POLITECNICO DI TORINO
Repository ISTITUZIONALE

RT Level vs. Microarchitecture-Level Reliability Assessment: Case Study on ARM(R) Cortex(R)-A9 CPU

Original
RT Level vs. Microarchitecture-Level Reliability Assessment: Case Study on ARM(R) Cortex(R)-A9 CPU / Chatzidimitriou, A., Kaliorakis, M., Gizopoulos, D., Iacaruso, M., Pipponzi, M., Mariani, R., Di Carlo, S.. - STAMPA. - (2017), pp. 117-120. (47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W) Denver, CO, USA, USA 26-29 June 2017) [10.1109/DSN-W.2017.16].

Availability:
This version is available at: 11583/2692797 since: 2017-11-20T10:29:11Z

Publisher:
IEEE

Published
DOI:10.1109/DSN-W.2017.16

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

16 September 2026

RT Level vs. Microarchitecture-Level Reliability Assessment: Case Study on ARM® Cortex®-A9 CPU

Athanasios Chatzidimitriou* Manolis Kaliorakis* Dimitris Gizopoulos*
Maurizio Iacaruso† Mauro Pipponzi† Riccardo Mariani† Stefano Di Carlo§

* University of Athens, {achatz | manoliskal | dgizop}@di.uoa.gr
† Yogitech s.p.a.[1], {maurizio.iacaruso | mauro.pipponzi | riccardo.mariani}@intel.com
§ Politecnico di Torino, stefano.dicarlo@polito.it

***Abstract*—Reliability assessment has always been a major concern in the design of computing systems. The results of the assessment highlight and guide enhancements which trigger re-design cycles; thus *early* and *accurate* reliability assessment is of profound importance. For the purposes of early reliability analysis, abstract models of the design (which are available in early design stages) are typically used. These models, however, may not be completely accurate compared to the actual final design. Existing literature has not quantified this inaccuracy, through a comparison between Register-Transfer-Level (RTL) and microarchitecture-level reliability assessment on the same commercial microprocessor design.**

In this paper, we perform reliability assessment using statistical fault-injection on the RTL and Microarchitectural models of the same commercial ARM® Cortex®-A9 processor. The assessment was performed using the same benchmark workloads and equivalent configurations of the hardware structures. The results show that, compared to RTL model, the almost 200x faster microarchitectural model reports an average difference of 0.7 percentile units (10%) on the vulnerability estimation of register file and 3 percentile units (20%) on the vulnerability estimation of L1 data cache.

Keywords-fault injection; reliability; microarchitecture-level; RTL; ARM® Cortex®-A9 microprocessor

I. INTRODUCTION

The increasing density of transistors per chip has raised significant reliability concerns for system designers. Nano-scale devices have become vulnerable to cosmic radiations, which introduce transient faults (soft errors) [1] [2]. These faults can harm the system operation and cause catastrophic results, especially when considering safety-critical systems. Designers adopt fault-tolerance techniques to address the increasing appearance of soft errors and to reduce the probability of system failures. However, fault-tolerance comes at certain costs in terms of performance, area and energy footprint. For instance, typical memory protection mechanisms require additional area ranging from 1% to 125% [3].

In order to avoid costly and unnecessary use of protection mechanisms, the actual reliability of the system must be accurately assessed. Existing techniques for reliability assessment can be grouped in three main categories: Architecturally Correct Execution (ACE) analysis [4], Statistical Fault Injection (SFI) [5] and Probabilistic-Statistical models [6] [7]. Each method offers a different trade-off between speed and accuracy. These techniques can be applied on simulation models of the target system. Depending on their abstraction level, the simulation tools can be grouped in three categories: (i) RTL simulators; the actual hardware design, (ii) Microarchitecture-level simulators; detailed cycle-accurate model of the microarchitecture and (iii) Architectural emulators; software-level emulation without hardware details.

The most detailed RTL models are extremely slow while the more abstract models are significantly faster. Microarchitectural simulators have been widely used to evaluate the impact of different design choices in terms of performance, power, energy and reliability, as they offer the following advantages compared to RTL simulators:

- Higher throughput (2 to 3 orders of magnitude).
- Complete system stack modeling, allowing reliability estimation of the hardware layer (Hardware Vulnerability Factor) [8], the software layer (Program Vulnerability Factor) [9], and the entire system stack (AVF) [8].
- Suitability for early reliability estimation since they are available earlier in the design chain.
- Support of full system capabilities that can also include the operating system.

Although many arguments exist on which model is better and in which terms, the literature lacks of an actual comparison between RTL and microarchitectural reliability assessment for a popular commercial CPU model. In this paper we make the following contributions:

1. We present a reliability assessment comparison between RTL and microarchitectural models of the ARM® Cortex®-A9 CPU using statistical fault injection.
2. We highlight capabilities and limitations that exist when modeling the same system on different abstraction layers.

Our findings show that the average difference on the vulnerability estimation between the two models is about 15%, which translates to a difference of about 2 percentile units. However, it is not always possible to model the exact same workloads on the exact same hardware, as some differences cannot be covered and may still introduce sources of error.

II. BACKGROUND AND RELATED WORK

A. Vulnerability

A common metric to evaluate the vulnerability of a CPU hardware structure when a particular program is executed on the CPU is the *Architectural Vulnerability Factor* (AVF) [4]

[1] Intel Corporation Italia has acquired Yogitech in 2016 www.yogitech.com

that defines the probability of a fault in a particular structure to result in a corruption of the program visible output. In the industrial domain a variant of this metric is used, called *Safeness*. *Safeness* is different than AVF as it evaluates the vulnerability of a structure by observing the pinouts of a hardware block, which in the case of a Cortex®-A9 CPU, includes the pipeline and the L1 cache memories. This observation point can only capture differences that are written-back in the lower memory hierarchy, which corresponds to neither the architectural nor the hardware vulnerability factors (AVF [4] or HVF [8]). For each run the trace of the pinout is compared to the golden trace of a fault-free run and a mismatch in the trace identifies an *unsafe* run. Given a number of N simulations, *safeness* is defined by the following formula:

$$Safeness = \frac{\#Safe\ runs}{N}$$

B. Array-based Structures and Logic Components

Modern systems consist of two main types of components: the storage elements (SRAM arrays, e.g., caches, register files, queues, buffers, etc.) that occupy the largest part of the chip's area and control/logic components (controllers, arithmetic units, etc.). The reliability assessment of both these types of components is of major importance in the early design phases of a system. Authors in [10] report that only 11% of processor AVF is caused by logic components, and this is along with the assumption that large SRAM arrays (such as the register file and cache memories) are protected; which is not a typical case. RTL models accurately describe all components, while microarchitectural models have only functional descriptions of the logic blocks and accurately model the major storage structures. This makes the analysis of the effect of faults on logic blocks possible only on RTL models. However, the majority of storage elements are modeled in micro-architecture level models. Storage elements constitute the functional inputs and outputs of the logic blocks. Since the functionality of logic is the same in both models, fault effects of storage elements are accurately modeled in both of them.

C. Related work

There are many reliability evaluation studies based on fault injection experiments either at the RTL [11] [12] or at the microarchitecture-level [13] [14]. Also, *Architecturally Correct Execution* (ACE) analysis [4] has been widely used in many reliability studies either at the RTL [15] [16] or at the microarchitecture-level [4] [16] [8]. None of these studies compares the reliability metrics estimated with the two levels of abstraction in terms of speed and accuracy. This is the goal of our study. In [17] the authors compare fault injection and the ACE analysis at the microarchitectural level, while in [18] the same comparison takes place at the RTL level.

In [19], the authors compare RTL models to software models using fault injection, concluding that RTL reliability assessment can offer significantly more accurate reliability estimations than software emulators. The ARM® Cortex® microprocessors family has been previously used for reliability estimation with fault injection either at the RTL [20] or at the architectural level [21].

Our study is the first that compares a state-of-the-art RTL fault injection methodology being used in the industry with the microarchitectural level injection on the same processor.

III. Experimental Setup

A. RTL Setup

To perform RTL simulation, we use an industrial workflow designed by Yogitech s.p.a (now Intel Corporation Italia s.p.a.) that is based on the Cadence NCSIM simulator and the Yogitech Safety Verifier. The former tool is used to perform RTL simulations, while the latter is used for managing the fault injection campaigns. The process is split in two steps: golden (fault-free) simulation, which is used as a reference of correct simulation and faulty simulations, where the faults are injected.

A commercial Cortex®-A9 RTL model on a bare-metal environment was used in this study. A standard methodology is used to compute the *safeness* of the processor, which uses design signals as observation points. For this study, Yogitech s.p.a. exceptionally created an injection model of the L1 data cache that is typically considered protected by safety-industry and extended their workflow to also allow AVF computation.

B. Microarchitectural model

For the microarchitecture-level reliability assessment, we used the GeFIN fault-injection framework [13] [14], which is based on a state-of-the-art microarchitecture-level simulator (Gem5) [22]. The simulator was configured to resemble the micro-architecture of the ARM® Cortex®-A9 core as close as possible. TABLE I summarizes some major attributes of the core that were used in the microarchitecture-level configuration. Similarly to the RTL flow, GeFIN initially performs a fault-free simulation, which produces the golden outputs that are used for comparison against fault-injection simulations.

TABLE I: Microarchitectural configuration of Cortex®-A9

Microarchitectural attribute	***Value***
ISA / Core	ARMv7 / Out-of-order
Data cache	32KB 4-way
Instruction cache	32KB 4-way
Physical Register File	56 registers
Instruction queue	32
Reorder buffer	40
Fetch/Execute/Writeback width	2/4/4

C. Point to point comparison

Both tools assess the ARM® Cortex®-A9 core, using the same workloads. Due to proprietary reasons, the same binary files could not be used and the benchmarks were built using the same source files with the same options, using different tool chains. The two tools were configured on an equivalent setup in all possible details.

GeFIN was originally developed to cover full-system reliability assessment, which includes both the operating system and peripheral devices. On the contrary, the RTL injection flow uses a bare-metal setup that does not involve operating system. Since Gem5 supports similar capabilities

(referred as syscall-emulation mode), in which an application can be simulated excluding the operating system code, we modified GeFIN to utilize this capability and to perform a reliability assessment much closer to the RTL injection flow. Another key difference that had to be addressed was on the observation points of the two flows. RTL fault injection computes the *safeness* by comparing the signals of the CPU pinout (as best known method), while on the other hand GeFIN offers observation points between the system layers (offering HVF and AVF estimations). We modified GeFIN accordingly to monitor the equivalent core pinout of the Cortex®-A9 in order to offer identical observation points to the RTL flow.

All remaining simulation details (window of opportunity, timeouts, timers etc.) were set up identically in both tools to offer a fair comparison of the same CPU, with equivalent setup, same workloads and equivalent observation points.

D. Benchmarks

We have used a subset of MiBench suite [23] as the target workloads for this assessment. MiBench are widely used in reliability studies [1] [14] [11] [15] [16] [17] [21] as they combine a wide range of common workloads/algorithms with relatively small datasets, which effectively translate to small execution time. This combination makes them perfect candidates for fault injection reliability assessment since RTL simulation suffers from slow simulation throughput.

TABLE II summarizes the selected benchmarks along with their simulation throughput (seconds per run) and time (million clock cycles). The ratio column shows that GeFIN is up to 2 orders of magnitude faster compared to RTL simulator. Although the benchmarks were built using the same compilation flags and architecture, we can see how some of them report significant differences in execution time, which implies that the workloads are executed differently. Despite our efforts to reach an equivalent setup, there are cases that cannot be covered and this is a major limitation to be considered.

TABLE II: AVERAGE SIMULATION TROUGHPUT AND TIME FOR A FAULT INJECTION RUN ON EACH FFRAMEWORK

Benchmark	***Throughput***			***million clock cycles***	
	RTL	***GeFIN***	***Ratio***	***RTL***	***GeFIN***
FFT	7001 s/run	39.1 s/run	178.9	27 m	45 m
qsort	3157 s/run	23.9 s/run	131.8	10 m	12 m
cAES	413 s/run	30.7 s/run	30.7	1 m	22 m
sha	3421 s/run	8 s/run	427.2	10 m	11 m
stringsearch	60 s/run	2.8 s/run	21.39	0.2 m	0.2 m
susan corners	1019 s/run	3.2 s/run	315.5	3 m	1.5 m
susan edges	874 s/run	3.6 s/run	242.1	3 m	1.5 m
susan smooth	893 s/run	3.7 s/run	241.4	3 m	1.8 m
Average			198.6		

IV. RESULTS

We performed SFI campaigns on the physical *register file (RF)* and *L1 Data Cache* (L1D) of the Cortex®-A9 core, using the two simulation models: RTL and microarchitecture-level. A single transient fault (bit-flip) was injected per run, on a normal distribution. Using the formulation presented by Leveugle et al. [5], we calculate the fault population for an error margin of 2% and confidence level of 99%. The resulting statistical fault sample consists of 4000 injections for each benchmark and each component.

A. Fault effect classification

Both frameworks were configured to measure *Safeness* and AVF. We define the following fault effect classes:

Masked/Safe: The fault did not cause any deviations on the simulation. The fault was either masked or unused.

Unsafe: A mismatch was observed compared to the fault-free simulation.

We report the *vulnerability* of each workload as the ratio of unsafe cases to the number of the injected faults.

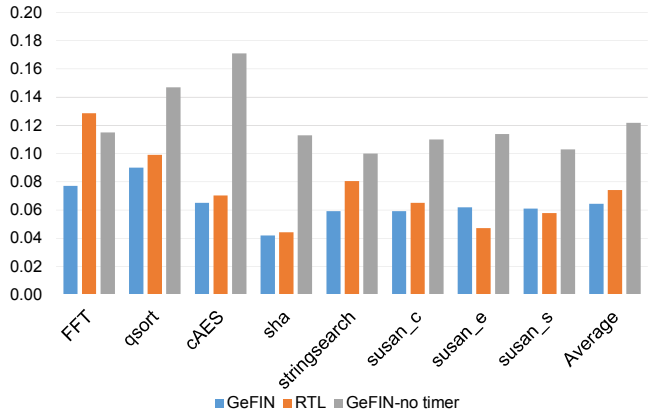

Fig. 1: Register File vulnerability (unsafeness)

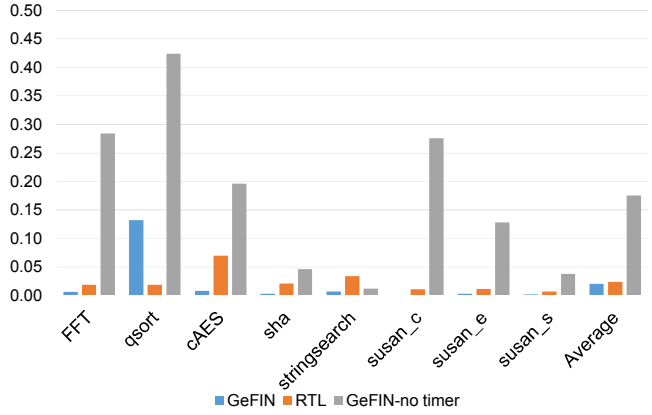

Fig. 2: L1D cache vulnerability (unsafeness)

B. Cortex®-A9 Safeness

Fig. 1 and Fig. 2 show the results for the RF and L1D cache respectively for the microarchitectural (blue bars) and the RTL level (orange bars) injections. Each run is terminated 20k cycles after the fault injection, since longer simulations are not feasible using RTL models. We have also included a GeFIN run until the end of the program (grey bars), to estimate the portion of mismatches that was not covered and to highlight the weakness of methodologies that use small timeout windows, which is typical for RTL. The results were obtained using the core pinout as observation point. In Fig. 1 we can see that, for the Register File, both frameworks report less than 10% different vulnerability in 5 benchmarks and a worst case of 60% for FFT. For L1D

cache, we can see much larger differences, which are caused by an optimization of the RTL framework, which moves the fault injection time closer to its consumption time. This increases the probability to observe the fault effect within the 20k time window. However, the window of 20k cycles (in both cases) is still very short to allow mismatch observation in the CPU core pinout (L1 write-back). The grey bar (run to-the-end) in Fig. 2 highlights that the methodology almost completely fails to capture the vulnerability of the component.

C. *Cortex®-A9 AVF*

In order to cover the limited capability of the core observation point mode for the L1D, we have extended the RTL framework to cover a software observation point (SOP). This directly monitors the program output for mismatches, which allows AVF calculation. Fig. 3 shows the results of L1D using SOP mode. Due to the limited throughput of the RTL flow, which is more than 2 orders of magnitude smaller than GeFIN (TABLE II), only the smaller benchmarks were analyzed; the longer benchmarks are impossible to be executed to their end using the RTL simulator. We can see that the results are again within the range of the Register File differences, with the exception of cAES benchmark, where RTL reports 2x higher vulnerability. It is not clear why the two models occasionally report significantly different results, but it should be pointed out that, due to the limitations on using the exact same binaries and toolchains, some workloads have significant differences, which does not allow further investigation around the source of error.

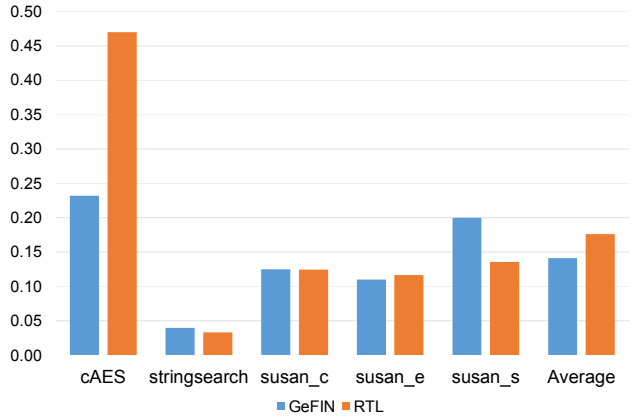

Fig. 3: L1D cache AVF using software observation point (AVF).

V. Conclusions

We have performed a statistical fault-injection-based reliability analysis on an ARM® Cortex®-A9 core, using an RTL and a microarchitecture-level model. We modified the two frameworks to resemble an equivalent setup. However, some of the differences that cannot be addressed still do not allow an exact point-to-point comparison, leaving open questions on how large is the accuracy loss when using a microarchitecture-level simulator. In our experiments, the average difference on the reported estimation is 10% for the register file (Fig. 1) and 20% for the L1 data cache (Fig. 3), which translates to 0.7 and 3 percentile points on the estimated vulnerability. This is significantly less than the accuracy loss of early stopping that is typically used in RTL models. Our comparison is limited to array-based structures, since the control logic is only functionally implemented on microarchitectural simulators. Due to the more than two orders of magnitude better throughput, micro-architecture level simulators appear to offer many advantages for assessing array-based structures, such as, the ability to run large workloads and the assessment of realistic full-system setups that also include the operating system. Both models have pros and cons that appear to be complementary to each other.

Acknowledgment

This work has been funded by the European Union through the CLERECO FP7 Project (Grant Agreement 611404), the UniServer H2020 Project (Grant Agreement 688540) and by a Cisco Research grant.

References

[1] J.F.Ziegler, et al., “IBM experiments in soft fails in computer electronics (1978 – 1994)”, IBM Journal of Research and Development,Volume 40, Number 1,pp. 3-18, 1996.
[2] C.Constantinescu, “Trends and challenges in VLSI circuit reliability”, IEEE Micro, vol. 23, pp. 14-19, July 2003.
[3] Y.Luo, et al.,“Characterizing application memory error vulnerability to optimize datacenter cost via heterogeneous-reliability memory”, DSN 2014.
[4] S.S.Mukherjee, C.T.Weaver, J.Emer, S.K.Reinhardt, and T.Austin, “A systematic methodology to compute the architectural vulnerability factors for high-performance microprocessors”, MICRO 2003.
[5] R.Leveugle, A.Calvez, P.Maistri, P.Vanhauwaert, “Statistical fault injection: quantified error and confidence”, DATE 2009.
[6] G.H.Asadi, V.Sridharan, M.Tahoori, D.Kaeli, “Balancing performance and reliability in the memory hierarchy”, ISPASS 2005.
[7] A.Vallero, et al., “Cross-Layer system reliability assessment framework for hardware faults”, ITC 2016.
[8] V.Sridharan, D.R.Kaeli, “Using hardware vulnerability factors to enhance AVF analysis”, ISCA 2010.
[9] V.Sridharan, D.R.Kaeli, “Eliminating microarchitectural dependency from architectural vulnerability”, HPCA 2009.
[10] S.Mitra, N.Seifert, M.Zhang, Q.Shi, K.S.Kim, “Robust system design with built-in soft-error resilience”, Computer, Volume 38, Issue 2, Feb. 2005 Page(s): 43 – 52.
[11] M.Ebrahimi, N.Sayed, M.Rashvand, M.B.Tahoori, “Fault injection acceleration by architectural importance sampling”, CODES 2015.
[12] R.Balasubramanian, K.Sankaralingam, "Understanding the impact of gate-level physical reliability effects on whole program execution", HPCA 2014.
[13] M.Kaliorakis, S.Tselonis, A.Chatzidimitriou, N.Foutris, D.Gizopoulos, “Differential fault injection on microarchitectural simulators”, IISWC 2015.
[14] A.Chatzidimitriou, D.Gizopoulos, “Anatomy of microarchitecture-level reliability assessment: throughput and accuracy”, ISPASS 2016.
[15] M.Ebrahimi, L.Chen, H.Asadi, M.B.Tahoori. “Class: combined logic and architectural soft error sensitivity analysis”, ASP-DAC 2013.
[16] S.Raasch, A.Biswas, J.Stephan, P.Racunas, J.Emer, "A fast and accurate analytical technique to compute the AVF of sequential bits in a processor", MICRO 2015.
[17] N.George, C.Elks, B.Johnson, J.Lach, “Transient fault models and AVF estimation revisited”, DSN 2010.
[18] N.Wang, A.Mahersi, S.J.Patel, “Examining ACE analysis reliability estimates using fault-injection”, ISCA 2007.
[19] H.Cho, S.Mirkhani, C.Y.Cher, J.A.Abraham, S.Mitra, "Quantitative evaluation of soft error injection techniques for robust system design", DAC 2013.
[20] X.Iturbe, B.Venu, E.Ozer. "Soft error vulnerability assessment of the real-time safety-related ARM Cortex-R5 CPU", DFTS 2016.
[21] G.S.Rodrigues, F.L. Kastensmidt, "Soft error analysis at sequential and parallel applications in ARM Cortex-A9 dual-core", LATS 2016.
[22] N.Binkert et al., “The Gem5 simulator”, ACM SIGARCH Computer Arch. News, vol. 39, no. 2, May 2011.
[23] M.R.Guthaus et al., “MiBench: A free, commercially representative embedded benchmark suite”, IWWC 2001.