UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

PABLO RAFAEL BODMANN

Reliability Assessment of Cores Integration and Operating System on Arm-Based Systems

Thesis presented in partial fulfillment of the requirements for the degree of Master of Computer Science

Advisor: Prof. Dr. Paolo Rech

Porto Alegre
March 2020

CIP — CATALOGING-IN-PUBLICATION

Bodmann, Pablo Rafael

Reliability Assessment of Cores Integration and Operating System on Arm-Based Systems / Pablo Rafael Bodmann. – Porto Alegre: PPGC da UFRGS, 2020.

59 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2020. Advisor: Paolo Rech.

1. Neutron Radiation Experiment. 2. Reliability. 3. ARM processor. 4. Fault Injection. I. Rech, Paolo. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Rui Vicente Oppermann
Vice-Reitora: Prof[a]. Jane Fraga Tutikian
Pró-Reitor de Pós-Graduação: Prof. Celso Giannetti Loureiro Chaves
Diretora do Instituto de Informática: Prof[a]. Carla Maria Dal Sasso Freitas
Coordenadora do PPGC: Prof[a]. Luciana Salete Buriol
Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

“Computers are like Old Testament gods; lots of rules and no mercy.”

— JOSEPH CAMPBELL

ACKNOWLEDGMENT

I would like to thanks my parents for all the support they gave me and for introducing me to the academic life. I also would like to thank my advisor Prof. Dr. Paolo Rech for patiently guiding me during this part of my academic life.

ABSTRACT

Reliability has become one of the main issues for computing devices employed in several domains. This concern only deepens with the increase of integration in the same chip of several peripherals and accelerators. To evaluate computational system reliability, fault injection and radiation experiments are used. Fault injection in microarchitectural models of the processor provides deep insights on faults propagation through the entire system stack, including the operating system. Beam experiments, on the other hand, estimate the device's expected soft error rate in realistic physical conditions by exposing it to the accelerated particle beam. Combining beam experiments and fault injection data can deliver deep insights about the device's expected reliability when deployed in the field. However, it is yet largely unclear if the fault injection error rates can be compared to those reported by beam experiments and how this comparison can lead to informed soft error protection decisions in early stages of the system design.

In this work, first, the data gathered with extensive beam experiments (on physical CPU hardware) and microarchitectural fault injections (on an equivalent CPU model on Gem5) performed with 13 different benchmarks executed on top of Linux on an ARM Cortex-A9 microprocessor are presented and analyzed. We then compare the soft error rate estimations that are based on neutron accelerated beam and fault injection experiments. We show that, for most benchmarks, fault injection can be very accurately used to predict the Silent Data Corruptions (SDCs) rate and the Application Crash rate. The System Crash rate measured with beam experiments, however, is much larger than the one estimated by fault injection due to unknown proprietary parts of the physical hardware platform that can't be modeled in the simulator. Overall, our analysis shows that the relative difference between the total error rates of the beam experiments and the fault injection experiments is limited within a narrow range of values and is always smaller than one order of magnitude. This narrow range of the expected failure rate of the CPU provides invaluable assistance to the designers in making effective soft error protection decisions in early design stages.

After that, the impact of cores integration and the OS interference on the reliability of Arm microprocessors is also analyzed and quantified. But in this analysis besides the same Arm Cortex-A9, as used in the previous analysis, a standalone Arm Cortex-A5 is also tested with both neutron beam and microarchitecture-level fault injections (on equiv-

alent CPU models of the A5 and A9 CPUs on Gem5 simulator). Correlating the beam experiments to the fault injection results it was found that due to the peripherals and interfaces, the integration of various cores significantly increases the System Crash rates but has a negligible impact on the SDC rate which is attributed to the CPU cores. Moreover, the OS has a beneficial impact on the Application Crashes but not on the System Crashes nor the SDC rates. The results of this second analysis firmly confirm, on two different CPU cores, the initial findings and speculations from the first analysis that the SDC part of the overall system failure rate is minimally affected by the SoC integration and the existence of the OS, while the Crashes parts are more severely affected by both aspects. The findings can be employed to support diligent design decisions for CPU cores error protection at the hardware or software level.

Keywords: Neutron Radiation Experiment. Reliability. ARM processor. Fault Injection.

Entendendo o Impacto de Integração de Núcleos e Sistema Operacional sobre a Confiabilidade em Sistemas Baseados em ARM

RESUMO

A confiabilidade se tornou um dos principais problemas em dispositivos de computação empregados em vários domínios. Essa preocupação apenas se aprofunda com o aumento da integração no mesmo chip de vários periféricos e aceleradores. Para avaliar a confiabilidade de um sistema computacional, são utilizados experimentos de injeção de falhas e de radiação. A injeção de falhas em modelos microarquiteturais do processador, por um lado, fornece informações detalhadas sobre a propagação de falhas em todo fluxo do sistema, incluindo o sistema operacional. Os experimentos com radiação, por outro lado, estimam a taxa de erro mais proximo de condições físicas realistas, expondo-o a fluxos acelerados de partículas. A combinação de experimentos de radiação e dados de injeção de falhas pode fornecer informações profundas sobre a confiabilidade esperada do dispositivo quando implantado em campo. No entanto, ainda não está claro se as taxas de erro de injeção de falha podem ser comparadas com as relatadas por experimentos com radiação e como essa comparação pode levar a decisões concientes sobre proteção de erros nos estágios iniciais do projeto de um sistema.

Neste trabalho, primeiro são apresentados e analisados, os dados coletados com extensos experimentos de radiação (no hardware físico da CPU) e injeções de falhas microarquiteturais (em um modelo de CPU equivalente no Gem5) realizadas com 13 benchmarks diferentes executados no Linux em um microprocessador ARM Cortex-A9. Em seguida, comparamos as estimativas de taxa de erro leve baseadas em experimentos de radiação de nêutrons e injeção de falhas. Mostramos que, para a maioria dos benchmarks, a injeção de falhas pode ser usada com muita precisão para prever a taxa de SDCs (Silent Data Corruptions) e a taxa de falha do aplicativo. A taxa de falha do sistema medida com experimentos de radiação, no entanto, é muito maior que a estimada por injeção de falha devido a partes proprietárias desconhecidas da plataforma de hardware físico que não podem ser modeladas no simulador. No geral, nossa análise mostra que a diferença relativa entre as taxas de erro total dos experimentos de radiação e as experiências de injeção de falha é limitada dentro de uma faixa estreita de valores e é sempre menor que uma ordem de magnitude. Esse intervalo estreito da taxa de falhas esperada da CPU fornece assistência inestimável aos projetistas na tomada de decisões eficazes de proteção contra erros de

software nos estágios iniciais do projeto.
Depois disso, o impacto da integração dos núcleos e a interferência do Sistema Operacional na confiabilidade dos microprocessadores Arm também são analisados e quantificados. Mas nessa segunda análise, além do mesmo Arm Cortex-A9 usado na análise anterior, um Arm Cortex-A5 também é testado com injeções de falha no nível de microarquitetura (em modelos de CPU equivalentes dos processadores A5 e A9 no Gem5 simulador) e na radiação de nêutrons. Correlacionando os experimentos de radiação com os resultados da injeção de falhas, verificou-se que, devido aos periféricos e outras interfaces, a integração aumenta significativamente as taxas de falha do sistema, mas tem um impacto insignificante na taxa de SDC atribuída aos núcleos da CPU. Além disso, o sistema operacional tem um impacto benéfico nos travamentos de aplicativos, mas não nos travamentos do sistema nem nas taxas de SDC. Os resultados desta segunda análise confirmam firmemente, em dois núcleos diferentes de CPU, as descobertas e especulações iniciais da primeira análise de que a parte SDC da taxa geral de falhas do sistema é minimamente afetada pela integração do SoC e pela existência do sistema operacional, enquanto os Crashes são mais severamente afetadas por ambos os aspectos. Ambas descobertas podem ser empregadas para apoiar decisões de projeto com o objetivo minimizar a taxa de erros tanto no nível de hardware quanto no de software.

Palavras-chave: Experimentos de radiação de neutrôns, Confiabilidade, Processadores ARM, Injeção de Falhas.

LIST OF ABBREVIATIONS AND ACRONYMS

ACE Architecturally Correct Execution

AES Advanced Encryption Standard

APSoC All-Programmable SoC

AVF Architecture Vulnerability Factor

CNN Convolutional Neural Networks

CPU Central processing Unit

CRC Cyclic Redundancy Check

ECC Error Correcting Code

FFT Fast Fourier Transform

FIT Failures-in-time

GPU Graphics Processing Unit

HPC High-performance computing

LANL Los Alamos National Laboratory

MMU Memory Management Unit

OS Operating System

RAL Rutherford Appleton Laboratory

RISC Reduced Instruction Set Computer

RTL Register-transfer level

SDC Silent Data Corruption

SECDED Single Error Correction Double Error Detection

SET Single Event Transient

SEU Single Event Upset

SoC System-on-Chip

SRAM Static Random-access memory

TLB	Translation Lookaside Buffer
UAVs	Unmanned Aerial Vehicles

LIST OF FIGURES

Figure 4.1 Neutron accelerated beam test setup at ChipIR. 27
Figure 4.2 Spectrum of neutron energy produced at ChipIR compared to the atmospheric and Los Alamos Nuclear Science Center (LANSCE) one. 28

Figure 5.1 neutron accelerated beam experiments FIT rates for SDCs, Application Crashes and System Crashes. 33
Figure 5.2 Fault injection effects classification for all 13 benchmarks in all 6 components. Effects are Masked, AppCrash, SysCrash, and SDC. The AVF corresponds to the sum of the three non-Masked cases. 33
Figure 5.3 Fault Injection FIT rates for SDCs, Application Crashes and System Crashes. 36
Figure 5.4 Multiplication Factor for SDC FIT comparison between neutron accelerated beam experiments and fault injection. 37
Figure 5.5 Multiplication Factor for Application Crash FIT comparison between neutron accelerated beam experiments and fault injection. 38
Figure 5.6 Multiplication Factor for System crash FIT comparison between neutron accelerated beam experiments and fault injection. 39
Figure 5.7 SDC and Application Crash Comparison between neutron accelerated beam experiments and fault injection. 40
Figure 5.8 Cortex A9 Linux neutron accelerated beam FIT rates for SDCs, Application Crashes and System Crashes. 42
Figure 5.9 Cortex A5 Bare-metal and Linux GeFIN FIT rates for SDCs, Application Crashes and System Crashes. 43
Figure 5.10 Cortex A5 Bare-metal and Linux neutron accelerated beam FIT rates for SDCs, Application Crashes and System Crashes. 44
Figure 5.11 Cortex A5 and A9 Bare-metal FIT rates estimated through GeFIN using the experimentally measured technology sensitivity. 45
Figure 5.12 SDC FIT comparison between the stand-alone Cortex-A5 and the integrated Cortex-A9. Susan c, e, and s could not be tested in the A5 for lack of neutron accelerated beam time. 47
Figure 5.13 Application and System Crash FIT comparison between the stand-alone Cortex-A5 and the integrated Cortex-A9. Susan c, e, and s could not be tested in the A5 for lack of neutron accelerated beam time. 48
Figure 5.14 FIT estimation of FFT across all configurations. 49
Figure 5.15 FIT estimation of qsort across all configurations. 50
Figure 5.16 Overall impact of cores integration and operating system on the CPU error rate as resulting from the data. On the average, the cores integration barely increases the SDC rates (about 3.5x) but significantly increases the Crash rate (174x, combining the increase in App crashes and Sys crashes. The operating system does not have a significant impact on SDCs (about 2.5x) and increases crashes of about 7.5x. 51

LIST OF TABLES

Table 2.1 Performance of different abstraction layer models (CHO et al., 2013; CHATZIDIMITRIOU et al., 2017a). 22

Table 4.1 Summary of setup attributes. 25

CONTENTS

1 INTRODUCTION **14**
1.1 Work Proposal **17**
1.2 Work Structure **17**
2 BACKGROUND **18**
2.1 ARM architecture **18**
2.2 Faults, Errors and Failures Concepts **18**
2.3 Radiation Effects in Electronic Devices on Ground Level **19**
2.4 Effects of radiation-induced errors on processors **19**
2.5 Functional Safety **20**
2.6 Reliability Evaluation Methodologies **21**
3 RELATED WORKS **23**
4 METHODOLOGY **25**
4.1 Neutron Beam Experiments **27**
4.2 Fault Injection **29**
4.3 Fault Injection vs. Neutron accelerated beam Experiment Setup **31**
5 RESULTS **32**
5.1 Neutron accelerated beam experiments vs fault injection: Linux on ARM A9 32
5.1.1 Neutron accelerated beam Experiments Results 32
5.1.2 Fault injection results using GeFIN 35
5.1.3 Discussion of the comparison between Fault Injection and Neutron Accelerated Beam Experiments Data on the A9 36
5.2 Operating System and Integration Impact: ARM A9 and ARM A5 .... **41**
5.2.1 Neutron accelerated beam Experiments Results 41
5.2.2 Fault injection results using GeFIN 43
5.2.3 Operating System impact 44
5.2.4 Integration Impact 44
5.2.5 Discussion on Operating System and Integration Impact 46
6 CONCLUSION **52**
REFERENCES **54**
APPENDIX A — PAPERS PUBLISHED AND SUBMITTED **59**
A.1 Paper published **59**
A.2 Paper submitted **59**

1 INTRODUCTION

Reliability has become one of the main concerns for computing devices employed in several domains, from High Performance Computing (HPC) to automotive, military, and aerospace applications (LUCAS, 2014; DONGARRA; MEUER; STROHMAIER, 2015; COHEN et al., 2018). Reliability has been identified by the U.S. Department of Energy (DOE) as one of the ten major challenges for exascale performance computing (LUCAS, 2014). In fact, a lack of understanding or the underestimation of devices and applications error rate may lead to lower scientific productivity of large scale HPC servers, resulting in significant monetary loss (SNIR et al., 2014). When the computing device is integrated in cyber-physical systems such as cars, airplanes, or Unmanned Aerial Vehicles (UAVs), high reliability becomes mandatory and unexpected errors should be strictly avoided (DONGARRA; MEUER; STROHMAIER, 2015; COHEN et al., 2018).

For example, the Titan super computer at Oak Ridge National Laboratory, which is composed of 18000 Graphics Graphics processing units (GPU), registered during 3.2 years 18000 errors. This means 14 errors per day. Another example, is the Jaguar super computer also located at Oak Ridge National Laboratory. With its 360 terabytes of main memory, it was measured 350 ECC error per minute.

This problem is also seen in self-driving cars, a emerging trend in the automotive market. One example would be the self-driving car from Google which, when adding all journeys made, traverse a distance of $1.5 * 10^6$ miles. During all these trips, it was detected 1 system error. Assuming a speed of 25 mph, this results in 60000 hours driven. This results in an error rate of 16000 errors per 10^9 hours of operation. However, if the cars speed would have been 50 mph, this would result in 30000 hours of operation resulting in an error rate of 23000 errors per 10^9 hours of operation.

Additionally, reliability concerns have become more crucial as highly integrated System on Chips (SoCs) move from consumer applications to the automotive, military, aerospace, and HPC markets as the the complexity of these systems increase and the transistor size keeps shrinking. The trend in the design of computing devices is to include multiple, heterogeneous, cores in the same chip. This allows flexibility and more computational power.

Modern SoCs usually integrate a central processing unit (CPU) and an accelerator, such as Graphic Processing Units (GPUs) or a Field Programmable Gate Array (FPGA). These SoCs typically allow the computing cores to share a common memory,

which significantly improves the performance and reduce the total power consumption by removing costly device-to-device data transfers. The improved efficiency of heterogeneous SoCs makes them widely common in portable devices such as smartphones, tablets, and laptops. Lately, embedded SoCs are become extremely attractive for autonomous vehicles and High Performance Computing. Arm announced the interest in entering the self-driving market based on the increased performance, improved efficiency, and considerable computing power, which is necessary to detect objects in real-time. Moreover, the US Department of Energy's (DOE's) labs at Sandia and Los Alamos, in NM, announced that their next clusters are going to be powered by Arm SoCs (HEMSOTH, 2018).

To be useful and effective any method for the chip's reliability evaluation must be both fine grain (i.e., providing full visibility and understanding of faults effect in the microarchitecture and the software stacks) and realistic. A *fine grain* evaluation is employed to capture a clear understanding of the causes and effects of faults, to identify the most vulnerable parts of the hardware or software, and to observe how raw bit flips propagate through the system stack. A *realistic* evaluation ensures a correct prediction of the expected error rate of the device when used in the field and provides realistic models of the faulty behavior. The fine grain and realistic evaluation of the reliability of Commercial Off-The-Shelf (COTS) devices, that became increasingly common in both safety-critical and HPC applications, is challenging as information about the architecture and device characteristics is typically very limited.

Neutron accelerated beam experiments are the most realistic way to measure the error rate of a code or device in conditions that are as close as possible to the actual ones after system deployment. However, neutron accelerated beam experiments are coarse grain, i.e., errors can be observed only when they manifest at the application output or when they compromise the system responsiveness. There is no information about the spatial/temporal location of the original fault and no information on its propagation pattern through the microarchitecture, the system software or the application software layers. It is, therefore, hard to identify the most vulnerable hardware resources or software code portions and to extend/generalize results to other applications or devices when relying on neutron accelerated beam experiments only.

Fault injection on models of a microprocessor at different detail levels (architecture, microarchitecture, register-transfer, transistor) has been extensively used to evaluate the vulnerability of hardware architectures and software codes. By injecting faults in the hardware blocks of a microprocessor or particular parts of a software code it is possi-

ble to measure the probability for faults to impact the application output or the system responsiveness, i.e., the Architectural Vulnerability Factor, AVF (MUKHERJEE et al., 2003). Fault injection can then identify codes, code portions, or architectural resources which are more likely, once corrupted, to affect the system reliability. This information is extremely useful to design dedicated and efficient hardening solutions. It has been shown that fault injection at the microarchitecture level is a very effective (fast and accurate method) for early assessment of the reliability of a microprocessor (CHATZIDIMITRIOU; GIZOPOULOS, 2016; KALIORAKIS et al., 2017). However, faults can typically be injected in a limited set of resources and, due to time constraints, simplified errors models (such as single bit flip) are usually adopted. In other words, if fault injection is not validated or tuned with physical experimental data, the efforts could potentially lead to imprecise results.

On one side a device with more integrated accelerator can have more functionalities, execute codes with higher speed and efficiency, however, this may result in loss of reliability. Although the reliability of standalone CPUs and FPGAs has been extensively studied (WANG; LIU; SUN, 2018; Vallero; Carelli; Di Carlo, 2018; Chatzidimitriou et al., 2019a) in previous works, which have also addressed the reliability of GPUs for HPC and automotive applications (Wilkening et al., 2018; Vallero et al., 2018; Santos et al., 2019; OLIVEIRA et al., 2017b), few works evaluate the impact of integration of a computing core in a SoC on reliability.

The increased hardware complexity required to integrate several resources and the increased software complexity necessary to manage these resources makes the reliability evaluation of embedded CPU extremely challenging. This setup suffers failures (either Silent Data Corruptions, SDCs or Crashes) that can only be caused by soft errors affecting the CPU core while running user instructions. This integration of the SoC can potentially modify the SDC and Crash FIT rates of the system.

Additionally, in order to use the full advantage of the resources available in an SoC, the use of an Operating System (OS) becomes essential. The OS is used to orchestrate the execution of multiple tasks, to manage resources utilization, and to allow codes portability. When using an OS, besides the main application running, there are also other applications running responsible for the OS management. This might influence the system reliability. Although previous works have evaluated the effect of the operating system on a code reliability and even designed neutron accelerated beam hardened OS (Iyer; Rossetti, 1985; Santini et al., 2015; SANTINI et al., 2016; SANTINI et al., 2017), it is still

unclear which are the architectural vulnerabilities that induce errors in operating systems.

1.1 Work Proposal

This work aims to refine and confirm initial findings and speculation about the sources of failures manifesting as SDCs and as Crashes as well as their magnitude variation when SoC integration and OS inclusion are employed.

More specifically, the main contributions of this work are:

- Compare fault injection and neutron accelerated beam experiments in order to evaluate at which level fault injection can be used to emulate neutron accelerated beam experiment. ("How much fault injection SDC and Crashes differ from reality?")
- An experimental evaluation of the impact of cores integration and Operating System on Arm microprocessors reliability, based on neutron accelerated beam experiments ("how much do SDC and Crashes differ?").
- A microarchitecture-level fault injection analysis of the vulnerability of programs executed on top of the Linux kernel and also bare-metal in Arm microprocessors ("how much do SDC and Crashes differ in this setup as well?").
- Hints and guidelines on how to estimate the reliability of a microprocessor when it is integrated on a SoC or when executing programs on top of an OS. ("how much does SDC and Crash rates change when SoC and OS integration is employed?").

1.2 Work Structure

The rest of this work is organized as follows: Chapter 2 presents some important definition and concepts used in this work. Chapter 3 presents some works already done in the reliability field. Chapter 4 shows the methodology used for this work. In Chapter 5, the results are presented and discussed. First, two methodologies, fault injection and neutron accelerated beam experiments, are compared and analysed. After, the results comparing the Operating System and integration impact are presented and discussed. Finallly, conclusions are presented in Chapter 6.

2 BACKGROUND

In this section, the main concepts of radiation effects and reliability, as the metrics used in the reliability evaluation are presented.

2.1 ARM architecture

ARM is one of the most used architectures on embedded systems. It started focusing as a low-power processor architecture. Most ARM processors design seek a Reduced Instruction Set Computer (RISC) architecture aiming to get the best clock cycle performance by simplifying the computing architecture. Today there are 3 main families of ARM processors, design for different applications, Cortex-A, Cortex-R and Cortex-M.

The Cortex-A family is a group of 32-bit and 64-bit processor cores, intended for application use. It includes several features found in general-purpose processors such as Memory Management Unit (MMU).

The Cortex-M is a group of 32-bit RISC ARM processor cores, intended for microcontroller use. The main difference between the M family to the A is that there is no FPU unit but on latter cores it can be added but it is optional. Since this family of cores is focused on microcontrollers there is no MMU unit, therefore, it cannot run a full general-purpose OS.

The Cortex-R is a family of ARM cores designed for high performance hard real-time and safety-critical applications. It is similar to the A profile for application processing but adds features that make it more fault-tolerant and suitable for use in hard real-time and safety-critical applications. Some of these features include non-overlapping memory regions, ECC on L1 cache and buses and Dual-core lockstep for CPU fault tolerance.

2.2 Faults, Errors and Failures Concepts

In the literature faults errors and failures have similar but with small differences in their definition. In order to establish a standard in this work, the definition used for these 3 concepts is described in (Avizienis et al., 2004). The definition of failure states that, if the system deviates from the correct function which it was designed. The failure is caused by an error, which is the state that deviates from the correct state of a system.

The cause of an error is called a fault. Faults can be classified as permanent, transient or intermittent. Permanent faults are ones that are continuous in time, transient faults are the ones which are temporary and occur randomly. Intermittent faults are temporary faults which happens repetitively. A fault can be active if it causes and error otherwise it is called dormant.

2.3 Radiation Effects in Electronic Devices on Ground Level

When a galactic cosmic ray interacts with the terrestrial atmosphere, it triggers a chain reaction that generates a flux of particles (mainly neutrons) that reach ground. About $13\ neutrons/((cm^2) \times h)$ reach ground (JEDEC, 2006). A neutron strike may perturb a transistor's state, generating bit-flips in memory or current spikes in logic circuits that, if latched, lead to an error (MAHATME et al., 2011). A transient error can have no effect on the program output (i.e., the fault is masked, or the corrupted data is not used) or propagate through the abstraction stack of the system leading to a Silent Data Corruption (SDC), or unrecoverable behaviors such as a program or system crashes.

The error rate of an application running on a computing device depends on both the memory/logic sensitivity (BAUMANN, 2005; NOH et al., 2015) and the probabilities for the fault to propagate through the architecture or program (MUKHERJEE et al., 2003; SRIDHARAN; KAELI, 2010). Hardening solutions can be applied at different levels of abstraction (from transistor level to software or system level) to reduce the probability of the fault occurrence or to avoid fault propagation. One of the most common and effective strategies to protect memory is the Single Error Correction Double Error Detection (SECDED) Error Correcting Code (ECC). However, many user-oriented embedded devices, such as ARM Cortex A5 and A9 do not implement ECC in their memory.

2.4 Effects of radiation-induced errors on processors

As discussed in the previous section, the radiation may disturb the transistor state causing a transient pulse. This pulse is called Single Event Transient (SET). If a SET happens in a memory cell it may result in a bit-flip, a Single Event Upset (SEU). If more than one bit is flipped in the same memory word, this event is defined as Multiple Bit Upset (MBU). Depending where a SET happens, it may lead to SDCs or crashes. If a

SET causes a SEU or a MBU in in the register file or the caches this may lead either to an SDC or to a Crash. If a byte containing data is modified this may lead to a SDC but if a control variable is modified this will lead to a Crash or in rare cases to an SDC if the modified value is a valid one in the context of the execution. In this case a crash maybe an explicit error detected by the CPU or and infinite loop, where the exit condition is never reached. The same behavior is expected on the data and instruction caches, or the data portion on the upper-level caches. In the specific case of the instruction caches, a SET can either cause an invalid instruction to executed leading to a crash or an execution of a different but valid instruction leading to a SDC. (OLIVEIRA, 2017). A SET on the control logic may lead to a crash or a SDC and due to the computation being wrongly done or the synchronization maybe. Also a SET in the ALU or in the FPU, may cause a SDC or a crash, due the computation being wrongly done.

2.5 Functional Safety

A safety-critical application is defined as an application whose failure may result in loss of life, property damage, or environmental damage (Knight, 2002). Some embedded systems are used in safety-critical systems, so they have specific dependability constraints. The main properties for embedded systems used in safety-critical applications are reliability, maintainability, availability, safety, and security. Reliability is the probability of a fail on the system. Maintainability is the probability when a failure occurs, it could be repaired on a time interval. Availability is the probability that the system will be available to service. Safety is the probability that the system will not cause any harm or property damage. Security is the probability if the system has confidential data it will keep this data confidential, and authentic communication is guaranteed (MARWEDEL, 2010).

It is worth noting that in some specific applications embedded systems are required to respect all the mentioned requirements. Self-driving cars, civilian and military aerospace sectors are examples in which the system must be reliable while manipulating a significant amount of data in real-time and be extremely energy efficient. In this scenario, design an embedded system that fills all requirements is not an easy task, due to the resource limitation, essentially on energy and performance, and the inhospitably of the operating environment which some times is not possible to emulate realistically.

2.6 Reliability Evaluation Methodologies

The evaluation of the error rate of a device is essential to understand if the device meets the project's reliability requirement. Additionally, an early pre-silicon prediction of the device error rate is useful to evaluate if the reliability needs to be improved and to identify possible design vulnerabilities.

Realistic error rates can be measured exposing the real hardware to controlled particles beams. By exposing the device to particle beams, it is possible to mimic the effect of natural radiation on the final system in the field. Thanks to the high particles flux intensity, a statistically significant amount of data is gathered in a short time. Moreover, the same kind of faults that would impact the device in its application in the field are injected in all physical hardware resources with realistic probabilities. Unfortunately, radiation experiments offer limited visibility of fault propagation as faults are observed only when they compromise the system functionality (corrupting the output or crashing the application/system). With radiation experiments it is then very hard to correlate the observed effects with their causes, limiting the identification of the system most vulnerable parts. Additionally, radiation experiments can obviously be performed only on real hardware, after the device project has been finalized.

Designs that need to comply with certain dependability constraints require decisions to improve the reliability of the system but without adding unnecessary overhead. Reliability evaluation is intended also to support such decisions. It is critical to have this analysis in time and as early as possible, since any additional re-design iteration can lead to catastrophic costs. As a result, early-reliability assessment is often performed in models that exist prior to silicon prototypes, which can be summarized as architecture level, microarchitecture level and Register-transfer Level (RTL). These vary in level of detail, with the most abstract being available earlier in the design chain while the most detailed (RTL) being available at the later stages. Architecture-level models often lack most, if not all, of the hardware details of the system, offering a software level functional emulation, while microarchitecture-level includes most functional and timing-accurate models of the microarchitecture (pipeline, cache memories etc.), offering clock cycle accuracy. In addition, most memory elements of the system (including (Static Random-access Memories (SRAMs), pipeline registers and flops/latches not related to logic; e.g. state machines) are accurately modeled in microarchitecture level. RTL offers a full description of the implemented hardware, including the logic and SRAM components. Simulation time for

Table 2.1: Performance of different abstraction layer models (CHO et al., 2013; CHATZIDIMITRIOU et al., 2017a).

Abstraction Layer	Model	Performance (Cycles/sec)
Software (native)	Modern processors	2×10^9
Architecture	Gem5 atomic model	2×10^7
Microarchitecture	Gem5 detailed out-of-order model	2×10^5
RTL	NCSIM simulation	6×10^2

each abstraction layer is proportional to the level of detail, with each detail step adding approximately 2 orders of magnitude more simulation time; the most detailed RTL model ends up being extremely slow. Table 2.1 illustrates the simulation throughput of each abstraction level.

Different reliability evaluation techniques can be applied on top of each model, delivering different levels of detail along with throughput. Probabilistic and statistical models (ASADI et al., 2005; SUH; ANNAVARAM; DUBOIS, 2012) often require a single simulation to deliver a rough estimation of the reliability, based on simulation statistics. Architecturally Correct Execution (ACE) analysis (FU; LI; FORTES, ; MUKHERJEE et al., 2003; WANG; MAHESRI; PATEL, 2007; GEORGE et al., 2010) on the other hand tries to capture more details on the residency and lifetime of workload critical data on each vulnerable component of the system, and weight their vulnerability against their sensitive exposure time. ACE analysis often requires one or a few simulations to quantify the vulnerability, but also requires additional development effort in order to capture all of the system's complexity. ACE analysis is claimed to have adjustable accuracy (proportional to the effort) but the tradeoff between effort and speed should always lean towards speed, otherwise more straight-forward approaches can be used (BISWAS et al., 2008). Statistical fault-injection is one of the most widely adopted approaches of reliability assessment. It offers the flexibility of variable accuracy (depending on the size of statistical sample) while at the same time, it delivers failure samples produced by simulation. On the drawback side, the requirement of multiple simulations requires significant amount of time and, depending on the model detail, it can often be considered as unfeasible. Therefore, one of the objectives of this work is to compare the FIT rate which fault injection estimated using neutron accelerated beam experiment, giving an estimation of how far is the former from reality.

3 RELATED WORKS

Particle accelerators have been used for many years to measure and study the reliability of devices and applications (BAUMANN, 2005; ZIEGLER; PUCHNER, 2010). Computing devices reliability has a strong tradition, motivated mainly by their use in safety-critical applications (SEIFERT; ZHU; MASSENGILL, 2002; NGUYEN et al., 2005; CONSTANTINESCU, 2002).

Arm processors have been exposed to accelerated particles beam and have been subjects to fault injection in previous studies. In (SANTINI et al., 2016), (OLIVEIRA; RODRIGUES; KASTENSMIDT, 2017), (MARTÍNEZ-ÁLVAREZ et al., 2016), and (FRATIN et al., 2018). authors present neutron accelerated 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. (RODRIGUES; KASTENSMIDT, 2016) and (ROSA et al., 2015) present results on architecture-level fault injection of the processor core, while (CHATZIDIMITRIOU et al., 2017b) includes a microarchitecture-level fault injection on A9. (CHATZIDIMITRIOU et al., 2017a) presents a comparative reliability evaluation between microarchitecture and RTL fault injection, for baremetal workloads running on Cortex-A9, while (ITURBE; VENU; OZER, 2016; BLOME et al., 2005) also includes results of RTL fault injection on Arm CPU cores. Apart from Arm processors, fault injection on RTL was also used in (CHO et al., 2013; MANIATAKOS et al., 2011; WANG; MAHESRI; PATEL, 2007). Some preliminary studies have proposed a comparison or combination of different reliability evaluation techniques (WANG; MAHESRI; PATEL, 2007; GEORGE et al., 2010; CHO et al., 2013; FRATIN et al., 2018; Santos et al., 2019; OLIVEIRA et al., 2017a). In (Chatzidimitriou et al., 2019b), a first attempt to compare the reliability evaluation of Cortex-A9 using neutron accelerated beam experiment and microarchitectural fault injection was made. According to authors, for SDCs the comparison can be very close, for Crashes the difference is significant. Some work has been done to evaluate the influence of an operating system on the reliability of codes executions (Iyer; Rossetti, 1985; Santini et al., 2015; SANTINI et al., 2016; SANTINI et al., 2017). 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 works do not perform fault injection, as it is done in this work, but are just based on neutron accelerated beam experiments, which might limit the insights that can be gathered.

Finally, none of these works address the impact of the integration of heterogeneous components in a single chip in the reliability of the device or of a code. This is one of the first works that uses both neutron accelerated beam experiments and microarchitectural fault injection to better understand how the operating system and the cores integration affect the reliability of a processor.

4 METHODOLOGY

This chapter aims to present the experimental setup for the neutron accelerated beam test and fault injection campaigns. First, it is described the Arm CPUs (the Cortex A5 and the Cortex A9) that were used, the operating system version running on the selected processors, and the benchmarks that were executed.

The study is performed on an Arm®Cortex™-A5 implemented in the Microchip SAMA5D2 XPLAINED ULTRA board and on the Arm®Cortex™-A9 that is embedded in a Xilinx Zynq™-7000 AP System on Chip (SoC). Both the A5 and A9 CPU configurations were resembled and simulated in Gem5 using the in-order and out-of-order CPU cores, respectively (details in Section 4.2). The Zynq device has two Arm cores operating at a maximum frequency of 667 MHz and the SAMA5D2 device has a single core operating at a maximum frequency of 500MHz. 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. The Gem5 model was tuned to resemble the physically available one and the second core of the SoC was disabled in order to make the two evaluation setups as close as possible. The A9 is implemented with with 28nm CMOS technology and the A5 is implemented with 65nm.

The two configurations were evaluated in both bare-metal and full-system environments. The Linux kernel versions that were used on the neutron accelerated beam setups is 3.14 for the A9 and 4.14 for the A5 while for Gem5, the kernel version used for both devices is 3.13. These were the closest kernel versions that have been ported on the two platforms. Despite the different versions between A9 and A5, all executables were compiled with the same compiler and static linked. This means that all the functions used during computation are exactly the same between the A5 and A9. 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. Table 4.1 presents the main characteristics of the two setups.

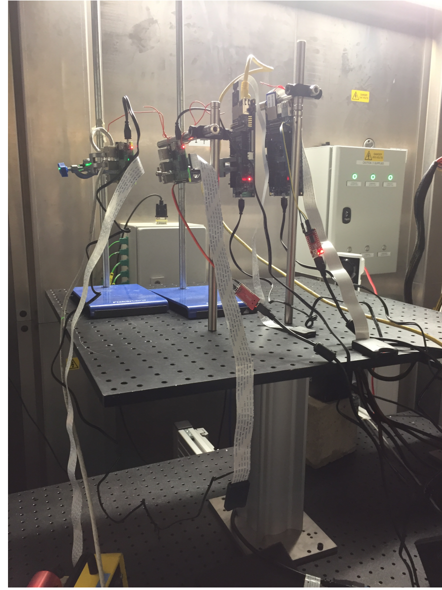
Table 4.1: Summary of setup attributes.

Property	Cortex-A5		Cortex-A9	
Setup	Neutron accelerated beam	Gem5	Neutron accelerated beam	Gem5
Platform	SAMA5D2	VExpress	Zynq 7000	VExpress
CPU cores	1	1	1*	1
L1 Cache	32 KB 4-way	32 KB 4-way	32 KB 4-way	32 KB 4-way
L2 Cache	128 KB 8-way	128 KB 8-way	512 KB 8-way	512 KB 8-way
Kernel version	4.14	3.13	3.14	3.13

To have a broad analysis and avoid the bias of results on specific applications, the chosen benchmarks have different computational characteristics. The applications chosen are part of *mibench* (GUTHAUS et al., 2001) testbench and are listed below.

- CRC32: It takes a file as input and calculate the corresponding Cyclic Redundancy check (CRC) with length of 32 bits. CRC is widely used in networks and storage devices in order to detect unwanted changes in the data. It calculates the corresponding 32-bit Cyclic Redundancy check (CRC) of a given file input. CRC is widely used in networks and storage devices in order to detect unwanted changes in the data.
- Dijkstra: This benchmark calculates the shortest path between 2 nodes using a adjacency matrix. 100 paths are calculated during each execution.
- FFT: It performs the Fast Fourier Transform (FFT). The FFT is widely used in digital signal processing.
- Jpeg C (Encode) and Jpeg D (Decode): This benchmark converts one PPM image to JPEG format (JPEG C) and from JPEG to PPM (JPEG_D). One input file is a PPM image with size of 512X512 pixels (JPEG C), the other one is a JPEG image (JPEG D). These benchmarks convert one PPM image to JPEG format (Jpeg C) and vice versa (Jpeg D).
- MatMul and MatMul_Big: It multiplies two matrices. This algorithm is used in image processing and Convolutional Neural Networks (CNN).
- Qsort: It sorts an array using the quick-sort algorithm implemented in the GNU C standard library. This algorithm was chosen in order to represent data sorting operations.
- Rijndael E (Encryption) and Rijndael D (Decryption): These two benchmarks use the Rijndael algorithm as defined in the Advanced Encryption Standard (AES). One encrypts an input file (E) and the other decrypts an encrypted input file (D).
- StringSearch: It searches a word in a sentence.
- Susan C: It uses the corner SUSAN algorithm in order to find the corners of the features. This algorithm is used to detect corners in features in an image.
- Susan E: It uses the edge SUSAN algorithm in order to find the edges of the features.
- Susan S: It uses the SUSAN algorithm in order to remove noise and preserve the image structure.

Figure 4.1: Neutron accelerated beam test setup at ChipIR.



Source: The author

It is worth to mention that the exact same input vector, i.e., the same values and same size for each corresponding benchmark were used in both neutron accelerated beam experiments and fault injection campaigns. Further discussion is presented in Section 4.3.

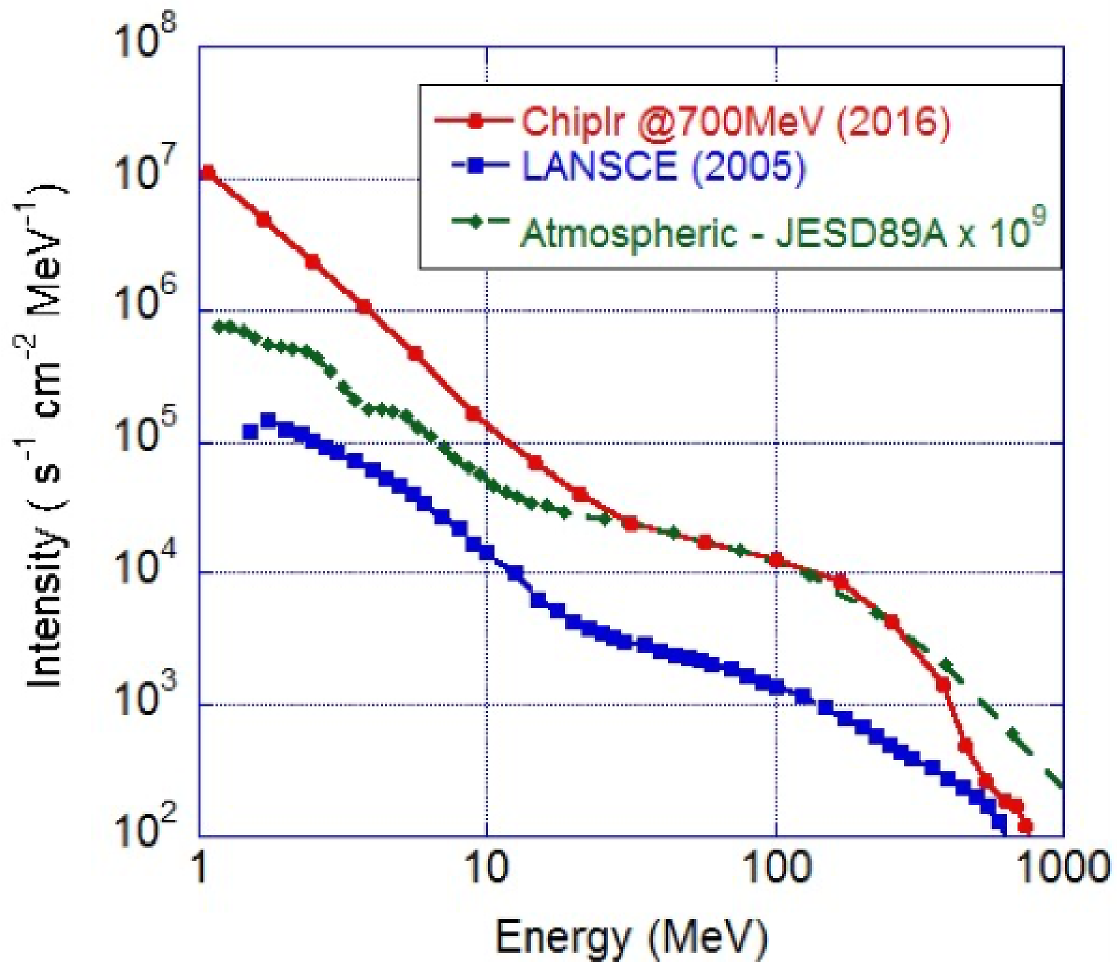
4.1 Neutron Beam Experiments

The neutron accelerated beam experiments were performed at the ChipIR facility of the Rutherford Appleton Laboratory (RAL) in Didcot, UK and at Los Alamos National Laboratory (LANL). As faults are induced with realistic probabilities and the whole chip is irradiated, faults are not restricted to a subset of accessible resources like for most software fault-injection frameworks. Neutron beam experiments are then a precise way to predict devices and applications error rates but can give little insights on the origin of faults, as errors are observed only when they appear at the output.

Figure 4.1 shows part of the setup at ChipIR. For the experiment two Xilinx Zed-boards and three Microchip boards with a 3×3 cm beam spot were irradiated, which is sufficient to irradiate the chip uniformly.

As shown in Figure 4.2, the spectrum of energy of neutrons produced at ChipIR is very similar to the atmospheric one as measured in (JEDEC, 2006). This means that the kind of neutrons produced at ChipIR (or LANL) is very similar to the neutrons that hit computing devices on realist applications. ChipIR, then, provides a neutron beam suitable

Figure 4.2: Spectrum of neutron energy produced at ChipIR compared to the atmospheric and Los Alamos Nuclear Science Center (LANSCE) one.



Source: (CAZZANIGA; FROST, 2018)

to mimic the atmospheric neutron effects in electronic devices (CAZZANIGA; FROST, 2018).

The available neutron flux was about $3.5 \times 10^6 n/(cm^2/s)$, about 6 orders of magnitude higher than the terrestrial flux ($13n/(cm^2 \times h)$ at sea level (JEDEC, 2006)). Since the terrestrial neutron flux is low, in a realistic application it is highly unlikely to see more than a single corruption during program execution. Experimental data, then, can be scaled to the natural radioactive environment without introducing artifacts.

A beam spot of 3cm of diameter was chosen, which is sufficient to uniformly irradiate the SoC without affecting the main memory or other on-board peripherals. This means that data in the DDR is not exposed to the neutron accelerated beam and is not corruptible. In the unlikely event of a fault in the DDR, this would be detected in the setup as it would affect multiple subsequent executions.

With the experiments, as the spectrum of neutrons energy at ChipIR resembles

the terrestrial one, the **Failures In Time (FIT)** rate of the device executing a code (failures per 10^9 hours of operation) can be measured. FIT depends only on the kind and amount of resources required for computation, without considering the code execution time (BAUMANN, 2005).

During the experiments, the Arm output is compared to a golden reference which contains the expected output (pre-computed in a fault-free environment). Any mismatch between the experimental and expected output is marked as an **SDC** and logged for later analysis. Additionally, during the execution the Arm sends an "Alive" message to a host PC to indicate the correct function of the application. When the code is running bare-metal, if no message is received after a given period (normally 10x higher the code execution time), the board is power-cycled and a **Crash** is logged. When the code is running on top of Linux, the Crashes are distinguished between two types. First an attempt is made to contact to the board and restart the application. If the attempt is successful, the event is logged as an **Application Crash** (Linux is still running and responding). If no connection with the board can be establish, the event is logged as a **System Crash**, as the operating system has crashed.

4.2 Fault Injection

The microarchitecture-level reliability assessment was performed on top of Gem5 simulator (BINKERT et al., 2011), using the GeFIN fault injection framework (CHATZIDIMITRIOU; GIZOPOULOS, 2016). The results of the fault injections were made by the reliability group at the University of Athens on a collaboration between the groups of both universities. GeFIN was used to quantify the Architectural Vulnerability Factor (AVF) (MUKHERJEE et al., 2003) 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 Arm Cortex microarchitectural configurations (GUTIERREZ et al., 2014). The in-order core was used to resemble A5 microarchitecture while the out-of-order core was used for A9. Both cores include a detailed model of the CPU pipeline along with cache memories and TLBs. 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.

The GeFIN was configured 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, Physical Register file, Data and Instruction Translation Lookaside Buffers (TLB). To achieve a statistical sample of 4% error margin and 99% confidence level, it was injected 1,000 single bit transient faults on each of the target components (LEVEUGLE et al., 2009). By refining the p variable of the formulation to correspond to the estimation result, the error margin is different for each workload and component and ranges between 1.7% and 4% with 99% confidence. Thus, the fault injection estimates first the Architectural Vulnerability Factor (AVF) which is used later to calculate the FIT rate of each benchmark.

The AVF is the probability that a fault in a specific hardware structure can result in a corruption in the execution of a program. The metric is independent of the components' technology or environmental factors, which are however related to the soft error rate. There is a direct connection between the failure rate and the vulnerability of a structure. The soft error rate quantifies how many faults are introduced in a hardware structure at a given period of time. As the structure's size increases, the error rate is also increased. In order to attribute the FIT of a component to its size, a per-bit FIT, which is called raw FIT, or FIT_{raw} must be measured. The FIT of a component can then be expressed as:

$$FIT_{component} = FIT_{raw}(bit) * Size(bits) * AVF_{component}$$

The size of each component is known a priory and the AVF is provided by GeFIN. The only missing attribute is the one that depends on the technology, which is the FIT_{raw}. To have a perfect emulation of the realistic error rate, the FIT_{raw} for all the resources in the processor should be measured, which is unfeasible mainly for COTS devices. However the L1 cache bit FIT_{raw} can be used as representative of the Cortex-A9 or the Cortex A5 technology as implemented in each board. The L1 cache was choosen because it is among the most vulnerable resources, while at the same time uses the same SRAM technology with rest of the CPU components. The L1 cache bit FIT was used as a common baseline for all the resources in the ARM CPU.

The sum of all $FIT_{component}$ equals to the FIT rate of the system. This can be calculated for all of the different fault-effects that were observed by the fault injection experiments. For the purpose of this work, the faults are classified in four fault effect classes: Masked, SDC, Application crash and System Crash (for the full system experiments).

4.3 Fault Injection vs. Neutron accelerated beam Experiment Setup

To avoid any difference not related to the reliability evaluation that can bias the results, the exactly same source codes were used, compiler, compiler options, and input vector (size and values) for both fault injection and neutron accelerated beam experiments. Still, the setups used for neutron accelerated beam experiments and fault injection are intrinsically different when executing on actual hardware vs. Gem5 simulation (GUTIERREZ et al., 2014). To evaluate if the benchmark execution shows any difference when running in the setups used for neutron accelerated beam experiments and fault injection, the execution of the same code are compared in the two setups using 7 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. The biggest difference is observed in the L1 Instruction TLB counters. Literature actually identifies certain design differences that exist in the implementation of TLB of Gem5 and Arm Cortex microarchitectures that support these observations (GUTIERREZ et al., 2014). Differences are to be expected as the Gem5 model is not exactly the same as the one implemented in hardware. One of the main goals of this work is to also understand if microarchitectural simulations can provide accurate insights on the corresponding hardware reliability.

To estimate the failure rate of a code using fault injection in order to compare it with the one measured with neutron accelerated beam experiments, it is necessary to know the raw (intrinsic) fault rate along with the probability of each fault to lead into a failure. In principle, multiplying the raw fault rate of each microarchitectural resource to its AVF would provide the failure (FIT) rate of the code executed on the device. However, measuring the raw fault rate for each hardware resource would require too much time and, when dealing with COTS, could be unfeasible due to visibility limitations. In this work, it was 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 also the targeted components in GeFIN are all implemented in the same SRAM technology as the L1 cache.

5 RESULTS

This chapter presents the results obtained with neutron accelerated beam experiments and fault injection on the A9 and A5. First, the fault injections and neutron accelerated beam experiments are compared only on the A9 in order to show the differences on the FIT rates between the neutron accelerated beam experiments the fault injection. After that, the results of the neutron accelerated beam and fault injection on the A5 are shown and compared with another run of experiments on the A9 in order to see the operating system and integration impact of the overall FIT rate.

5.1 Neutron accelerated beam experiments vs fault injection: Linux on ARM A9

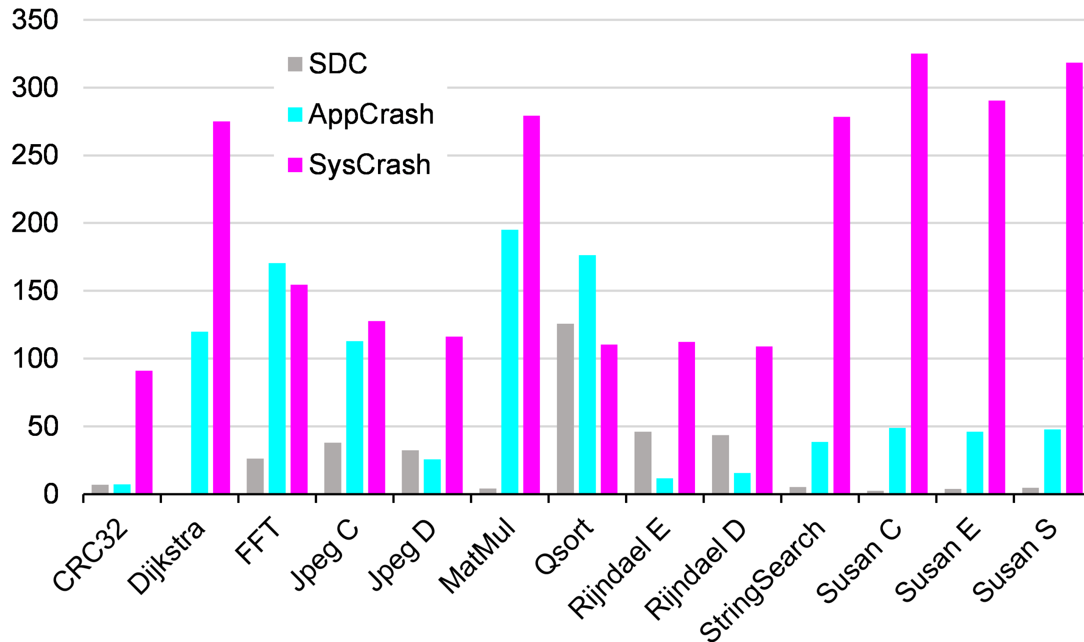
In this Section, neutron accelerated beam experiment results and fault injection analysis are presented and discussed, highlighting the similarities and differences each methodology provides on the reliability of the ARM A9.

5.1.1 Neutron accelerated beam Experiments Results

Figure 5.1 shows the FIT rates for the 13 benchmarks tested at LANSCE following the experimental procedures detailed in Section 4. It is shown the FIT rate for SDC, Application Crash, and System Crash. It is worth noting that these three types of events are uncorrelated and independent, in the sense that at most one of the three events occurs in one execution and its occurrence will not affect the probability of errors in the next executions.

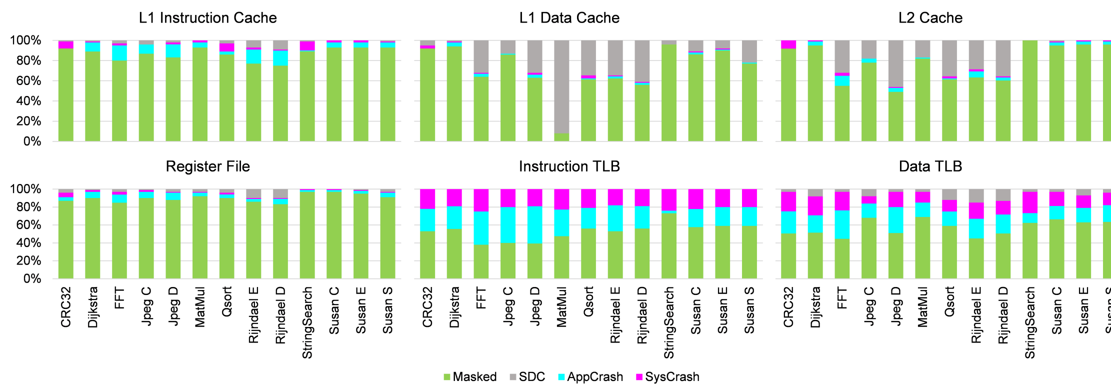
From Figure 5.1 it is clear that a **System Crash** is the most likely event, for all the benchmarks but FFT and Qsort. As discussed in the following, FFT and Qsort have a higher Application Crash FIT compared to other benchmarks; their Application Crash FIT is even higher than their System Crash rate. As shown in (FRATIN et al., 2018), System Crashes have a component which is intrinsic to the particular hardware platform, only. Even resilient codes (i.e., with very low SDC or Application Crash rates), then, could experience a relative high number of System Crashes. This is the case of CRC32, Rijndael D, and Rijndael E in Figure 5.1. The benchmarks with the highest System Crashes FIT are Dijkstra, MatMul, StringSearch, and the three Susans benchmarks. These are actually

Figure 5.1: neutron accelerated beam experiments FIT rates for SDCs, Application Crashes and System Crashes.



Source: The author

Figure 5.2: Fault injection effects classification for all 13 benchmarks in all 6 components. Effects are Masked, AppCrash, SysCrash, and SDC. The AVF corresponds to the sum of the three non-Masked cases.



Source: The author

the benchmarks with the smallest input size, which is not even sufficient to fill all caches. Since there is space available in the caches, the Linux kernel code will not be evicted when the context is switched to the application. When Linux is in idle, its data is then exposed and vulnerable to neutron accelerated beam. An error in the kernel code is likely to lead to a System Crash, exacerbating the System Crash vulnerability. This is aligned with previous studies that show a higher operating system error rate in the absence of cache conflicts (SANTINI et al., 2016).

There is also a significant variation among the **Application Crash** FIT of the different benchmarks. The benchmarks with highest Application Crash rate are MatMul, Qsort, and FFT while the lowest (about 2 orders of magnitude lower than MatMul) is CRC32. The codes with the higher Application Crash FIT are found to be the ones with higher presence of control-flow operations, higher use of stack for memory or nested loops. Additionally, these applications are the ones with non-coherent memory accesses. This requires several accesses (reads and writes) to the main memory, which is outside the irradiated chip. To access the main memory it is necessary to use an interface between the core and the main memory. It has already been demonstrated that errors during intra-chip communications are likely to lead to an Application Crash as an error in the interface or in one of the cores involved in the communication makes the application to wait indefinitely (FRATIN et al., 2018). While Rijndael has similar Application Crash FIT rate for both encoding and decoding, for Jpeg C the coding has a much higher (almost 1 order of magnitude) Application Crash FIT rate than the decoding. This can be explained by the fact that in the case of Rijndael the process to encode and decode is algorithimcally almost identical, but in Jpeg D the decoding is achieved by doing the reverse steps from the encoding. This means that the program flow is different and behaves differently when an error occurs.

For **SDC** FIT rate, the difference between the lowest (Dijkstra) and the highest (Qsort) is around 2 orders of magnitude. This result confirms previous studies showing that the SDC rate is strongly application dependent (FRATIN et al., 2018). The codes with lower SDC FIT rate are the three Susans, StringSearch, MatMul, Dijkstra, and CRC32. These benchmarks either have a small input (Susans, StringSearch, MatMul) or have long memory latency (CRC32). This translate to either small irradiation area or the information is being fetched from the main memory which is not irradiated. It is also observed that for SDC rate for both encoding and decoding of Jpeg and Rijndael benchmarks have similar FIT SDC rate. This can be explained noting that, for Rijndael, both encoding and decoding have equal input size and, for Jpeg, the combination of input and output have equal size (the input of the encoding and the output of the decoding have almost identical sizes).

5.1.2 Fault injection results using GeFIN

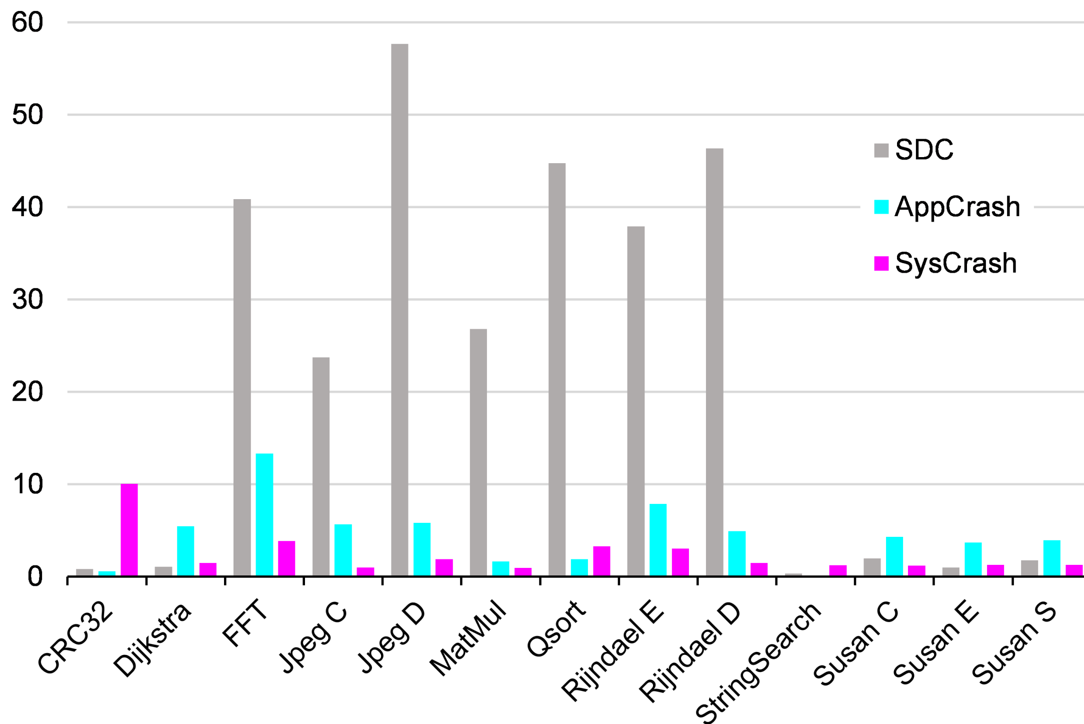
Unlike neutron accelerated beam experiments, fault injection campaigns were executed for each hardware component separately. Figure 5.2 illustrates the AVF estimation reported by GeFIN. With fault injection the faults that were benign (masked) and the fault was either overwritten or did not affect the execution in any observable way can be counted. Figure 5.2 presents the AVF distribution of each fault class, SDC, Application Crash or System Crash, while the summary of all non-masked cases expresses the vulnerability of the structure (AVF). Notice that AVF does not only depend on a structure's size or organization, and is aggregately related to the significance of the stored data to the correct operation of the system.

As one would intuitively expect, the majority of SDC results (grey areas) are related with the structures that mostly contain data, which are the L1 Data Cache and L2 Cache. In contrast, it can be seen how most of the abnormalities reported by faults in the L1 Instruction Cache are crashes. Interestingly, most of the benchmarks have higher Application Crashes than System Crashes, with CRC32, Qsort and StringSearch being the only outliers.

The TLBs are consistently highly vulnerable. The reported fault injections refer to the Physical page (target) of the tables as they mostly lead to either incorrect memory translations or wrong permission flags. Incorrect translations will lead to use of wrong data in all references of the particular page. In contrast, the virtual part (tag) has almost zero vulnerability as corruption in the tag can mainly result to tag misses and thus invoke unnecessary page walks, which introduces a performance penalty. The Register file is involved in both control and data processing and the vulnerability is evenly distributed in all classes, without particular trends. However, it can be seen how both Rijndael benchmarks report high probability of SDCs, and this can be attributed to the high level of instruction level parallelism of the algorithms, which results in high utilization of the register file for data processing.

Although the AVF measurement is not related only to the size of a component, the probability that a fault is introduced at a particular structure (by a particle) is highly related to its size. Each TLB for instance, has a size of 512 bytes (4,096 bits) while the L1 Cache memories have a size of 32KB (252,144 bits). Consequently, the probability that a fault will strike the TLB is only $1/64^{th}$ of the Cache's probability. That being said, the L2 Cache, which covers more than 80% of the modeled memory cells of the system,

Figure 5.3: Fault Injection FIT rates for SDCs, Application Crashes and System Crashes.



Source: The author

suffers the most by the striking of faults.

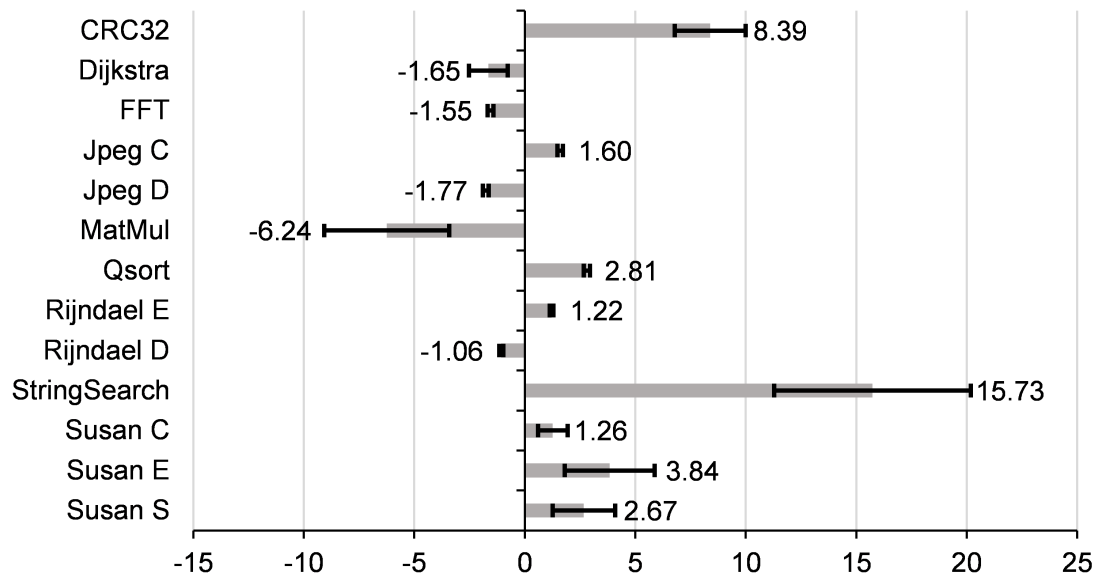
5.1.3 Discussion of the comparison between Fault Injection and Neutron Accelerated Beam Experiments Data on the A9

To experimentally measure the FIT_{raw}, a specific benchmark was designed in order to test the L1 data cache. This is achieved filling byte-by-byte the L1 data cache with a known pattern (0xA5) and read it after a period of time, comparing the read values with the pattern and counting the bit-flips. This gives us the FIT rate for the cache and, diving it by the tested cache size, it gives us the cache FIT per bit. For the L1 cache, the measured error rate is $2.76x10^{-5}$ FIT per bit, very close to other publicly available data for the same technology (BAGGIO et al., 2004).

Using the FIT_{raw} the FIT rate of applications can be estimated based on the AVF analysis, as shown in Figure 5.3. In order to compare the FIT rate predicted with neutron accelerated beam experiments and fault injection, for each benchmark the highest FIT rate between both methodologies are divided by the lowest FIT rate. Whenever the fault injection FIT rate is lower than the one estimated with neutron accelerated beam experiments

the value is represented as positive, if not it is represented as a negative number.

Figure 5.4: Multiplication Factor for SDC FIT comparison between neutron accelerated beam experiments and fault injection.

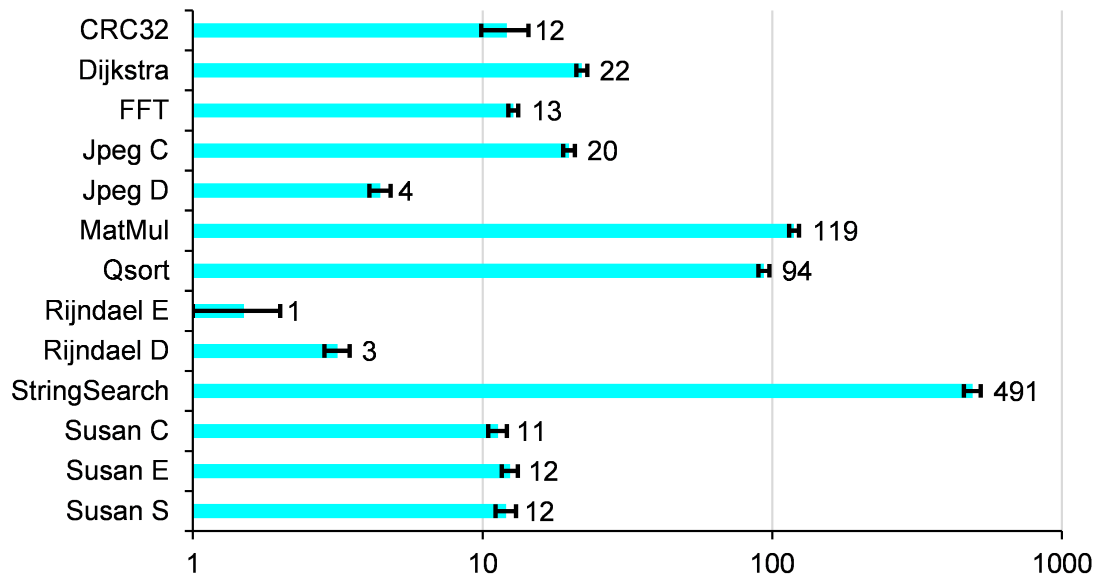


Source: The author

Figure 5.4 shows the comparison between neutron accelerated beam experiments and fault injection SDC FIT rates for all 13 benchmarks. The positive values (towards the right) of the horizontal axis indicate how many times the FIT rate measured with neutron accelerated beam experiments is higher than the FIT rate calculated with fault injection while negative numbers indicate the opposite (fault injection FIT rates are higher). For most benchmarks neutron accelerated beam and fault injection give very close FIT rates (for 10 out of 13 codes the difference is smaller than 4x, while for 7 of them it is less than 2x). MatMul, StringSearch, and CRC32 have the largest difference between the two FIT rate measurements. However, these benchmarks have very low SDC FIT rate, for instance, StringSearch has 5.45 SDC FIT on the neutron accelerated beam experiment and only 0.34 on the fault injection, meaning that the absolute difference between FIT rates is very small and such differences are within the statistical error. As expected from the discussion in Section 2, for most of the benchmarks the FIT rate measured with neutron accelerated beam experiment is higher than the fault injection one. However, for 5 benchmarks (Rijndael, Jpeg D, FFT, Dijkstra, and, mainly, MatMul), fault injection reports a higher SDC FIT rate than neutron accelerated beam experiments. As discussed in the following sections, these are also the benchmarks that show a much higher Application Crash and System Crash rate for neutron accelerated beam experiments compared to fault

injection and this difference implies that some faults propagate differently to generate SDCs or Crashes in the two setups but still result in a corruption of the correct execution.

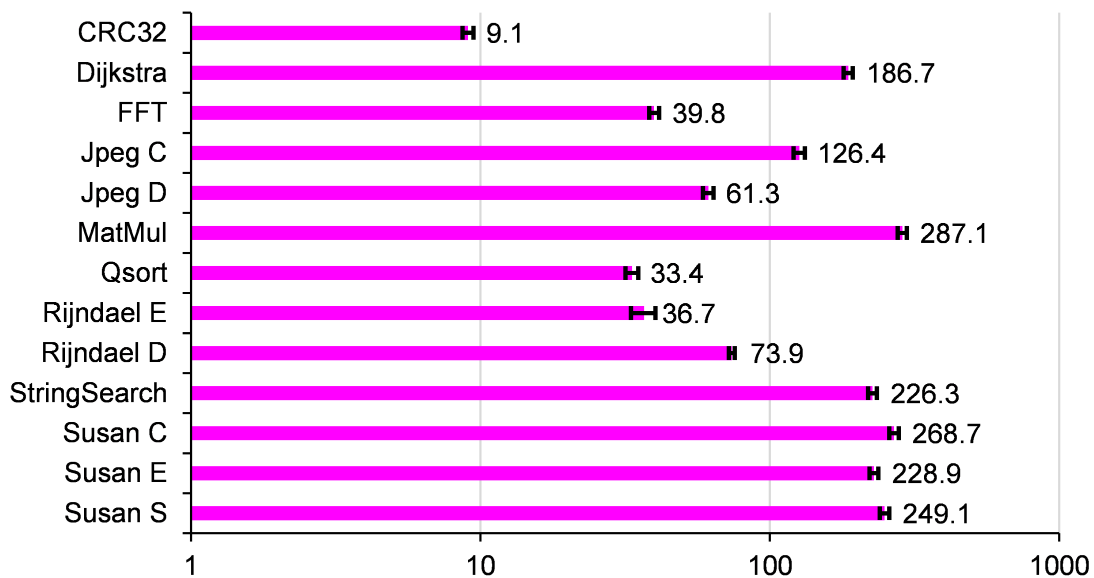
Figure 5.5: Multiplication Factor for Application Crash FIT comparison between neutron accelerated beam experiments and fault injection.



Source: The author

When comparing the Application Crash FIT rate calculated with the two setups, as shown in Figure 5.5, the differences between fault injection and neutron accelerated beam experiments are much higher than for SDCs, ranging from 1.5x to almost 500x (horizontal axis is in logarithmic scale). It is worth noting that neutron accelerated beam experiments FIT is always higher than the fault injection estimation, which is expected since Application Crashes could be triggered by corruption in logic/control hardware elements which are difficult to simulate (FRATIN et al., 2018). For three benchmarks, StringSearch, MatMul, and Qsort, the difference between neutron accelerated beam experiments and fault injection is close or bigger than two orders of magnitude (while for all others the difference is smaller than 22x). The reason behind this may be attributed to differences between the two setups. While fault injection experiments output is downloaded and compared offline against the fault-free output to detect SDCs, neutron accelerated beam experiments require an additional routine for on-line SDC checking. As during neutron accelerated beam experiments most executions are error-free, downloading all outputs would be an unnecessary waste of space and time. These checks are almost transparent to the workload characteristics and, to avoid the corruption of SDC details, they are intentionally designed to hold pointer references instead of actual data. Application Crashes are mostly sourcing in abnormalities caused in the program flow (i.e., irregular branches, wrong memory ref-

Figure 5.6: Multiplication Factor for System crash FIT comparison between neutron accelerated beam experiments and fault injection.

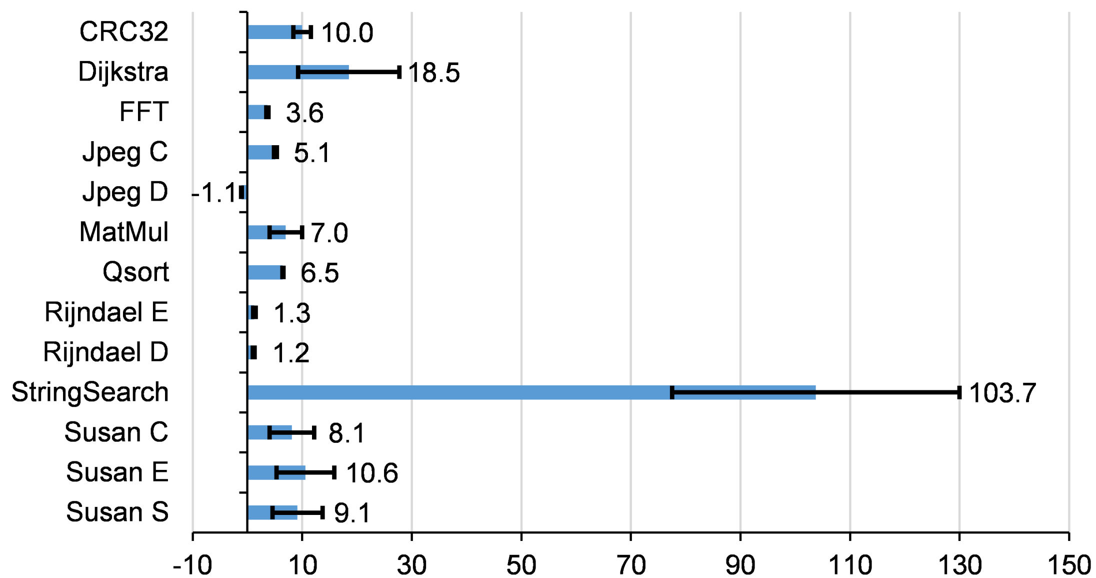


Source: The author

erences, etc.). These have roots in the executed code of a program (of what is placed in the .text section of a program). The common property of the 3 workloads with greater differences between fault injection and irradiation data (StringSearch, MatMul, and Qsort) is the relatively small code size, which fits inside the Instruction Cache. As a result, the code sits in the cache for the whole experiment, being exposed to neutrons. Additionally, there is enough space for the SDC check routines to remain in the cache hierarchy, instead of being evicted during the program execution (as L2 is shared) during the neutron accelerated beam experiments. The exposure of these routines to the neutron accelerated beam (which mainly consist of pointers) would result in segmentation faults that translate to Application Crashes. This inevitable difference between the two setups could explain the observed behavior for the three outliers.

The System Crash FIT difference, as shown in Figure 5.6, does not follow the same behavior of the Application Crash FIT. In this case there is a high difference between neutron accelerated beam and fault injection for all benchmarks, with the neutron accelerated beam FIT being always higher than the injection. The difference ranges from about 9 times (CRC32) to about 287 times (MatMul). The benchmarks with the largest difference are MatMul, Dijkstra, StringSearch, and the three Susans. These workloads also happen to have the smallest inputs. As a result, they actually leave a large part of the cache hierarchy unused. These differences can also be attributed to by differences on the

Figure 5.7: SDC and Application Crash Comparison between neutron accelerated beam experiments and fault injection.



Source: The author

setups.

In fault injection experiments, the unused portion of the cache hierarchy remains empty as the caches are reset on every experiment, while in neutron accelerated beam experiments this space is used by the kernel for other system operations (e.g. scheduling routines, timer handlers etc.). The introduction of faults in these regions that will most likely result to system crashes only in the neutron accelerated beam experiments. The rest of the benchmarks that use most of the cache hierarchy do evict the kernel from the caches and do not suffer from this scenario.

As it is shown in Figures 5.4, 5.5, and 5.6, the magnitude of the differences between neutron accelerated beam experiment and fault injection FIT rate are application dependent while there exists a clear trend of larger FIT rates measured by neutron accelerated beam experiment compared to fault injection for all FIT rates (SDC, Application Crash, and System Crash FIT rates). As discussed in Section 2, there are various reasons for fault injection and neutron accelerated beam experiments to provide different FIT rates and this is the reason why they have to be considered complementary to each other. One very likely reason for the differences and particularly of the larger number of corruptions in the neutron accelerated beam experiments are the resources that are not modeled in the simulator. This high System Crash FIT rate is a peculiar characteristic of the Xilinx Zynq platform, specifically the FPGA-ARM interface based on interrupts, which cannot

be further investigate without detailed (proprietary) information.

An indirect way to correlate the results and focus on the same hardware is by attributing the effects that different hardware parts cause. While System Crashes, exceptions and wrong memory accesses can be caused by most of the system's components (including CPU cores, peripherals, controllers, bridges and interconnections), SDCs can only occur at the components that produce the output, which is the CPU core. This also partially applies to the majority of Application Crashes.

In Figure 5.7, the relative difference of the *sum* of the FIT rates for SDCs and Application Crashes measured with neutron accelerated beam and fault injection are shown. StringSearch has the highest relative difference (of about 100x). This is due to the *extremely* small number of SDCs observed in both setups, having much less events observed in the injection than in the neutron accelerated beam setup. It is also interesting that the MatMul and Qsort, that show a difference of 100 times in Application Crash FIT (Figure 5.5), now have a difference lower than 10 times when comparing SDC and Application Crash FIT rates (Figure 5.7). This means that, the overall FIT rate is only 10x higher in the neutron accelerated beam case. It is likely that some of the Application Crashes observed in the neutron accelerated beam experiment are observed as SDCs in the injection. This is probably caused by the corruption of some hardware resource not modeled in the injection setup. The other benchmarks are less affected by the code characteristics discussed previously. For three benchmarks, Jpeg D and the two Rijndael, the overall FIT difference is very small, from 1.08x up to 1.26x.

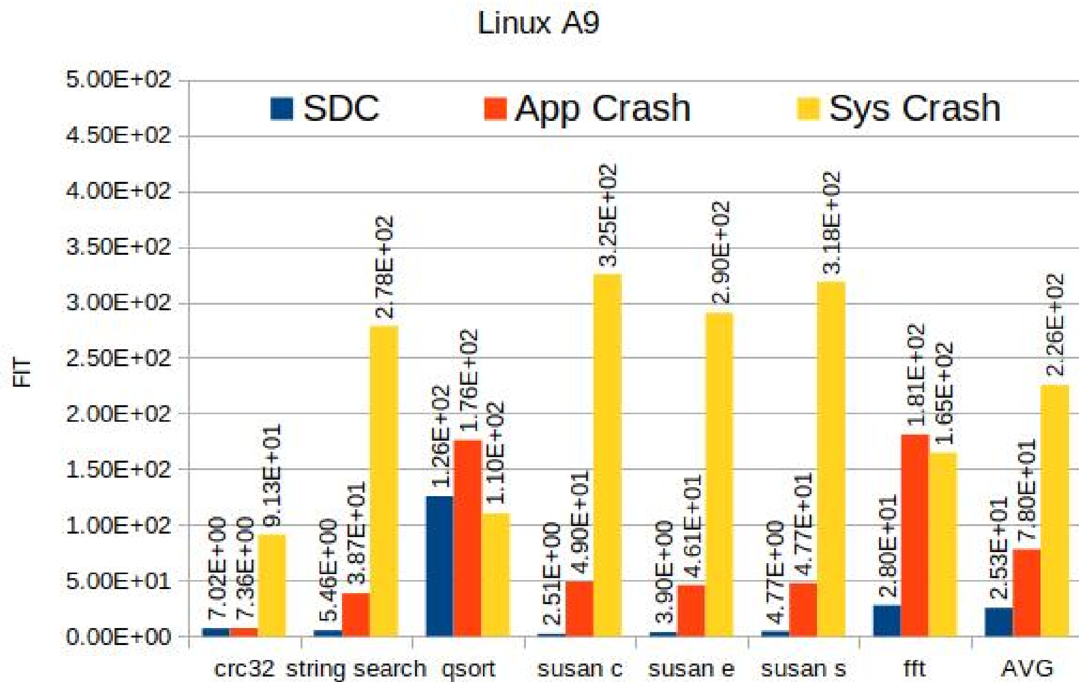
5.2 Operating System and Integration Impact: ARM A9 and ARM A5

In this Section it is presented and discussed neutron accelerated beam experiment results and fault injection analysis on both the A9 and A5, but this second comparison aims to also compare the operating system and SoC integration impact on the overall FIT rate.

5.2.1 Neutron accelerated beam Experiments Results

Figures 5.8 and 5.10 show the FIT rates measured with neutron accelerated beam experiments for the A9 (Linux) and for the A5 (bare-metal and Linux), respectively. What

Figure 5.8: Cortex A9 Linux neutron accelerated beam FIT rates for SDCs, Application Crashes and System Crashes.



Source: The author

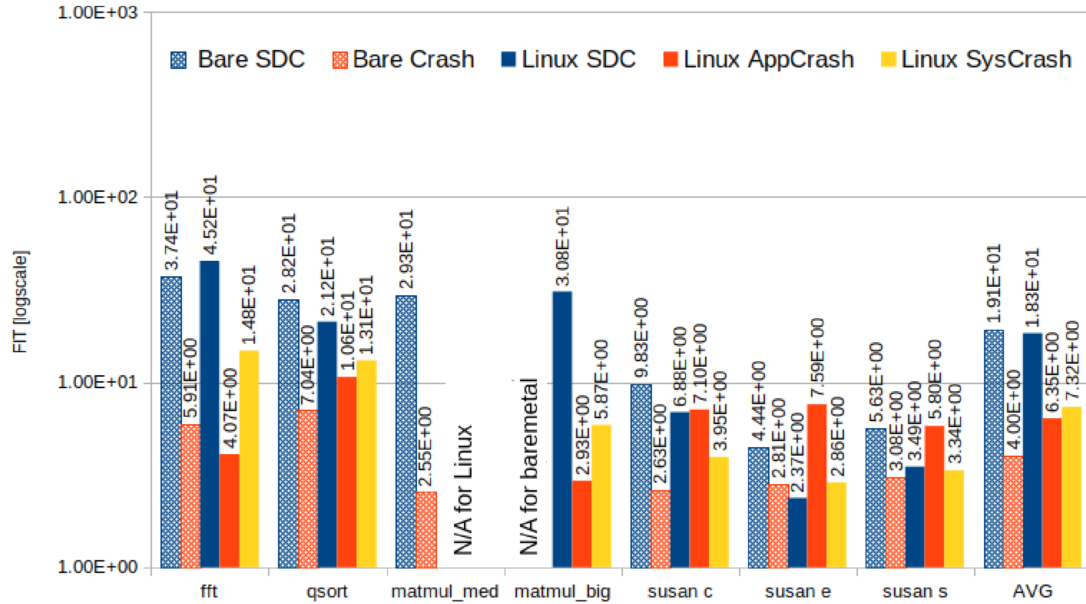
can be observed from the neutron accelerated beam experiment data is that there is a significant dependence of the FIT rate on both the device, program executed, and system stack configuration (Linux vs bare-metal, on the A5). Next the reasons for these dependencies are analyzed and understood.

The SDC FIT variation from benchmark to benchmark is of about 1 order of magnitude for both the A5 and the A9. Interestingly, qsort benchmark is the one with the highest FIT rate in both devices, probably because how the code access its input data stored in the caches and memory. While the main memory of the A5 is kept out of the neutron accelerated beam, and then fault-free, as the core is waiting for new data to be fetch the elements kept in the caches are exposed and can be corrupted.

The average SDC rate is 3.5x higher on the A5 than on the A9. Nevertheless, to have a fair comparison of the CPU vulnerability, the data measured in the two Silicon devices needs to be normalized by the technology sensitivity (raw FIT rates per bit). This is done in Section 5.2.4.

Interestingly, both the Application Crash and the System Crash FIT rates, are much higher (more than 2 orders of magnitude) on the A9 than on the A5. Additionally, the reported experimental data shows that, while the SDC rate and Application Crash rate varies significantly across benchmarks (over 1 order of magnitude for SDC and Appli-

Figure 5.9: Cortex A5 Bare-metal and Linux GeFIN FIT rates for SDCs, Application Crashes and System Crashes.



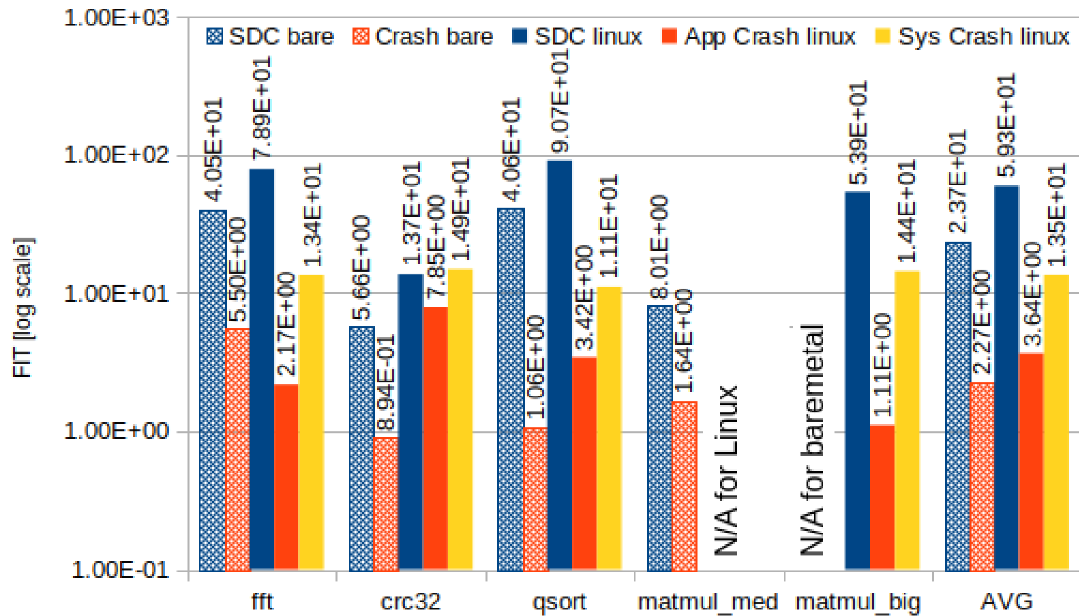
Source: The author

cation Crashes), the System Crash rate is almost constant on the A5 and on the A9 (the variation for system crashes is, on the average, 30%). This observation confirms previous studies that demonstrate that the System Crash has a stronger component that depends on the hardware alone (FRATIN et al., 2018). And additional insight this works data provides is that the OS does not play a significant role in the System Crash of processors.

5.2.2 Fault injection results using GeFIN

Figure 5.9 shows the equivalent A5 estimation using fault injection. The fault injection was provided with the experimentally measured raw FIT for caches. In this second experiment run the FIT rate measured for the Cortex-A5 is $2.37 \times 10^{-4} cm^2$ and $2.59 \times 10^{-5} cm^2$ for the Cortex-A9 which is similiar as the previous experiment run. This raw FIT is used to predict, from the Architectural Vulnerability Factor, the error rate of applications. When compared to neutron accelerated beam data, the SDC FIT estimation appears to be extremely close and in all cases less than 3x. When the technology dependence of the error rate is removed(fault injection uses the same fault probability as neutron accelerated beam experiments), then, the SDC rate can be predict with good precision. Interestingly, GeFIN crash rates appear to be slightly higher compared to irradiated. The fault effect classes follow similar trends and behavior.

Figure 5.10: Cortex A5 Bare-metal and Linux neutron accelerated beam FIT rates for SDCs, Application Crashes and System Crashes.



Source: The author

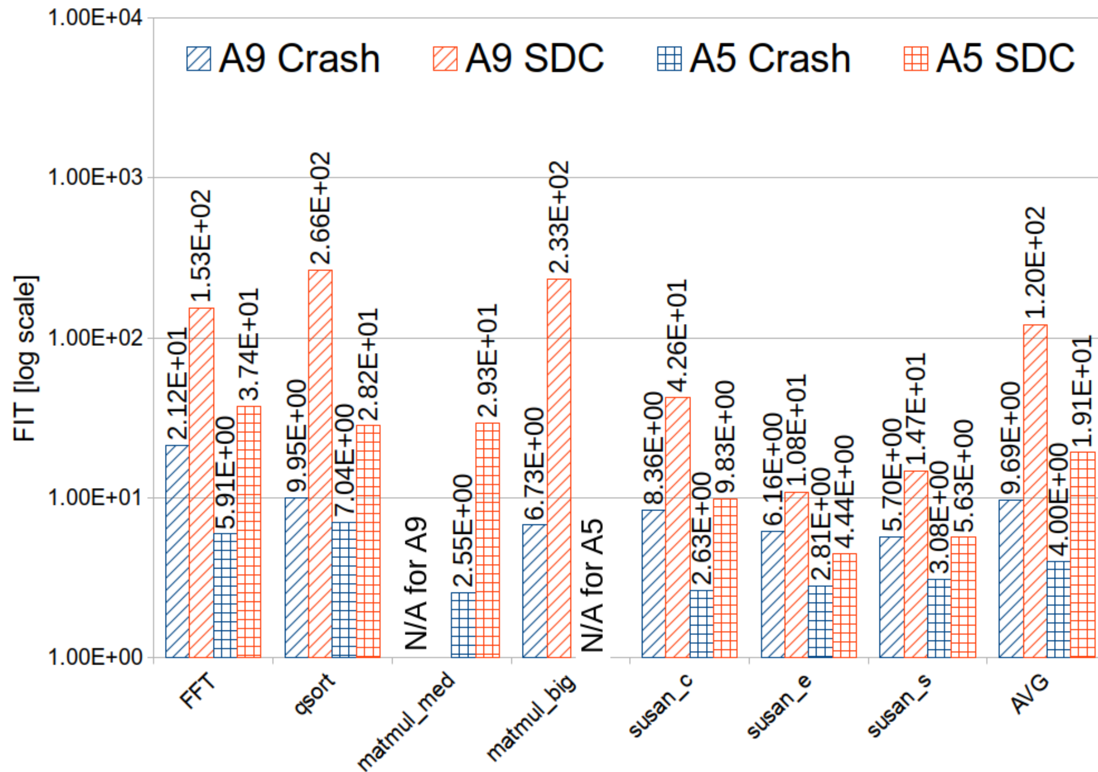
5.2.3 Operating System impact

Figure 5.9 and 5.10 show, respectively for fault injection and neutron accelerated beam experiments, the comparison between the FIT rates of the different codes executed on the A5 bare-metal or on top of Linux. As described in Section 4, the very same codes with the very same input vectors in the two configurations were executed to reduce any possible bias. The observed differences between bare-metal and Linux configurations are then due to the operating system presence, only. Unlike application crashes, system crashes can occur when the CPU operates in kernel mode. A fault that has affected OS data structures (e.g. kernel heap) or kernel code, including system calls and drivers can result to a system crash.

5.2.4 Integration Impact

There are three variable factors that can affect the vulnerability of each CPU core: (a) Manufacturing Technology, (b) Microarchitecture and (c) System integration. To understand the impact of SoC integration in the reliability of the two cores, it is required to isolate as many variables that influence the observed differences between the Cortex-A5 and Cortex-A9 error rates, as possible. The GeFIN fault injection setup allows control

Figure 5.11: Cortex A5 and A9 Bare-metal FIT rates estimated through GeFIN using the experimentally measured technology sensitivity.



Source: The author

over these variables and can be used to evaluate the impact of (b); it receives (a) in terms of the raw FIT rate as an input. However, there is no Gem5 description available for complex and integrated SoC. So GeFIN lacks the possibility of evaluating the impact of (c). neutron accelerated beam experiments data, as the whole silicon chip is irradiated, is affected by all three factors. However, as errors can be observed only at the output, neutron accelerated beam experiments offer very limited visibility and cannot be used to differentiate between the three factors. It is only combining GeFIN and ChipIR data that allows us to quantify what is the impact of each of these attributes to the overall vulnerability estimation.

Figure 5.11 shows, for both the A5 and the A9, the FIT rates estimated using GeFIN. To estimate the FIT rates the same technology raw FIT measured through neutron accelerated beam experiments on the real hardware is used. The average A9 difference is 6x for SDC and 2.4x for Crashes, or 3.3x in total FIT.

Since the Cortex-A5 and Cortex-A9 are implemented in two different technologies, the raw probability for a neutron to generate a fault in the memory or during computation is different in the two devices. To measure the technology sensitivity the same

static test in each device was performed exactly as discussed previously on both cores.

The difference between the per-bit sensitivity of the Cortex-A5 and Cortex-A9 states that a neutron is about 8.6x more likely to generate a fault in the Cortex-A5 than in the Cortex-A9. The difference in raw sensitivities is not surprising, there are several reasons for the cross section of two devices to be highly different, including transistor dimensions, transistor layout, voltage, charge, etc. (BAUMANN, 2005; ZIEGLER; PUCHNER, 2010).

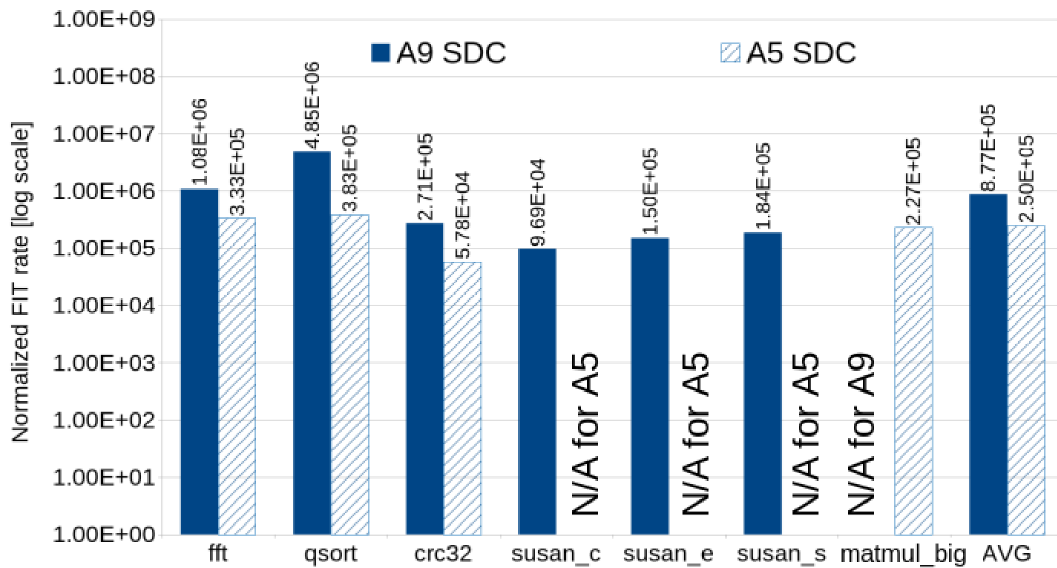
This means that on the ChipIR setup, the difference of the estimated FIT rate that can be attributed to the variables of technology and microarchitecture is expected to be within the same order of magnitude, as the two variables negate each other. Thus, any differences that exceed an order of magnitude can be safely attributed to the system integration of the A9 CPU. In a first approximation, to compare the error rate of the two devices,all the FIT rate of the Cortex-A5 and Cortex-A9 are divided by the respective technology sensitivity. This is shown in Figure 5.12 for SDCs and Figure 5.13 for Application and System Crashes.

Data depicted in Figure 5.12 shows that the SDC FIT rates for the A5 and A9, normalized by the respective technology sensitivity, are very similar (well inside the order of magnitude). Figure 5.13, on the contrary, highlights that the integration of different cores have a strong impact (well over two orders of magnitude) on the Application and System Crashes.

5.2.5 Discussion on Operating System and Integration Impact

The experimental analysis provides useful insights that allow us to quantify how different attributes impact the reliability evaluation. The combination of fault injection results along with the normalized A9 and A5 neutron accelerated beam experiments results allows us to see how either the SoC integration or the OS influences the overall FIT rate. Figure 5.16 shows a high level summary of the results that were observed using neutron accelerated beam experiments and validated using microarchitecture level fault injection analysis. It shows the impact of the integration of the CPU core in the SoC as well as the impact of the OS in the system software stack, in the SDC FIT rates as well as the Crashes FIT rates that. Confirming the initial findings and general speculations of previous work, it is clearly shown that the SDC part of the FIT rate is minimally affected by both the SoC integration and the OS existence while the Crashes FIT rates are severely affected by both

Figure 5.12: SDC FIT comparison between the stand-alone Cortex-A5 and the integrated Cortex-A9. Susan c, e, and s could not be tested in the A5 for lack of neutron accelerated beam time.



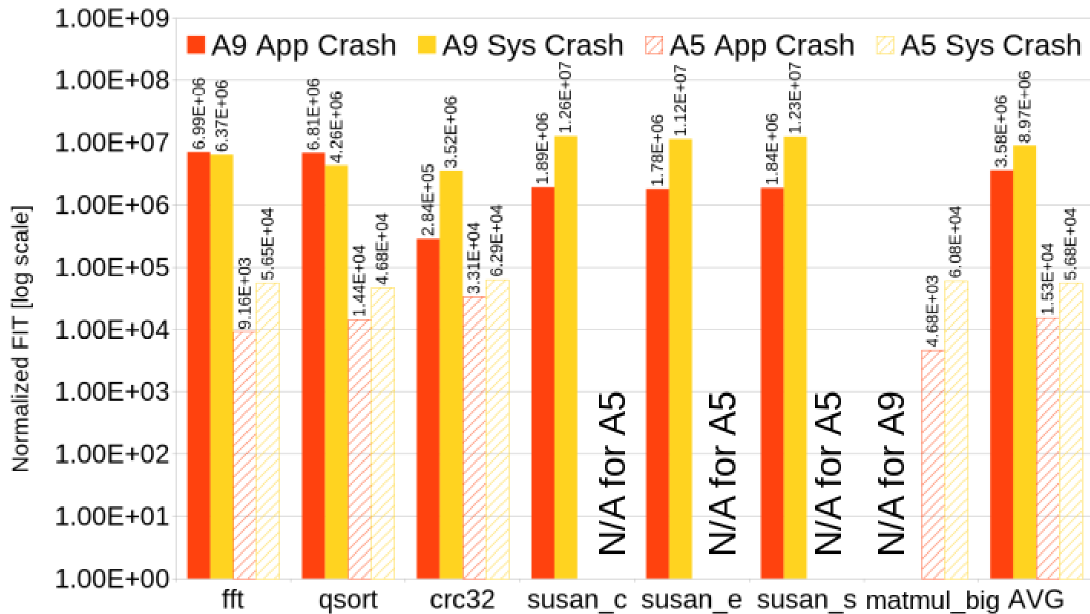
Source: The author

parameters.

The most dominant variable is the SoC integration, which was present only in the neutron accelerated beam experiments. The different microarchitecture itself was quantified to be 5x more vulnerable on A9 using fault injection. Along with the different fabrication technology which results to 8.6x higher fault rate on A5, the product of these two variables (which are self-neutralized) results to an expected difference of 2-3x. It can be seen that the FIT rate of crashes is increased, on the average, by 174 times when the CPU is integrated in the SoC, and this difference can only be attributed to the integration variable. In practice, this translates to a different and highly complex interconnection inside the chip. In the particular case of the A9 on the Zynq board, apart from the normal DDR memory controllers and other peripherals, the SoC also includes the programmable logic arrays and other programmable connections to the FPGA. The increased complexity is proved with this analyzes to have a massive impact in the failure rate of the system but not on the SDC rate of the applications.

The impact of cores integration and operating system can be further evaluated by considering, in Figures 5.14 and Figure 5.15, fft and qsort, which are the benchmarks estimated across all of the setups of this work. These setups cover several combinations of the system variables and only A9 Linux involves SoC integration. Interestingly, the SDC estimation is close in all cases with an average difference of 6.8x, which also matches the expected difference due to the different technology and microarchitecture. When

Figure 5.13: Application and System Crash FIT comparison between the stand-alone Cortex-A5 and the integrated Cortex-A9. Susan c, e, and s could not be tested in the A5 for lack of neutron accelerated beam time.



Source: The author

considering the A5 microprocessor, which is not integrated to a complex SoC in both neutron accelerated beam and fault injection experiments, the estimation between the two methodologies is very close.

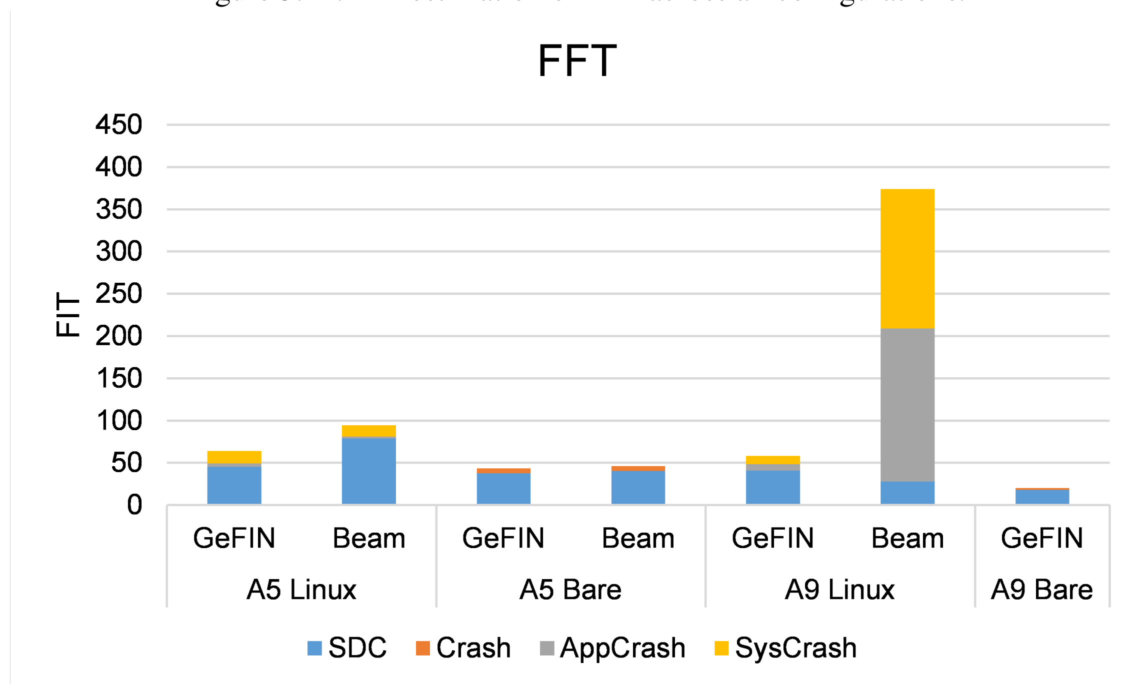
When focusing on the normalized A5 and A9 FIT neutron accelerated beam experiment estimations, it can be seen that, as expected (due to the larger cross section), A9 reports slightly higher vulnerability. The SDCs are very little influenced, where observed an increase of 3.5x on the average FIT rate between the A9 and the A5 on the neutron accelerated beam experiments, the A9 being higher. Since all benchmarks running on Linux on both the A9 and A5 use common I/O like Ethernet, the SDC FIT rate is little influenced by the A9-FPGA interface.

Apart from the SoC components, the OS also affects the overall crash FIT rate. A difference of 7.5x (the setup with Linux being higher) on the averages crash FIT rates between A5 bare-metal and A5 Linux on neutron accelerated beam experiments is observed, while the difference is only 3.5x on fault injection experiments. The increase in crashes can be explained by the fact that besides the benchmarks several other system processes responsible for the OS management are also running and since there is only a single core these processes coded are susceptible to irradiation. In addition, since FI only targets SRAM-based components, it is expected to underestimate vulnerability that is associated

to the microprocessor logic and is expected to lead to crashes. Faults in logic can only lead to SDCs only in data processing components, such as the functional units (e.g. ALU) of the microprocessor. This can explain why it is observed a 2x difference in the crash estimation between neutron accelerated beam and fault injection experiments.

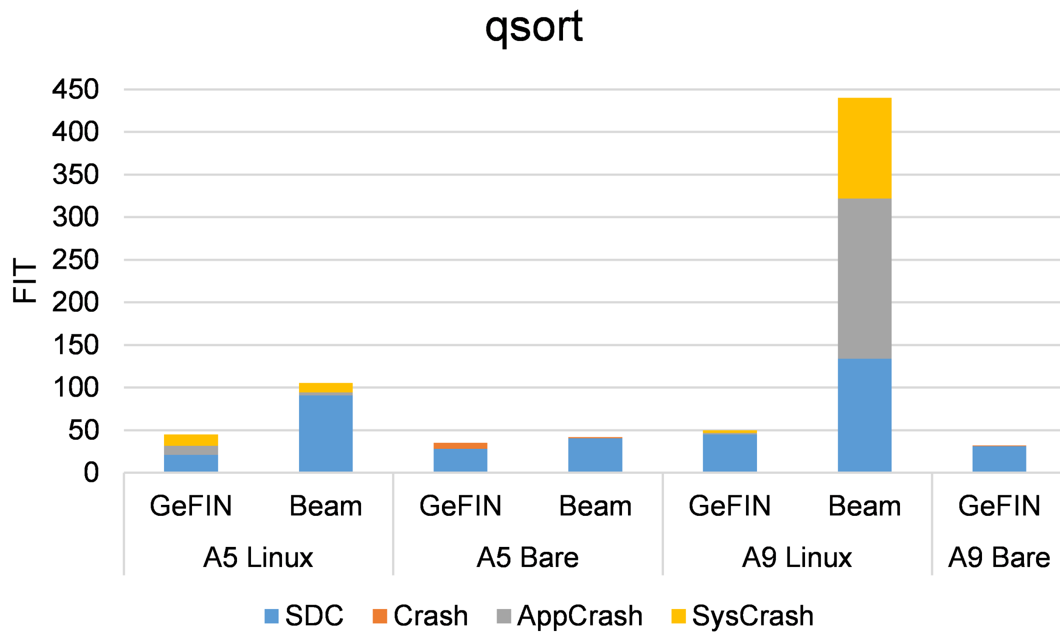
Again, just like with the previous analysis the SDC are only slightly affected, since output is only being processed under the actual workload and any OS interference does not participate or relate the the generated output, unless it is serviced by a system call. This is observed in both irradiation experiments as well as fault injection.

Figure 5.14: FIT estimation of FFT across all configurations.



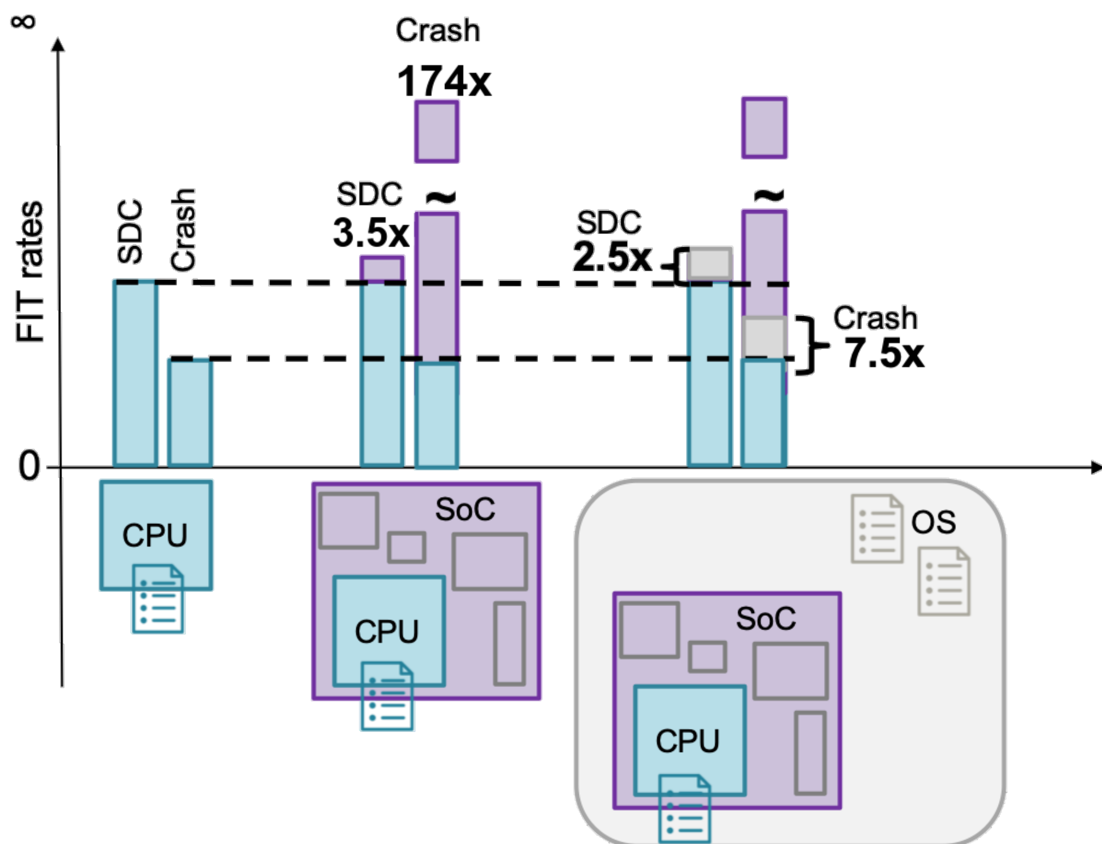
Source: The author

Figure 5.15: FIT estimation of qsort across all configurations.



Source: The author

Figure 5.16: Overall impact of cores integration and operating system on the CPU error rate as resulting from the data. On the average, the cores integration barely increases the SDC rates (about 3.5x) but significantly increases the Crash rate (174x, combining the increase in App crashes and Sys crashes. The operating system does not have a significant impact on SDCs (about 2.5x) and increases crashes of about 7.5x.



Source: The author

6 CONCLUSION

This work presents first a detailed analysis that aims to report a head-to-head comparison of two very popular reliability assessment methods: (a) physical accelerated beam test of an ARM Cortex-A9 CPU and (b) fault injection on the corresponding model of the ARM Cortex-A9 CPU on the state-of-the-art microarchitectural simulator Gem5. For an as-close-as-possible comparison, this work maximize the equivalence of the physical system setup and the simulated system setup: hardware configuration, application software, and operating system.

Although accelerated beam testing is both very fast and accurate, it suffers from the very limited observability level and can only deliver the summary of these variables; the eventual result of programs execution when hit by the neutron beam is either a crash or a silent data corruption (output mismatch) but there is no way to attribute the failure to a specific part of the hardware or a specific portion of the software code. Aided by microarchitecture-level fault injection using a state-of-the-art fault injector built on top of Gem5, this work have been able to quantify the portion of the differences that can be attributed to each of these variables. The fault injector has been fed by different raw FIT rate per bit to estimate different manufacturing nodes, and has been used to model the two different microarchitectures as well as the impact of the operating system (the injector was employed in both bare metal and OS setup).

The comparison of the two reliability assessment approaches helps in bounding the range of the expected FIT rates of a CPU when it is deployed in a final system in the field. This work have shown that for the diverse set of benchmarks employed in these experiments, the FIT rates differences between accelerated beam test and microarchitectural fault injection can be extremely small (when only the SDC FIT rate is considered) and does not exceed one order of magnitude when all types of errors (including Application and System Crashes) are considered for the Total FIT rate of the system. The insights of this study can assist CPU designers in making informed decisions about the soft error protection mechanisms best suited to a particular hardware and software combination.

After, it was performed an extensive reliability evaluation of 2 widely used ARM microprocessors, Cortex-A5 and Cortex-A9 with the primary objective of identifying the failure rates impact the SoC integration and OS inclusion can have on their operation. The two cores are fabricated using different technologies and their SoC had a different organization; A5 was an individual CPU core whereas the A9 was integrated on a Xilinx Zynq

SoC along with an FPGA. The two microprocessors are associated with 3 variables in the setup, namely: Technology, Microarchitecture, and SoC integration. Using accelerated beam testing, this work have evaluated the two platforms on baremetal and full system setups, in order to quantify how each of these variables contributes to the FIT rate of the device.

This works analysis showed that the SDC FIT rate is only slightly affected by the SoC integration and the existence of the OS, and is measured to only increase up to 3.5x due to the microarchicture difference. On the other hand, Crashes (both application and system) are dramatically increased (up to 2 orders of magnitude) on the SoC that includes the FPGA, showcasing how this attribute can really influence the FIT rate. The OS influence on the overall setup is also evaluated and is quantified to increase the crash rate up to 7.5x.

The outcomes of this analysis confirm on two different Arm CPU cores the initial findings and speculations from earlier work that the majority of the SDC failure rates can be safely attributed to the CPU core itself while executing the user codes. On the other hand the System and Application Crashes parts of the overall system failure rates are significantly affected by the complexity of the SoC integration as well as the inclusion of the operating systems and are significantly increased. Using a combination of neutron beam testing and microarchitecture level fault injection this work have quantified the magnitude of these differences which can be used for the integration of diligent soft error protection methods at the hardware and microarchitecture level as well as at the software level.

REFERENCES

ASADI, G.-H. et al. Balancing performance and reliability in the memory hierarchy. In: **Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software, 2005**. Washington, DC, USA: IEEE Computer Society, 2005. (ISPASS ’05). ISBN 0-7803-8965-4.

Avizienis, A. et al. Basic concepts and taxonomy of dependable and secure computing. **IEEE Transactions on Dependable and Secure Computing**, v. 1, n. 1, p. 11–33, Jan 2004. ISSN 2160-9209.

BAGGIO, J. et al. Analysis of proton/neutron SEU sensitivity of commercial SRAMs-application to the terrestrial environment test method. **IEEE Transactions on Nuclear Science**, Institute of Electrical and Electronics Engineers (IEEE), v. 51, n. 6, p. 3420–3426, dec. 2004. Available from Internet: <https://doi.org/10.1109/tns.2004.839135>.

BAUMANN, R. Radiation-induced soft errors in advanced semiconductor technologies. **Device and Materials Reliability, IEEE Transactions on**, v. 5, n. 3, p. 305–316, Sept 2005. ISSN 1530-4388.

BINKERT, N. et al. The gem5 simulator. **SIGARCH Comput. Archit. News**, ACM, New York, NY, USA, v. 39, n. 2, p. 1–7, Aug 2011. ISSN 0163-5964. Available from Internet: <http://doi.acm.org/10.1145/2024716.2024718>.

BISWAS, A. et al. Computing accurate AVFs using ACE analysis on performance models: A rebuttal. **IEEE Computer Architecture Letters**, Institute of Electrical and Electronics Engineers (IEEE), v. 7, n. 1, p. 21–24, Jan 2008. Available from Internet: <https://doi.org/10.1109/l-ca.2007.19>.

BLOME, J. et al. A microarchitectural analysis of soft error propagation in a production-level embedded microprocessor. In: **In Proceedings of the First Workshop on Architecture Reliability**. [S.l.: s.n.], 2005.

CAZZANIGA, C.; FROST, C. D. Progress of the scientific commissioning of a fast neutron beamline for chip irradiation. **Journal of Physics: Conference Series**, IOP Publishing, v. 1021, p. 012037, may 2018. Available from Internet: <https://doi.org/10.1088\%2F1742-6596\%2F1021\%2F1\%2F012037>.

Chatzidimitriou, A. et al. 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)**. [S.l.: s.n.], 2019. p. 26–38. ISSN 1530-0889.

Chatzidimitriou, A. et al. 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)**. [S.l.: s.n.], 2019. p. 26–38. ISSN 1530-0889.

CHATZIDIMITRIOU, A.; GIZOPOULOS, D. Anatomy of microarchitecture-level reliability assessment: Throughput and accuracy. In: **2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)**. IEEE, 2016. Available from Internet: <https://doi.org/10.1109/ispass.2016.7482075>.

CHATZIDIMITRIOU, A. et al. 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, 2017. Available from Internet: <https://doi.org/10.1109/dsn-w.2017.16>.

CHATZIDIMITRIOU, A. et al. Performance-aware reliability assessment of heterogeneous chips. In: **2017 IEEE 35th VLSI Test Symposium (VTS)**. IEEE, 2017. Available from Internet: <https://doi.org/10.1109/vts.2017.7928940>.

CHO, H. et al. 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. Available from Internet: <https://doi.org/10.1145/2463209.2488859>.

COHEN, A. et al. **Inter-Disciplinary Research Challenges in Computer Systems for the 2020s**. USA, 2018.

CONSTANTINESCU, C. Impact of deep submicron technology on dependability of vlsi circuits. In: **Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on**. [S.l.: s.n.], 2002. p. 205–209.

DONGARRA, J.; MEUER, H.; STROHMAIER, E. **ISO26262 Standard**. 2015. Available from Internet: <https://www.iso.org/obp/ui/\#iso:std:iso:26262:-1:ed-1:v1:en>.

FRATIN, V. et al. Code-dependent and architecture-dependent reliability behaviors. In: **2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)**. [S.l.: s.n.], 2018. p. 13–26.

FU, X.; LI, T.; FORTES, J. A. B. **Sim-SODA: A Unified Framework for Architectural Level Software Reliability Analysis**.

GEORGE, N. et al. Transient fault models and AVF estimation revisited. In: **2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)**. IEEE, 2010. Available from Internet: <https://doi.org/10.1109/dsn.2010.5544276>.

GUTHAUS, M. R. et al. 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)**. [S.l.: s.n.], 2001. p. 3–14.

GUTIERREZ, A. et al. Sources of error in full-system simulation. In: **2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)**. IEEE, 2014. Available from Internet: <https://doi.org/10.1109/ispass.2014.6844457>.

HEMSOTH, N. **ARM it the NNSA'S new secret weapon**. 2018. Available from Internet: <https://www.nextplatform.com/2018/11/07/arm-is-the-nnsas-new-secret-weapon/>.

ITURBE, X.; VENU, B.; OZER, E. 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, 2016. Available from Internet: <https://doi.org/10.1109/dft.2016.7684076>.

Iyer, R. K.; Rossetti, D. J. Effect of system workload on operating system reliability: A study on ibm 3081. **IEEE Transactions on Software Engineering**, SE-11, n. 12, p. 1438–1448, Dec 1985. ISSN 2326-3881.

JEDEC. **Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices**. [S.l.], 2006.

KALIORAKIS, M. et al. MeRLiN. In: **Proceedings of the 44th Annual International Symposium on Computer Architecture - ISCA '17**. ACM Press, 2017. Available from Internet: <https://doi.org/10.1145/3079856.3080225>.

Knight, J. C. Safety critical systems: challenges and directions. In: **Proceedings of the 24th International Conference on Software Engineering. ICSE 2002**. [S.l.: s.n.], 2002. p. 547–550.

LEVEUGLE, R. et al. Statistical fault injection: Quantified error and confidence. In: **2009 Design, Automation Test in Europe Conference Exhibition**. [S.l.: s.n.], 2009. p. 502–506. ISSN 1530-1591.

LUCAS, R. Top ten exascale research challenges. In: **DOE ASCAC Subcommittee Report**. [S.l.: s.n.], 2014.

MAHATME, N. et al. Comparison of Combinational and Sequential Error Rates for a Deep Submicron Process. **Nuclear Science, IEEE Transactions on**, IEEE Press, v. 58, n. 6, p. 2719–2725, 2011.

MANIATAKOS, M. et al. Instruction-level impact analysis of low-level faults in a modern microprocessor controller. **IEEE Transactions on Computers**, Institute of Electrical and Electronics Engineers (IEEE), v. 60, n. 9, p. 1260–1273, Sep 2011. Available from Internet: <https://doi.org/10.1109/tc.2010.60>.

MARTÍNEZ-ÁLVAREZ, A. et al. A hardware-software approach for on-line soft error mitigation in interrupt-driven applications. **IEEE Trans. Dependable Sec. Comput.**, v. 13, n. 4, p. 502–508, 2016. Available from Internet: <https://doi.org/10.1109/TDSC.2014.2382593>.

MARWEDEL, P. **Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems**. [S.l.: s.n.], 2010. ISBN 9789400702561.

MUKHERJEE, S. S. et al. 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: IEEE Computer Society, 2003. p. 29–. ISBN 0-7695-2043-X.

NGUYEN, H. T. et al. Chip-level soft error estimation method. **IEEE Transactions on Device and Materials Reliability**, v. 5, n. 3, p. 365–381, Sept 2005. ISSN 1530-4388.

NOH, J. et al. Study of neutron soft error rate (ser) sensitivity: Investigation of upset mechanisms by comparative simulation of finfet and planar mosfet srams. **Nuclear Science, IEEE Transactions on**, v. 62, n. 4, p. 1642–1649, Aug 2015. ISSN 0018-9499.

OLIVEIRA, A. B. d. **Applying dual core lockstep in embedded processors to mitigate radiation induced soft errors**. Dissertation (Master), 2017. Available from Internet: <http://hdl.handle.net/10183/173785>.

OLIVEIRA, A. B. de; RODRIGUES, G. S.; KASTENSMIDT, F. L. 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**. New York, NY, USA: ACM, 2017. (SBCCI '17), p. 84–89. ISBN 978-1-4503-5106-5. Available from Internet: <http://doi.acm.org/10.1145/3109984.3110008>.

OLIVEIRA, D. et al. Experimental and analytical study of xeon phi reliability. In: **Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis**. New York, NY, USA: ACM, 2017. (SC '17), p. 28:1–28:12. ISBN 978-1-4503-5114-0.

OLIVEIRA, D. et al. Radiation-Induced Error Criticality in Modern HPC Parallel Accelerators. In: ACM. **Proceedings of 21st IEEE Symp. on High Performance Computer Architecture (HPCA)**. [S.l.], 2017.

RODRIGUES, G. S.; KASTENSMIDT, F. L. Soft error analysis at sequential and parallel applications in ARM cortex-a9 dual-core. In: **2016 17th Latin-American Test Symposium (LATS)**. IEEE, 2016. Available from Internet: <https://doi.org/10.1109/latw.2016.7483359>.

ROSA, F. et al. 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, 2015. Available from Internet: <https://doi.org/10.1109/dft.2015.7315164>.

SANTINI, T. et al. Effectiveness of software-based hardening for radiation-induced soft errors in real-time operating systems. In: KNOOP, J. et al. (Ed.). **Architecture of Computing Systems - ARCS 2017**. Cham: Springer International Publishing, 2017. p. 3–15. ISBN 978-3-319-54999-6.

SANTINI, T. et al. Reliability analysis of operating systems and software stack for embedded systems. **IEEE Transactions on Nuclear Science**, v. 63, n. 4, p. 2225–2232, Aug 2016. ISSN 0018-9499.

Santini, T. et al. 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)**. [S.l.: s.n.], 2015. p. 49–58. ISSN null.

Santos, F. F. d. et al. Analyzing and increasing the reliability of convolutional neural networks on gpus. **IEEE Transactions on Reliability**, v. 68, n. 2, p. 663–677, June 2019. ISSN 1558-1721.

SEIFERT, N.; ZHU, X.; MASSENGILL, L. W. Impact of scaling on soft-error rates in commercial microprocessors. **Nuclear Science, IEEE Transactions on**, IEEE, v. 49, n. 6, p. 3100–3106, 2002.

SNIR, M. et al. Addressing failures in exascale computing. **International Journal of High Performance Computing Applications**, SAGE Publications, p. 1–45, 2014.

SRIDHARAN, V.; KAELI, D. R. Using hardware vulnerability factors to enhance avf analysis. In: **Proceedings of the 37th Annual International Symposium on Computer Architecture**. New York, NY, USA: ACM, 2010. (ISCA '10), p. 461–472. ISBN 978-1-4503-0053-7. Available from Internet: <http://doi.acm.org/10.1145/1815961.1816023>.

SUH, J.; ANNAVARAM, M.; DUBOIS, M. MACAU: A markov model for reliability evaluations of caches under single-bit and multi-bit upsets. In: **IEEE International Symposium on High-Performance Comp Architecture**. IEEE, 2012. Available from Internet: <https://doi.org/10.1109/hpca.2012.6168940>.

Vallero, A.; Carelli, A.; Di Carlo, S. Trading-off reliability and performance in fpga-based reconfigurable heterogeneous systems. In: **2018 13th International Conference on Design Technology of Integrated Systems In Nanoscale Era (DTIS)**. [S.l.: s.n.], 2018. p. 1–6. ISSN null.

Vallero, A. et al. Multi-faceted microarchitecture level reliability characterization for nvidia and amd gpus. In: **2018 IEEE 36th VLSI Test Symposium (VTS)**. [S.l.: s.n.], 2018. p. 1–6. ISSN 2375-1053.

WANG, G.; LIU, S.; SUN, J. A dynamic partial reconfigurable system with combined task allocation method to improve the reliability of fpga. **Microelectronics Reliability**, v. 83, p. 14 – 24, 2018. ISSN 0026-2714. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0026271418300350>.

WANG, N. J.; MAHESRI, A.; PATEL, S. J. Examining ACE analysis reliability estimates using fault-injection. **ACM SIGARCH Computer Architecture News**, Association for Computing Machinery (ACM), v. 35, n. 2, p. 460, Jun 2007. Available from Internet: <https://doi.org/10.1145/1273440.1250719>.

Wilkening, M. et al. Evaluating the resilience of parallel applications. In: **2018 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)**. [S.l.: s.n.], 2018. p. 1–6. ISSN 1550-5774.

ZIEGLER, J. F.; PUCHNER, H. **SER–history, Trends and Challenges: A Guide for Designing with Memory ICs**. [S.l.]: Cypress, 2010.

APPENDIX A — PAPERS PUBLISHED AND SUBMITTED

A.1 Paper published

- *Chatzidimitriou, Athanasios & *Bodmann, Pablo & Papadimitriou, George & Gizopoulos, Dimitris & Rech, Paolo. (2019). Demystifying Soft Error Assessment Strategies on ARM CPUs: Microarchitectural Fault Injection vs. Neutron Beam Experiments. 10.1109/DSN.2019.00018. *The first two authors equally worked on this paper.
    - Published at DSN 2019

A.2 Paper submitted

- *Bodmann, Pablo & *Chatzidimitriou, Athanasios & Papadimitriou, George & Gizopoulos, Dimitris & Rech, Paolo. (2019) Understanding the Impact of Cores Integration and Operating System on Arm Processors Reliability
    - Submitted at DSN 2020