Estimating the Failures and Silent Errors Rates of CPUs Across ISAs and Microarchitectures

Dimitris Gizopoulos George Papadimitriou Odysseas Chatzopoulos
University of Athens
Department of Informatics and Telecommunications
{dgizop, georgepap, od.chatzopoulos}@di.uoa.gr

***Abstract*—Silent data corruptions (SDCs) pose a significant challenge to the reliable operation of modern microprocessors. As the need for enhanced performance and reliability continues to grow, it becomes essential to gain insight into the potential malfunctions and the occurrence of unnoticeable errors that microprocessors might encounter across different Instruction Set Architectures (ISAs) and microarchitectures. This study delves into assessing failures and rates of silent data corruptions within CPUs, shedding light on the variables that impact these rates and their consequences on system dependability. In this context, we present a comprehensive comparative investigation of SDC susceptibilities in CPU hardware structures, mainly targeting the L1 data cache, L1 instruction cache, physical register file, and a modern CPU's primary functional units (FUs). We carry out this investigation across three prominent CPU architectures: x86, Arm, and RISC-V. Our aim is to analyze both transient and permanent faults to evaluate the susceptibility of these architectures to SDCs.**

***Index Terms*—Reliability, soft errors, transient faults, permanent faults, silent data corruptions, failure rates, instruction set architectures, x86, Arm, RISC-V, CPU, microarchitecture-level fault injection**

I. Introduction

As microprocessors become increasingly dominant in critical computing systems, ensuring their reliability and fault tolerance becomes of utmost importance [1]. Silent data corruptions (SDCs) represent a specific class of errors that can lead to incorrect program execution without triggering any immediate error notifications [2]–[4]. These errors are particularly insidious as they can propagate through the computational pipeline, potentially leading to catastrophic consequences.

Modern microprocessors are composed of intricate structures, subsystems, and hardware elements that have the potential to impact both system functionality and performance [5]–[8]. Evaluating the reliability of these hardware components in isolation is a challenging task, influenced by numerous factors related to design, environment, technology, and workloads. The complexity increases further when considering the entire system stack and the interaction patterns between software layers and hardware elements [9]–[11]. Key design attributes, such as size and complexity, play a pivotal role in determining the reliability of hardware components. Additionally, the utilization of a hardware component and its involvement in executing operations are critical parameters affecting reliability estimation [12], [13]. Components that are frequently used, in particular, are more susceptible to wear-out effects, leading to device degradation. Moreover, the interplay of frequency, voltage, and temperature conditions is closely tied to a computer system's reliability [14]–[17]. Specifically, microprocessors operating in near-threshold voltage mode (a widely adopted technique for power consumption reduction) are more prone to experiencing timing errors, while overclocking and elevated temperatures can accelerate chip wear-out effects [18]–[25].

When evaluating the reliability threat chain, the concept of AVF (Architectural Vulnerability Factor) serves as a quantification of the overall susceptibility of a system. This encompasses the entire sequence from activation to propagation, ultimately affecting program output. AVF takes into consideration the combined impact and interplay of both hardware and software components. To illustrate, if a fault is present within the physical register file of an out-of-order (OoO) microprocessor, the hardware might utilize the fault speculatively, only to discard the result later due to a pipeline flush. In such instances, the fault does not result in an actual failure. However, even if the fault is activated and makes its way to the software level, the software algorithm may not use the corrupted data (discarding the fault), leading to the same end result. The SRAM arrays located within CPUs, including the L1 data cache, L1 instruction cache, and physical register file, as well as the functional units (FUs) are crucial components that are susceptible to SDCs. In this paper, our objective is to investigate and compare the vulnerability of these SRAM arrays and FUs to both transient and permanent fault types in three prominent CPU architectures: x86, Arm, and RISC-V. By identifying the factors contributing to SDC susceptibility, we can formulate effective strategies for mitigating these risks.

II. Background: Concepts & Definitions

Modern computing systems are expected to provide a high degree of reliability and accuracy in their operations. However, the increasing complexity of central processing units (CPUs) has brought to the forefront a range of challenges related to failures and silent data corruptions (SDCs). This section provides an overview of SDCs, introduces the architectural vulnerability factor (AVF), and outlines the concept of Failures in Time (FIT) rates, laying the foundation for understanding the need to estimate these metrics across different instruction set architectures (ISAs) and microarchitectures.

Silent Data Corruptions (SDCs): Silent data corruptions are elusive and potentially catastrophic errors that can occur within a CPU, leading to incorrect computational results. Unlike traditional hard faults that cause system crashes or immediate error detection, SDCs stealthily corrupt data without any apparent signs of a problem. These errors can go undetected by both hardware and software mechanisms, leading to a false sense of system integrity. SDCs can result from a wide range of sources, including electrical noise, manufacturing defects, and even radiation-induced soft errors. Their insidious nature makes them particularly challenging to identify and mitigate, posing a significant threat to the reliability of modern computing systems.

Architectural Vulnerability Factor (AVF): Early identification of weak hardware structures that are more vulnerable to faults can prevent system failures and data corruptions, guiding effective countermeasures. Calculating the Architectural Vulnerability Factor (AVF) for each microarchitectural component is the comprehensive way to assess the vulnerability of the entire system stack [7], [10], [26], [27]. Two prevailing methods to estimate the AVF are Architecturally Correct Execution (ACE) analysis [28] and statistical fault injection (SFI) [26]. While ACE analysis is fast, it can overestimate AVF and has implementation difficulties. SFI, although slower, provides accurate results [7], [10]. Still, both methods operate at the microarchitecture-level and calculate the cross-layer AVF [10] of the system.

Failures in Time (FIT) Rate: The Failures in Time (FIT) rate is a fundamental measure of a CPU's reliability. It quantifies the expected number of failures (which can include both hard and soft errors) that a component or system can experience per billion hours of operation. FIT rates provide an essential basis for estimating the overall reliability of a CPU. A lower FIT rate implies a more reliable component, while a higher FIT rate indicates a greater likelihood of failures occurring within a given timeframe. FIT rates are crucial for evaluating the robustness of CPUs in safety-critical applications

III. Experimental Results & Analysis

A. Fault Injection Methodology

The microarchitecture-level reliability assessment is performed on top of gem5 simulator [29]–[31] using the GeFIN fault injection framework [9], [32]. GeFIN was used to quantify the Architectural Vulnerability Factor (AVF) [28] of the system, which expresses the probability for a fault to lead to a failure. Combined with the raw fault rate, AVF can be used to estimate the FIT rate of a system. gem5 has been demonstrated to accurately resemble several microarchitectural configurations. Unlike RTL models, microarchitecture-level simulation can achieve a faster simulation throughput by two orders of magnitude, allowing simulation of realistic workloads, both in bare metal and with an operating system, as well as evaluation of multiple hardware components [7], [10], [13].

We have configured GeFIN to inject single-event transient and permanent faults during system simulation in the following components: L1 Data and Instruction Cache and the Physical Register File. To achieve a statistical sample of 4% error margin and 99% confidence level, we have injected 1,000 single bit transient faults and 1,000 single-bit permanent faults on each of the target components. According to this methodology, we present in the next sections the SDC AVF results. Note that we do not present SDC results for permanent fault on the Physical Register File, since our experiments show zero probability of SDCs in this component.

B. Transient Faults SDC Rates for 3 ISAs

Transient faults, also known as soft errors or single-event upsets (SEUs), are temporary and non-destructive errors in a CPU. They occur when external factors, such as cosmic rays, electrical noise, or voltage fluctuations, disrupt the normal operation of the CPU. These faults are often unpredictable and may result in wrong results after a computation, system crash, or data corruptions. Transient faults can be mitigated through error-correcting codes (ECC), e.g., parity or SECDED, which can detect and/or correct errors in memory arrays, or by designing fault-tolerant systems that can recover from these errors.

In this subsection, we present the contribution of Silent Data Corruptions (SDCs) to the overall AVF regarding transient faults (i.e., soft errors), considering three major microarchitectural components of an OoO core (i.e., the Physical Register File, the L1 Instruction Cache, and the L1 Data Cache).

1) Physical Register File: Figure 1 presents the SDC AVF results for the Integer Physical Register File (RF) across fifteen benchmarks of the MiBench [33] suite for the three prevailing ISAs (Arm, x86, RISC-V). As shown in Figure 1, the AVF varies from 0% to 6.9% for Arm, 0% to 3.7% for x86 and 0.1% to 9.9% for RISC-V. On average for all benchmarks, RISC-V ISA shows the highest SDC AVF among all ISAs studied in this paper (i.e., most benchmarks provide the highest SDC AVF), while the x86 ISA shows the lowest SDC AVF among all ISAs.

2) L1 Instruction Cache: Figure 2 presents the SDC AVF results for the L1 Instruction Cache across fifteen benchmarks of the MiBench [33] suite for the three ISAs (Arm, x86, RISC-V). As shown in Figure 2, the AVF varies from 0.3% to 9.9% for Arm, 0.3% to 4.6% for x86 and 0.2% to 5.7% for RISC-V.

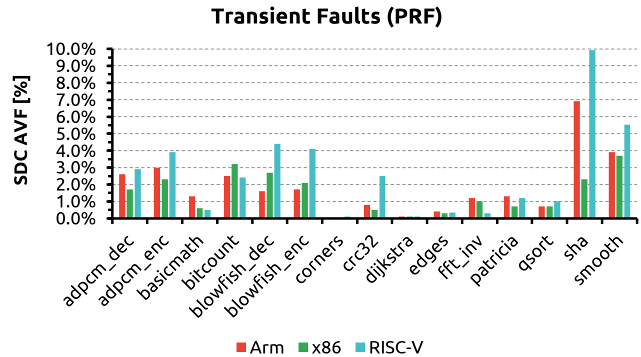

Fig. 1. SDC AVF for transient faults of the physical register file (PRF).

On average for all benchmarks, Arm ISA shows the highest SDC AVF among all ISAs studied in this paper (i.e., most benchmarks provide the highest SDC AVF), while in most of the benchmarks we can see that the x86 ISA shows the lowest SDC AVF.

3) L1 Data Cache: Figure 3 presents the SDC AVF results for the L1 Data Cache across fifteen benchmarks of the MiBench [33] suite for the three ISAs (Arm, x86, RISC-V). As shown in Figure 3, the AVF varies from 1.2% to 43% for Arm, 0.7% to 32.6% for x86 and 0.8% to 40.2% for RISC-V. On average for all benchmarks, Arm ISA shows the highest SDC AVF among all ISAs studied in this paper (i.e., most benchmarks provide the highest SDC AVF), while in most of the benchmarks we can see that the x86 ISA shows the lowest SDC AVF. The same observation exists for the L1 Instruction Cache as well.

Overall, SDCs are much rarer in the Physical Integer Register File and L1 Instruction Cache than in the L1 Data Cache, where they are the dominant fault effect. Specifically, wrong values in registers are very likely to result in illegal memory accesses, and corrupted blocks in the L1 Instruction cache will most likely result in an illegal instruction being executed. Data cache corruptions, on the other hand, are less likely to cause a crash and can easily propagate to the program output resulting in an SDC.

C. Permanent Faults SDC Rates for 3 ISAs

Permanent faults, also known as hard errors, are long-lasting or permanent defects in the CPU's hardware or hardware structures, such as SRAM arrays. These faults typically result from manufacturing defects, physical damage (e.g., from overheating or electrical overloads), or wear and tear over time. Permanent faults are persistent and usually require hardware repair or replacement to resolve. Permanent faults may have serious consequences for CPU operation, potentially resulting in system instability, data corruption, or even severe system failures, depending on their severity and frequency.

In this subsection, we present the contribution of Silent Data Corruptions (SDCs) to the overall AVF regarding permanent faults (i.e., hard errors), considering two major microarchitectural components of an OoO core (i.e., the L1 Instruction Cache, and the L1 Data Cache).

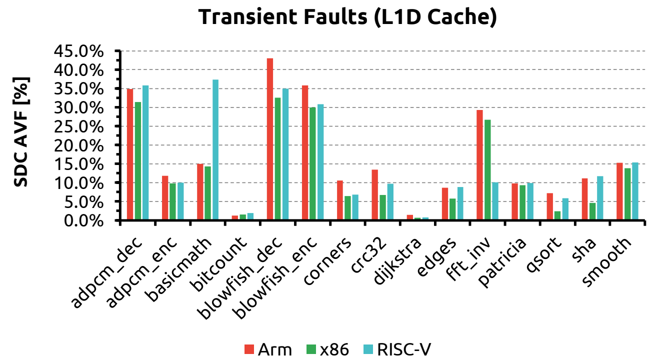

Fig. 3. SDC AVF for transient faults of the L1 data cache.

1) L1 Instruction Cache: Figure 4 presents the SDC AVF results for permanent faults in the L1 Instruction Cache across fifteen benchmarks of the MiBench [33] suite for the three ISAs (Arm, x86, RISC-V). As shown in Figure 4, the AVF varies from 0.1% to 2.3% for Arm, 0.1% to 1.3% for x86 and 0.3% to 2.7% for RISC-V. On average for all benchmarks, x86 ISA shows the lowest SDC AVF among all ISAs studied in this paper (i.e., most benchmarks provide the lowest SDC AVF), while in most of the benchmarks we can see that the RISC-V ISA shows the highest SDC AVF.

2) L1 Data Cache: Figure 5 presents the SDC AVF results for permanent faults in the L1 Data Cache across fifteen benchmarks of the MiBench [33] suite for the three ISAs (Arm, x86, RISC-V). As shown in Figure 5, the AVF varies from 5.1% to 53.3% for Arm, 4.4% to 64.7% for x86 and 4.4% to 70.8% for RISC-V. On average for all benchmarks, the RISC-V ISA shows the highest SDC AVF among all ISAs studied in this paper (i.e., most benchmarks provide the highest SDC AVF).

Overall, the RISC-V ISA exhibits a significantly higher probability of SDCs due to permanent faults compared to the other ISAs, namely Arm and x86.

D. Functional Unit SDC Rates for x86 and RISC-V

Functional units in modern OoO microprocessors are also a source of SDCs. In order to model permanent gate-level faults in these units, we create statistical models describing the

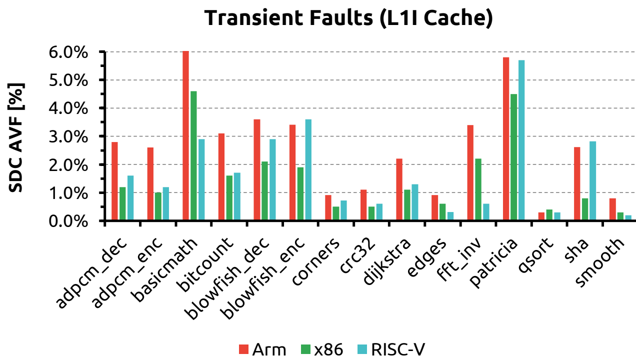

Fig. 2. SDC AVF for transient faults of the L1 instruction cache.

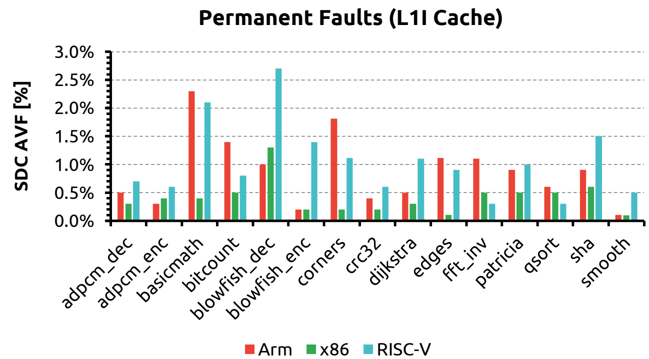

Fig. 4. SDC AVF for permanent faults of the L1 instruction cache.

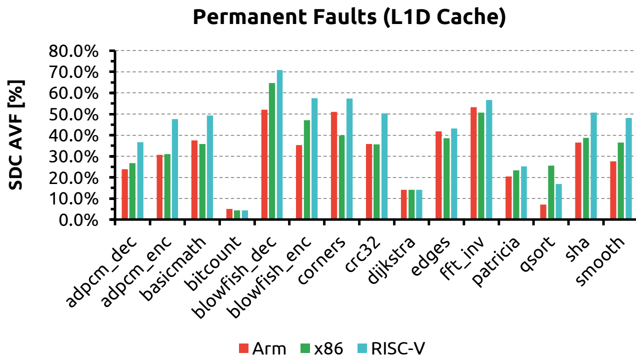

Fig. 5. SDC AVF for permanent faults of the L1 data cache.

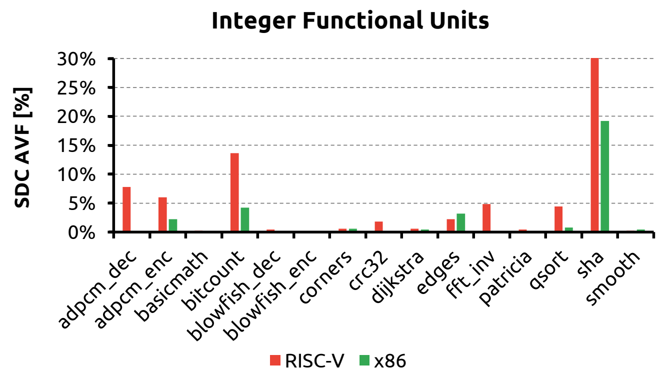

Fig. 6. SDC AVF for permanent faults in an integer adder.

propagation of these faults from the faulty gate to the output. In their simplest form these models constitute of a frequency at which a functional unit presents a corruption of its output (stuck-at-0, stuck-at-1 or bit-flip). In this section we present the SDC AVF of the first of six integer adders[1] in our OoO core for fifteen different benchmarks. For these experiments the output error frequency has been set to 1 output error per 100,000 operations and the output corruption model is set to bit-flip.

Figure 6 presents the SDC AVF results for the aforementioned fault model in integer adder #1 across fifteen benchmarks of the MiBench [33] suite for x86 and RISC-V. As shown in Figure 6, the AVF varies from 0% to 19.2% for x86 and 0% to 30.4% for RISC-V. We notice significant workload variability between the benchmarks, with *sha* and *bitcount* having the largest AVFs. Several benchmarks have an SDC AVF that is either very small or even zero. However, the total AVF is substantially high, and it is mainly attributed to Crashes. This means that in such units the microarchitectural and software masking effects are significantly low.

IV. Related Work

Numerous methodologies exist for estimating, predicting, or measuring system reliability, each differing in terms of their level of detail, accuracy, estimation time, and availability during various stages of the design and product lifecycle.

In situations where silicon prototypes become available, typically in the later stages of design and production, one of the quickest and most efficient reliability assessment methods is accelerated particle beaming. Particle accelerators have a long history of use in studying device and application reliability [34], [35]. During this process, the device is exposed to a particle beam while being monitored for operational anomalies. In references [36]–[39], authors provide experimental data on beam testing of embedded ARM Cortex-A9 processors, propose solutions for enhancing reliability, and discuss the influence of operating systems on application and device reliability. While particle beaming yields precise system-level FIT rates because it targets the actual chip and introduces the actual particles responsible for flips, it only offers coarse-grain system-level observations and does not uncover the root causes of failures. Furthermore, it becomes available late in the design cycle, close to the final product stage, making its results more suitable for design validation than for making critical design decisions. Substantial chip design revisions are nearly impossible (or prohibitively expensive) at this stage. However, one significant advantage of assessing physical chips is the ability to operate them at native speeds.

Software-level fault injection tools are also widely employed to assess systems at native speeds [40]–[47]. Despite their capacity to evaluate systems under realistic workloads, they fall short in capturing hardware vulnerabilities since they start with a corrupted instruction, not a hardware structure, and provide limited information on the Hardware Vulnerability Factor (PVF). It should be noted that PVF, while valuable for enhancing software-side reliability, should be used cautiously as it can provide misleading insights into the AVF estimation of a system.

FPGAs have also been utilized for prototyping and reliability analysis [48]–[52]. FPGA-based reliability assessment offers high throughput and accuracy levels. However, it requires a complete design, which is only available in the late design stages. When dealing with early design stages where only models are accessible, two abstraction levels of microprocessor design can be used for reliability evaluation: RTL (Register Transfer Level) and performance (microarchitecture level) models. The RTL model provides detailed hardware representation, while the microarchitecture-level model offers a more abstract implementation. Both models provide high observability and are available during the design cycle, with microarchitecture-level models being accessible early in the design process and RTL models available just before design signoff. These models help overcome the limitations associated with silicon prototypes, such as the lack of observability. However, early design stage models may suffer from limited throughput and potential inaccuracies, with these drawbacks being inversely proportional—greater accuracy comes at the expense of throughput, and vice versa. The following sections provide a detailed analysis of the methodologies applicable at these two abstraction levels.

[1]load/store address generation and branch target calculation is performed in separate functional units

Microprocessor reliability evaluation is a broad and complex endeavor, involving various methodologies across different abstraction levels and stages of system design and lifetime. Each approach has its advantages and disadvantages, and no single method can cover all aspects of reliability estimation trade-offs. These methodologies complement each other and can contribute to different extents in guiding sound design decisions. However, it remains challenging to determine which of these techniques is closest to the ground truth and how they interrelate due to the complexity involved. Therefore, the aim of this dissertation is to enhance the overall assessment of reliability at the microarchitecture level and provide additional insights to refine the taxonomy of the reliability evaluation ecosystem.

V. Conclusion

This paper provides a comprehensive analysis of silent data corruptions in CPU hardware structures, specifically focusing on the L1 data cache, L1 instruction cache, physical register file, and functional units, across the three prevailing CPU architectures: x86, Arm, and RISC-V. This research contributes to our understanding of SDC risks in modern microprocessors and guides future efforts in designing more robust and reliable CPU architectures.

Acknowledgment

This work is supported by research gifts from Meta and AMD and by the European Union's Horizon Europe research and innovation programme under grant agreements No 101070238 (NEUROPULS) and No 101097224 (REBECCA). Views and opinions expressed are those of the authors only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, "The impact of technology scaling on lifetime reliability," in *International Conference on Dependable Systems and Networks, 2004*, 2004, pp. 177–186.

[2] G. Papadimitriou and D. Gizopoulos, "Silent data corruptions: Microarchitectural perspectives," *IEEE Transactions on Computers*, pp. 1–13, 2023.

[3] G. Papadimitriou, D. Gizopoulos, H. D. Dixit, and S. Sankar, "Silent data corruptions: The stealthy saboteurs of digital integrity," in *2023 IEEE 29th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2023, pp. 1–7.

[4] A. Singh, S. Chakravarty, G. Papadimitriou, and D. Gizopoulos, "Silent data errors: Sources, detection, and modeling," in *2023 IEEE 41st VLSI Test Symposium (VTS)*, 2023, pp. 1–12.

[5] P. R. Bodmann, G. Papadimitriou, R. L. Rech Junior, D. Gizopoulos, and P. Rech, "Soft error effects on arm microprocessors: Early estimations versus chip measurements," *Computer*, vol. 56, no. 7, pp. 4–6, 2023.

[6] P. R. Bodmann, G. Papadimitriou, R. L. R. Junior, D. Gizopoulos, and P. Rech, "Soft error effects on arm microprocessors: Early estimations versus chip measurements," *IEEE Transactions on Computers*, vol. 71, no. 10, pp. 2358–2369, 2022.

[7] G. Papadimitriou and D. Gizopoulos, "Avgi: Microarchitecture-driven, fast and accurate vulnerability assessment," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023, pp. 935–948.

[8] ——, "Characterizing soft error vulnerability of cpus across compiler optimizations and microarchitectures," in *2021 IEEE International Symposium on Workload Characterization (IISWC)*, 2021, pp. 113–124.

[9] ——, "Anatomy of on-chip memory hardware fault effects across the layers," *IEEE Transactions on Emerging Topics in Computing*, vol. 11, no. 2, pp. 420–431, 2023.

[10] ——, "Demystifying the system vulnerability stack: Transient fault effects across the layers," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 902–915.

[11] P. Bodmann, G. Papadimitriou, D. Gizopoulos, and P. Rech, "Impact of cores integration and operating system on arm processors reliability: Micro-architectural fault-injection vs beam experiments," in *2020 20th European Conference on Radiation and Its Effects on Components and Systems (RADECS)*, 2020, pp. 1–4.

[12] 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)*, 2019, pp. 26–38.

[13] P. Bodmann, G. Papadimitriou, D. Gizopoulos, and P. Rech, "The impact of soc integration and os deployment on the reliability of arm processors," in *2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2021, pp. 223–225.

[14] D. Gizopoulos, G. Papadimitriou, A. Chatzidimitriou, V. J. Reddi, B. Salami, O. S. Unsal, A. C. Kestelman, and J. Leng, "Modern hardware margins: Cpus, gpus, fpgas recent system-level studies," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2019, pp. 129–134.

[15] G. Papadimitriou, A. Chatzidimitriou, and D. Gizopoulos, "Adaptive voltage/frequency scaling and core allocation for balanced energy and performance on multicore cpus," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2019, pp. 133–146.

[16] G. Papadimitriou, M. Kaliorakis, A. Chatzidimitriou, D. Gizopoulos, P. Lawthers, and S. Das, "Harnessing voltage margins for energy efficiency in multicore cpus," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-50 '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 503–516. [Online]. Available: https://doi.org/10.1145/3123939.3124537

[17] G. Papadimitriou, M. Kaliorakis, A. Chatzidimitriou, D. Gizopoulos, G. Favor, K. Sankaran, and S. Das, "A system-level voltage/frequency scaling characterization framework for multicore cpus," in *IEEE Silicon Errors in Logic – System Effects (SELSE 2017)*, 2017. [Online]. Available: https://doi.org/10.48550/arXiv.2106.09975

[18] I. Tsiokanos, G. Papadimitriou, D. Gizopoulos, and G. Karakonstantis, "Boosting microprocessor efficiency: Circuit- and workload-aware assessment of timing errors," in *2021 IEEE International Symposium on Workload Characterization (IISWC)*, 2021, pp. 125–137.

[19] A. Chatzidimitriou, G. Papadimitriou, D. Gizopoulos, S. Ganapathy, and J. Kalamatianos, "Assessing the effects of low voltage in branch prediction units," in *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2019, pp. 127–136.

[20] ——, "Analysis and characterization of ultra low power branch predictors," in *2018 IEEE 36th International Conference on Computer Design (ICCD)*, 2018, pp. 144–147.

[21] K. Tovletoglou, L. Mukhanov, G. Karakonstantis, A. Chatzidimitriou, G. Papadimitriou, M. Kaliorakis, D. Gizopoulos, Z. Hadjilambrou, Y. Sazeides, A. Lampropulos, S. Das, and P. Vo, "Measuring and exploiting guardbands of server-grade armv8 cpu cores and drams," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2018, pp. 6–9.

[22] P. Koutsovasilis, C. D. Antonopoulos, N. Bellas, S. Lalis, G. Papadimitriou, A. Chatzidimitriou, and D. Gizopoulos, "The impact of cpu voltage margins on power-constrained execution," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 1, pp. 221–234, 2022.

[23] G. Papadimitriou, A. Chatzidimitriou, D. Gizopoulos, V. J. Reddi, J. Leng, B. Salami, O. S. Unsal, and A. C. Kestelman, "Exceeding conservative limits: A consolidated analysis on modern hardware margins," *IEEE Transactions on Device and Materials Reliability*, vol. 20, no. 2, pp. 341–350, 2020.

[24] A. Chatzidimitriou, G. Papadimitriou, and D. Gizopoulos, "Healthlog monitor: Errors, symptoms and reactions consolidated," *IEEE Transactions on Device and Materials Reliability*, vol. 19, no. 1, pp. 46–54, 2019.

[25] ——, "Healthlog monitor: A flexible system-monitoring linux service,"

in *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*, 2018, pp. 183–188.

[26] 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*, 2009, pp. 502–506.

[27] S. Mukherjee, C. Weaver, J. Emer, S. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, 2003, pp. 29–40.

[28] ——, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, 2003, pp. 29–40.

[29] 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, no. 2, p. 1–7, aug 2011. [Online]. Available: https://doi.org/10.1145/2024716.2024718

[30] J. Lowe-Power, A. M. Ahmad, A. Akram, M. Alian, R. Amslinger, M. Andreozzi, A. Armejach, N. Asmussen, B. Beckmann, S. Bharadwaj, G. Black, G. Bloom, B. R. Bruce, D. R. Carvalho, J. Castrillon, L. Chen, N. Derumigny, S. Diestelhorst, W. Elsasser, C. Escuin, M. Fariborz, A. Farmahini-Farahani, P. Fotouhi, R. Gambord, J. Gandhi, D. Gope, T. Grass, A. Gutierrez, B. Hanindhito, A. Hansson, S. Haria, A. Harris, T. Hayes, A. Herrera, M. Horsnell, S. A. R. Jafri, R. Jagtap, H. Jang, R. Jeyapaul, T. M. Jones, M. Jung, S. Kannoth, H. Khaleghzadeh, Y. Kodama, T. Krishna, T. Marinelli, C. Menard, A. Mondelli, M. Moreto, T. Mück, O. Naji, K. Nathella, H. Nguyen, N. Nikoleris, L. E. Olson, M. Orr, B. Pham, P. Prieto, T. Reddy, A. Roelke, M. Samani, A. Sandberg, J. Setoain, B. Shingarov, M. D. Sinclair, T. Ta, R. Thakur, G. Travaglini, M. Upton, N. Vaish, I. Vougioukas, W. Wang, Z. Wang, N. Wehn, C. Weis, D. A. Wood, H. Yoon, and Éder F. Zulian, "The gem5 simulator: Version 20.0+," 2020. [Online]. Available: https://arxiv.org/abs/2007.03152

[31] "gem5 GitHub Repository," https://github.com/gem5/gem5, accessed: 2023-07-25.

[32] 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)*, 2019, pp. 119–130.

[33] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. 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)*, 2001, pp. 3–14. [Online]. Available: https://doi.org/10.1109/WWC.2001.990739

[34] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *Device and Materials Reliability, IEEE Transactions on*, vol. 5, no. 3, pp. 305–316, Sept 2005.

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

[36] 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, no. 4, pp. 2225–2232, Aug 2016.

[37] 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*, ser. SBCCI '17. New York, NY, USA: ACM, 2017, pp. 84–89. [Online]. Available: http://doi.acm.org/10.1145/3109984.3110008

[38] 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)*, 2018, pp. 13–26.

[39] 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 Transactions on Dependable and Secure Computing*, vol. 13, no. 4, pp. 502–508, 2016.

[40] T. Tsai, R. Iyer, and D. Jewitt, "An approach towards benchmarking of fault-tolerant commercial systems," in *Proceedings of Annual Symposium on Fault Tolerant Computing*, 1996, pp. 314–323.

[41] S. Han, K. Shin, and H. Rosenberg, "Doctor: an integrated software fault injection environment for distributed real-time systems," in *Proceedings of 1995 IEEE International Computer Performance and Dependability Symposium*, 1995, pp. 204–213.

[42] J. Carreira, H. Madeira, J. Silva, and D. Informtica, "Xception: Software fault injection and monitoring in processor functional units," *Proceedings of the 5th IFIP Working Conference on Dependable Computing for Critical Applications*, 03 2001.

[43] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: Building customized program analysis tools with dynamic instrumentation," in *Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 190–200. [Online]. Available: https://doi.org/10.1145/1065010.1065034

[44] H. Madeira, D. Costa, and M. Vieira, "On the emulation of software faults by software fault injection," in *Proceeding International Conference on Dependable Systems and Networks. DSN 2000*, 2000, pp. 417–426.

[45] R. Natella, D. Cotroneo, J. A. Duraes, and H. S. Madeira, "On fault representativeness of software fault injection," *IEEE Transactions on Software Engineering*, vol. 39, no. 1, pp. 80–96, 2013.

[46] J. Wei, A. Thomas, G. Li, and K. Pattabiraman, "Quantifying the accuracy of high-level fault injection techniques for hardware faults," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2014, pp. 375–382.

[47] Q. Lu, M. Farahani, J. Wei, A. Thomas, and K. Pattabiraman, "Llfi: An intermediate code-level fault injection tool for hardware faults," in *2015 IEEE International Conference on Software Quality, Reliability and Security*, 2015, pp. 11–16.

[48] A. Sari and M. Psarakis, "A fault injection platform for the analysis of soft error effects in fpga soft processors," in *2016 IEEE 19th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, 2016, pp. 1–6.

[49] M. Alderighi, F. Casini, S. D'Angelo, M. Mancini, S. Pastore, G. Sechi, and R. Weigand, "Evaluation of single event upset mitigation schemes for sram based fpgas using the flipper fault injection platform," in *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007)*, 2007, pp. 105–113.

[50] S. Di Carlo, P. Prinetto, D. Rolfo, and P. Trotta, "A fault injection methodology and infrastructure for fast single event upsets emulation on xilinx sram-based fpgas," in *2014 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2014, pp. 159–164.

[51] N. A. Harward, M. R. Gardiner, L. W. Hsiao, and M. J. Wirthlin, "Estimating soft processor soft error sensitivity through fault injection," in *2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines*, 2015, pp. 143–150.

[52] G. L. Nazar and L. Carro, "Fast single-fpga fault injection platform," in *2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2012, pp. 152–157.