Impact of Cores Integration and Operating System on ARM Processors Reliability: Micro-Architectural Fault-Injection vs Beam Experiments

Pablo Bodmann* George Papadimitriou† Dimitris Gizopoulos† Paolo Rech*
†*Dept. of Informatics and Telecommunications, University of Athens,* Athens, Greece
**PPGC, Institute of Informatics, Federal University of Rio Grande do Sul,* Porto Alegre, Brasil

***Abstract*—We compare and correlate neutron beam and micro-architectural fault-injection data on ARM Cortex-A5 and Cortex-A9 running codes bare-metal and on top of Linux. Cores integration exacerbates crashes while Linux does not significantly impact SDCs rate.**

I. Introduction

The trend in the design of computing devices is to include multiple, heterogeneous, cores in the same chip. Modern System on Chips (SoCs) usually integrate a central processing unit (CPU) and an accelerator, such as Graphic Processing Units (GPUs) or a Field Programmable Gate Array (FPGA). Although the integration of these components has unquestionable benefits in terms of performance, efficiency, and flexibility, it is still unclear how it impacts the reliability of the main CPU. Moreover, the Operating System (OS), which is required to orchestrate the different cores and to manage the available hardware and software resources can also potentially impact the system reliability [1], [2].

Heterogeneous SoCs, once common mainly in portable devices such as smartphones, tablets, and laptops, are extremely attractive for both autonomous vehicles and High Performance Computing (HPC). NVIDIA Xavier platform for real-time object detection embeds an ARM core and ARM itself has expressed interested in powering autonomous vehicles with their platforms. Moreover, the US Department of Energy's labs at Sandia and Los Alamos, in NM, announced that their next clusters are going to be powered by ARM SoCs [3].

In this paper, we analyze, quantify, and qualify the impact of cores integration and the OS interference on the reliability of ARM microprocessors. We focus mainly on cores integration and OS are they are the most likely sources of reliability uncertainly and the least studied ones. We target a standalone Cortex-A5 and a Cortex-A9 integrated in a SoC. Through *accelerated neutron beam experiments* that count for more than 15 million years of natural exposure, we have measured the FIT rates of the A5 and A9 cores running representative benchmarks bare-metal and on top of Linux. We have also performed extensive *micro-architecture-level evaluation* (on equivalent CPU models of the A5 and A9 CPUs on Gem5 simulator), injecting more than 200,000 faults. Correlating and comparing the beam experiments with the fault injection results of the standalone (A5) and integrated (A9) cores, we can provide a realistic and detailed reliability evaluation. We found that due to peripherals and interfaces, the integration of various cores significantly increases the System Crash rates but has negligible impact on the Silent Data Corruption (SDC) rate which is attributed to the CPU cores. Moreover, we extend previous work [2] showing that the OS has a beneficial impact on the Application Crashes but not on the SDC rates.

Beam experiments (and software fault-injection) must be performed on the final product, when eventual modifications to improve the architecture reliability would be extremely costly. Micro-architectural models are available in the early stages of the project and they could provide valuable information to improve its reliability, once they are found to be sufficiently accurate to predict the product FIT. One of the main goal of this work is exactly to evaluate how close beam and micro-architectural reliability evaluations are. As we show, the SDC portion of the overall system failure rate can be accurately predicted with early-stage fault-injection for both stand alone or integrated cores, independently of the OS presence. The Crash rates, on the contrary, are severely affected by both aspects and might be underestimated with fault-injection alone.

II. Background and Related Work

ARM processors have been exposed to accelerated particles beam and have been subjects and fault injection in previous studies. In [2], [4]–[6] authors present beam experimental data on embedded ARM Cortex-A9, propose hardening solutions, and discuss the impact of the presence of an operating system in the application and device reliability. Some work has been done to evaluate the influence of an operating system on the reliability of codes executions [1], [2]. The available data shows that the operating system can be beneficial in the presence of cache conflicts, as the application data is written back when the operating system takes control. However, these papers do not perform fault injection, as we do in this paper, but are just based on beam experiments, which might limit the insights that can be gathered. Finally, none of these works address the impact of cores integration in the reliability of a device or of a code. This is the first paper that uses both beam experiments and micro-architectural fault injection to better understand how the operating system and the cores integration affect the reliability of a processor.

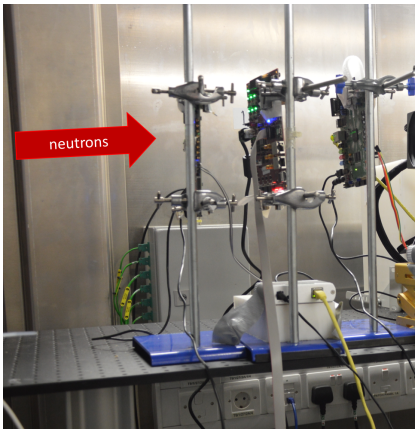

Fig. 1. Part of the radiation test setup at ChipIR.

III. Evaluation Methodologies

To have an accurate and detailed analysis of the impact of cores integration and OS in ARM devices reliability we compare and combine beam experiments and micro-architectural fault-injection. With the goal of moving a step forward in the validation of fault-injection data, we also predict the FIT rates of the A5 and A9 with fault-injection and compare them with the experimentally measured FIT rate.

Our study is performed on the ARM®Cortex™-A5 CPU implemented in a 65nm CMOS technology in the Microchip SAMA5D2 XPLAINED ULTRA board and on the ARM®Cortex™-A9 embedded in a Xilinx Zynq™-7000 AP SoC implemented in a 28nm CMOS technology. The Zynq has two ARM cores (we disabled one) and the SAMA5D2 has a single core. Each core has a 32 KB 4-way set-associative instruction and data caches and a unified 8-way set-associative Level 2 cache of 128kB and 512kB for A5 and A9 respectively.

We have chosen three benchmarks with different computational characteristics from the reliability benchmark suite [7]: FFT, MxM, Quicksort. The reliability of the two devices was evaluated in both bare-metal and on the top of Linux.

The **Silent Data Corruption (SDC)** detection, identical for the bare-metal and Linux setup, is performed comparing the experimental output (for either beam or fault-injection) with the expected output (calculated in a fault-free environment). 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 stuck the application and, then, the device. We call this event **Crash**. When the application runs on top of Linux, a fault can stuck the application but not the OS (Linux is still responsive and a new instance of the application can be launched) or can stuck the OS, requiring a hard reboot. We call **AppCrash** the former and **SysCrash** the latter.

A. Neutron Beam Experiments

Our radiation experiments were performed at the ChipIR, Rutherford Appleton Laboratory (RAL) in Didcot, UK (Figure 1). The available neutron flux was about $3.5 \times 10^6 n/(cm^2/s)$. Using a total of four A5 and three A9 boards, each of the 12 configurations (3 codes on each architecture, bare-metal and on top of Linux) were tested for more than 100 effective hours (i.e., not considering setup, initialization, and recover from crash times). The 1,200 hours of test, when scaled to the natural exposure, account for more than 15 million years.

B. Micro-Architecture Fault-Injection

The microarchitecture-level reliability assessment for both the A5 and A9 CPUs was performed on top of Gem5 simulator [8], using the GeFIN fault injection framework [9]. GeFIN quantifies the Architectural Vulnerability Factor (AVF) of the system, which expresses the probability for a fault to lead to a failure. The in-order core was used to resemble A5 microarchitecture while the out-of-order core was used for A9. Unlike beam experiments or software fault injection, microarchitecture-level simulation is available in the early stage of the chip project, before the RTL description. If found to be sufficiently accurate, GeFIN could allow a prompt estimation of the product error rate.

We have configured GeFIN to inject single-event transient faults during system simulation in the following components, which cover the vast majority (>90%) of resources inside the CPU core: CPU pipelines, L2 Cache, L1 Data and Instruction Caches, Physical Register file, Data and Instruction Translation Lookaside Buffers (TLB). To achieve a statistical sample of 2% error margin and 95% confidence level, we have injected 1,000 single bit transient faults on each of the target components, for a total of more than 200,000 faults injected.

C. Beam vs Fault Injection

We remove any other source of variation in our beam vs fault-injection comparison keeping the same input size, the same input vectors, the same compiler options, and a very close OS distribution between the tested configurations (3.14 for the A9 and 4.14 for the A5 on the beam setups, while for Gem5 version used is 3.13). These were the closest kernel versions that have been ported on the two platforms. Despite the different versions, all executables were compiled with the same compiler and static linked. All the OS functions used during computation are then exactly the same between the A5 and A9, between bare-metal and Linux execution. The only difference would be the communication with the board which uses the Ethernet adapter and the appropriate system calls which are only used at the end of each iteration of the benchmark (less than 0.1% of execution time).

To evaluate if the benchmark execution shows any difference when running in the setups used for beam experiments and fault injection, we compare the execution of the same code in the two setups using different counters: CPU cycles, branch misses, L1 data cache accesses, L1 data cache misses, L1 data TLB misses, L1 instruction cache misses, and L1 Instruction TLB misses. About 70% of the counters report acceptable deviations between the two setups (lower than 1%).

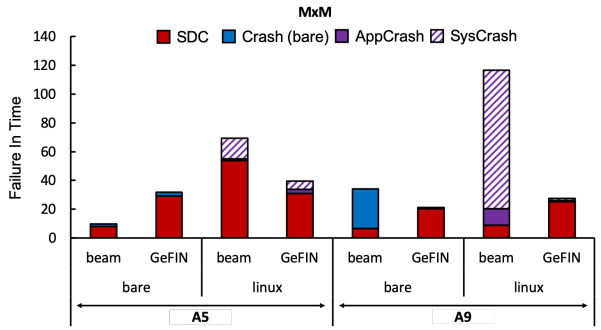

Fig. 2. MxM A5 and A9 bare-metal and Linux beam FIT rates measured with beam and fault-injection.

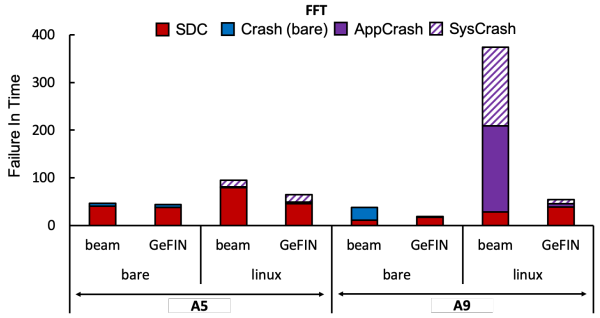

Fig. 3. FFT A5 and A9 bare-metal and Linux beam FIT rates measured with beam and fault-injection.

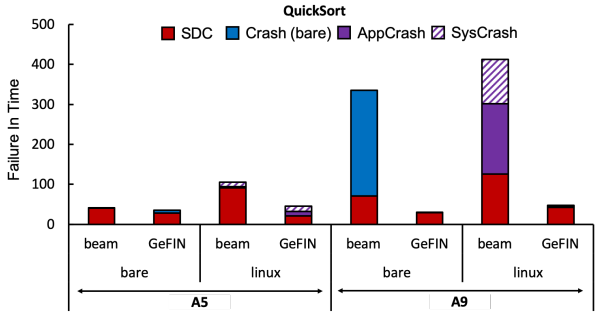

Fig. 4. QuickSort A5 and A9 bare-metal and Linux beam FIT rates measured with beam and fault-injection.

The biggest difference is observed in the L1 Instruction TLB counters.

To estimate the FIT rate of a code using fault injection, it is necessary to multiply the raw fault rate of each micro-architectural resource (i.e., the probability for a fault to occur) to its AVF (the probability for the fault to propagate to the output). However, measuring the raw fault rate for each resource would require too much time and, when dealing with COTS, could be unfeasible due to visibility limitations. We decided to use the experimentally measured L1 Cache raw FIT rate, as a reference raw FIT for the technology of each microprocessor. This simplification is justified by the fact that caches are normally the most vulnerable resource in a microprocessor and most CPU resources are likely to be implemented in the same technology as the L1 cache.

The biggest difference between the setups is that GeFIN does not include any other cores or peripherals besides the ARM CPU, while in the silicon devices those are necessary for interfacing. This difference is particularly remarkable for the A9, as the tested Zynq integrates 2 CPUs (one disabled), several other cores, and implements various communication interfaces while the A5 chip only includes ethernet and usb interfaces. The comparison between A5 and A9, between beam and GeFIN can then also help in understanding the impact of cores integration in the CPU error rate.

IV. Fault Injection and Beam Testing Data

In this section we present the FIT rates measured with beam experiments and predicted with fault-injection on the A5 CPU and on the A9-based SoC. To predict the FIT rate of the A5 and A9 we multiply the AVF with the technology sensitivity (L1 cache bit FIT rate), as discussed in Section III-C. With a dedicated experiment, we have measured the neutron FIT rate of the L1 cache, which results being 2.37×10^{-4} FIT/bit for the A5 and 2.59×10^{-5} FIT/bit for the A9. A neutron has then about a 8.6x higher probability of generating a fault in the A5 than in the A9.

Figures 2, 3, and 4 show for MxM, FFT, and QuickSort, the SDCs and the DUEs (Crash for bare-metal, AppCrash, and SysCrash for Linux) FIT rates measured at ChipIR (beam) and predicted with micro-architectural fault-injection (GeFIN). In each Figure we show the results obtained executing the codes bare-metal and on the top of Linux for both the A5 and for the A9. 95% confidence intervals (not shown) are lower than 15% of reported values for beam, lower than 2% for GeFIN.

As a first quantitative comparison between the A5 and A9 we can observe that the average SDC rates are very similar for the two devices (the average SDC rates difference between the A5 and A9 is lower than 40%). Interestingly, the average DUE rate (Crash, AppCrash, SysCrash) is at least one order of magnitude higher on the A9 than on the A5. This data, correlated with the fact that A5 has a 8.6x higher technology sensitivity than the A9, suggests that the A9 architecture is much more vulnerable than the A5, mostly for DUEs. With the micro-architectural analysis we will investigate the A5 and A9 vulnerabilities further.

From Figures 2, 3, and 4 we can observe that, for SDCs, there is quite a good correlation between the FIT rates measured with beam experiments and predicted with GeFIN. The average difference is less than 1.2x, which is pretty accurate considering the simplifications that come with a micro-architectural simulation. The A9 has a much higher AVF (not shown for lack of space) than the A5, mostly due to the larger memory arrays and higher amount of resources available. This is a first promising results towards the estimation of the error rate of a processor with micro-architectural fault-injection. The order of magnitude of the SDC rate can be predicted knowing just the sensitivity of the

technology in which the device is going to be implemented, allowing to identify specific vulnerabilities in the early stages of the project. The most vulnerable resource we identified with micro-architectural analysis depends on the executed code and, not surprisingly, normally is the cache or the caches tags.

The OS presence increases the SDC FIT rate of about 2.4x for the A5 and 1.9x for the A9, on the average. Interestingly, the trend observed for beam experiments is maintained with GeFIN, for all codes, showing that micro-architectural fault-injection can also be used to predict the OS effects on SDCs and for better understanding the reasons for the observed behaviors. Micro-architectural analysis confirms previous work that suggested that the slight increase of SDC FIT rates caused by Linux is due to longer waiting times of data in the caches [2].

When it comes to DUEs, as detailed in Section III, 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 Application Crash (i.e., the application does not respond but Linux is still up an running) and System Crash (i.e., Linux is no longer responding).

We observed that Crashes in a bare-metal environment are mainly caused by exceptions raised by the CPU. Theses can be an undefined instruction, i.e, an invalid instruction code, a prefetch abort, an instruction fetched from an illegal address and a data abort, a data load or store from 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. While in bare-metal the CPU itself terminates the application, the Linux AppCrash results from a kernel terminating the application. The reason for the kernel terminating the application may be the result from a CPU exception handled by the kernel (i.e., errors in the control flow, bad memory accesses, etc... as for the bare-metal Crash), but also from a signal sent by the kernel to the application. This explains why, as shown in Figures 2, 3, and 4, 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 drivers or services. The reported beam experimental data shows that, while the SDC rate and Crash/AppCrash rates varies significantly across benchmarks, the SysCrash rate is almost constant for the A5 and for the A9 (the SysCrash variation measured with beam between codes is, on the average, 30%), but not across devices. This observation also confirms previous studies that demonstrate that the System Crash has a stronger component that depends on the hardware alone and is almost independent on the code being executed [6].

A very interesting result that derives from Figures 2, 3, and 4 is that, while for the A5, for both Linux and bare-metal, beam and GeFIN provide similar Crash, AppCrash, and SysCrash rates (the average differences are lower than 1.2x), for the A9 the two reliability measurements diverge significantly. For bare-metal, beam experiments for the A9 shows, on the average, a more than 2 orders of magnitude higher Crash rate than GeFIN. For Linux, the difference on AppCrash and SysCrash is even higher, reaching almost 3 orders of magnitude. The only difference between the two setups, as detailed in Section III-C, is that for the A9 the tested silicon device is embedded in a SoC that includes an FPGA and many others interfaces while the A9 simulated in GeFIN is stand-alone, as stand-alone is also the A5 GeFIN model and the A5 silicon device. As there is a good correlation between GeFIN and beam experiments for all the other configurations, and as only the A9 under the beam shows an extremely high DUE rate we can derive that the integration of various cores has the side effect of exacerbating the probability of crashes but has negligible impact on the SDCs rate.

Microarchitectural fault injection, then, provides a good estimation of the SDC FIT rate *before* implementing the device in silicon while a large portion of Crashes is generated by faults *outside* the CPU core. Whenever the FIT rate estimate is too high for the project requirements, taking advantage of ARM flexibility in modifying the microarchitecture, it is possible to include hardware reliability solutions to the resources that GeFIN found to be responsible for the majority of failures, avoiding the introduction of unnecessary overhead or inefficiencies. It is worth noting that, according to our results, these CPU architectural hardening solutions may not be effective in reducing the Crash rate if the CPU is integrated in a SoC, as most Crashes are caused by faults in external resources. A re-design of the cores interfaces, including specific software procedures to handle synchronizations or hangs problems, could be necessary to significantly reduce Crashes.

V. Conclusions

In this work, we have performed an extensive reliability evaluation of two ARM microprocessors, a Cortex-A5 and a Cortex-A9, fabricated using different technologies and with different SoC organizations. Our analysis shows that the SDC FIT rate is only slightly affected by the SoC integration and the existence of the OS. On the other hand, Crashes (both application and system) are dramatically increased (up to 3 orders of magnitude) on the SoC that includes the FPGA, showcasing how this attribute can really influence the FIT rate.

References

[1] R. K. Iyer and D. J. Rossetti, *IEEE Transactions on Software Engineering*, vol. SE-11, no. 12, pp. 1438–1448, 1985.

[2] T. Santini *et al.*, *IEEE TNS*, no. 4, pp. 2225–2232, Aug.

[3] N. Hemsoth. (2018) Arm it the nnsa's new secret weapon.

[4] A. B. de Oliveira *et al.*, ser. SBCCI '17, 2017.

[5] A. Martínez-Álvarez *et al.*, *IEEE Trans. Dependable Sec. Comput.*, vol. 13, no. 4, pp. 502–508, 2016.

[6] V. Fratin *et al.*, in *DSN*, 2018, pp. 13–26.

[7] H. Quinn *et al.*, *IEEE TNS*, vol. 62, no. 6, pp. 2547–2554, 2015.

[8] N. Binkert *et al.*, *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug 2011.

[9] A. Chatzidimitriou and D. Gizopoulos, in *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, Apr 2016.