *In Proceedings of the First Workshop on Architecture Reliability*, 2005.

[35] T. Santini, C. Borchert, C. Dietrich, H. Schirmeier, M. Hoffmann, O. Spinczyk, D. Lohmann, F. R. Wagner, and P. Rech, "Effectiveness of software-based hardening for radiation-induced soft errors in real-time operating systems," in *Architecture of Computing Systems - ARCS 2017* (J. Knoop, W. Karl, M. Schulz, K. Inoue, and T. Pionteck, eds.), (Cham), pp. 3–15, Springer International Publishing, 2017.

[36] N. J. Wang, A. Mahesri, and S. J. Patel, "Examining ACE analysis reliability estimates using fault-injection," *ACM SIGARCH Computer Architecture News*, vol. 35, p. 460, Jun 2007.

[37] N. George, C. R. Elks, B. W. Johnson, and J. Lach, "Transient fault models and AVF estimation revisited," in *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*, IEEE, Jun 2010.

[38] H. Cho, S. Mirkhani, C.-Y. Cher, J. A. Abraham, and S. Mitra, "Quantitative evaluation of soft error injection techniques for robust system design," in *Proceedings of the 50th Annual Design Automation Conference on - DAC '13*, ACM Press, 2013.

[39] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization. WWC-4 (Cat. No.01EX538)*, pp. 3–14, Dec 2001.

[40] H. Quinn, W. H. Robinson, P. Rech, M. Aguirre, A. Barnard, M. Desogus, L. Entrena, M. Garcia-Valderas, S. M. Guertin,

D. Kaeli, F. L. Kastensmidt, B. T. Kiddie, A. Sanchez-Clemente, M. S. Reorda, L. Sterpone, and M. Wirthlin, "Using benchmarks for radiation testing of microprocessors and fpgas," *IEEE Transactions on Nuclear Science*, vol. 62, no. 6, pp. 2547–2554, 2015.

[41] C. Cazzaniga and C. D. Frost, "Progress of the scientific commissioning of a fast neutron beamline for chip irradiation," *Journal of Physics: Conference Series*, vol. 1021, p. 012037, may 2018.

[42] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, pp. 1–7, Aug 2011.

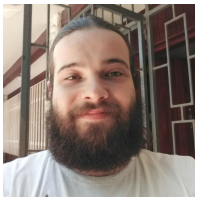
[43] A. Chatzidimitriou and D. Gizopoulos, "Anatomy of microarchitecture-level reliability assessment: Throughput and accuracy," in *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, IEEE, Apr 2016.

[44] A. Gutierrez, J. Pusdesris, R. G. Dreslinski, T. Mudge, C. Sudanthi, C. D. Emmons, M. Hayenga, and N. Paver, "Sources of error in full-system simulation," in *2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, IEEE, Mar 2014.

[45] R. Leveugle, A. Calvez, P. Maistri, and P. Vanhauwaert, "Statistical fault injection: Quantified error and confidence," in *2009 Design, Automation Test in Europe Conference Exhibition*, pp. 502–506, April 2009.

[46] J. F. Ziegler and H. Puchner, *SER–history, Trends and Challenges: A Guide for Designing with Memory ICs*. Cypress, 2010.

[47] N. Seifert, B. Gill, K. Foley, and P. Relangi, "Multi-cell upset probabilities of 45nm high-k + metal gate sram devices in terrestrial and space environments," in *2008 IEEE International Reliability Physics Symposium*, pp. 181–186, 2008.



Pablo R. Bodmann received his MSc degree from Universidade Federal do Rio Grande do Sul (UFRGS) in 2017 and received his master degree in 2019. Currently, he is a Ph.D. student at UFRGS working on fault tolerance on Arm architectures. His main research focus is reliability of Arm architectures when exposed to radiation.

George Papadimitriou is a Postdoctoral Researcher in the Department of Informatics & Telecommunications at National & Kapodistrian University of Athens in Greece. He received his PhD in Computer Science from the same University in 2019. His research focuses on dependability and energy-efficient computer architectures, microprocessor reliability, functional correctness of hardware designs and design validation of microprocessors, in which he has published more than 30 papers in international conferences and journals.

Rubens Luis Rech Junior is currently Computer Engineering student at Universidade Federal do Rio Grande do Sul (UFRGS), Brazil. He is currently working on radiation-induced effects in neural networks and safety-critical applications.

Dimitris Gizopoulos is Professor at the Department of Informatics & Telecommunications of the National & Kapodistrian University of Athens in Greece where he leads the Computer Architecture Laboratory. The group's research focuses on the dependability, the energy-efficiency and the performance of computer architectures. Gizopoulos has published more than 180 papers in top-tier conferences and journals, has served and is currently serving as Associate Editor for several IEEE and ACM Transactions and Magazines and as member of Program, Organizing and Steering Committees of IEEE and ACM conferences. He served as the General Chair of the 53rd and the 54th editions of the IEEE/ACM International Symposium on Microarchitecture (MICRO). Gizopoulos is an IEEE Fellow, a Golden Core member of the IEEE Computer Society and a Senior ACM member.

Paolo Rech received his master and Ph.D. degrees from Padova University, Padova, Italy, in 2006 and 2009, respectively. He is an associate professor at UFRGS in Brazil and a Marie Curie Fellow at Politecnico di Torino, Italy. His main research interests include the reliability of radiation-induced effects in large-scale HPC, autonomous vehicles for automotive applications and space exploration.