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

LUCAS ANTUNES TAMBARA

Analyzing the Impact of Radiation-induced Failures in All Programmable System-on-Chip Devices

Thesis presented in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Microelectronics

Advisor: Dr. Fernanda Lima Kastensmidt

Porto Alegre
2017

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Antunes Tambara, Lucas
Analyzing the Impact of Radiation-induced Failures in All Programmable System-on-Chip Devices / Lucas Antunes Tambara. – 2017.
192 f.

Orientadora: Fernanda Lima Kastensmidt.

Tese (Doutorado) -- Universidade Federal do Rio Grande do Sul, Instituto de Informática, Programa de Pós-Graduação em Microeletrônica. Porto Alegre, BR-RS, 2017.

1. APSoC. 2. SRAM-based FPGA. 3. Radiation effects. I. Lima Kastensmidt, Fernanda, orient. II. Analyzing the Impact of Radiation-induced Failures in All Programmable System-on-Chip Devices.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Rui Vicente Oppermann
Vice-Reitor: Prof. Jane Fraga Tutikian
Pró-Reitor de Pós-Graduação: Prof. Celso Giannetti Loureiro Chaves
Diretor do Instituto de Informática: Prof. Luís da Cunha Lamb
Coordenador do PGMICRO: Prof. Fernanda Lima Kastensmidt
Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

ACKNOWLEDGMENTS

I would like to start by thanking my wife, Renata Borella Venturini, for always supporting me unconditionally and for always walking along by my side, no matter the country.

To my parents, Lúcia Regina Antunes Tambara and Pedro Jorge Tadiello Tambara, for my existence, unconditional support, and everything else that involve the relation between parents and sons.

To UFRGS, Institute of Informatics, PGMICRO, and the Brazilian research agencies CAPES, CNPq, and FAPERGS, for the financial support and for putting their facilities at my disposal so I could develop my research.

To my advisor, Fernanda Lima Kastensmidt, for the confidence, lessons, and patience. Thank you for pushing me forward and encouraging me in my academic and personal decisions.

To Matteo Sonza Reorda, from Politecnico di Torino, and Salvatore Danzeca, from CERN, for having accepted me as a visiting researcher in their research institutes during my Ph.D.

To Paolo Rech, Nilberto Medina, Nemitala Added, Marcilei Guazzelli, and several other professors and researchers that contributed to this thesis.

To all my colleagues from laboratories 230/232 from UFRGS and others that are already in other rooms, cities, or countries.

To all of you, my sincere thanks.

ABSTRACT

The recent advance of the semiconductor industry has allowed the integration of complex components and systems' architectures into a single silicon die. Nowadays, state-of-the-art FPGAs include not only the programmable logic fabric but also hard-core parts, such as hard-core general-purpose processors, dedicated processing blocks, interfaces to various peripherals, on-chip bus structures, and analog blocks. These new devices are commonly called of All Programmable System-on-Chip (APSoC) devices. One of the major concerns about radiation effects on APSoCs is that radiation-induced errors may have different probability and criticality in their heterogeneous hardware parts at both device and design levels. For this reason, this work performs a deep investigation about the radiation effects on APSoCs and the correlation between hardware and software resources sensitivity in the overall system performance. Several static and dynamic experiments were performed on different hardware parts of an APSoC to better understand the trade-offs between reliability and performance of each part separately. Results show that there is a trade-off between design cross section and performance to be analyzed when developing a system on an APSoC. Therefore, today it is mandatory to take into account each design option available and all the parameters of the system involved, such as the execution time and the workload of the system, and not only its cross section. As an example, results show that it is possible to increase the performance of a system up to 5,000 times by changing its architecture with a small impact in cross section (increase up to 8 times), significantly increasing the operational reliability of the system. This work also proposes a reliability analysis flow based on fault injection for estimating the reliability trend of hardware-only designs, software-only designs, and hardware and software co-designs. It aims to accelerate the search for the design scheme with the best trade-off between performance and reliability among the possible ones. The methodology takes into account four groups of parameters, which are the following: area resources and performance; the number of output errors and critical bits; radiation measurements, such as static and dynamic cross sections; and, Mean Workload Between Failures. The obtained results show that the proposed flow is a suitable method for estimating the reliability trend of system designs on APSoCs before radiation experiments.

Keywords: APSoC, SRAM-based FPGA, Processor, Reliability, Radiation effects, Radiation experiment, Fault injection, SEE, SEU, Execution time, Trade-off.

Avaliação do Impacto de Falhas Induzidas pela Radiação em Dispositivos Sistemas-em-Chip Totalmente Programáveis

RESUMO

O recente avanço da indústria de semicondutores tem possibilitado a integração de componentes complexos e arquiteturas de sistemas dentro de um único chip de silício. Atualmente, FPGAs do estado da arte incluem, não apenas a matriz de lógica programável, mas também outros blocos de hardware, como processadores de propósito geral, blocos de processamento dedicado, interfaces para vários periféricos, estruturas de barramento internas ao chip, e blocos analógicos. Estes novos dispositivos são comumente chamados de Sistemas-em-Chip Totalmente Programáveis (APSoCs). Uma das maiores preocupações acerca dos efeitos da radiação em APSoCs é o fato de que erros induzidos pela radiação podem ter diferente probabilidade e criticalidade em seus blocos de hardware heterogêneos, em ambos os níveis de dispositivo e projeto. Por esta razão, este trabalho realiza uma investigação profunda acerca dos efeitos da radiação em APSoCs e da correlação entre a sensibilidade de recursos de hardware e software na performance geral do sistema. Diversos experimentos estáticos e dinâmicos inéditos foram realizados nos blocos de hardware de um APSoC a fim de melhor entender as relações entre confiabilidade e performance de cada parte separadamente. Os resultados mostram que há um comprometimento a ser analisado entre o desempenho e a área de choque de um projeto durante o desenvolvimento de um sistema em um APSoC. Desse modo, é fundamental levar em consideração cada opção de projeto disponível e todos os parâmetros do sistema envolvidos, como o tempo de execução e a carga de trabalho, e não apenas a sua seção de choque. Exemplificativamente, os resultados mostram que é possível aumentar o desempenho de um sistema em até 5.000 vezes com um pequeno aumento na sua seção de choque de até 8 vezes, aumentando assim a confiabilidade operacional do sistema. Este trabalho também propõe um fluxo de análise de confiabilidade baseado em injeções de falhas para estimar a tendência de confiabilidade de projetos somente de hardware, de software, ou de hardware e software. O fluxo objetiva acelerar a procura pelo esquema de projeto com a melhor relação entre performance e confiabilidade dentre as opções possíveis. A metodologia leva em consideração quatro grupos de parâmetros, os quais são: recursos e performance; erros e bits críticos; medidas de radiação, tais como seções de choque estáticas e dinâmicas; e, carga de trabalho média entre falhas. Os resultados obtidos mostram

que o fluxo proposto é um método apropriado para estimar tendências de confiabilidade de projeto de sistemas em APSoCs antes de experimentos com radiação.

Palavras-chave: APSoC, FPGA baseado em SRAM, Processador, Confiabilidade, Efeitos da radiação, Experimentos com radiação, Injeção de falhas, SEE, SEU, Tempo de execução, Correlação.

LIST OF FIGURES

Figure 2.1 – Bock diagram of a processor with a SIMD engine embedded (a) and a simplified block diagram of a SIMD engine (b). 22
Figure 2.2 - Processor models: (a) Symmetric, (b) Asymmetric, and (c) Heterogeneous........ 24
Figure 2.3 – Generic architecture of an APSoC. 26
Figure 2.4 – Block diagram of the Zynq-7000 (XILINX, 2015d). 28
Figure 2.5 – Block diagram of the PS part of the Zynq-7000 (XILINX, 2015d). 29
Figure 2.6 – Zynq-7000's memory hierarchy. 30
Figure 2.7 – Abstraction layers of a generic SRAM-based FPGA (TARRILLO, 2014). 34
Figure 2.8 – Basic structure of the PL part of the Zynq-7000 (CROCKETT, 2014). 34
Figure 2.9 – Example of a 3-input LUT implementing a majority voter. 35
Figure 2.10 – Block diagram of a Xilinx 7-Series slice (XILINX, 2014). 36
Figure 2.11 – Relationship between CLBs and Slices in Xilinx 7-Series FPGAs (XILINX, 2014). 36
Figure 2.12 – Example of a generic Xilinx FPGA floorplan and frame structure. 37
Figure 2.13 – AXI interconnects and interfaces connecting the PS and PL parts of the Zynq-7000. 39
Figure 2.14 – EMIO interface between the PS and PL parts of the Zynq-7000. 40
Figure 2.15 – Obtained best methods for transferring different numbers of data items between PS and PL in Zynq-7000 (SILVA, SKLYAROV, SKLIAROVA, 2015). 41
Figure 2.15 – Example of parallel versus sequential execution. 43
Figure 2.16 – Estimated overall system speed-up provided by the addition of different numbers of hardware accelerators. 44
Figure 2.17 – Example of a control and data flow extraction during the HLS process. 45
Figure 2.18 – The essential three HLS phases and their relations. 45
Figure 2.19 – Xilinx Vivado HLS design flow. 46
Figure 2.20 – Function pipelining behavior. 48
Figure 2.21 – Loop unrolling behavior. 48
Figure 2.22 – Loop merging behavior. 49
Figure 2.23 – Influence of the hardware accelerator granularity on the execution time of a system (LAFOND, LILIUS, 2008). 53
Figure 2.24 – Impact of the cache architecture on the average execution time (speed-up) relative to a baseline system, which employs a single accelerator running sequentially using a 1-way (direct-mapped) 2KB cache with a 32-byte line size (CHOI et al., 2012). 54
Figure 2.25 – Processing bandwidth comparison of different acceleration methods in Zynq-7000. Data size sweeps from 4 KB to 2048 KB (SADRI et al., 2013). 54
Figure 3.1 – Fault, error, and failure propagation. 56
Figure 3.2 – The space environment and its sources of ionizing particles. 58
Figure 3.3 – Particle cascade generated from cosmic rays in the Earth's atmosphere (STEFAN, 2001). 61
Figure 3.4 – General view of the CERN/LHC/ATLAS detector (ATLAS, 2016). 63
Figure 3.5 – Electron-hole pairs track generated by an ionized particle in a CMOS transistor (SOOS, 2009) on the left and the charge collection mechanism in an inverter gate on the right. 65
Figure 3.6 – Main radiation effects on programmable circuits. Adapted from (QUINN et al., 2015a). 65
Figure 3.7 – Example of Single Event Upset (KASTENSMIDT, CARRO, REIS, 2006). 66
Figure 3.8 – MCU events as a percentage of SEUs for different families of Xilinx FPGAs (WIRTHLIN et al., 2014a). 66
Figure 3.9 – Example of a Single Event Transient (KASTENSMIDT, CARRO, REIS, 2006). 67
Figure 3.10 – View of the surface of a Zynq-7000 device, part XC7Z020-CLG484. 68
Figure 3.11 – Possible effects of SEEs in Zynq-7000 and similar APSoCs. 69
Figure 3.12 – Possible effects of SEUs in SRAM-based FPGAs (TONFAT, 2015). 70
Figure 3.13 – Possible effects of SEEs in processors. 72
Figure 4.1 – CERN's CHARM particle spectra. Adapted from (ALÍA, 2016). 75

Figure 4.2 – BRAM of a Xilinx Virtex-5 FPGA scanned during a laser test campaign (KASTENSMIDT et al., 2014). 76
Figure 4.3 – Block diagram of the fault injection platform used in this thesis in (a) and an example of the FPGA floorplanning with the fault injector and the DUT placed in (b). 79
Figure 4.4 – Flow diagram illustrating the hardware fault injection procedure in the Zynq-7000's PL and Artix-7. 80
Figure 4.5 – Platform setup of the software fault injector. 81
Figure 4.6 – Flow diagram illustrating the software fault injection procedure in the Zynq-7000's PS. 82
Figure 4.7 – SEU cross section for static and dynamic tests in a memory (SCHWANK, SHANEYFELT, DODD, 2013). 84
Figure 4.8 – Static test flow diagram. Adapted from (IROM, 2008). 84
Figure 4.9 – Dynamic test flow diagram. Adapted from (IROM, 2008). 87
Figure 4.10 – Neutron cross section per bit for different Xilinx devices (XILINX, 2016c). 91
Figure 4.11 – SEL occurrence in the PL part of Zynq-7000 during heavy ion experiments (AMRBAR et al., 2015). 92
Figure 4.12 – Heavy ion SEU static cross section for the PL part of the Zynq-7000 configuration memory and BRAMs (ALLEN, IROM, AMRBAR, 2015; AMRBAR et al., 2015). 92
Figure 4.13 – (a) SEU cross section and (b) SET and SEFI cross section in neutrons for different SoCs (QUINN et al., 2014b). 93
Figure 4.14 – Measured Zynq-7000 SDC and SEFI cross sections of applications running bare to the metal (Bare SDC and Bare SEFI) and on the top of Linux (Linux SDC and Linux SEFI) compared to the expected standalone Linux SEFI cross section (dashed line). (SANTINI et al., 2016). 95
Figure 4.15 – Heavy ion SEFI cross sections for the MSS, GPIO, and FIC blocks of the SmartFusion2 (REZZAK et al., 2015). 96
Figure 5.1 – Integrated block diagram of the static tests architectures. 98
Figure 5.2 – Heavy ion experiment setup mounted outside and inside the vacuum chamber at the LAFN-USP. 101
Figure 5.3 – Microscopic section of a Zynq-7000 device, part XC7Z020-CLG484. 102
Figure 4.4 – Cross section results from the heavy ion irradiations in the embedded memories OCM and BRAM of the Zynq-7000. 104
Figure 5.5 – Obtained SDC and SEFI cross sections from the heavy ion irradiations. 107
Figure 5.6 – Comparison between the obtained SDC cross section and MWBF values from the heavy ion irradiations. 109
Figure 6.1 – Cross section results from the heavy ion irradiations in the configuration memory of the PL part of Zynq-7000. 113
Figure 6.2 – Cross section results from the proton irradiations in the configuration memory of the PL part of Zynq-7000. 113
Figure 6.3 – Proposed reliability analysis flow for hardware-only designs. 115
Figure 6.4 – Architecture of the (a) processor-based design and the (b) HLS-based design. 118
Figure 6.5 – SDC, SEFI, and TOTAL dynamic cross section results obtained for the MxM designs from both fault injections and radiation experiments. 123
Figure 6.6 – SDC, SEFI, and TOTAL dynamic cross section results obtained for the AES designs both fault injections and radiation experiments. 124
Figure 6.7 – SDC, SEFI, and TOTAL dynamic cross section results obtained for the ADPCM designs both fault injections and radiation experiments. 125
Figure 6.8 – SDC, SEFI, and TOTAL MWBF results obtained for the MxM designs both fault injections and radiation experiments. 127
Figure 6.9 – SDC, SEFI, and TOTAL MWBF results obtained for the AES designs both fault injections and radiation experiments. 128
Figure 6.10 – SDC, SEFI, and TOTAL MWBF results obtained for the ADPCM designs both fault injections and radiation experiments. 129
Figure 6.11 – Comparison between SDC, SEFI, and TOTAL dynamic cross sections and their respective performance rates for the MxM designs from fault injection and radiation experiment results. 130

Figure 6.12 – Comparison between SDC, SEFI, and TOTAL dynamic cross sections and their respective performance rates for the AES designs from fault injection and radiation experiment results. 131
Figure 6.13 – Comparison between SDC, SEFI, and TOTAL dynamic cross sections and their respective performance rates for the ADPCM designs from fault injection and radiation experiment results. 132
Figure 7.1 – Block diagram of the case-study multiprocessor-based heterogeneous system. 137
Figure 7.2 – Matrix multiplication algorithm. 138
Figure 7.3 – Generic representation of the case-study benchmark algorithms. 142
Figure 7.4 – Block diagram of the architecture developed for evaluating the case-study hardware and software co-designs in Zynq-7000. 142
Figure 7.5 - Heavy ion experiment setup mounted at the new beam line of the LAFN-USP. 145
Figure 7.6 – SDC, SEFI, and TOTAL dynamic cross section results obtained for the MxM designs from radiation experiments. 147
Figure 7.7 – SDC, SEFI, and TOTAL dynamic cross section results obtained for the AES designs from radiation experiments. 148
Figure 7.8 – SDC, SEFI, and TOTAL MWBF results obtained for the MxM designs from radiation experiments. 149
Figure 7.9 – SDC, SEFI, and TOTAL MWBF results obtained for the AES designs from radiation experiments. 150
Figure 7.10 – Comparison between SDC, SEFI, and TOTAL dynamic cross sections and their respective performance rates for the MxM designs from radiation experiment results. 151
Figure 7.11 – Comparison between SDC, SEFI, and TOTAL dynamic cross sections and their respective performance rates for the AES designs from radiation experiment results. 152
Figure 8.1 – Proposed reliability analysis flow for hardware and software co-designs. 155
Figure 8.1 – Block diagrams of the architectures developed with the hardware (a) and software (b) fault injectors embedded. 158
Figure 8.2 – Comparison between the total SDC, SEFI, and TOTAL (SDC + SEFI) AVF values obtained from both hardware and software fault injection campaigns with the respective dynamic cross sections obtained from radiation experiments for the MxM benchmark. 161
Figure 8.3 – Comparison between the total SDC, SEFI, and TOTAL (SDC + SEFI) AVF values obtained from both hardware and software fault injection campaigns with the respective dynamic cross sections obtained from radiation experiments for the AES benchmark. 162
Figure 8.4 – Comparison between the total SDC, SEFI, and TOTAL (SDC + SEFI) MWBF values obtained from both hardware and software fault injection campaigns with the respective MWBF values obtained from radiation experiments for the MxM benchmark. 164
Figure 8.5 – Comparison between the total SDC, SEFI, and TOTAL (SDC + SEFI) MWBF values obtained from both hardware and software fault injection campaigns with the respective MWBF values obtained from radiation experiments for the AES benchmark. 165
Figure 8.6 – Comparison between the total SDC, SEFI, and TOTAL (SDC + SEFI) Performance rate values obtained from both hardware and software fault injection campaigns with the respective MWBF values obtained from radiation experiments for the MxM benchmark. 166
Figure 8.7 – Comparison between the total SDC, SEFI, and TOTAL (SDC + SEFI) Performance rate values obtained from both hardware and software fault injection campaigns with the respective MWBF values obtained from radiation experiments for the AES benchmark. 167

LIST OF TABLES

Table 2.1 – Commercially-available APSoCs characteristics. 26
Table 5.1 – Characteristics if the heavy ion beams used at LAFN-USP. 100
Table 5.2 – Characteristics of the ROSCOSMOS heavy ion facility. 101
Table 5.3 – Characteristics if the heavy ion beams used at ROSCOSMOS. 102
Table 5.4 – Characteristics of the Russian proton facility. 103
Table 5.5 – Heavy ion and proton test schemes performed in the embedded memories of the PS part of Zynq-7000. 103
Table 5.6 – Application information running on Zynq-7000's ARM Cortex-A9 Core 0 with different cache schemes (D = Disabled, E = Enabled). 106
Table 5.7 – Obtained SDC and SEFI cross sections from the heavy ion irradiations. 107
Table 5.8 – Obtained MTBF, MEBF, and MWBF from the heavy ion irradiations. 108
Table 6.1 – Heavy ion and proton test schemes performed in the configuration memory of the PL part of Zynq-7000. 112
Table 6.2 – Optimization strategies applied in each HLS-based design for each benchmark program. 119
Table 6.3 – Resource usage and performance results of each case-study design. 120
Table 7.1 – Resource usage and performance of each case-study architecture implemented. 137
Table 7.2 – Experimental results from the neutron radiation tests for the four case-studied system in Zynq-7000. 139
Table 7.3 – Case-study hardware and software co-designs and their respective configurations, resource usage, and performance results. 144
Table 8.1 – Number of critical bits and AVF of the hardware part of each hardware and software co-design obtained from the hardware fault injection campaigns. 159
Table 8.2 – AVF of the software part of each hardware and software co-design obtained from the software fault injection campaigns. 160

LIST OF ABBREVIATIONS AND ACRONYMS

ABFT	Algorithm-Based Fault Tolerance
ACP	Accelerator Coherency Port
ADAS	Advanced Driver Assistance Systems
ADPCM	Adaptive Differential Pulse-Code Modulation
AES	Advanced Encryption Standard
ALICE	A Large Ion Collider Experiment
AMBA	Advanced Microcontroller Bus Architecture
AMP	Asymmetric Multi-Processing
APSoC	All Programmable System-on-Chip
APU	Application Processing Unit
ARM	ARM Cortex-A9
ASIC	Application Specific Integrated Circuit
ATLAS	A Toroidal LHC Apparatus
AXI	Advanced Extensible Interface
AXI-S	Advanced Extensible Interface Stream
AVF	Architectural Vulnerability Factor
BCE	Base-Core Equivalent
BRAM	Block Random Access Memory
CERN	European Organization for Nuclear Research
CHARM	CERN High Energy Accelerator Mixed-Field
CLB	Configurable Logic Block
CMOS	Complementary Metal Oxide Semiconductor
CMS	Compact Muon Solenoid
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CRAM	Configuration Memory
CRC	Cyclic Redundancy Check
D	Disabled
DD	Displacement Damage
DDR	Double Data Rate
DFT	Discrete Fourier Transform

DMA	Direct Memory Access
DRAM	Dynamic Random Access Memory
DSP	Digital Signal Processing
DUT	Design Under Test
DUT	Device Under Test
E	Enabled
ECC	Error Correcting Code
EMIO	Extended Multiplexed Input/Output
ESA	European Space Agency
EXP	Radiation Experiment
FF	Flip-Flop
FFT	Fast Fourier Transform
FI	Fault Injector
FIC	Fabric Interface Controller
FIFO	First In, First Out
FIT	Failure in Time
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
FSM	Finite State Machine
GEO	Geostationary Orbit
GIC	General Interrupt Controller
GP	General Purpose
GPGPU	General Purpose Graphics Processing Unit
GPIO	General Purpose I/O
GPR	General-Purpose Register
GPU	Graphics Processing Unit
HDL	Hardware Description Language
HEP	High-Energy Physics
HLS	High-Level Synthesis
HP	High Performance
I/O	Input/Output
ICAP	Internal Configuration Access Port
IOB	Input/Output Block
ISA	Instruction Set Architecture

ITRS	International Technology Roadmap for Semiconductors
JPEG	Joint Photographics Experts Group
JTAG	Joint Test Action Group
L1	Level 1
L1D	Level 1 Data
L1I	Level 1 Instruction
L2	Level 2
LAFN	Laboratório Aberto de Física Nuclear
LANL	Los Alamos National Laboratory
LANSCE	Los Alamos National Science Center
LEO	Low Earth Orbit
LET	Linear Energy Transfer
LHC	Large Hadron Collider
LHCb	LHC-Beauty
LLVM	Low Level Virtual Machine
LUT	Look-Up Table
MB	Microblaze
MBU	Multiple Bit Upset
MCU	Multiple Cell Upset
MEBF	Mean Execution Between Failures
MEPhI	Moscow Engineering Physics Institute
MIO	Multiplexed Input/Output
MIPS	Microprocessor without Interlocked Pipeline Stages
MMU	Memory Management Unit
MSR	Millimeter Square Radian
MSS	Microcontroller Subsystem
MTBF	Mean Time Between Failures
MxM	Matrix Multiplication
MWBF	Mean Workload Between Failures
NASA	National Aeronautics and Space Administration
NSREC	Nuclear and Space Radiation Effects Conference
OCM	On-Chip Memory
OS	Operating System
PC	Program Counter

PCR	Primary Cosmic Radiation
PL	Programmable Logic
PLL	Phase-Locked Loop
PoR	Power-on Reset
PS	Processing System
PSI	Paul Scherrer Institut
RAL	Rutherford Appleton Laboratory
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
ROM	Read-Only Memory
ROSCOSMOS	Russian Federal Space Agency
RTL	Register Transfer Level
RTOS	Real-Time Operating System
SAA	South Atlantic Anomaly
SCU	Snoop Control Unit
SDC	Silent Data Corruption
SECDED	Single-Error Correct/Double-Error Detect
SEE	Single Event Effect
SEFI	Single Event Functional Interrupt
SEL	Single Event Latchup
SEM	Soft Error Mitigation
SER	Soft Error Rate
SET	Single Event Transient
SEU	Single Event Upset
SIHFT	Software Implemented Hardware Fault Tolerance
SIMD	Single Instruction Multiple Data
SMP	Symmetric Multi-Processing
SoC	System-on-Chip
SRAM	Static Random Access Memory
TID	Total Ionizing Dose
TMR	Triple Modular Redundancy
TSMC	Taiwan Semiconductor Manufacturing Company
UART	Universal Asynchronous Receiver/Transmitter
USA	United States of America

USP	Universidade de São Paulo
WR	Worst Result

CONTENTS

1 INTRODUCTION 16

1.1 Objectives and contributions 18

1.2 Thesis organization 20

2 HETEROGENEOUS PROGRAMMABLE HARDWARE 22

2.1 All Programmable System-on-Chip devices 25
2.1.1 Xilinx Zynq-7000 APSoC 27
2.1.1.1 The Processing System (PS) 28
2.1.1.2 The Programmable Logic (PL) 33
2.1.1.3 The interfaces between PS and PL 38

2.2 Hardware/Software co-design 42
2.2.1 High-Level Synthesis 44
2.2.2 Xilinx Vivado High-Level Synthesis 46
2.2.3 Related works about HLS 49

2.3 Implementation metrics 51

2.4 Related works about APSoCs 52

2.5 Summary 54

3 RADIATION EFFECTS ON APSOCS 56

3.1 Fault, error, and failure 56

3.2 Radiation environments 57
3.2.1 Space environment 57
3.2.2 Terrestrial environment 59
3.2.3 Particle accelerators environment 61

3.3 Radiation effects on integrated circuits 63

3.4 Radiation effects on APSoCs 68

3.5 Summary 73

4 METHODS AND METRICS FOR EVALUATING APSOCS UNDER RADIATION 74

4.1 Accelerated radiation tests 74

4.2 Fault injection by emulation 77
4.2.1 Hardware fault injection platform used 78
4.2.2 Software fault injection platform used 81

4.3 Test methods and metrics 82
4.3.1 Static test method and metrics 84
4.3.2 Dynamic test method and metrics 87

4.4 Related works about APSoCs under radiation 90

4.5 Summary **96**

5 ANALYSING SINGLE EVENT EFFECTS ON THE PS PART OF ZYNQ-7000 .... **97**

5.1 Static tests **97**
5.1.1 Tests procedures 97
5.1.2 Tests setups 99
5.1.3 Tests results 103

5.2 Dynamic tests **104**
5.2.1 Tests procedures 104
5.2.2 Tests setup 106
5.2.3 Tests results 106

6 ANALYSING SINGLE EVENT EFFECTS ON THE PL PART OF ZYNQ-7000 .... **110**

6.1 Static tests **111**
6.1.1 Tests procedures 111
6.1.2 Tests setups 111
6.1.3 Tests results 112

6.2 Dynamic tests and proposed reliability analysis for hardware-only designs **113**
6.2.1 Proposed reliability analysis for hardware-only designs 114
6.2.1.1 Resources and performance 115
6.2.1.2 Errors and critical bits 116
6.2.1.3 Radiation measurements 116
6.2.1.4 Mean Workload Between Failures 116
6.2.1.5 Xilinx analysis tools 117
6.2.1.6 Fault injection method and analysis 117
6.2.2 Case-study designs and resources and performance results 117
6.2.3 Cross section and MWBF results 120

6.3 Summary **133**

7 EXPLORING BOTH PS AND PL PARTS OF ZYNQ-7000 UNDER SINGLE EVENT EFFECTS **135**

7.1 Reliability of hard- and soft-cores heterogeneous processing in the Zynq-7000 .... **135**

7.2 Reliability of heterogeneous processing through hardware and software co-designs in the Zynq-7000 .... **141**
7.2.1 Case-study designs and resources and performance evaluations 141
7.2.2 Cross section, MWBF, and performance evaluation 145

7.3 Summary **153**

8 PROPOSED RELIABILITY ANALYSIS FOR HARDWARE AND SOFTWARE CO-DESIGNS **154**

8.1 Analysis flow **154**

8.2 Case-study designs **157**

8.3 Obtained results **158**

8.4 Summary **167**

9 CONCLUDING REMARKS **169**

9.1 Main contributions **169**

9.1.1 Extensive review about APSoCs, possible radiation effects on them, and the methods and metrics for evaluating them under radiation 169
9.1.2 Original static data about Xilinx Zynq-7000 under radiation 170
9.1.3 Original dynamic analysis and data about Xilinx Zynq-7000 under radiation 170
9.1.4 Reliability analysis flow for hardware-only designs, software-only designs, and hardware and software co-designs 171

9.2 Future works 171
9.2.1 Completing the static measurements of Zynq-7000 171
9.2.2 Improving the reliability analysis flow 172
9.2.3 Analyzing the use of fault-tolerant techniques in APSoCs 172
9.2.4 Evaluation of other APSoCs 173

9.3 Publications 173
9.3.1 Book chapters 173
9.3.2 Journals 175
9.3.3 Conferences and workshops 176

REFERENCES 180

1 INTRODUCTION

The recent advances in silicon technology have allowed the integration of complex components and systems' architectures into a single silicon die. Today, state-of-the-art complex embedded systems include Field Programmable Gate Array (FPGA) together with hard-core parts, such as general-purpose embedded processors (hereafter shortened to only "processors"), dedicated processing blocks, interfaces to various peripherals, on-chip bus structures, and analog blocks. These new devices are commonly called All Programmable Systems-on-Chip (APSoCs) or, more generically, Heterogeneous Hardware.

APSoCs are designed to provide higher system performances and programmable flexibility at lower costs compared to standard FPGAs and processors. According to ITRS (2013), heterogeneous architectures such as APSoCs will dominate the next generation of computing systems. In general, APSoCs are composed of two main parts: the Programmable Logic (PL), which is basically an embedded FPGA, and the Processing System (PS), which is formed around of a hard-core processor. The PL is adopted to implement high-speed logic, arithmetic, data processing subsystems, etc. The PS supports software routines and operating systems. The overall functionality and workload of any system design can then be appropriately distributed between hardware and software. Some recent examples of commercially available APSoCs are Zynq-7000 (XILINX, 2015d) from Xilinx, SmartFusion (MICROSEMI, 2015a) and SmartFusion2 (MICROSEMI, 2015b) from Microsemi, and Cyclone V (ALTERA, 2015) from Altera.

The mentioned characteristics of being programmable, flexible, and extremely efficient, make APSoCs very suitable and attractive for safety-critical markets such as space, avionics, automotive, biomedical, and high-energy physics experiments. Moreover, Commercial Off-The-Shelf (COTS) products have been widely employed in these safety-critical areas in recent years. The Large Hadron Collider (LHC) at the European Organization for Nuclear Research (CERN) is a clear example. There are several areas of LHC in which commercial programmable devices not specifically designed to be radiation-tolerant are used (MUSA, 2008). As an example, ALICE (A Large Ion Collider Experiment), which is one of the LHC experiments, makes use of hundreds of SmartFusion2 devices in its Time Projection Chamber (TPC). The TPC is the main particle tracking detector in the central barrel of ALICE. The CHREC Space Processor (CSPv1) is an example of APSoC employment in space applications (RUDOLPH et al., 2014). CSPv1 is a small computer designed to operate

in low-cost space missions powered by a Zynq-7000. CSPv1 relies on the use of a combination of commercial and radiation-hardened components, in which commercial components perform critical computations but are supervised by the radiation-hardened components. Another example is the APEX-SoC proposed in (ITURBE et al., 2015), which is a generic platform based on a Zynq-7000 device and intended to control science instruments in future NASA missions. At terrestrial level, for example, APSoCs are very suitable to implement the Advanced Driver Assistance Systems (ADAS), aimed at increasing vehicles safety (XILINX, 2016b). In ADAS, the hardware and software combined programmability eases the performance and efficiency optimization distributing between hardware and software operations like sensing, environmental characterization, and feature implementation.

Unfortunately, although APSoCs offers a plethora of advantages, their high complexity and density increase the system's susceptibility to transient errors that are present in the environment, such as the ones caused by radiation. Radiation effects known as Single Event Effects (SEEs) are a well-known issue at device level in standard FPGA (DODD et al., 2010; WIRTHLIN, 2015) and processor-based (DODD et al., 2010) devices. SEEs result from the interaction of high-energy particles with circuit elements in integrated circuits. When a high-energy particle passes through the silicon substrate of a device, charged particles are created as the result of sub-atomic particle collisions. These particles are generated by an ionization trail along the path of the incoming particle. As an example, if a charged particle impacts at or near a transistor junction, the collected charge can temporally charge or discharge the stroke node inducing a transient pulse, known as Single Event Transient (SET). If the SET width of is wide enough, the pulse can propagate through the circuit and be latched by a memory cell. If the SET occurs inside a memory cell such as a latch or a flip-flop, the transient pulse can change the state of that memory cell. This effect is known as bit-flip or Single Event Upset (SEU). With the dimensions of the transistors shrinking to below 28 nm, the operating voltages and the element capacitance decreasing to very low levels, and the clock speed increasing, the concerns about SET and SEU in FPGAs, processors, and APSoCs, have increased in the last years.

One of the most challenging concerns about radiation effects on APSoCs is that radiation-induced errors may have different probability and criticality in the PL and PS parts at both device and design levels. APSoCs are programmed by configuring a large set of SRAM memory cells and, consequently, they are very susceptible to bit-flips. SEUs in the configuration memory bits of the PL part have a persistent effect and reconfiguration is needed to correct them. When an application is executed on the PS, it may mask eventual

SEUs according to the application Architectural Vulnerability Factor (AVF) (MUKHERJEE et al., 2003) and the sensitivity of the resources in use. Memories of the PS part such as L1 and L2 caches, embedded SRAM and Block RAM memories are also very sensitive to SEU, and each one has a distinct sensitivity, which may contribute differently to the overall system failure rate and performance overhead.

At design level, APSoCs enable many possibilities for implementing a system due to their heterogeneous architectures. However, each implementation will impose a different amount of resources usage and a different resource utilization efficiency, which may impact the vulnerability of the system. Therefore, the correlation between hardware and software resources sensitivity and the overall benefits brought to the system is essential to evaluate its efficiency.

1.1 Objectives and contributions

This thesis aims at performing a deep investigation about the radiation effects on APSoCs and the correlation between hardware and software resources sensitivity in the overall system reliability and performance. Therefore, the main two topics addressed in this thesis are:

1. Which is the behavior of an APSoC device under radiation? Moreover, considering a hardware and software co-design implementing a high-performance system that runs on both PS and PL parts of an APSoC like Xilinx Zynq-7000, how much is it necessary to accelerate it to compensate the sensitivity increase and improve the Mean Workload Between Failures (MWBF)?
2. Is it possible to estimate the reliability trend of APSoC-based systems like Velazco, Foucard, and Peronnard (2010) did for FPGAs in the past?

As this thesis shows, state-of-the-art complex devices and technologies such as APSoCs and hardware and software co-designs have created many challenges for the radiation effects field. That is because radiation-induced failures in such devices and architectures may result in a complex chain of effects due to their heterogeneous nature. Additionally, the components testing methodologies have not changed over the years for taking into account such heterogeneity.

Another important critical point is the cross section metric. Today, the growing computational need, whether in a spacecraft or high-energy physics experiments, for example, has pushed the need to deploy high-performance computing in harsh environments, such as satellites and the LHC detectors. Thus, adopting only the cross section metric for estimating the reliability of a device, system, or design, is no longer enough. Besides the sensitivity of a resource, it is also essential to evaluate the benefit that this resource brings to the system. Therefore, to compare the reliability of heterogeneous and high-performance systems such as APSoCs, it is essential to take into account not only the cross section but also at least the execution time and workload of the system. Consequently, one of the main metrics adopted in this thesis is the Mean Workload Between Failures (MWBF), previously introduced in (RECH et al., 2014), but not in the APSoC context. The capability of a system to provide correct data depends on several factors, as sensitive area and the time required to complete computations. The MWBF metric identifies the workload that can be correctly computed by the system before experiencing a failure. Moreover, the MWBF considers all these aspects and is of particular interest in safety-critical applications as it provides the realistic impact of a given APSoC configuration on the system reliability.

The first objective of this thesis is to try to answer the following questions: Which is the behavior of an APSoC device under radiation? Moreover, considering a hardware and software co-design implementing a high-performance system that runs on both PS and PL parts of an APSoC like Zynq-7000, how much is it necessary to accelerate it to compensate the dynamic cross section increase and improve the MWBF? One of the main issues concerning radiation effects in an APSoC is that radiation-induced failures have different probability and criticality in its heterogeneous parts. In addition, at design level, the heterogeneous architecture of an APSoC enables a plethora of possibilities for implementing a project. Each implementation imposes a different amount of resources usage and a different resource utilization efficiency, which impact the vulnerability of the system. Therefore, the correlation between hardware and software resources sensitivity and the overall benefits brought to the system are essential to evaluate the system efficiency.

The second objective of this thesis is to develop a methodology to estimate the reliability trend of APSoC-based systems like Velazco, Foucard, and Peronnard (2010) did in the past for FPGA-based designs. Accelerated radiation tests are mandatory to obtain the sensitivity of the target device by determining its static cross section. However, static cross section significantly overestimates the sensitivity of the final application, as the next chapters show, and accelerated radiation tests are scarce, making prohibitive the test of any design

developed. In the case of FPGAs, in (VELAZCO, FOUCARD, PERONNARD, 2010), authors demonstrated that the dynamic cross section of a design can be predicted combining the static cross section with the results of fault injection campaigns, in which SEUs are emulated by a suitable approach, such as hardware/software fault injection. Base on this context, and together with the fact that APSoC devices have been increasingly used in safety-critical markets, a methodology aiming to estimate the APSoC-based design with the best trade-off between performance and reliability among the ones available becomes fundamental.

1.2 Thesis organization

This thesis is organized as follows:

- Chapter 2 - Heterogeneous programmable hardware: introduces heterogeneous programmable hardwares, Xilinx Zynq-7000 as an example of APSoC and the case-study device of this thesis, hardware/software co-design concepts, implementation metrics, and related works about Zynq-7000 and similar APSoCs;
- Chapter 3 - Radiation effects on APSoCs: introduces the concepts of fault, error, and failure; the main radiation environments; and the radiation effects on integrated circuits and APSoCs;
- Chapter 4 - Methods and metrics for evaluating APSoCs under radiation: presents the main methods and metrics for evaluating APSoCs under radiation and related works about APSoCs under radiation;
- Chapter 5 - Analyzing single event effects on the PS part of the Zynq-7000: presents static and dynamic tests procedures, setups, and results for the Zynq-7000's PS part;
- Chapter 6 - Analyzing single event effects on the PL part of the Zynq-7000: presents static and dynamic tests procedures, setups, and results for the Zynq-7000's PL part. This chapter also proposes a reliability analysis flow for estimating the reliability of hardware-only designs based on fault injections;
- Chapter 7 - Exploring both PS and PL parts of Zynq-7000 under single event effects: presents an analysis of the impact of using both PS and PL parts of Xilinx Zynq-7000 in the overall failure rate of a system;

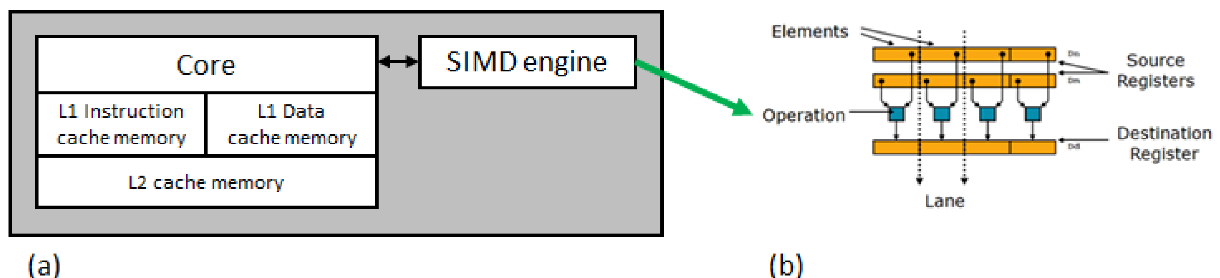
- Chapter 8 - Proposed reliability analysis for hardware and software co-designs: presents the proposed reliability analysis flow for estimating the reliability trend of hardware and software co-designs, the case-study designs, and the obtained results;
- Chapter 9 - Concluding remarks: presents the concluding remarks of this thesis, such as its main contributions and future works.

2 HETEROGENEOUS PROGRAMMABLE HARDWARE

Following Moore's law (MOORE, 1965), the processors' frequency doubled at each every 18 to 24 months until the middle of 2000's decade. However, due to the ever increasing core design complexity of the high performance processors and the power consumption caused by the high frequencies, researches started to look at other strategies to continue increasing systems performance. According to Shen and Pétrot (2011), three possible solutions were proposed until that year. More important, we already can see all of them being commercialized today.

The first solution was to optimize the instruction set for certain application classes. The generalization of the Single Instruction Multiple Data (SIMD) extensions (Fig. 2.1), which first appeared in the general purpose high-performance processors in the early of 90's decade to all processors including the embedded ones (GOODACRE, SLOSS, 2005) after the 2000's decade is an evidence of this trend. The SIMD technique allows multiple data to be processed in one or a few CPU cycles by assuming that registers are considered as vectors of elements of the same data type. A today's example is the NEON engine (ARM, 2015b), which is the SIMD implementation present in ARMv7-A processors (Cortex-A9 family). The drawback of this solution is that these resources can accelerate only part of the application, which make them useless for other parts of the execution.

Figure 2.1 – Bock diagram of a processor with a SIMD engine embedded (a) and a simplified block diagram of a SIMD engine (b).



The second solution, straightforward from the point of view of the hardware designer, was to integrate several symmetric cores, based on the first solution, into the same silicon die, as Fig. 2.2(a) shows. Due to power dissipation issues, the integrated cores should feature a high performance per watt ratio and an overall current consumption acceptable for the

application. The drawback of this solution is that all cores are symmetric in both performance and function, which limits the speed-up that can be obtained by the parallel execution of the cores, since all applications intrinsically have sequential phases.

In the general and high-performance computing fields, there is a clear trend towards to chips with multiprocessor architectures. In 1967, G. M. Amdahl stated that the performance improvement (*S*) to be gained from using some faster mode of execution is limited by the fraction of the time (*f*) the faster mode can be used (Eq. 2.1) (AMDAHL, 1967).

(Equation 2.1) $$Speedup_{enhanced} = \frac{1}{(1-f)+\frac{f}{S}}$$

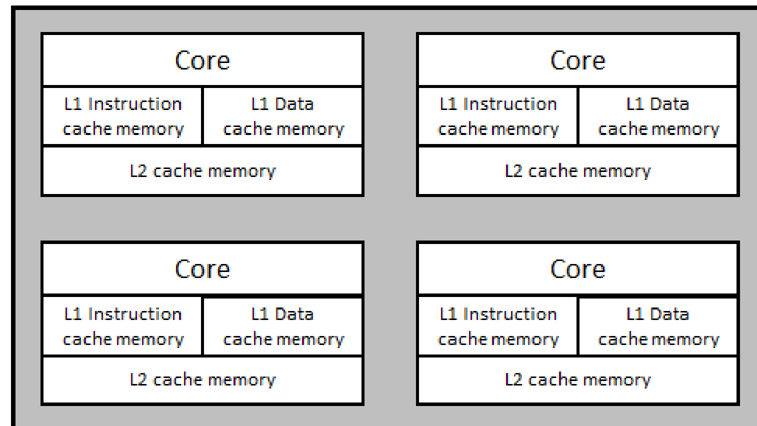
Based on Amdahl's law, during the last decade, the semiconductor industry has defined high-performance asymmetric architectures which accelerate the sequential execution by using fast cores and the parallel execution by a massive usage of small cores (Base-Core Equivalent – BCE) having lower performance but better high performance per watt ratio than the fast ones. This was the third solution and it is shown in Fig. 2.2(b). This kind of architecture can improve the overall system performance by accelerating some critical parts of a parallel application while still providing very good flexibility. Based on this concept, Hill and Marty (2008) extended the Amdahl's law for symmetric (Eq. 2.2) and asymmetric (Eq. 2.3) multicore devices by introducing two additional parameters, *n* and *r*, to represent the total number of resources available and those dedicated to sequential processing (BCE), respectively. Hill and Marty used the Pollack's Law (POLLACK, 1999) as input to their model, which observes that the sequential processing performance from a microarchitecture alone grows roughly with the square root of transistors used ($perf_{seq}(r) = \sqrt{r}$).

(Equation 2.2) $$Speedup_{symmetric}(f,n,r) = \frac{1}{\frac{1-f}{perf_{seq}(r)}+\frac{f}{\left(\frac{n}{r}\right) x\, perf_{seq}(r)}}$$

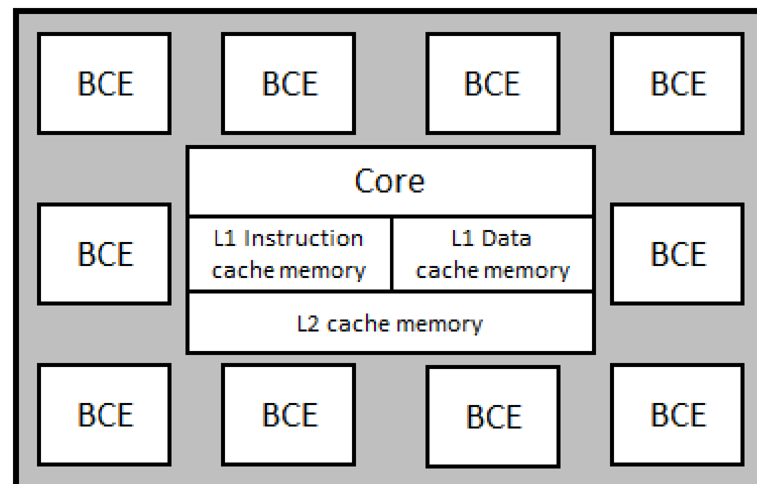
(Equation 2.3) $$Speedup_{asymmetric}(f,n,r) = \frac{1}{\frac{1-f}{perf_{seq}(r)}+\frac{f}{perf_{seq}(r)+n-r}}$$

However, today the semiconductor industry already went a step further in the asymmetric architectures solution by using heterogeneous architectures to achieve greater energy efficiency. Thus, the present metrics are no longer totally applied to such devices. Heterogeneous architectures (Fig. 2.2(c)) combine traditional processors with other hardware

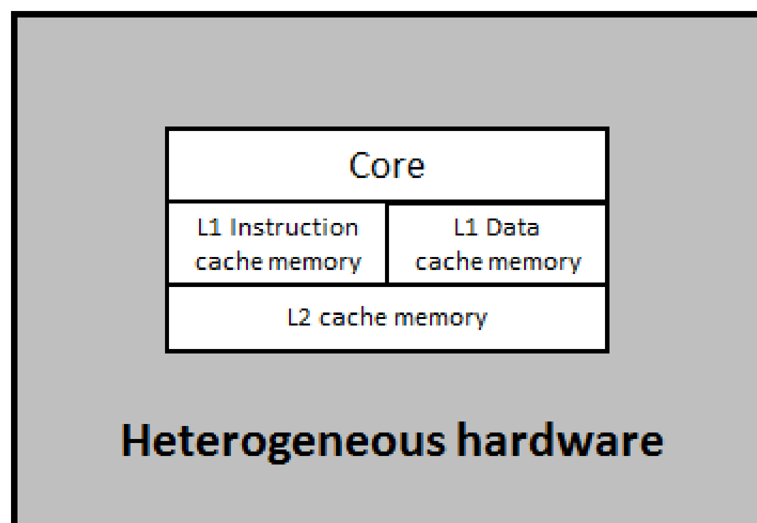
Figure 2.2 - Processor models: (a) Symmetric, (b) Asymmetric, and (c) Heterogeneous.



(a)



(b)



(c)

categories, such as custom logic, General Purpose Graphics Processing Units (GPGPUs), or FPGAs. Custom logic can provide the most energy-efficient form of computation (100-1000x improvement in either efficiency or performance) through ASICs customized to a specific task (DALLY et al., 2008). However, it is costly to develop and cannot be easily re-used for new applications. GPGPUs have also been shown to significantly outperform conventional microprocessors in target applications (LEE et al., 2010). GPGPUs derive their capabilities through SIMD vectorization and through multithreading to hide long memory latencies. GPGPUs are very suitable for homogeneous data parallel tasks. Finally, the third option of heterogeneous architecture is the FPGAs. Unlike custom logic and GPGPUs, FPGAs enable flexibility through programmable Look-Up Tables (LUTs) cells that can be used to implement arbitrary logic circuits. In exchange for this flexibility, a typical 10-100x gap in area and power exists between custom logic and FPGAs (KUON, ROSE, 2007). Heterogeneous FPGAs, which are the focus of this thesis, are the most suitable devices to perform pipeline irregular data flows as well as data parallel tasks.

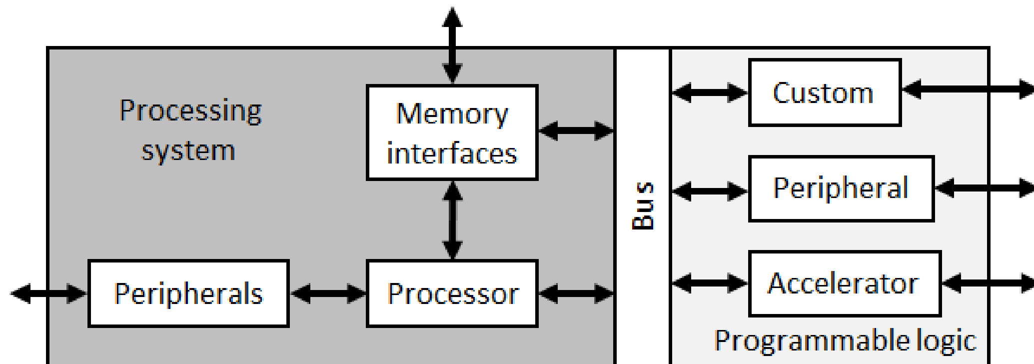
2.1 All Programmable System-on-Chip devices

Although traditional processors have been coupled with FPGAs before, it has never been quite the same proportion as now. Today, heterogeneous FPGAs, called of APSoC devices hereafter, have embedded hard-core processors capable of running full operating systems.

In general, APSoCs are composed of two main parts (Fig. 2.3): the Programmable Logic (PL), which is FPGA-based, and the Processing System (PS), which is formed around of a hard-core processor. The architecture is usually completed by embedding several peripherals and industry bus interfaces, which provide high bandwidth and low latency between PS and PL. The PL section is ideal for implementing high-speed logic, arithmetic, and data processing subsystems, while the PS supports software routines and/or operating systems. Thus, the overall functionality of any designed system can be appropriately partitioned between hardware and software. Finally, it is worth mentioning that in APSoCs, the processor embedded into the PS part can be regarded as the central element of the hardware system. The software system running on the processor comprises applications that run or in bare-metal mode with a lower level of software functionality for interfacing with the hardware system, or on the top of an Operating System (OS).

There are several APSoCs currently available on the market. Some recent examples are Zynq-7000 (XILINX, 2015d) from Xilinx, SmartFusion (MICROSEMI, 2015a) and SmartFusion2 (MICROSEMI, 2015b) from Microsemi, and Cyclone V (ALTERA, 2015) from Altera.

Figure 2.3 – Generic architecture of an APSoC.



Zynq-7000 and the Cyclone V devices use a dual-core ARM Cortex-A9 application processor. SmartFusion2 devices are based around the ARM Cortex-M3 embedded processor, primarily targeting microcontroller applications. The main characteristics of the three devices are summarized in Table 2.1. Further information about the architecture of the devices can be found in their respective datasheets, which were referred in the previous paragraph. This thesis uses the Xilinx Zynq-7000 APSoC as case-study platform. Nevertheless, the methodologies and the achieved results are capable to be extendable to other APSoCs.

Table 2.1 – Commercially-available APSoCs characteristics.

Functional unit	Xilinx Zynq-7000	Altera Cyclone V	Microsemi SmartFusion2
Processor	ARM Cortex-A9	ARM Cortex-A9	ARM Cortex-M3
Processor class	Application processor	Application processor	Microcontroller
Single or dual core	Dual	Dual	Single
Processor max. freq.	1.0 GHz	1.0 GHz	166 MHz
L1 cache	Data: 32 KB Instruction: 32 KB	Data: 32 KB Instruction: 32 KB	No data cache Instruction: 8 KB
L2 cache	Unified: 512 KB	Unified: 512 KB	Not available
Memory Management Unit (MMU)	Yes	Yes	Yes
Floating-Point Unit (FPU)	Yes	Yes	Not available

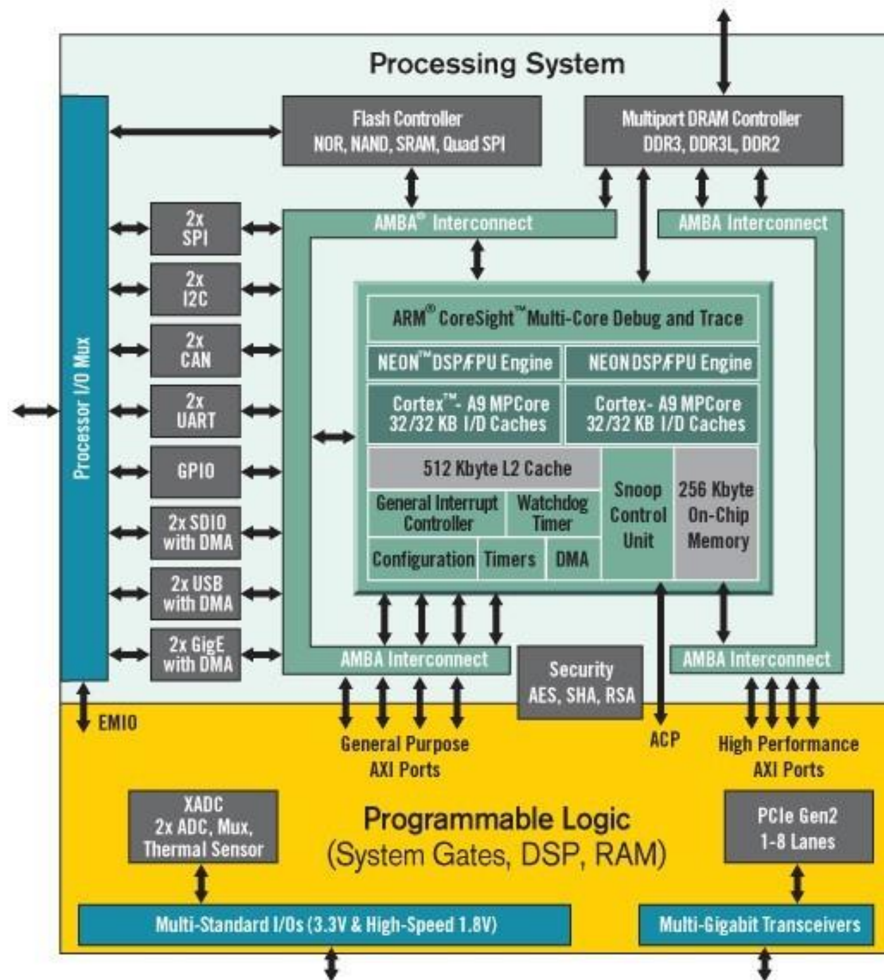
Accelerator Coherency Port (ACP)	Yes	Yes	Not available
Interrupt controller	Generic	Generic	Nested, Vectored
On-Chip Memory (SRAM)	256 KB	64 KB	64 KB
Direct Memory Access (DMA)	8-channel 4 requests	8-channel 32 requests	1-channel HPDMA 4 requests
External Memory Controller (EMC)	Yes	Yes	Yes
Memory types supported	LPDDR2, DDR2, DDR3L, DDR3	LPDDR2, DDR2, DDR3L, DDR3	LPDDR, DDR2, DDR3
Memory ECC	16 bit	16 bit, 32 bit	8 bit, 16 bit, 32 bit
External memory bus max. Frequency	533 MHz	400 MHz	333 MHz
Processor peripherals	1x quad SPI or dual quad SPI controller with 2 chip selects; 1x static memory controller (NAND-SLC, NOR, or SSRAM); 2x 10/100/1G Ethernet controller; 2x USB 2.0 OTG controller; 2x SD/SDIO controller; 2x UART; 2x I^2C controller; 2x CAN controller; 2x SPI controllers (master or slave); 2x 16 bit triple-mode timer/counters; 1x 24 bit watchdog timer	1x quad SPI controller with 4 chip selects; 1x NAND controller (single-and multilevel cell - MLC or SLC); 2x 10/100/1G Ethernet controller; 2x USB 2.0 On-the-Go (OTG) controller; 1x SD/MMC/SDIO controller; 2x UART; 4x I^2C controller; 2x CAN controller; 2x SPI master, 2x SPI slave controller; 4x 32 bit general-purpose timers; 2x 32 bit watchdog timers	1x 10/100/1G Ethernet controller; 2x USB 2.0 OTG controller; 2x UART; 2x I^2C controller; 1x CAN controller; 2x SPI; 2x general-purpose timers; 1x watchdog timer; 1x real-time clock (RTC)
FPGA Fabric	Artix-7, Kintex-7	Cyclone V, Arria V	Fusion2
FPGA logic density range	28 K to 444 K logic elements	25 K to 462 K logic elements	6 K to 146 K logic elements
High-speed transceivers	Higher-density devices only	Available in all devices	Higher-density devices only

2.1.1 Xilinx Zynq-7000 APSoC

Zynq-7000 is a commercial-off-the-shelf device designed by Xilinx in a Taiwan Semiconductor Manufacturing Company's (TSMC) 28 nm technology node. Fig. 2.4 shown

shows its block diagram. In this thesis, it was used an XC7Z020-CLG484 part embedded in a commercially available ZedBoard Development Board.

Figure 2.4 – Block diagram of the Zynq-7000 (XILINX, 2015d).



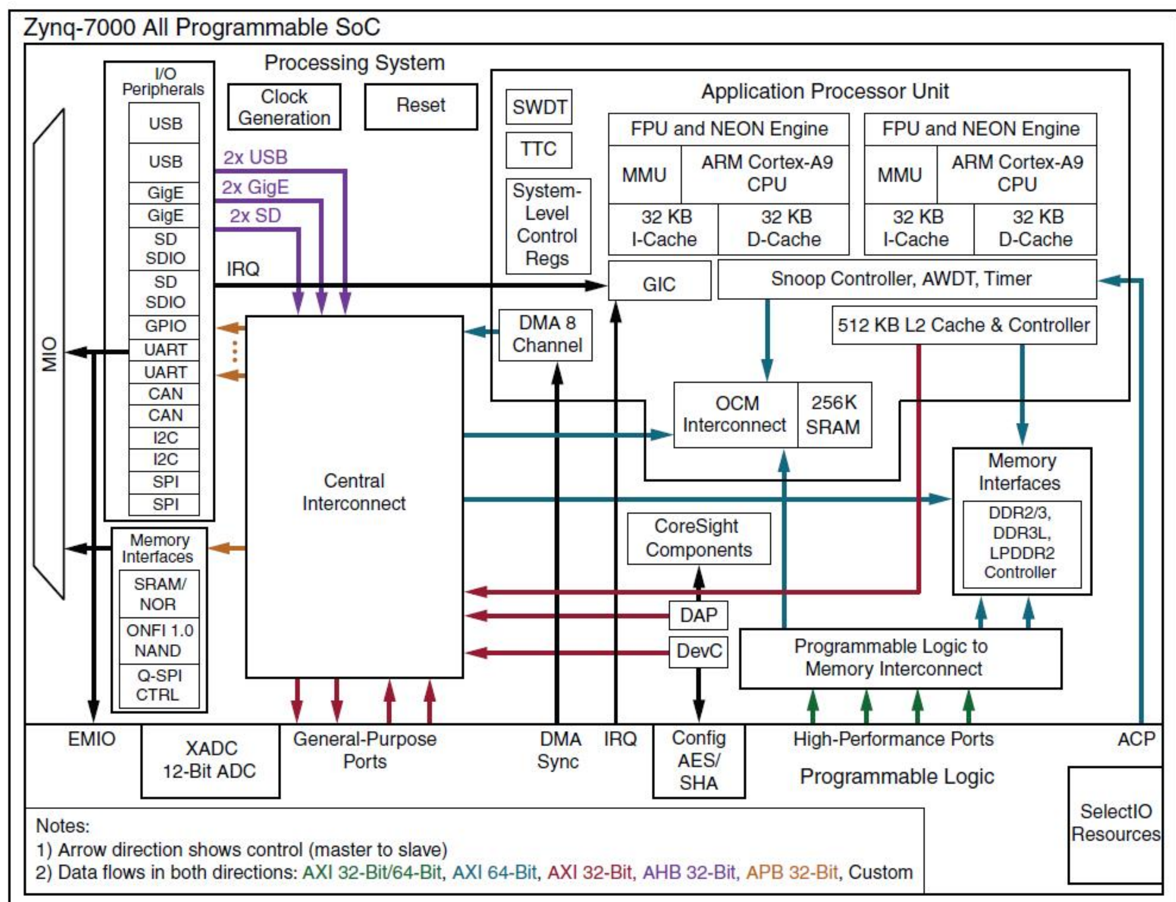
2.1.1.1 The Processing System (PS)

Zynq-7000 has a 32-bit 8-11 pipeline stages dual-core ARM Cortex-A9 hard-core processor as the PS basis. However, a set of associated processing resources surrounds it forming an Application Processing Unit (APU) (Fig. 2.5).

Each ARM core has associated several computational units, such as: a NEON Media Processing Engine (SIMD), a Floating Point Unit (FPU), a Memory Management Unit (MMU), and a Level 1 Cache 4-way set-associative divided in two sections for instructions and data. The APU also contains a Level 2 Cache 8-way set-associative memory and an SRAM On-Chip Memory (OCM). Timers and a General Interrupt Controller (GIC) are

further functional blocks located in the APU. Finally, a Snoop[1] Control Unit (SCU) forms a bridge between the ARM cores, the Level 2 cache, and the OCM. The SCU undertakes several tasks relating to interfacing between the processors and Level 1 and 2 cache memories. The SCU also has the capability of interfacing with the PL through an Accelerator Coherency Port (ACP).

Figure 2.5 – Block diagram of the PS part of the Zynq-7000 (XILINX, 2015d).



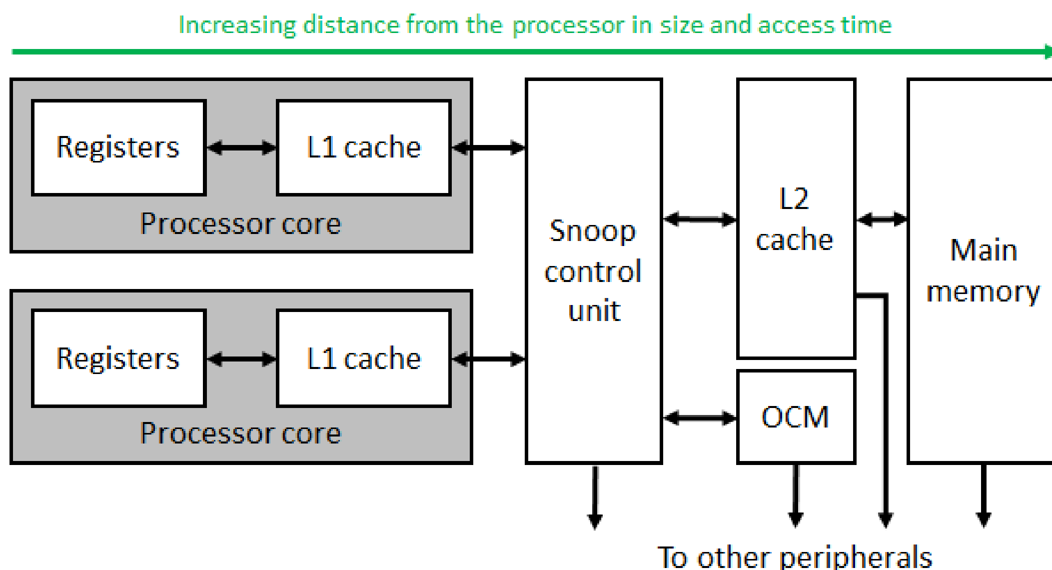
In the context of this thesis, special attention must be given to the embedded memories of the Zynq-7000 and their hierarchy (Fig. 2.6), because they play an important role in the overall reliability and system performance. Embedded memories represent around 60% of the chip area in current processors, and to be fast and efficient, their memory cells are built as small as possible and with very low capacitance, especially caches (MANOOCHEHRI,

[1] Snooping is one of the several mechanisms for ensuring cache coherency, i.e. managing the consistency of data across shared cache resources.

ANNAVARAM, DUBOIS, 2011). As consequence, such characteristics make them very vulnerable to SEEs, as will be possible to notice later.

Typically, every processor has inside its data path a set of General-Purpose Registers (GPRs) called of Register File ("Registers", in Fig. 2.6). In case of Zynq-7000's ARM Cortex-A9 dual-core processor, each core has a register file consisted of 15 GPR plus the Program Counter (PC) (ARM, 2012). The register file is the smallest addressable memory and it is located at the top of the memory hierarchy. Thus, a register file must be fast: a GPR must be able to drive its data onto an internal bus in a single clock cycle. The register file is an important microarchitectural component used for storing operands and results of instructions. It is visible only to programs that address registers directly, so it is part of the Instruction Set Architecture (ISA).

Figure 2.6 – Zynq-7000's memory hierarchy.



The L1 cache is the smallest and faster cache memory of the Zynq-7000. It is implemented in the form of SRAM cells, which are built into the fabric of the ARM core and, as such, operates at the same clock frequency. Similarly to the register file, L1 cache is not a general memory, not being accessible via any system bus. As any cache memory, the L1 cache of Zynq-7000 is used to store data that is frequently accessed by the processor from main memory. The basic mechanism of a cache memory is the following: when the access to a memory address is requested, the cache verifies if the block (a set of adjacent memory words) containing the address is present in this level. If the referred block is present within this cache level (a cache hit occurred), the address is fulfilled. Otherwise, if the referred block

is not found (a cache miss occurred), the cache copies the relevant block from the next lower memory level for then, fulfilling the request (PATTERSON, HENNESSY, 2005). Therefore, a cache miss implies in a penalty on the access time equal to the access time of the next memory level. In contrast, operations that make use of cached data are faster (lower access time) than those where the data is only in lower memory levels. Cache memories also operate on the principle of temporal and spatial locality (PATTERSON, HENNESSY, 2005). Temporal locality states that if a data is referenced, it will tend to be referenced again soon. Thus, after the first time a data is accessed, it worth keeping that data cached. Spatial locality states that if an item is referenced, items whose addresses are close by will tend to be referenced soon. Thus, when a memory address is accessed, it worth caching nearby positions (cache block).

The L2 cache of Zynq-7000 is external to the processing cores, but is located extremely close to them. It is larger than L1 cache, but has slower access speeds. L2 cache is in the form of Dynamic RAM (DRAM) and is unified in a single section (unlike L1, which is split into two sections). In the standard APU configuration, which means all cache levels enabled, larger quantities of data are constantly read in by L2 cache from main memory before being fed to L1.

The OCM contains 256 KB of RAM and 128 KB of ROM (where the boot of the processor resides, the BootROM). The RAM part of the OCM can be considered a general memory, so it is accessible via system buses. It supports two 64-bit slave interface ports, one dedicated for processor access via the SCU and the other is shared by all other bus masters within PS and PL. The OCM address range is not a cacheable memory region by default. However, it is possible to turn the OCM a cacheable memory by modifying the configuration of the MMU of the processor. Although this configuration does not provide a significant performance improvement comparing to a standalone L2 cache configuration, once both memories are practically in the same level of the memory hierarchy, it is an interesting approach when the caches are enabled and the OCM is used as a shared memory between PS and PL. The BootROM memory region is not visible to the user as it is reserved for exclusive use by the boot process of the device.

It is worth noting that in the Zynq-7000, the L1 cache, L2 cache, and OCM, have byte-parity support (XILINX, 2015d). Parity is commonly used as the simplest form of error detecting code. This method counts the number of logic one states (or "ones") in a data structure, such as a byte, and then adds a bit in the end of that data structure, stating whether if an odd or even number of ones is present in that structure. Parity detects an error if an odd

number of bits are in error, but if an even number of errors occurs, the parity is still correct (i.e. the parity is the same whether 0 or 2 errors occur). This is a "detect only" method of mitigation and does not attempt to correct the error that occurs. Therefore, in case a parity error is detected in a cached data, for example, it will only result in a cache miss. The parity support in L2 cache and OCM is configurable. However, the parity support in the L1 cache is always enabled by default, not being possible to disabled it through standard ways. No information was found about the presence of parity support in the register file of the processor cores.

Externally to the APU, the PS part features a variety of interfaces, both between the PS and PL, and between the PS and external components, as shown in Fig. 2.5. The communication between the PS and external interfaces is achieved primarily via the Multiplexed Input/Output (MIO), which provides 54 pins of flexible connectivity, meaning that the mapping between peripherals and pins can be defined as required. Connections can also be made via the Extended MIO (EMIO), which is not a direct path from the PS to external connections, but instead passes through and shares I/O resources of the PL. The complete set of I/O peripherals is stated in Table 2.1 and more information can be found in (XILINX, 2015d). It is important to highlight that such interfaces and peripherals are configured through the use of registers. Hundreds of 32-bit registers are used to determine the functionality of all the peripherals and they are also vulnerable to SEEs.

At system level, an important feature of APSoCs that have embedded a multi-core processor or are able to implement a multi-core architecture in their PL parts, is the option to configure the processor cores in an Asymmetric Multi-Processing (AMP) or in a Symmetric Multi-Processing (SMP) mode.

AMP can be used on a system that utilizes multiple processor cores, which may be of different architectures, such as hard- or soft-core. In this case, each processor core can run its own software system (OS or bate-metal), which can either be homogeneous or completely different. An example of this would be a system running a Linux OS on one core while a bare-metal application runs on another one. The communication between the cores is facilitated by a shared memory, which provides a level of software abstraction.

In SMP, each core has the same hardware architecture. They share the main memory space and have full access to all I/O devices. Although, each core has its private resources such as L1 cache memory, private timers, and memory management unit. The whole system is controlled by a single OS instance, which treats all cores equally. Due to the shared memory architecture, the OS has to provide some common interfaces for all cores to access

the main memory, as well as some communication mechanisms for task synchronization. Typically, SMP solutions are employed when an embedded application simply needs more CPU power to manage its workload.

2.1.1.2 The Programmable Logic (PL)

The PL part of the Zynq-7000 device is based on the Artix-7 and Kintex-7 families of Xilinx FPGAs. Therefore, the PL part has the same internal architecture of the mentioned FPGA families.

An FPGA can be viewed as a two-layer device, the Design Layer and the Configuration Layer, as depicted in Fig. 2.7. The Design Layer of the Zynq-7000's PL part and some of its user application resources are depicted in Fig. 2.8. It composed of Configurable Logic Blocks (CLBs); other specialized circuits, such as embedded memory blocks (Block RAM – BRAM), Digital Signal Processor (DSP) blocks, the Internal Configuration Access Port (ICAP), Phase-Locked Loop (PLL) blocks, clock trees, Power-on Reset (PoR) circuitry, and others; and they are surrounded by programmable Input/Output Blocks (IOBs). They are interconnected in a matrix structure by a set of programmable interconnections, creating an array of programmable logic blocks of different types. The Configuration Layer is composed of all SRAM memory cells responsible to configure all the Design Layer, such as the CLBs, content of BRAMs, DSPs, routing structures, clock trees, PLLs, IOBs, and others. Such programmable blocks and interconnections are configured by the bitstream, which is a group of configuration bits that are loaded in the configuration memory during the device power-up for defining a specific circuit previously described with a Hardware Description Language (HDL).

The purpose of the CLBs is to implement the combinational and sequential logic of the user's design synthesized into the FPGA. An CLB is composed of one or more slices and each slice is composed of one or more Look-up Tables (LUTs), Flip-Flops (FFs), and routing structures. An LUT is the basic structure for implementing the truth table of a logic function and it is usually implemented by a multiplexer with 2^n inputs and n selectors. The inputs are connected to SRAM cells that are part of the configuration memory. Then, with this architecture is possible to implement any combinational circuit with n inputs. In case of Zynq-7000's PL, it has six inputs LUTs in its CLBs. Fig 2.9 shows an example of an LUT implementing a 3-input majority voter. LUTs can also be configured as distributed memories ROM, a small RAM, or shift registers.

Figure 2.7 – Abstraction layers of a generic SRAM-based FPGA (TARRILLO, 2014).

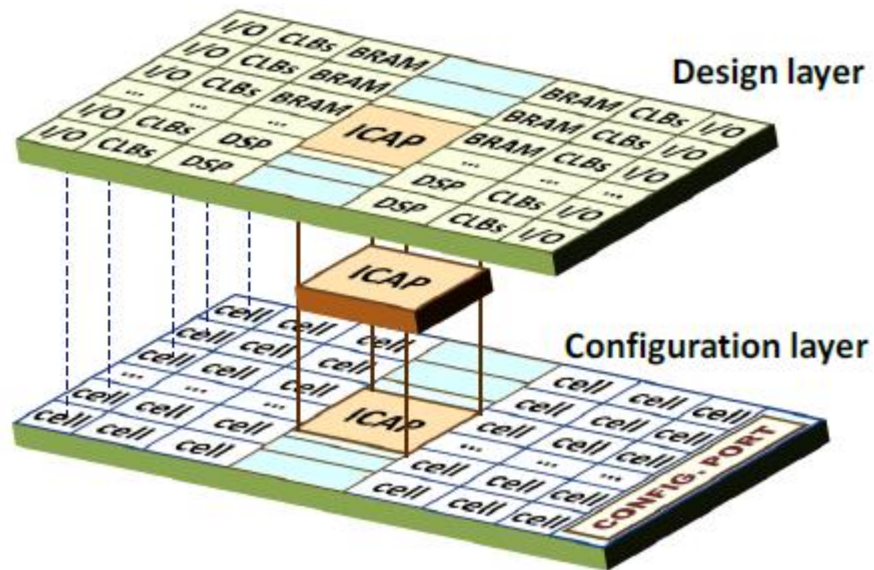


Figure 2.8 – Basic structure of the PL part of the Zynq-7000 (CROCKETT, 2014).

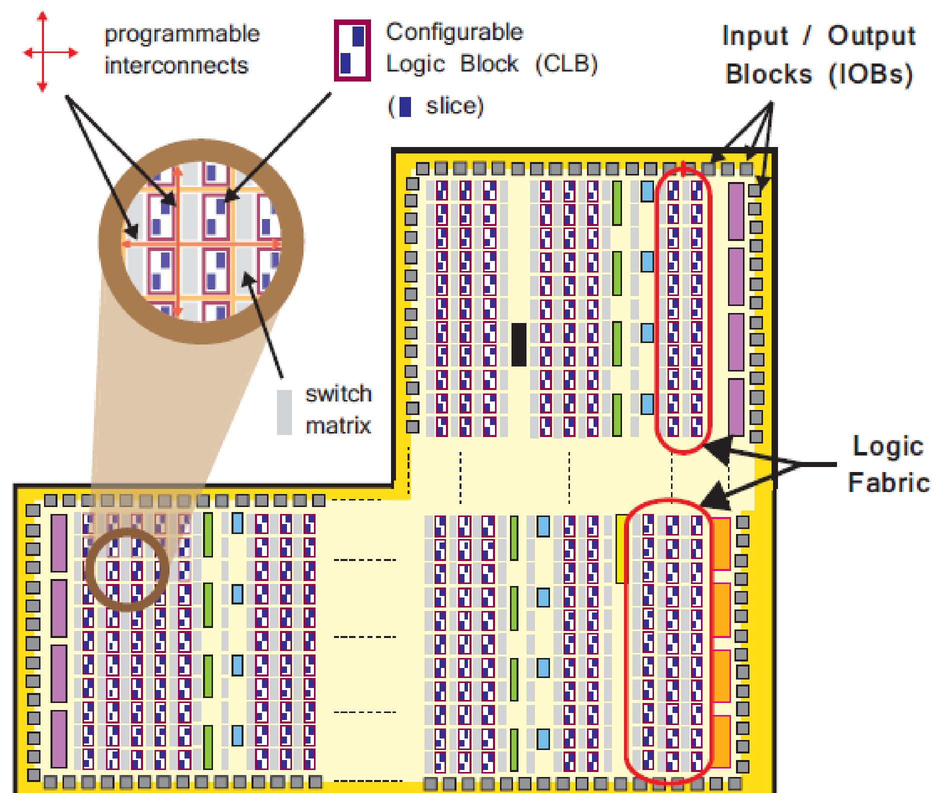
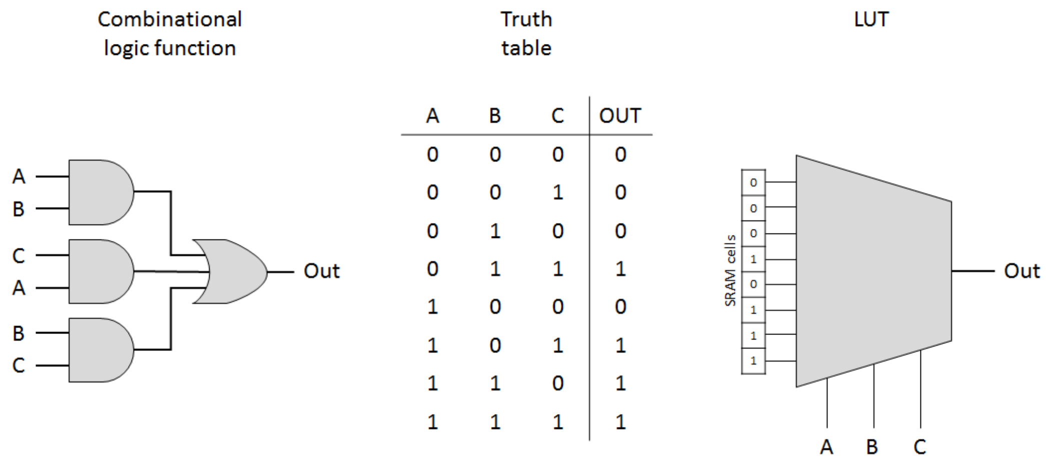


Figure 2.9 – Example of a 3-input LUT implementing a majority voter.



A	B	C	OUT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

In Xilinx 7-Series architecture, which Zynq-7000 is part, each CLB's slice contains four 6-input LUTs, eight flip-flops, multiplexers to interconnect the LUTs, flip-flops, and one carry propagation chain. Fig 2.10 shows the block diagram of a 7-Series slice. In turn, slices are grouped into CLBs. Each CLB has two slices, as shown in Fig. 2.11.

In addition to CLBs, FPGAs have blocks of embedded memory (BRAM). These blocks are based on SRAM cells and dedicated for the user circuit. They are more efficient implementing large memories or FIFOs than flip-flops in CLBs. Flip-flops are mainly used for implementing registers or pipeline barriers, for example. They support Error Correcting Codes (ECCs).

DSP blocks are also present in the FPGA fabric. These blocks contain hard-core multipliers and adders to implement arithmetic operations at high-speeds.

I/O Blocks provide an interface between the FPGA resources and the physical device pads used to connect to external circuitry. In the Zynq-7000's PL, it is possible to configure some features of the IOBs, such as the voltage level, signal direction, and programmable delay. Some devices also incorporate transceiver blocks to enable high-speed communications, such as the bigger Zynq-7000 devices, which are based on the Kintex-7 family.

The clock distribution in the FPGA is done by dedicated global and local clocks routing wires and buffers. These signals divide the FPGA into clock regions, and these regions are controlled by clock buffer primitives (XILINX, 2014). Such primitives enable the user to apply clock gating to an entire clock region, for example. There are also specialized clock management blocks where it is possible to multiply or divide the reference clock frequency.

Figure 2.10 – Block diagram of a Xilinx 7-Series slice (XILINX, 2014).

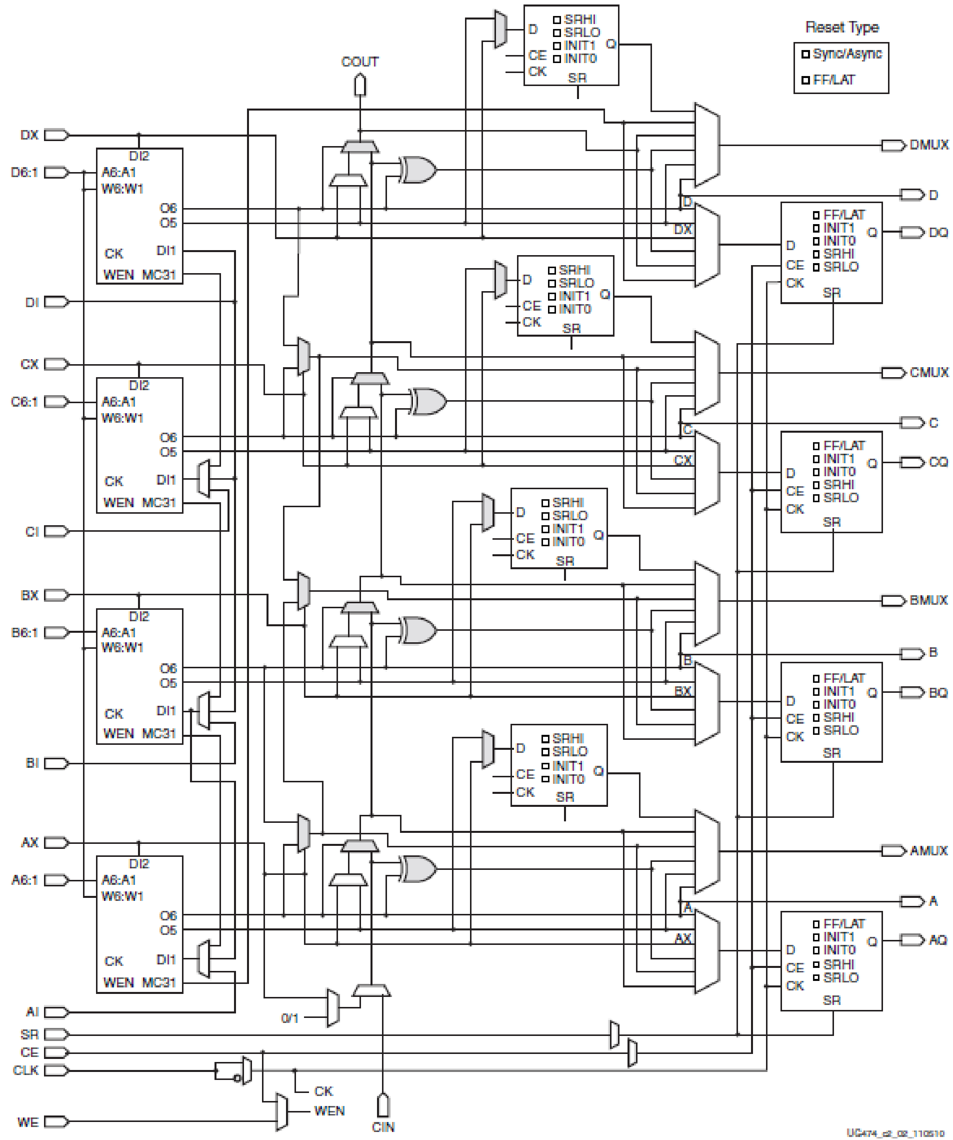
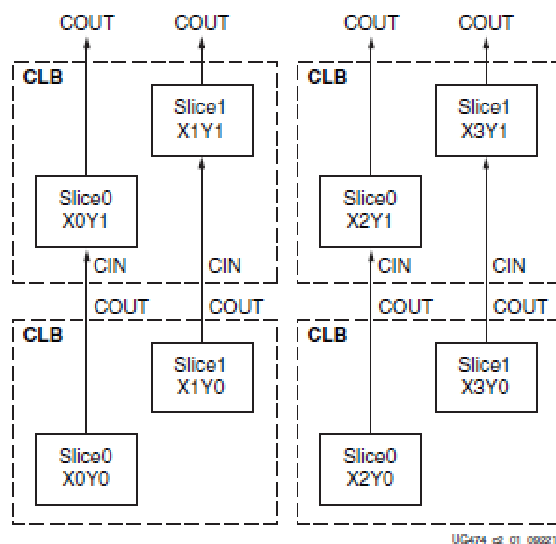
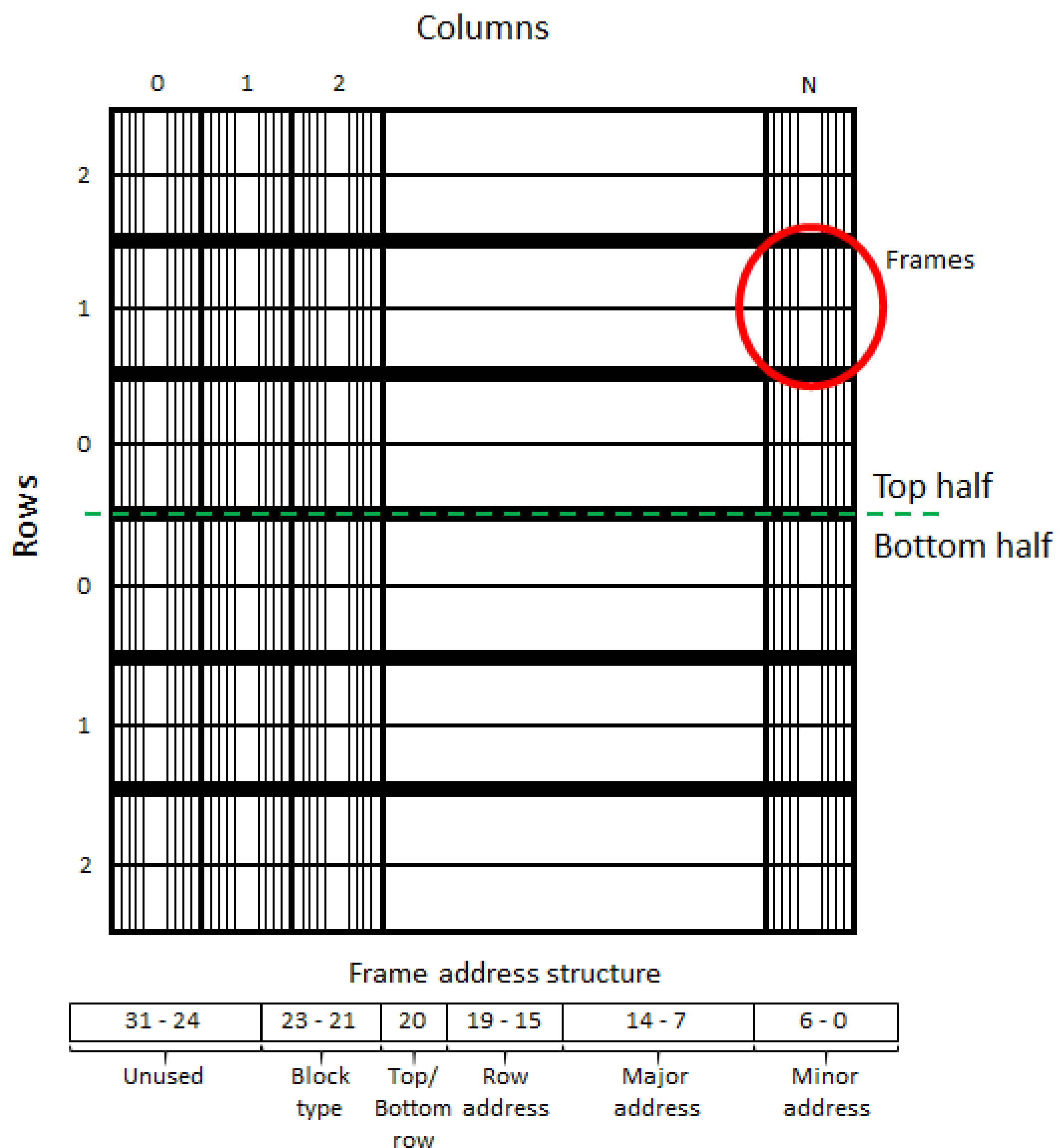


Figure 2.11 – Relationship between CLBs and Slices in Xilinx 7-Series FPGAs (XILINX, 2014).



With regard to the Configuration Layer, it consists mainly of the configuration bits. However, they have different functions. Some of them define the function of LUTs, others define the configuration of embedded resources, like BRAMs, DSPs, IOBs, and others define the interconnection of the CLBs. The FPGA configuration memory is composed of small memory segments called Configuration Frames. A configuration frame is the smallest addressable portion of the FPGA configuration memory, and the frame size varies among FPGA families. In case of Zynq-7000's PL, a configuration frame is composed of 101 32-bit words (XILINX, 2014). Each frame has a unique address that is related to the physical position in the FPGA floorplanning. The frame address is composed of five fields as follows and shown in Fig. 2.12:

Figure 2.12 – Example of a generic Xilinx FPGA floorplan and frame structure.



- Type – Define the type of the frame. It can be a configuration frame (type 0); BRAM content (type 1); and there is also type 2, but this one is not well documented in Xilinx's literature.
- Top/Bottom – Define the half part (top or bottom) of the FPGA where the frame is located.
- Row – Defines the frame row.
- Column – Define the frame column. A column is defined by the type of resource (CLB, BRAM, DSP, etc.).
- Frame in column – Define the frame position inside a column.

As can be seen in Fig. 2.12, the floorplanning of a Xilinx FPGA is divided into two main regions: top and bottom. Each region is organized in rows and columns. Each frame has the height of a row, and the columns are arranged according to the type of resource (e.g. CLB, BRAM, DSP, etc.). Each column contains a group of frames. The number of frames on each column depends on the type of the resource that it configures.

The access to the configuration memory is possible through several interfaces and it can be performed externally or internally to the device. Example of external interfaces are: JTAG and SelectMAP. The internal interface is the Internal Configuration Access Port (ICAP). It has the same interface as the SelectMAP, with the only difference that the ICAP can be accessed from the programmable logic.

2.1.1.3 The interfaces between PS and PL

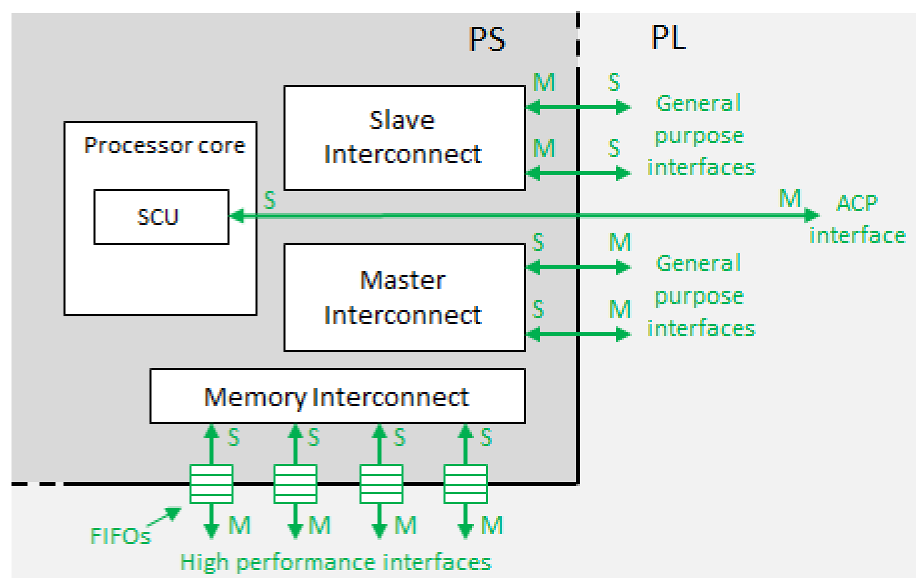
One of the main benefits of Zynq-7000 and other similar APSoCs is the ability to use both PS and PL parts in tandem to form complete and integrated systems. The key enabler in this regard is the set of highly specified interconnects and interfaces forming a bridge between the two parts. In case of Zynq-7000, the two main interfaces are the Advanced Extensible Interface (AXI), which is part of the ARM Advanced Microcontroller Bus Architecture (AMBA) standard (ARM, 2015a), and the Extended Multiplexed Input/Output (EMIO).

In case of AXI, Zynq-7000 uses AXI4 standard (ARM, 2015a), which is focused on memory-mapped links and provides the highest performance. The primary interface between the PS and PL is implemented via a set of nine AXI interfaces. Each interface is composed of multiple channels and make dedicated connections between the PL and the interconnects within the PS. Here, it is useful to define that an interconnect is effectively a switch which

manages and directs traffic among attached AXI interfaces and an interface is a point-to-point connection for passing data, addresses, and hand-shaking signals between master and slave components within the system. Concerning AXI interfaces, Zynq-7000 devices have three different types of PS-PL AXI interfaces (Fig. 2.13):

- General Purpose AXI (GP-AXI) - A 32-bit data bus suitable for low and medium communications rate between PS and PL. The interface is direct and does not include buffering. There are four general purpose interfaces in total: the PS is the master of two and the PL is the master of the other two.
- Accelerator Coherency Port (ACP) - A single asynchronous connection between the PL and the SCU within the APU with a bus width of 64 bits. This port is used to achieve coherency between the APU caches and elements within the PL. In this case, the PL is always the master.
- High Performance AXI (HP-AXI) Ports - There are four HP-AXI ports. They include FIFO buffers to accommodate read and write bursts and to support high rate communications between the PL and memory elements in the PS. The data width is either 32 or 64 bits and the PL is the master of all four interfaces.

Figure 2.13 – AXI interconnects and interfaces connecting the PS and PL parts of the Zynq-7000.



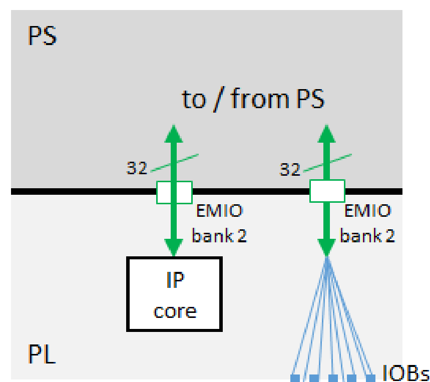
Concerning EMIO, it involves signal transfer between the two domains and is achieved through a simple set of wires connections. Interfaces routed through the EMIO can be connected to peripheral blocks in the PL or directly to external pins of the PL. EMIO can

provide an additional of 64 inputs and 64 outputs with corresponding output enables. An example of its architecture is shown in Fig. 2.14.

Other signals crossing the PS-PL boundary include watchdog timers, reset signals, interrupts, and DMA interfacing signals.

There are many issues when addressing the PS-PL interfaces of Zynq-7000. The performance of the communication between PS and PL is usually a bottleneck that restricts the hardware acceleration in the PL. In this sense, Silva, Sklyarov, and Skliarova (2015), analyzed and compared the PS-PL interfaces of Zynq-7000. They performed several experiments for evaluating the data exchange between PS and PL through GP, HP, and ACP ports, and using different memory hierarchies. The following scenarios of data transfer between PS and PL were evaluated: AXI ACP, allowing access to DDR/OCM and supporting coherency with the CPU cache using the SCU; AXI HP, allowing access to OCM; and, AXI HP, allowing access to external DDR. The following scenarios of data transfer between the processor and the memories were evaluated: processor and cache; processor and OCM; and, processor and DDR.

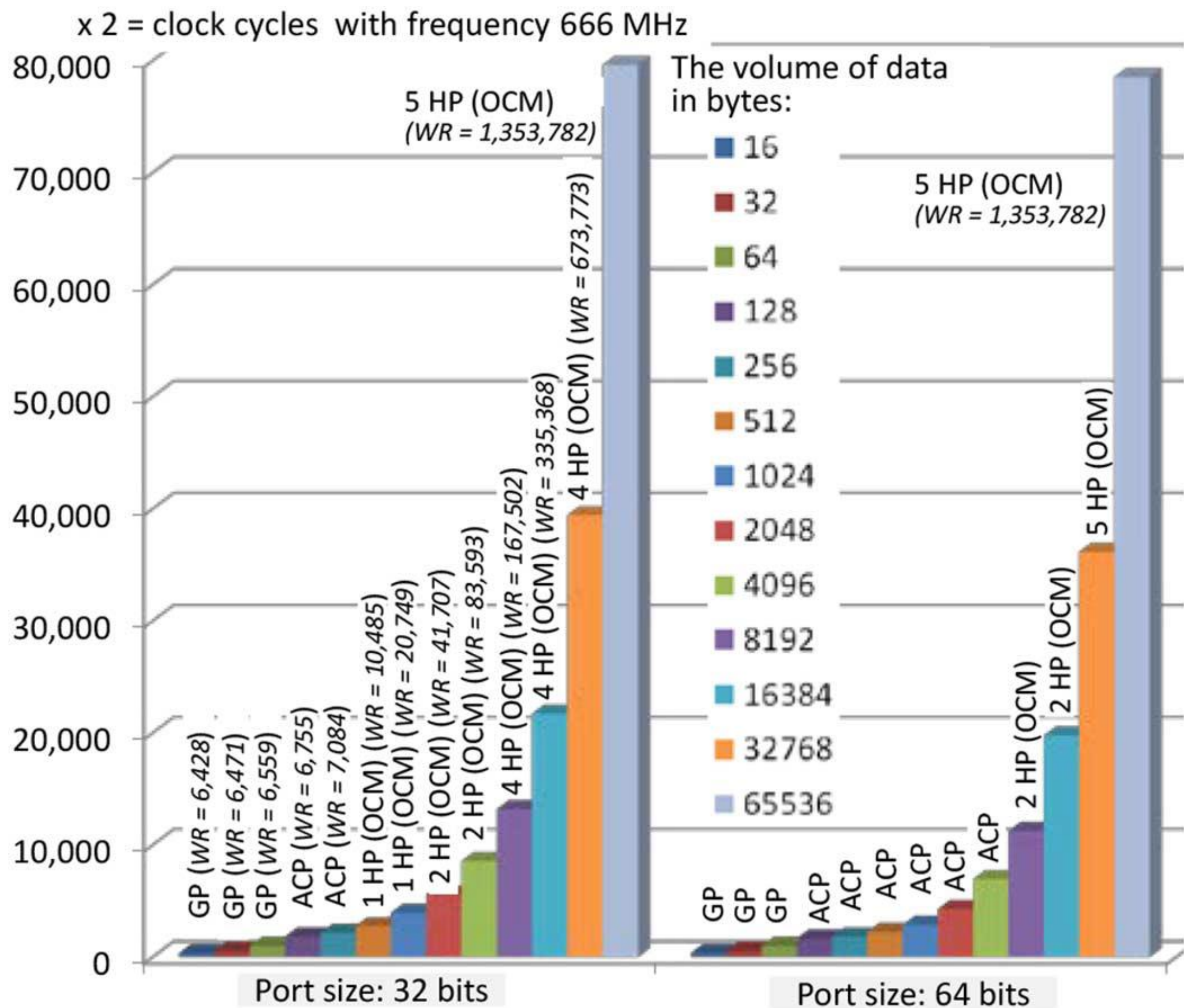
Figure 2.14 – EMIO interface between the PS and PL parts of the Zynq-7000.



It is worth to highlight that the SCU provides support for coherent access to memory that is automated for AXI ACP. If AXI HP ports are used, the cache memory is not able to participate in data transfers between the PS and the PL. However, it participates in data transfers between the processor and memories. Thus, enabling or disabling the cache affects such transfers. In addition, if the cache is enabled, processor operations may occur over data in the cache and it is necessary to provide coherency with data saved in other memories (OCM or DDR).

Fig. 2.15 shows the fastest ports for different volumes of data items (from 16 B to 64 KB) and the Worst Result (WR), which is the slowest interaction method found for each case. Results show that selecting the best port is important. For example, the GP port is always the best option for transferring a small number of data items (from 16 to 64 bytes). This implies that the GP port is very appropriate for supplying control signals from the PS to PL and providing additional information (such as interrupts from the PL to the PS). AXI ACP with the cache enabled gives the best results for a modest number of data items and if this number is increased, data transfer through OCM with the cache disabled is the fastest. This is because memory accesses through the AXI ACP utilize the same interconnecting paths as the processor, potentially decreasing the processor performance.

Figure 2.15 – Obtained best methods for transferring different numbers of data items between PS and PL in Zynq-7000 (SILVA, SKLYAROV, SKLIAROVA, 2015).



2.2 Hardware/Software co-design

The process of hardware/software co-design involves deciding which components should be implemented in hardware, which should be implemented in software, and how they will communicate each other. This partitioning process relies on the fact that hardware components, such as the ones residing in the PL, are typically faster due to the parallel processing nature of FPGA devices and have bigger design exploration space. However, it also tends to be more expensive if it is done manually. Software components implemented on a processor, on the contrary, are cheaper to both create and maintain, but they are also slower due to the inherent sequential processing. In order to achieve a good trade-off between performance and cost (and reliability, as it will be possible to notice in the next chapters), high-performance components can be implemented in hardware in the form of hardware accelerators, while less intensive processes can be implemented in software. Examples of suitable applications for PL implementation include intensive math operations, digital filtering, and image processing. These tasks are repetitive and quite static. On the contrary, problems that are more dynamic and unpredictable, are better suited to be implemented on a processor-based system.

In this context, the PL part of Zynq-7000 is a suitable platform for implementing functions which can be efficiently divided into parallel and/or multiple tasks. Due to the parallel execution nature of the programmable logic, multiple operations can be processed concurrently to calculate the final result in a shorter time than if processed sequentially. Fig. 2.15 shows the advantage of parallel hardware executions. Whereas the software execution requires 12 clock cycles (sequential execution) to produce the output G, the parallel implementation only requires 2 clock cycles (parallel execution) to produce the same result.

It is also possible to speed-up a function by using multiple instances of a hardware accelerator, creating a multi-core architecture. In this context, a system can run faster by executing several steps of a given function simultaneously. The architecture consists basically of several instances of one or more types of hardware accelerators that work in parallel and an interconnection network or a system bus that connects them together and with the main processor, which can be the PS part in case of Zynq-7000. As a practical example, it is possible to apply Eq. 2.3 for estimating the system speed-up provided by using identical hardware accelerators. To do so, assume a system implemented in Zynq-7000, such as a Fast Fourier Transform (FFT) operation, with the PS as the main processor, and that there are parts of the system that can be implemented as hardware accelerators in the PL, such as the internal

Discrete Fourier Transform (DFT) blocks. In addition, consider that is possible to accommodate at most 16 (n) hardware accelerators in the PL and that there are three versions of them, each one providing different speed-up factors (f) of 0.25, 0.50, and 0.75, compared to the main processor. Thus, Fig. 2.16 shows the estimated overall system speed-up provided by the addition of different numbers of hardware accelerators (r) in the PL, according to Eq. 2.3.

Results show that the availability of the main processor (higher f) makes it possible to run the sequential part of the system faster. Results also show that there is a point for each level of parallelism beyond which the overall system performance will decline. For example, for the 75% parallel type of system (f = 0.75), this point is reached with 8 hardware accelerators. The reason for this behavior relies on the fact that a higher number of hardware accelerators may lead to some overhead in the sequential part of the system, which is not taken into account in the present theoretical analysis, such as communication bottlenecks, memory accesses, and synchronization of the hardware accelerators.

Figure 2.15 – Example of parallel versus sequential execution.

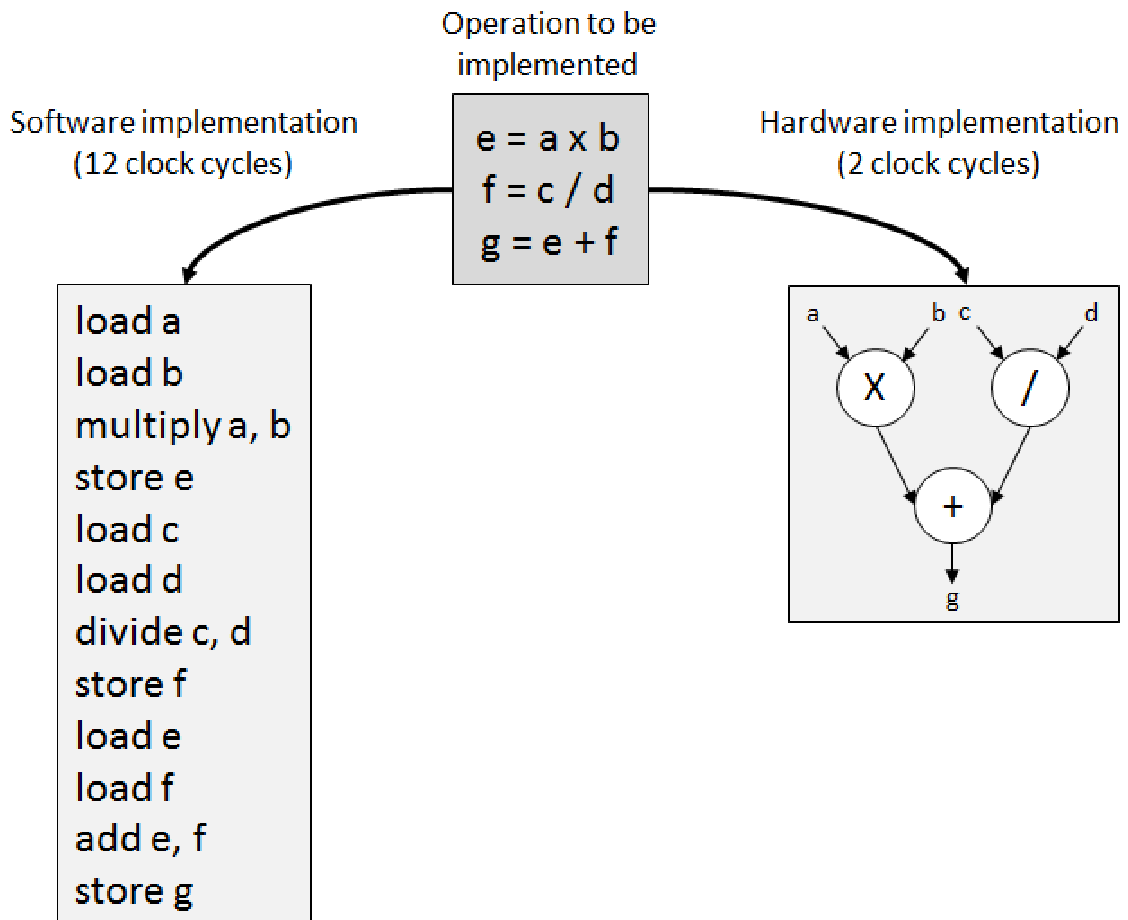
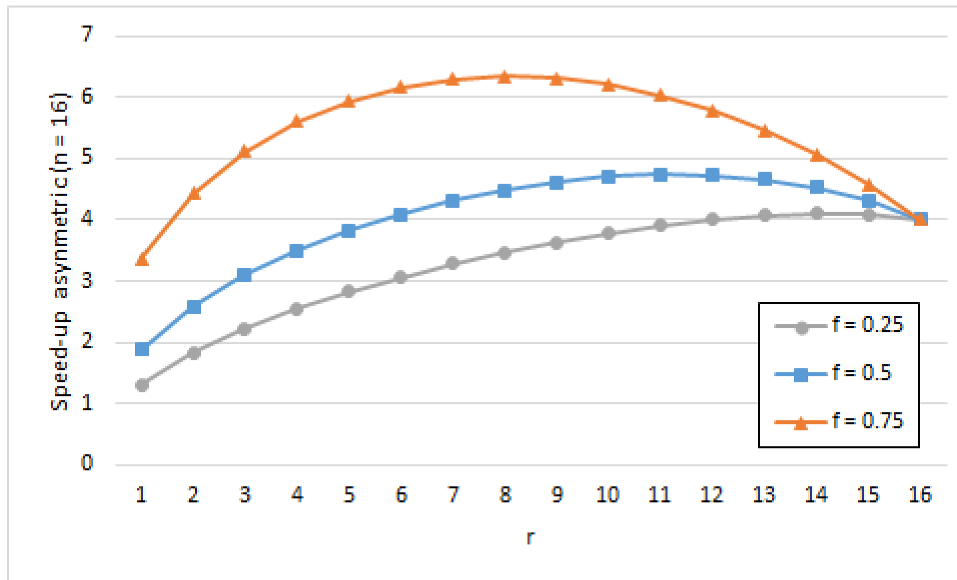


Figure 2.16 – Estimated overall system speed-up provided by the addition of different numbers of hardware accelerators.



2.2.1 High-Level Synthesis

The increasing systems' complexity and heterogeneity of hardware designs embedded in state-of-the-art APSoC devices and the shortening time-to-market have motivated the development of new designing methodologies focused to address the today's need of high-performance and energy-efficient circuits. In this context, the use of High-Level Synthesis (HLS) tools is an example of strategy that can be addressed during the development of a design for exploring the design space, such as performance improvement and resource utilization. This is crucial during the design of complex systems and especially suitable for projects targeting APSoC and FPGA devices, where many alternative implementations can be easily generated, deployed onto the target device, and compared. HLS tools have significantly evolved in the last years, providing very optimized results in area and performance with a very short development time.

HLS tools start from a high-level software programmable language (e.g. C, C++, SystemC) to automatically produce a hardware accelerator in HDL (e.g. VHDL or Verilog) that performs the same function. Thus, CPU-intensive tasks can be offloaded to dedicate hardware accelerators within an APSoC or a system-on-chip implemented into an FPGA.

A high-level synthesis process consists essentially of three phases: scheduling, allocation, and binding (XILINX, 2013). Scheduling extracts the control and data flow graphs from the high-level source code to implement the hardware design based on defaults and user-

applied special directives. Such directives force the high-level synthesis to focus on particular objectives, such as performance, throughput, area, or power consumption. Fig. 2.17 illustrates the control and data flow extraction process. The resource allocation and binding select the necessary RTL resources to implement behavioral functionalities based on a technology library, generated-component delays, and user directives, for example. They also determine the mapping relation between the behavioral constructs to the allocated RTL resources (DE MICHELI, 1994). Typically, the allocation and binding processes can be further divided into subtasks regarding functional units as hardware-specific cores and storage elements as memories, both based on a specific technology library (NANE et al., 2015). Fig. 2.18 illustrates the three phases and their relations.

Figure 2.17 – Example of a control and data flow extraction during the HLS process.

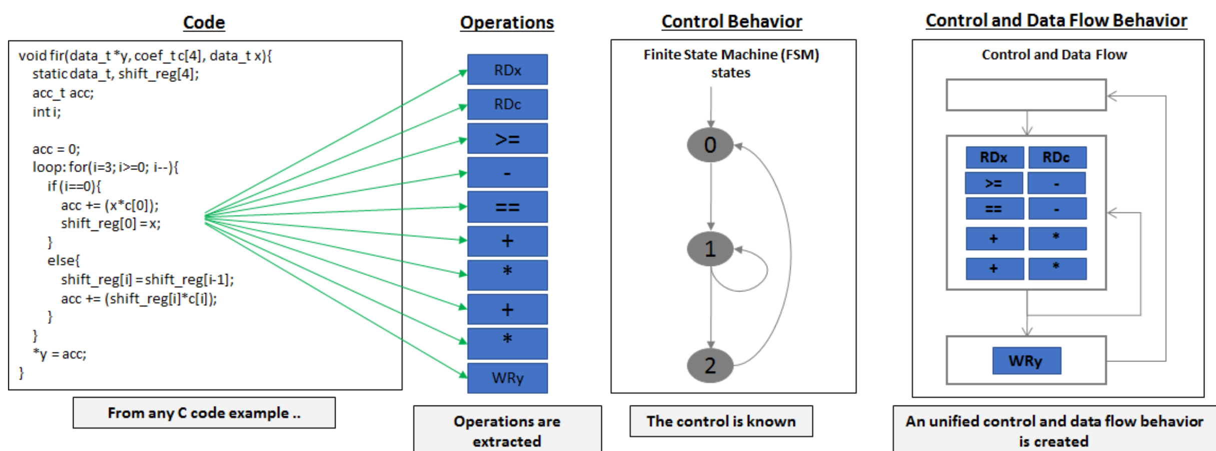
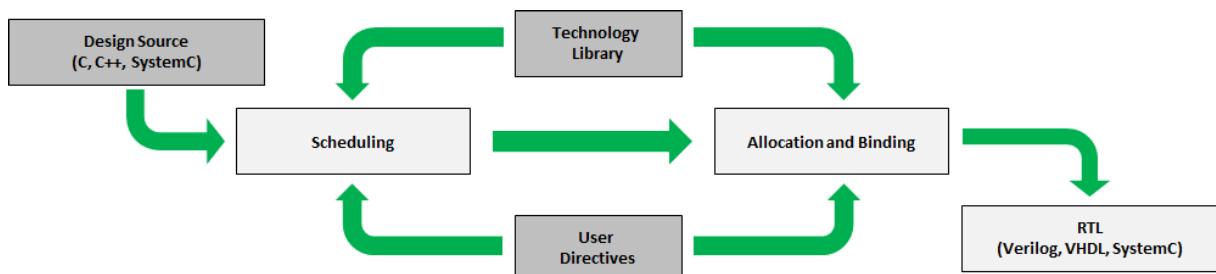


Figure 2.18 – The essential three HLS phases and their relations.



There are a large variety of HLS tools. They range from commercial products such as Vivado HLS (XILINX, 2016e) from Xilinx to open source tools developed from academic research initiatives such as LegUp (CANIS et al., 2011) from University of Toronto. An in-depth analysis and discussion of several commercial and academic HLS tools regarding performance and resources usage was performed in (NANE et al., 2015). In this thesis, Xilinx

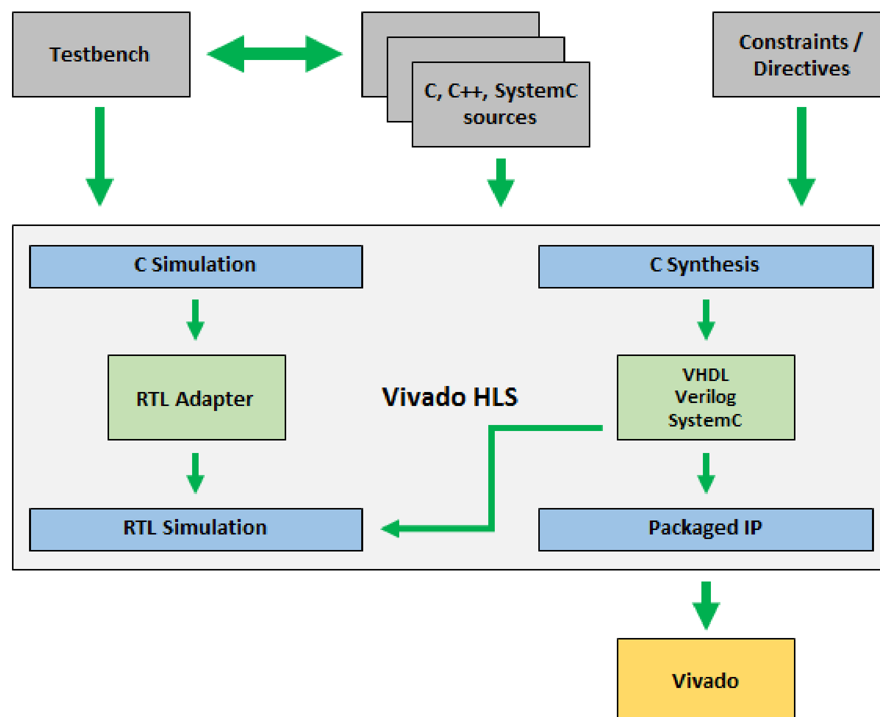
Vivado HLS was chosen for generating the HLS-based case-studies designs because as a Xilinx tool, it provides a significant support for different Xilinx devices (7-Series, such as Virtex, Artix, Kintex, and Zynq-7000), which are the main targets of this thesis.

2.2.2 Xilinx Vivado High-Level Synthesis

Vivado High-Level Synthesis is a complete HLS environment from Xilinx. It is built using the Low Level Virtual Machine (LLVM) compiler framework (LLVM, 2016). As such, it has access to many software optimizations (e.g., *loop unroll*, *loop rotation*, *dead code elimination*, etc.). However, hardware and software programing paradigms are inherently different, so it is not possible to expect that all of LLVM's optimizations work seamlessly for HLS (WINDH et al., 2015).

The typical Vivado HLS design flow (Fig. 2.19) starts with a high-level source code compiled to a pure software implementation and a self-validating testbench to verify its correctness. The user must specify the top function of the code that he wishes to synthesize to hardware. The interface provides to the user a list of code regions (targeted at loops, function bodies, and other regions) that can be optimized using synthesis directives to guide the RTL generation.

Figure 2.19 – Xilinx Vivado HLS design flow.



With regard to the optimization strategies, which is one of the main resources of any HLS tool, Vivado HLS have four main strategies for optimizing a design: clock, throughput, latency, and area.

The clock frequency along with the target device is the primary constraint which drives optimization. Vivado HLS seeks to place as many operations from the target device into each clock cycle.

Optimizing for throughput implies in pipeline the tasks to improve performance, improve the data flow between tasks, and optimize structures to improve address issues which may limit performance. Task pipelining allows operations to happen concurrently, which means that the task does not have to complete all operations before it begins the next operation. Pipelining can be applied to functions and loops. In case of a loop, for example (Fig. 2.20), if each iteration contains three operations, one read, one computation, and one write, the loop can be pipelined to read on every clock cycle instead of every three. Thus, a new loop iteration begins on every clock cycle before the previous iteration is finished. It is also possible to improve pipelining by partitioning arrays to improve the data flow between tasks. Arrays are implemented as BRAMs, which only have a maximum of two data ports. This can limit the throughput of a read/write (or load/store) intensive algorithm. The bandwidth can be improved by splitting the array (a single BRAM resource) into multiple smaller arrays (multiple BRAMs), effectively increasing the number of ports. Concerning the structural optimizations, the main one is the loop unrolling to improve the parallelism and/or pipelining, as shown in Fig. 2.21. By default, loops are kept rolled in Vivado HLS. That is to say that the loops are treated as a single entity: all operations in the loop are implemented using the same hardware resources for iteration of the loop. It is possible to unroll loops by a factor of N so the loop operations can be performed N times faster. However, unrolling a loop is directly related to how much resources of the FPGA are used. For example, unrolling a loop by a factor of three (N=3) means triple the speed, but also triple the resource cost.

Optimizing for latency uses the techniques of latency constraints and the removal of loop transitions to reduce the number of clock cycles required to complete. One option is to merge sequential loops, as shown in Fig 2.22. Rolled loops imply and create at least one state in the design Finite State Machine (FSM). When there are multiple sequential loops, they can create additional unnecessary clock cycles and prevent further optimizations. Thus, merging sequential loops allows the logic within the loops to be optimized together.

Optimizing for area focus on how the operations are implemented, such as controlling the number of operations and how those operations are implemented in hardware. One of the

primaries concerns is related to data types and bit-widths of the high-level languages. For example, if a variable only requires 12-bits but is specified as an integer type (32-bit), it will result in larger and slower 32-bit operators being used, reducing the number of operations that can be performed in a clock cycle and potentially increasing the initiation interval and latency. In this context, the use of appropriate precision for the data types is mandatory. Another option is inlining functions. Function inlining removes the function hierarchy aiming to improve area by allowing the components within the function to be better shared or optimized with the logic in the calling function. It is also possible to reshape arrays aiming to reduce the number of BRAMs while still allowing the beneficial attributes of partitioning such as parallel access to the data.

Figure 2.20 – Function pipelining behavior.

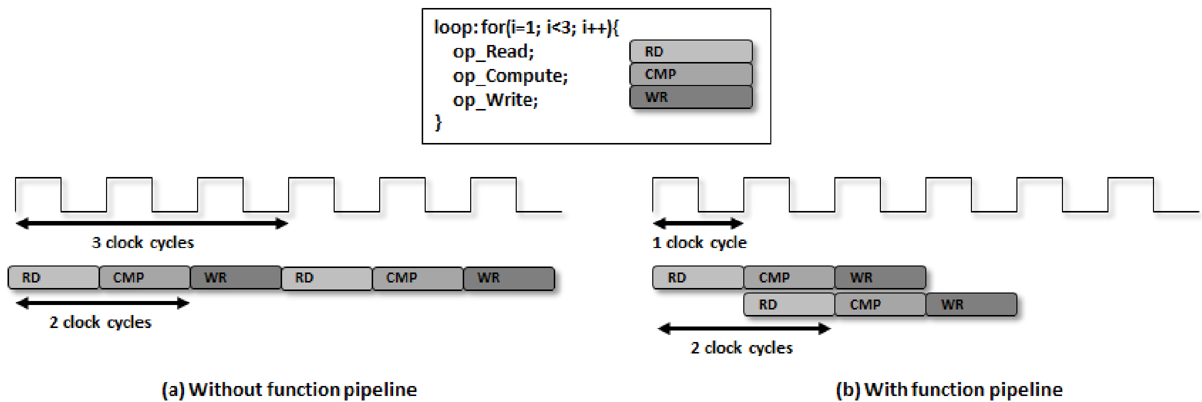


Figure 2.21 – Loop unrolling behavior.

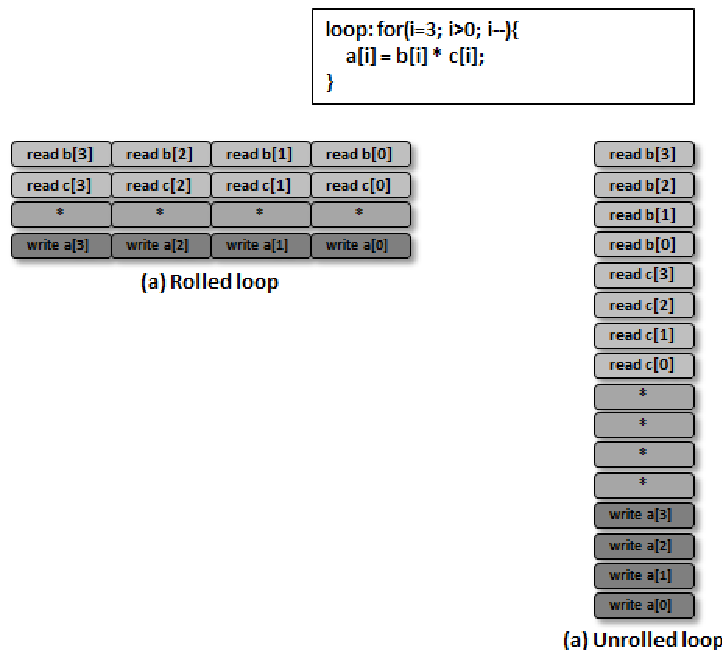
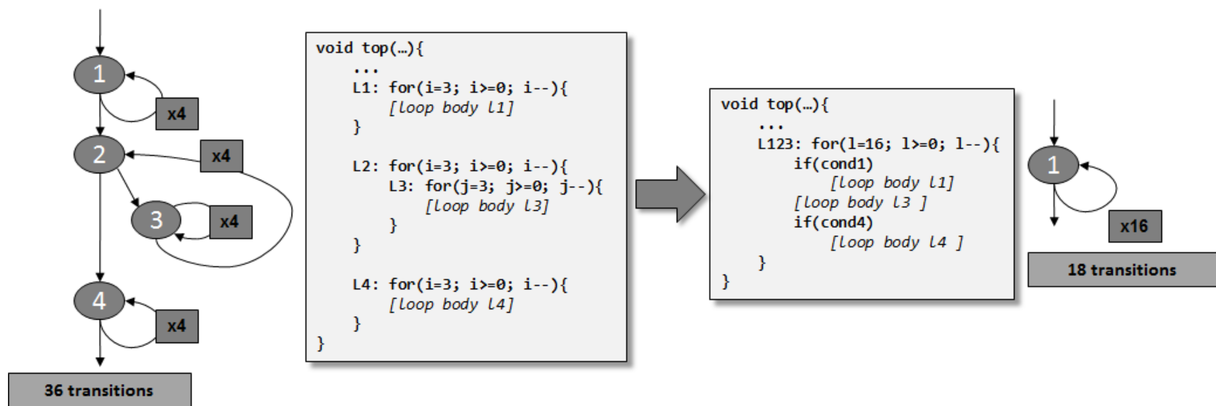


Figure 2.22 – Loop merging behavior.



There are also the interfaces optimizations, which affect both throughput and latency. In high-level programming languages such as C and C++, all input and output operations are performed in zero time, through formal function arguments. In a RTL design, these same input and output operations must be performed through a port in the design interface and typically operates using a specific I/O protocol. Vivado HLS supports two solutions for specifying the type of I/O protocol used: interface synthesis, where the port interface is created based on industry standard interfaces, such as AXI4; and, manual interface specification where the interface behavior is explicitly described in the input source code, which allows any arbitrary I/O protocol to be used. As an example, specifying input and output arguments of a function as an AXI4 interface is a common practice when another device, such as a CPU (Zynq-7000's PS, Microblaze soft-core, etc.), is used to configure and control when this block starts and stops its operation.

2.2.3 Related works about HLS

In (HARA et al., 2008), authors proposed the CHStone, a suite of benchmark programs for C-based high-level synthesis. CHStone consists of dozen of large, easy-to-use programs written in standard C programming language, which were selected from various application domains, such as arithmetic, microprocessor, media processing, and security. This thesis makes use of some of the CHStone benchmark programs.

In (HUANG et al., 2013), authors studied the effect of compiler optimizations on the hardware metrics of circuit area, execution cycles, frequency of the circuit, and wall-clock time. They made use of the LegUp academic HLS tool (CANIS et al., 2011). Among the

several results achieved, they observed that the hardware quality is affected by the optimization parameter values, as well as the order in which optimizations are applied.

Nane et al. (2015) presented the first-published methodology to evaluate different HLS tools. They compared one commercial and three academic HLS tools on a common set of C benchmarks in terms of performance and use of resources. Windh et al. (2015) also analyzed several HLS tools, but more qualitatively. They analyzed the tool flow, the optimizations they provide, and their hardware implementations of the high level code.

In (HUSEJKO, EVANS, DA SILVA, 2015), authors performed a feasibility investigation to verify if HLS tools are capable and mature enough to be applied for building critical data acquisition systems. Their case-study was the CERN CMS detector ECAL Data Concentrator Card (DCC). They concluded that the Vivado HLS fills all the constraints and can be used for building critical data acquisition systems.

In (WINTERSTEIN, BAYLISS, CONSTANTINIDES, 2013), authors performed a comparative study between two alternative algorithms that perform the same compute-intensive machine learning technique (clustering), but with significantly different computational properties. They compared a data flow centric implementation to a recursive tree traversal implementation, which incorporates complex data-dependent control flow and makes use of pointer-linked data structures and dynamic memory allocation. As results, they observed similar performances between the hand-written and automatically generated RTL designs for the first test case and also degradation in latency by a factor greater than 30 times if the source code is not altered prior to high-level synthesis.

Monson, Wirthlin, and Hutchings (2013), compared the performance of CPU and FPGA based implementations of a complex optical-flow algorithm. For the FPGA based implementation, the Vivado HLS was utilized. Using Vivado HLS, the designers were able to develop an implementation of the algorithm with comparable performance to the CPU implementation that operated at a fraction of the energy cost. The authors came to several important conclusions regarding the use of the Vivado HLS tool: little modification is necessary to prepare existing C language designs for conversion using the HLS tool; it is possible quickly optimize a design for different goals, and, it is easy to compare different versions of the algorithm in C and determine resource usage and performance.

With regard to reliability, several works have been published in recent years. Chen et al. (2016) evaluated by fault injection the sensitivity of HLS-based designs protected with fine grain module redundancy or gate sizing, targeting ASIC applications in satellite communications systems. By mixing both techniques, authors reached an area and power

reduction of 70% and 64%, respectively, with a reliability level over 99.97%. Fleming and Thomas (2016) proposed an approach which distinguishes between tolerable errors in data flow, such as arithmetic, and intolerable errors in control flow, such as branches and their data-dependencies. The approach is demonstrated in a new HLS compiler pass called StitchUp, which precisely identifies the control critical parts of the design, then automatically replicates only that part. They applied StitchUp to the CHStone benchmark suite and performed exhaustive hardware fault injection in each case, finding that all control flow errors were detected while only requiring 1% circuit area overhead in the best case. In (CHEN, EBRAHIMI, TAHOORI, 2016), authors propose a novel reliability-aware allocation and binding technique to explore more effective soft error mitigation during HLS processes. They perform a vulnerability analysis at behavior level by considering error propagation and masking in both control and data flows. Then optimizations based on integer linear programming, as well as heuristic algorithm, were employed to incorporate the behavioral vulnerabilities into the register and functional unit binding phases to achieve cost-efficient error mitigation. Their experimental results reveal that compared with previous techniques which ignored behavioral vulnerabilities, the proposed approach can achieve up to 85% of reliability improvement with the same amount of area budget in the RTL design. In (DOS SANTOS et al., 2017), authors evaluated the use of module redundancy with different granularities at the higher level of HLS processes, the high-level programming language. The advantage of this approach is that it is easily applicable in any HLS tool, including the commercial ones in which it is not possible to alter the HLS flow. Results show that by using a coarse grain module redundancy with triplicated inputs, voters, and outputs, it is possible to reach 95% of reliability by accumulating up to 61 bit-flips in the configuration memory bits of an SRAM-based FPGA.

2.3 Implementation metrics

In Zynq-7000 and other APSoCs, the area of an implemented design in the PL can be expressed in terms of the number of used resources such as LUTs, flip-flops, BRAMs, DSPs, etc. It is also possible to express the area in terms of configuration bits and configuration frames. In the PS, the resource usage of a program can be expressed mainly in terms of the type and amount of the used memories, such as L1 cache, L2 cache, OCM, and BRAMs. In

terms of reliability, the resource information is important since it is used to determine how much the design is physically exposed to radiation.

The performance of a design can be expressed in terms of the execution time, operational frequency, and the processed workload. The execution time can be defined by the number of clock cycles needed to perform an operation. According to the FPGA and the design embedded in it, or in case of a processor, a maximum clock frequency is achieved. Another important parameter is the workload processed by the design. The workload is the amount of data computed at each design execution. In terms of reliability, performance information is important to determine how much time the design is exposed to soft errors during the execution of the implemented function.

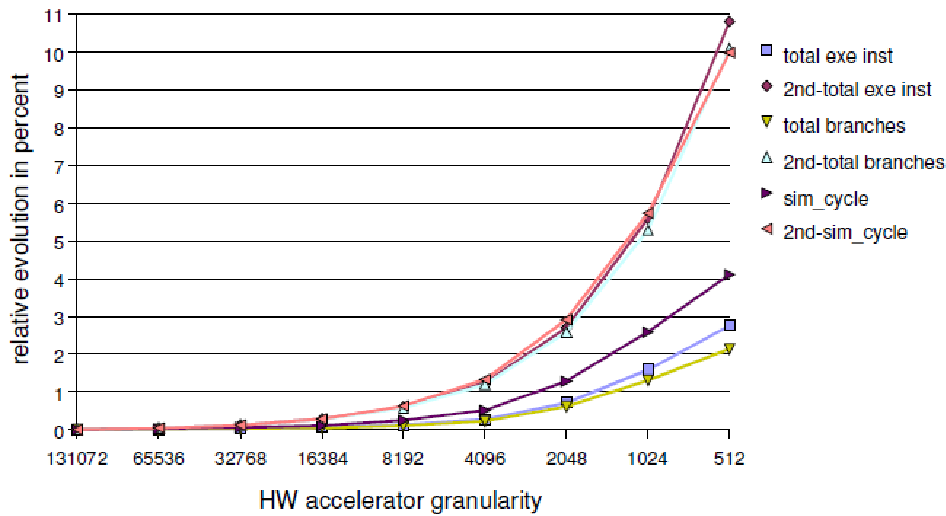
2.4 Related works about APSoCs

Most of the works that use an APSoC as case-study device take into account only performance measurements, since its main goal is to provide a high system-level performance through a flexible platform, as already exemplified before by citing the work of Silva, Sklyarov, and Skliarova (2015). However, other interesting works are worth to be cited.

A methodology for analyzing the impact of the hardware accelerator data transfer on the performance of a typical embedded system is presented in (LAFOND, LILIUS, 2008). Such work is particularly interesting because it shows that the granularity of the data transfer between memory and accelerator, and thus, the interrupt rate to the CPU has a direct impact on the system performance, as Fig. 2.23 shows. In this figure, all the results presented are relative to the measurements using the values obtained for the coarse-grained system. Results show an exponential improvement when the hardware granularity is reduced. This improvement can be explained by a decrease in data cache misses due to a decrease in the number of data accesses. It is worth noticing that all data were obtained using a simulation framework based on an ARM processor running a Real-Time Operating System (RTOS).

In (ALTERA, 2009), it is shown the benefits of using heterogeneous architectures to reduce power consumption and speed-up performance in FPGAs. The paper shows practical comparisons for the power consumption and performance enhancement of sample computational tasks, when they are executed by a CPU or by a hardware accelerator (generated by an HLS tool). Results show that is possible to speed-up a task up to 435 times with a small increase in power consumption of only 1.9 times.

Figure 2.23 – Influence of the hardware accelerator granularity on the execution time of a system (LAFOND, LILIUS, 2008).



The idea of using a portion of CPU sub-system caches as buffers for accelerators is studied in (FAJARDO et al., 2011). Such approach results in a smaller silicon area since each accelerator does not instantiate its own buffer. The basic idea of dedicating a shared memory space to accelerators is interesting because Zynq-7000 devices provide a dedicated On-Chip Memory (OCM), which can be used for the same purpose.

The impact of cache architecture on the performance and area of FPGA-based processor and parallel accelerator systems is discussed in (CHOI et al., 2012). The paper proposes a simple hardware containing one MIPS core, multiple accelerator units (generated by an HLS tool), a multi-port shared L1 cache and a DRAM controller. It considers different structural parameters for the L1 cache (such as number of ports, associativity, etc.) and defines a set of computational tasks to be done only by accelerators. It then quantifies the impact of cache structure on the overall speed of accelerators connected to the L1 cache. Fig. 2.24 shows that larger cache sizes generally provide higher speed-ups, as authors expected. For the parallel cases, six (6) accelerators were used. The detailed configuration of all the configurations evaluated can be found in (CHOI et al., 2012).

In (SADRI et al., 2013), authors systematically evaluated the performance that is achievable in practice with a Zynq-7000 device by comparing the HP-AXI port and ACP. However, the experiments were exclusively for applications running under a Linux operating system distribution. The main results of this work, which are shown in Fig. 2.25, can be summarized as follows: if a CPU collaborates with an accelerator in the PL, then the speed of the CPU ACP and the CPU OCM methods is always better than the CPU HP-AXI; and, if an

accelerator in the PL is entirely responsible for the problem and the CPU only uses its results, then the ACP or the OCM are recommended if the amount of data to be processed is smaller than the size of the cache memory or the OCM. Otherwise, it is supposed that the HP-AXI ports give better results. As mentioned in (LAFOND, LILIUS, 2008), authors also state that the size of the packets that are transferred and the burst length have a substantial effect on the overall data transfer bandwidth.

Figure 2.24 – Impact of the cache architecture on the average execution time (speed-up) relative to a baseline system, which employs a single accelerator running sequentially using a 1-way (direct-mapped) 2KB cache with a 32-byte line size (CHOI et al., 2012).

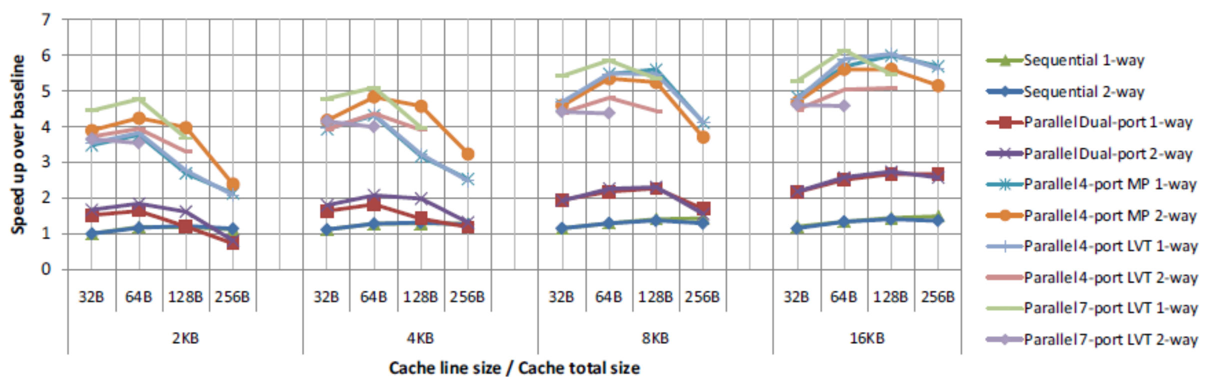
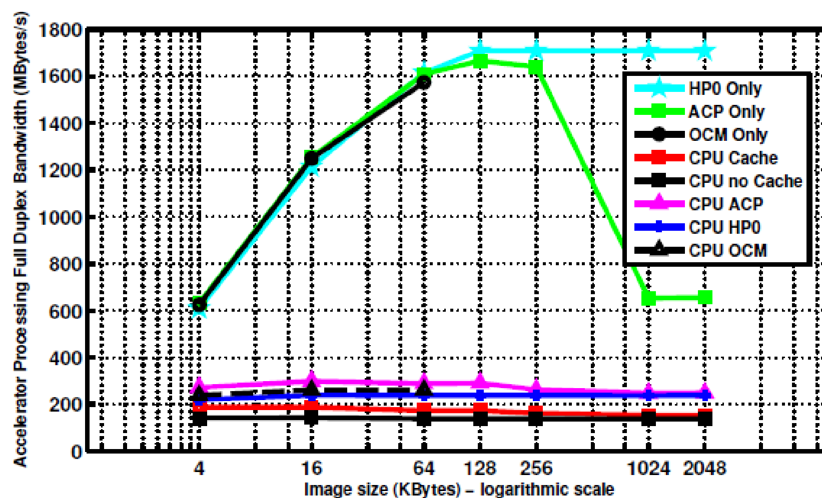


Figure 2.25 – Processing bandwidth comparison of different acceleration methods in Zynq-7000. Data size sweeps from 4 KB to 2048 KB (SADRI et al., 2013).



2.5 Summary

In summary, based on the information presented in this chapter, one can see that programmable devices have evolved very rapidly in the last decade, mainly because of performance pressure in the high-volume commercial marketplace. As consequence, several APSoC devices were introduced in the market providing higher programmable flexibility and overall system performance at lower costs than standalone processors and FPGAs, as the related works show. However, as the next chapters show, the high complexity and density of these devices increase the system's susceptibility to noises that are present in the environment, such as the ones caused by radiation.

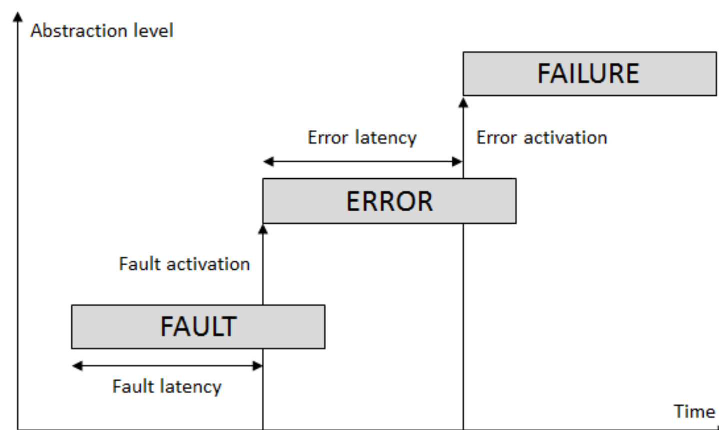
3 RADIATION EFFECTS ON APSOCS

This chapter introduces the main definitions and background knowledge for understanding the context of this thesis. It presents the cause-effect definitions of fault, error, and failure; the main radiation environments; the general effects of radiation on programmable devices; the standard metrics and methods for evaluating programmable devices under radiation; and, related works.

3.1 Fault, error, and failure

A system is an entity that is composed by one or more subsystems. All subsystems interact with each other, while the system interacts with other systems in its environment (AVIZIENIS et al., 2004). In this context, a system or a subsystem can be hardware-based, such as a processor, or software-based, such as a running application. The service delivered by a system is its behavior as perceived by other systems using it. According to (MUSHTAQ, AL-ARS, BERTELS, 2011), a system fails when its behavior deviates from the expected one. In this context, *fault*, *error*, and *failure* are three concepts related by a *cause-effect* link, as shown in Fig. 3.1.

Figure 3.1 – Fault, error, and failure propagation.



One failure may occur due to one or more faults within the system or external to it. When a fault becomes active, it can affect the total state of one or more subsystems of the system. The deviation of the total state of a component from the correct state is known as an error. When an error propagates to affect the external state of the system, it is said that the

error was activated. Once the error is activated, it is said that the failure of the system occurred. In other words, a fault might lead to an error, which in turn might lead to the failure of the system. However, it is worth noticing that not every fault generates an error and not every error generates a failure.

A fault is defined as a logic abstraction of a physical defect. A physical defect is an unexpected difference between the implemented hardware and the planned function of it. Faults can be classified with respect to persistence as transient, intermittent, or permanent. Transient fault is random and occurs for only a short period of time. Intermittent fault is a repetitive malfunction of a device or system that occurs at intervals and a permanent fault is continuous in time. This thesis targets transient faults caused by ionizing particles that pass through the device, which are generally named Single Event Effects (SEEs).

3.2 Radiation environments

For the purposes of this thesis, radiation is the transmission of energy through atomic and subatomic particles with very high kinetic energy. Radiation is a natural phenomena and is generated from the sun, cosmic sources, materials on Earth, and man-made environments, such as particle accelerators (for high-energy physics experiments and cancer treatment) and nuclear reactors.

With the increasing interest in using programmable systems in safety-critical markets where radiation is a major concern, researchers have investigated the suitability of commercially available programmable devices in such markets due to their low-cost compared to radiation-tolerant devices. This section reviews three of these markets, space, terrestrial, and particle accelerators, which are the ones of interest to this thesis.

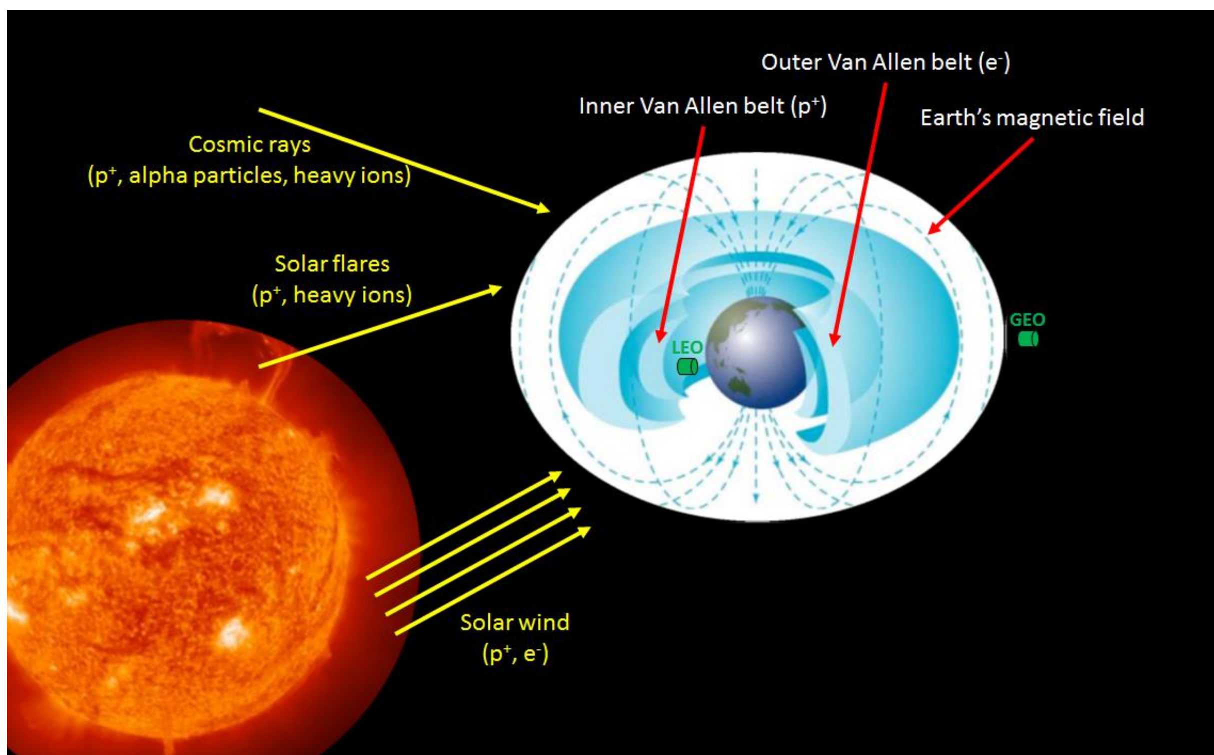
3.2.1 Space environment

Programmable devices have been used in the space environment for many years (KATZ et al., 1994). Today, the use of programmable devices such as FPGAs and APSoCs within modern spacecraft is motivated by the growing computational needs associated with modern sensors adopted (WIRTHLIN, 2015). Because of the huge amount of data generated by modern sensors, it is no longer possible to send all sensor data back to Earth for processing. Today, much of that processing must be done on the spacecraft.

The space environment is complex and includes a large spectrum of particles of different mass each with a different energy range. The primary challenge for using FPGA and APSoC devices in a spacecraft is to address the effects of radiation on the device operation. The radiation experienced by satellite electronics in space is generated from several different sources, as Fig. 3.2 shows. In summary, they are the following (CLAEYS, SIMOEN, 2002; BOUDENOT, 2007):

- Protons and heavy ions from solar flares;
- Cosmic ray protons and heavy ions;
- Protons and electrons trapped in the Van Allen Belts;
- Heavy ions trapped in the Earth's magnetosphere.

Figure 3.2 – The space environment and its sources of ionizing particles.



The radiation level of these sources strongly depends on the activity of the sun. The solar cycle is normally divided in two main activity phases, the solar minimum and the solar maximum. On the average, the cycles last for eleven years with approximately four years of solar minimum and seven years of solar maximum (NASA, 2011). The activity of the Sun can alter the spatial scenario during the period of solar maximum due to the increasing number of solar flares. In contrast, during the period of solar minimum, the flux of cosmic rays tends to increase, once the interplanetary magnetic field is weaker during this phase.

Solar flares cause protons and heavy ions. These flares, which can last for several hours to a few days, have protons with energies higher than 100 MeV that are attenuated by the Earth's magnetosphere.

The galactic cosmic ray particles originate outside the solar system and include a large range of elements. The fluxes are low, but because they include high energetic particles (from dozens of MeV up to hundreds of GeV) of heavy elements, they may produce ionization effects by passing through the materials. The composition of particles consists of about 90% protons, 9% alpha particles, and 1% heavy ions and other elements (GOLDHAGEN, 2003).

The Van Allen belts consist mainly of electrons up to a few MeV in energy and protons of up to several hundred MeV trapped in the Earth's magnetic field (GUSSENHOVER, MULLEN, BRAUTIGAM, 1996). The inner belt is situated at low altitudes (from hundreds of kilometers to 6,000 kilometers), while at high altitudes (up to 60,000 kilometers) the outer belt with high-energy electrons is observed. Once the charged particles are trapped, the Lorentz force controls their motion in the Earth's magnetic field (CLAEYS, SIMOEN, 2002). The South Atlantic Anomaly (SAA) is an important an important anomaly that happens on the region of the inner belt. It results of the offset and tilt of the Earth's magnetic field with respect to the Earth rotation axis, in which the field lines containing significant energetic-particle fluxes approach the Earth's surface (ESA, 1993). As consequence, the flux of energetic protons at low altitudes in the SAA can be two orders of magnitude higher than in other regions of the Earth's magnetosphere. The SAA is especially important for southern Brazilians because it concentrates mainly over the Brazilian state Rio Grande do Sul.

The altitude of the orbit is also essential to evaluate the radiation environment that the spacecraft will encounter during a space mission. In Low Earth Orbits (LEOs), the spacecraft passes several times per day through the Van Allen Belts, i.e., trapping protons and electrons. In Geostationary Orbits (GEOs), trapped protons with energy levels below the threshold for initiating nuclear reactions are present. In case of deep space missions, the radiation environment is more complex and it depends on the number of times the spacecraft passes through the Earth's radiation belt and on how close it will be to the sun (CLAEYS, SIMOEN, 2002). In addition, the solar maximum and minimum has to be taken into account.

3.2.2 Terrestrial environment

Another important radiation environment for programmable systems is the Earth's terrestrial environment. The earth environment is usually not considered a "harsh" radiation environment such as space. However, electronic circuits operating in terrestrial environments are exposed to radiation that can negatively impact their operation. While upsets within FPGAs and APSoCs due to terrestrial radiation are not so common, they do occur and are easily detectable using conventional error detection techniques.

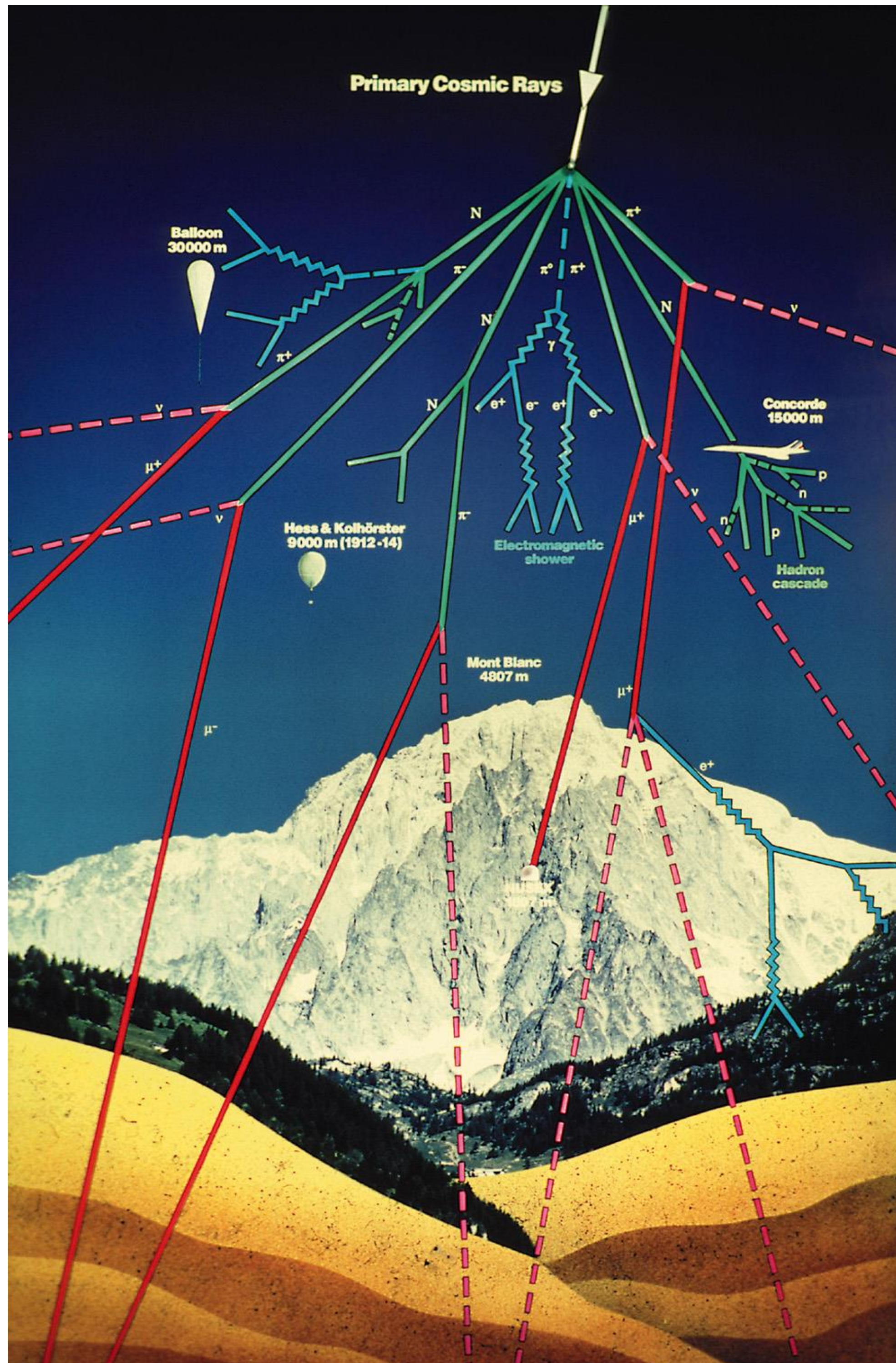
The Earth is constantly showered by high energetic particles that come from the sun and the outer space, such as the galactic cosmic rays. Such particles are usually referred as Primary Cosmic Radiation (PCR) and they are illustrated in Fig. 3.3.

When a PCR enters in the Earth's atmosphere, it will interact with other particles, such as nitrogen and oxygen atoms. These interactions trigger a process called of spallation, in which the atoms are divided into a broad spectrum of different particles, both stable and unstable. This spallation process generates a chain of reactions that produces exponentially more particles until the Pfotzer maximum is reached at about 14-25 km. Below this point, the particle flux starts decreasing due to energy loss, absorption, and decay process (GRIEDER, 2001). Nevertheless, some high energetic particles are still able to reach the ground. The result of this process is a shower of secondary particles in the atmosphere, mostly high-energy neutrons that interact with electronic systems (XILINX, 2012a).

The dose rate of cosmic radiation varies throughout the world and depends on the magnetic field and altitude of the location. The higher the altitude of the system, the higher is the terrestrial soft error rate (JEDEC, 2006). Researches have shown that, at altitudes below 18 km, high energy neutrons are the dominant factor in radiation-induced failures, while over 21 km, cosmic ray heavy ions begin to dominate these rates (TSAO, SILBERBERG, LETAW, 1984). Therefore, high altitude applications of FPGAs and APSoCs (including avionics) and high reliable systems, such as communication, power, medical, automotive, and industrial applications, must carefully estimate the effects of radiation and provide proper error detection and correction capabilities. Industrial standards have been created for measuring and reporting these errors in semiconductor devices (JEDEC, 2006).

There are also terrestrial sources of radiation, which include naturally occurring materials in the earth, such as soil, rocks, water, and the air. Most naturally occurring terrestrial radiation is relatively low energy and has little impact on electronic systems. The exception to this is the ionized particles that are sometimes found within the packaging materials used to manufacture semiconductor systems (XILINX, 2012a; WIRTHLIN, 2015).

Figure 3.3 – Particle cascade generated from cosmic rays in the Earth's atmosphere (STEFAN, 2001).



3.2.3 Particle accelerators environment

FPGAs and APSoCs are increasingly being used within particle accelerators, as in their readout electronics, for performing High-Energy Physics (HEP) experiments. HEPs use

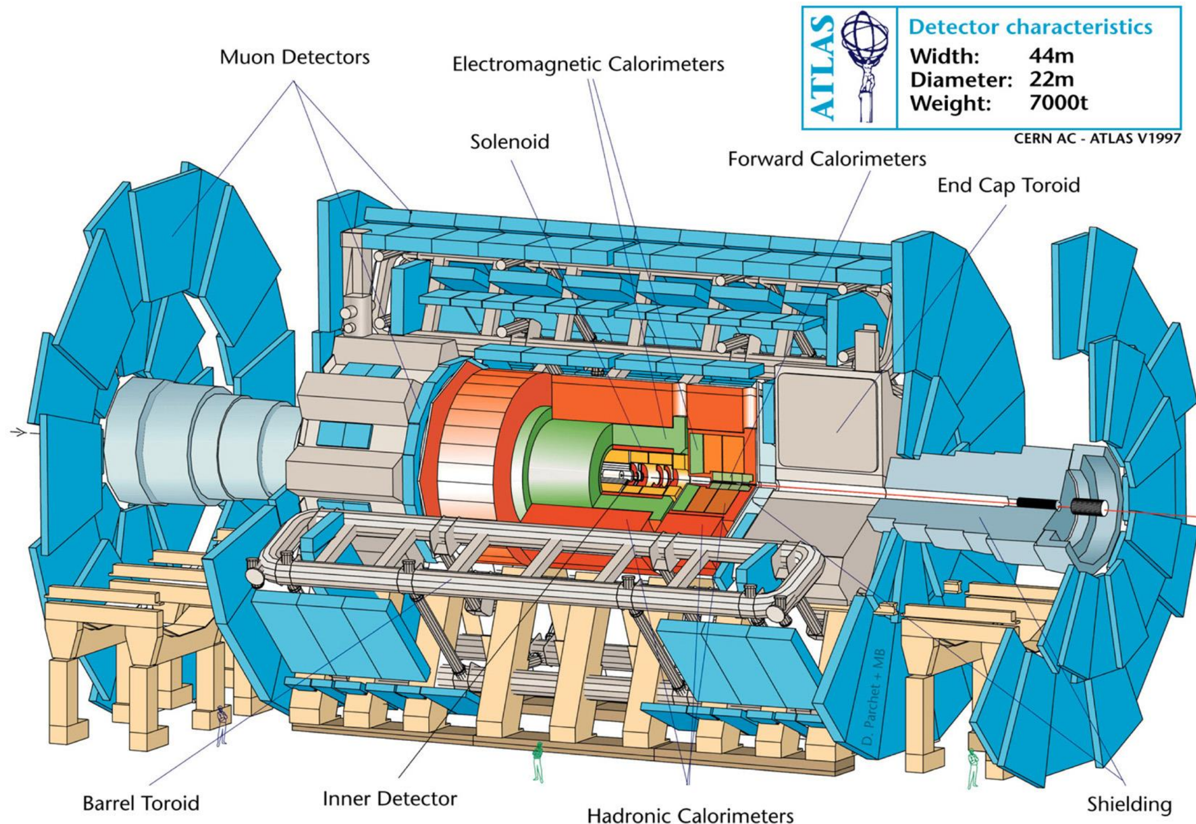
particle accelerators to accelerate charged particles to very high speeds (and, thus, high energy) and in opposite directions for then to form a particle collision. This collision generates a number of byproducts that are studied to learn more about the subatomic structure (hadrons, quarks, leptons, muons, etc.) and fundamental laws of nature.

An important part of HEP experiments is the detectors that measure the byproducts of high-energy particle collisions. A variety of particle detectors has been developed over the years and can be used to measure the energy, direction, spin, charge, etc. of a variety of particles. High-speed electronics are used within detectors to capture the particle data and send these data to external computer systems for post-processing. FPGAs are often used in the detectors of HEP experiments for interfacing with sensors, performing simple calculations, measuring sub-nanosecond time differences, and streaming the data outside of the experiment through high-speed interfaces (WIRTHLIN, 2015). Thousands of FPGAs can be used within large HEP experiments, such as the ATLAS (A Toroidal LHC Apparatus), CMS (Compact Muon Solenoid), ALICE (A Large Ion Collider Experiment), and LHCb (LHC-beauty) experiments that operate within the Large Hadron Collider (LHC) at the European Organization for Nuclear Research (CERN) (GRASSI, 2014). Recently, CERN is planning the use of APSoCs within its detectors in their next upgrade, planned to happen in 2020 (WU et al., 2016). Fig. 3.4 shows a general view of the CERN/LHC/ATLAS detector.

An intense radiation field is generated from the high-energy particle collisions within the experiments. The actual radiation environment depends heavily on the experiment itself and on the location within the experiment (in general, the radiation field is higher closer to the center of the particle detector). At some locations within the experiment, such as the inner detector, the radiation field is so high that FPGAs and APSoCs cannot be used (GRASSI, 2014). For many locations, however, the radiation environment is modest and FPGAs and APSoCs are appropriate with proper SEU mitigation methods.

The radiation environment encountered at the LHC as well as its experiments at CERN is composed of a complex mixed field of charged and neutral hadrons, photons, muons, and electrons, with energies ranging from GeVs down to thermal energies (MEKKI et al., 2016).

Figure 3.4 – General view of the CERN/LHC/ATLAS detector (ATLAS, 2016).



3.3 Radiation effects on integrated circuits

Radiation has long-term damaging effects on integrated circuits. Such effects are cumulative and its intensity is related to the energy and the exposure time of the incident radiation on the device. The exposure to high-energy ionizing radiation generates electron-hole pairs within the oxide of Complementary Metal Oxide Semiconductor (CMOS) transistors, the most common technology manufacturing until today. The generated carriers cause a buildup of charge within the oxide. This buildup of charge will change the threshold voltage, increase the leakage current, and modify the timing (mobility effects) of the CMOS transistor, leading to the parametric degradation and/or functional failure of the electronic device. Furthermore, high-energy particles can damage semiconductor materials by displacing atoms in the lattice, a process known as Displacement Damage (DD). Such displacement damage also changes the electrical parameters of the device. Lastly, the radiation will cause functional failures within the device. The amount of radiation dose that a device can tolerate

before failing to meet standard parameters specifications is called Total Ionizing Dose (TID) (QUINN, GRAHAM, 2005a).

In addition to long-term effects, the radiation from individual high-energy particles can cause immediate effects within the device that are generally called Single Event Effects (SEE) (DOOD, MASSENGILL, 2006) and are the major concern of this thesis. SEEs result from the interaction of high-energy particles with circuit elements in integrated circuits. When a high-energy particle passes through the silicon substrate of a device, charged particles are created as the result of sub-atomic particle collisions and Coulomb interaction. These particles are generated by an ionization trail along the path of the incoming particle (WANG, AGRAWAL, 2008), as illustrated on the left in Fig. 3.5. The charge deposited or generated by the energetic particle is collected in a region surrounding the drain (approximately 1 μm) and an extended region known as funnel is generated, with a depth that depends on the electrical field configuration. In general, it is accepted that this region is of approximately 2-3 μm for logic devices. In case of an energetic particle passes through a reverse biased pn-junction, a short low resistance path is momentarily created between the substrate and the struck drain terminal. The amount of charge that is collected produces a transient current that lasts until the deposited charge disappears by recombination or is conducted away via open current paths to V_{DD} or ground, returning the logic node to its original value. Fig. 3.5 on the right shows a collected charge occurring in the drain junction of the p-channel transistor. In this example, the node held the value “0”. As the current flows through the pn-junction of the struck transistor, from the bulk connected to V_{DD} and the drain, the transistor in the on-state (n-channel transistor in Fig. 3.5) conducts a current that attempts to balance the current induced by the particle strike. If the collected charge induced by the particle strike is high enough that the on-transistor cannot balance the current before the node capacitance is charged, a voltage change at the node will occur. This voltage change lasts until the charge is conducted away by the current feed through the on-transistor.

There are a variety of SEEs that must be considered before using a device in a radiation environment. SEEs can be divided into two categories: Soft SEEs versus Hard SEEs. Soft SEEs are those events that have no damaging effects and are cleared by normal device operation. Hard SEEs are events that generally result in lasting damage to the circuitry. The most relevant effects for programmable circuits are Single Event Upset (SEU), Single Event Transient (SET), Single Event Functional Interrupt (SEFI), and Single Event Latchup (SEL) (KASTENSMIDT, CARRO, REIS, 2006; DUZELLIER, BERGER, 2007; WANG, AGRAWAL, 2008; WIRTHLIN, 2015). Fig 3.6 summarizes the mentioned effects.

Figure 3.5 – Electron-hole pairs track generated by an ionized particle in a CMOS transistor (SOOS, 2009) on the left and the charge collection mechanism in an inverter gate on the right.

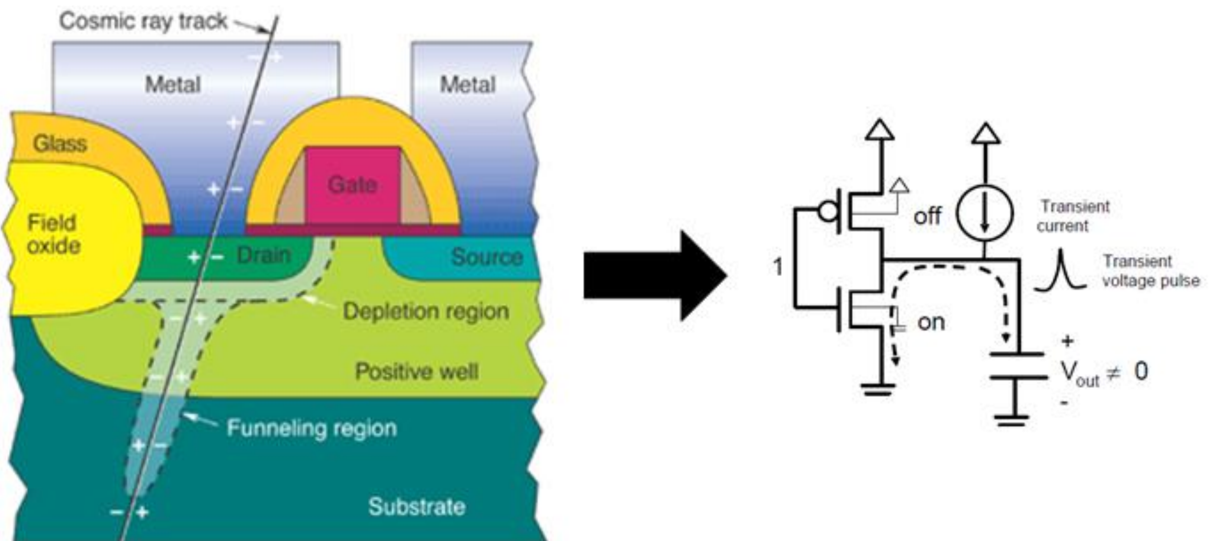
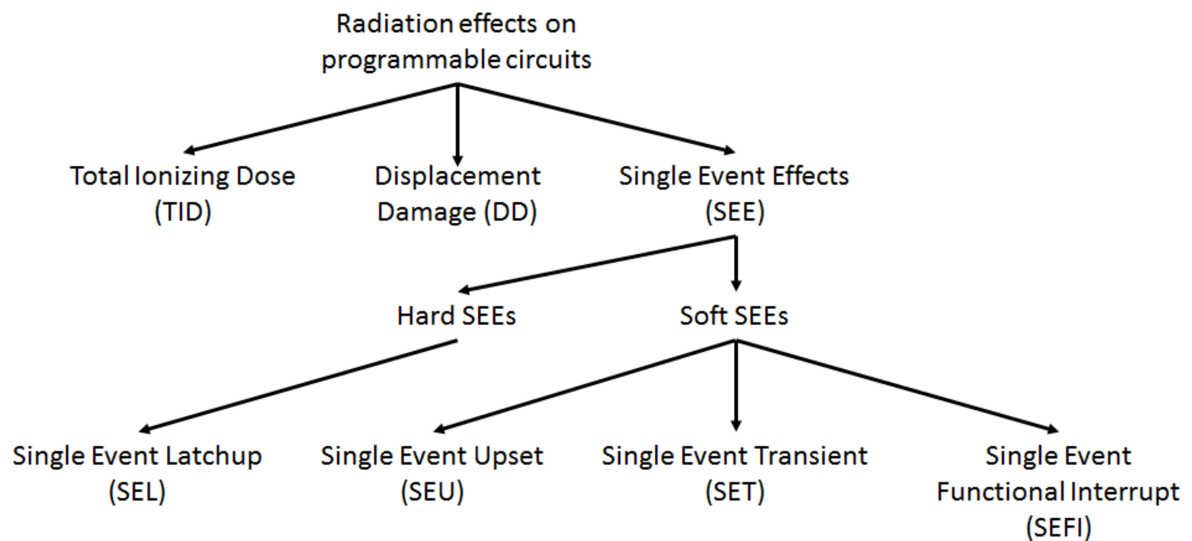


Figure 3.6 – Main radiation effects on programmable circuits. Adapted from (QUINN et al., 2015a).



An SEU is a change in the state of a memory cell (SRAM, flash, flip-flop, or latch) caused by an ionizing particle, as Fig. 3.7 illustrates. Since an ionizing particle passes through the device, charge can be transferred from one node to another. If the charge is greater than a device-specific critical charge, this charge transfer can change the voltage level of transistors within a memory cell such that the modified voltage level reflects the opposite state of the cell (i.e., changing a logic “1” to a logic “0” or a logic “0” to a logic “1”). The feedback nature of static latches, such as SRAM-based memory cells, will preserve this new value and the original value will be lost. SEUs usually refer to single-bit errors. However, as Fig. 3.8 shows,

with the dimensions of the transistors shrinking to below 28 nm, a single ionizing particle is capable of producing multiple-bit errors, an event called of Multiple Cell Upset (MCU), if more than one cell is affected, or Multiple Bit Upset (MBU), if two or more bit-flips occur in the same word. Fig. 3.8 compares the percentage of events that cause MCUs (more than one bit upset) in different families of Xilinx FPGAs, such as Virtex (180 nm), Virtex-II (150 nm), Virtex-4 (90 nm), Virtex-5 (65 nm), and Kintex-7 (28 nm) (WIRTHLIN et al., 2014a), at different energies (Linear Energy Transfer – LET).

Figure 3.7 – Example of Single Event Upset (KASTENSMIDT, CARRO, REIS, 2006).

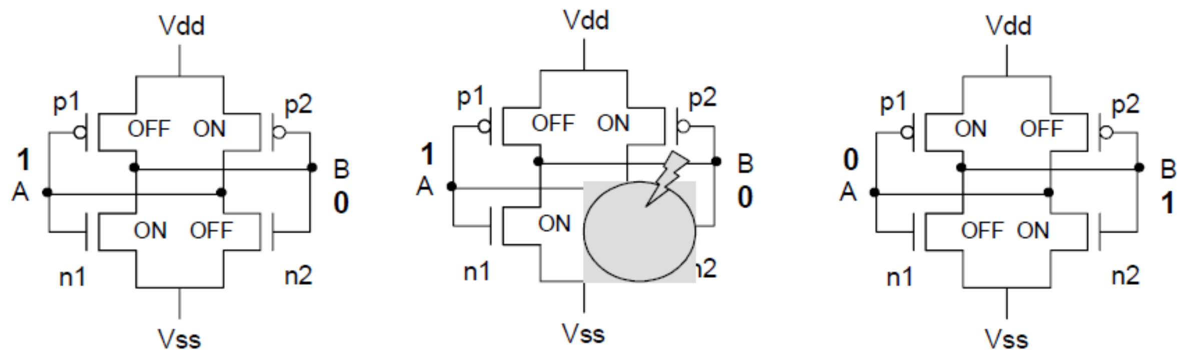
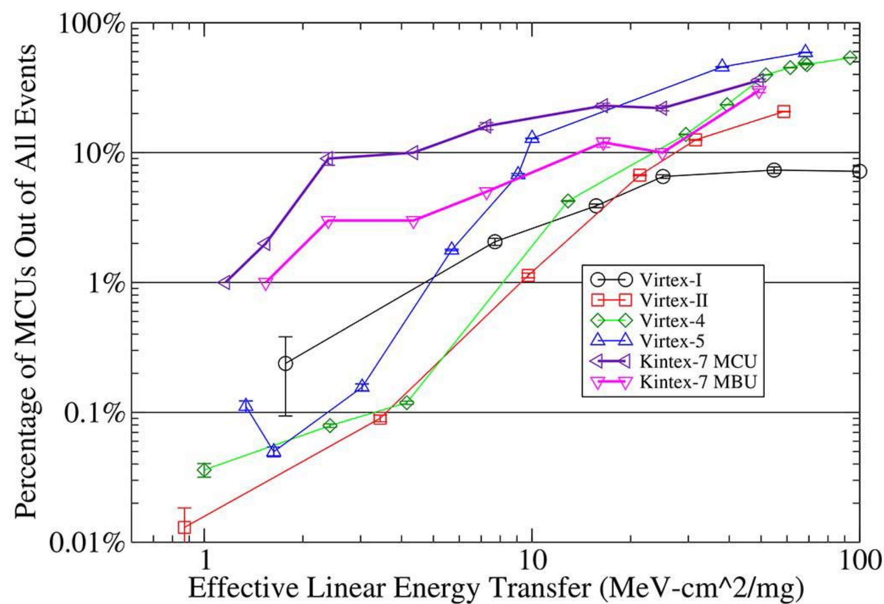


Figure 3.8 – MCU events as a percentage of SEUs for different families of Xilinx FPGAs (WIRTHLIN et al., 2014a).

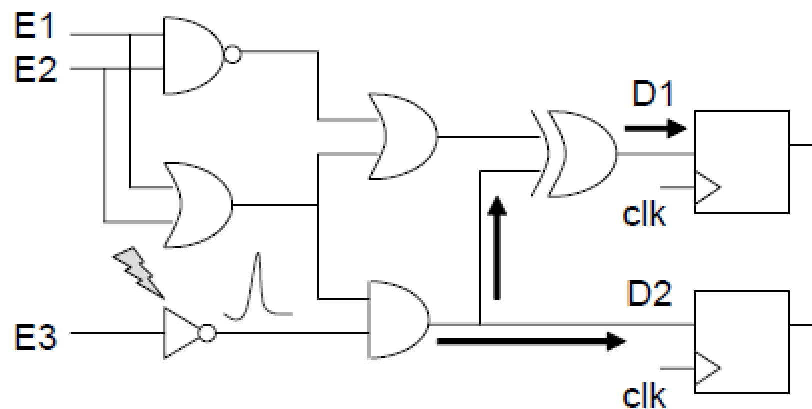


A charged particle can also induce a current and voltage spike in a combinatory circuit, which is referred to as a SET and is illustrated in Fig. 3.9. As Fig. 3.9 also shows, if

the pulse width of the spike is wide enough, the spike can propagate through the circuit and be latched, looking like a SEU. However, an SET in a combinatorial circuit might not be captured in a memory circuit because it can be masked by one of the following three phenomena:

- Logical masking - Occurs when a particle strikes a portion of the combinational logic that is blocked from affecting the output due to a subsequent gate whose result is completely determined by its other input values;
- Electrical masking - Occurs when the pulse resulting from a particle strike is attenuated by subsequent logic gates due to the electrical properties of the gates to the point that it does not affect the result of the circuit;
- Latching-window masking - Occurs when the pulse resulting from a particle strike reaches a latch, but not at the clock transition where the latch captures its input value.

Figure 3.9 – Example of a Single Event Transient (KASTENSMIDT, CARRO, REIS, 2006).



SEUs and SETs do not cause permanent damage, but they introduce unwanted transient behaviors into a circuit and can lead to errors.

SEFI is a broad term referring to an SEE that causes a significant change in the functional operation of a device (beyond a simple corruption of the user data).

SEL is a potentially destructive condition in which a single charged particle induces a parasitic *p-n-p-n* structure (WIRTHLIN, 2015). This structure produces a low-impedance path between V_{DD} and ground, resulting in large currents flowing through the parasitic bipolar transistors. In many cases, this current is high enough to destroy the device. Any device

considered for high radiation environments must be tested for latchup, since a latchup event is a potentially catastrophic failure.

3.4 Radiation effects on APSoCs

Chapter 2 has shown that the architecture and internal organization of modern devices such as Zynq-7000 have become more complex than standalone FPGAs and with faster clock speeds. Consequently, the layouts of these types of components are heterogeneous and have different radiation sensitivities (QUINN, 2014a). Fig. 3.10 shows the chip surface of Zynq-7000, where it is possible to clearly distinguish between its PL and PS parts. Therefore, Zynq-7000 and similar devices suffer the same problems of FPGAs and processors with respect to radiation effects, once they have both architectures embedded into their die. Fig. 3.11 illustrates possible SEEs occurrences in Zynq-7000 to help visualizing the effects described in the next paragraphs.

Figure 3.10 – View of the surface of a Zynq-7000 device, part XC7Z020-CLG484.

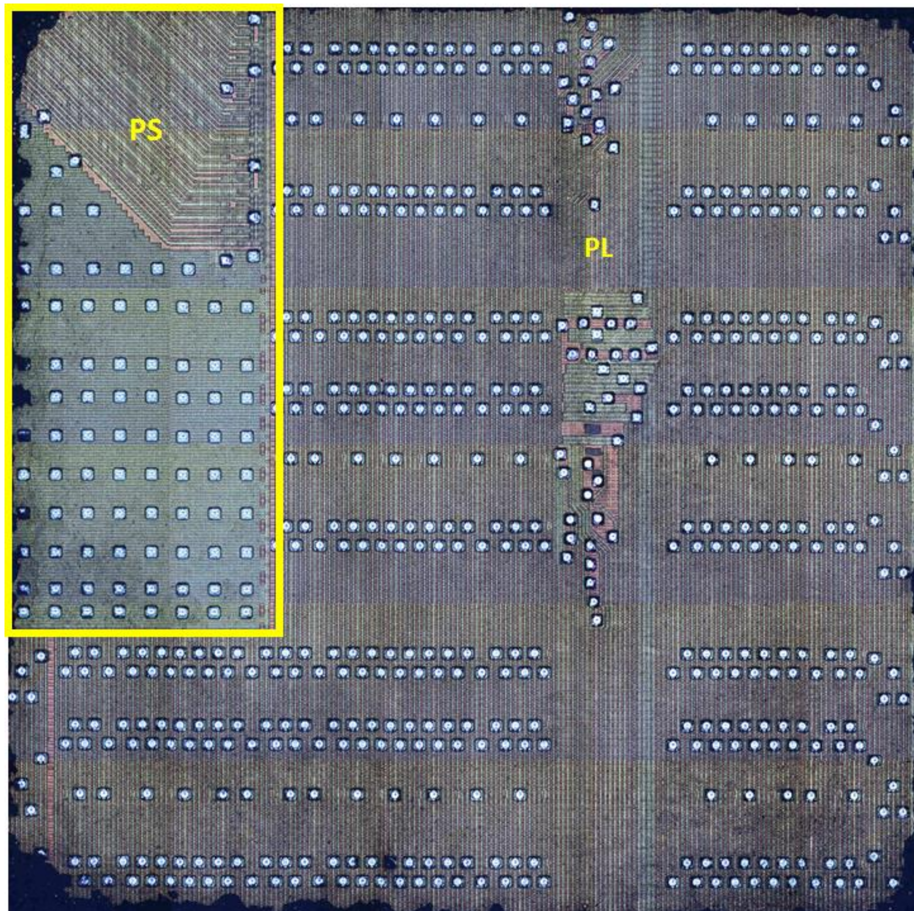
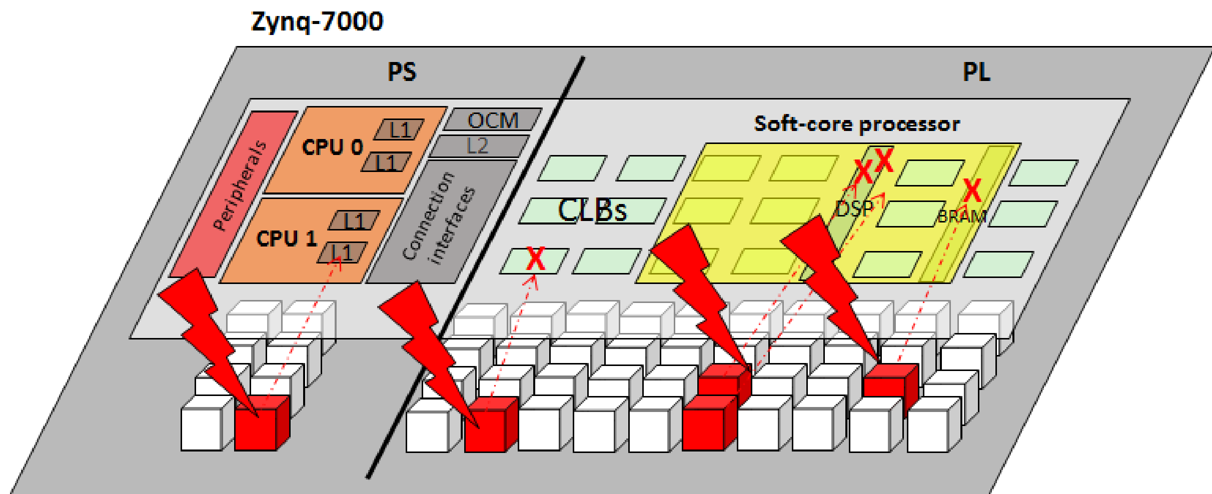


Figure 3.11 – Possible effects of SEEs in Zynq-7000 and similar APSoCs.

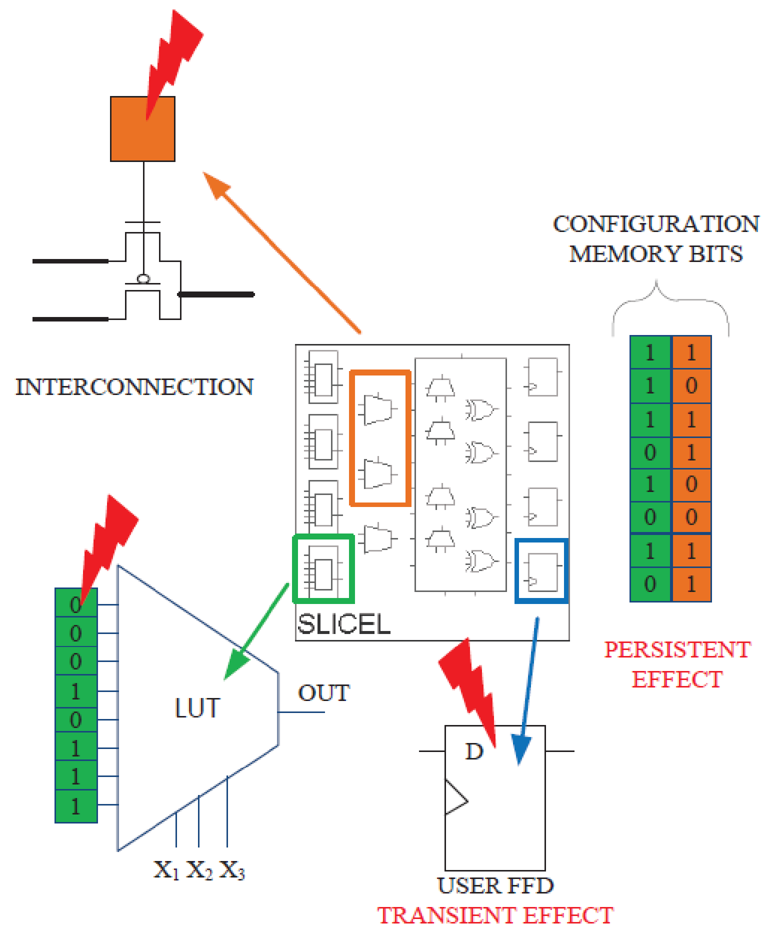


With regard to the PL part, SRAM-based FPGAs are mainly susceptible to SEUs in their configuration memory bits and embedded memory cells. A bit-flip (SEU) can occur in a configuration memory bit of the FPGA bitstream when the sensitive junction of a SRAM cell transistor collects energy deposited by an ionized particle. SEUs can alter the bits that define the combinational function of the LUTs, altering the original implemented function, as illustrated in Fig. 3.12. SEUs can also alter routing connections, generating open connections and short circuits between connections. Data stored in BRAMs can also be affected by SEUs. It is very important to highlight that such modifications caused by bit-flips are persistent until some action is taken to correct the configuration memory. Such fault persistence is the main difference between the SEUs effects on ASIC-based devices, such as processors, and SRAM-based FPGAs.

Bit-flips can also occur in the flip-flops of CLBs used to implement the user's sequential logic, as also illustrated in Fig. 3.12. In this case, the bit-flip has a transient effect and the next load of the affected flip-flop can correct it. Thus, the majority of the persistent errors observed in harsh environments come from bit-flips in the configuration memory bits. It is also worth remembering that once state-of-the-art SRAM-based FPGAs are built with cutting-edge manufacturing processes (sub-28 nm) and they are composed of millions of SRAM cells to store their configuration, they are very susceptible to MBUs and MCUs.

Another important point is that for a typical design loaded into an FPGA, only a fraction of the total number of configurable memory cells are used. Thus, depending on the design, a different number of configuration bits are used and a different number of susceptible bits may be responsible for provoking an error in the design output. According to Xilinx

Figure 3.12 – Possible effects of SEUs in SRAM-based FPGAs (TONFAT, 2015).



(2012b), the first number is defined as the Essential Bits and the second number as the Critical Bits of a design. Essential bits are the bits associated with the circuitry of the design, and are a subset of the device configuration bits. However, bit-flips in essential bits might not affect the function of the design. Only bit-flips in critical bits, which are the bits that cause a functional failure if they change state, and are a subset of the design essential bits, may corrupt a design implemented into an FPGA. In Xilinx devices, essential bits are calculated using a proprietary algorithm of the Xilinx tool after the bitstream is generated. On the contrary, generating a complete list of critical bits for a specific design is a time-consuming process that involves validating the correct design behavior while moving an upset through all the configuration memory bits in the design by using some fault-injection platform.

With regard to the PS part, radiation effects such as SEUs and SETs can have complex effects on a processor and its executing software (QUINN, 2014a). Such effects affect processors by modifying values stored in memory elements (such as registers or embedded memories), leading the processor to incorrectly execute an application, producing a wrong

output, or even entering into a loop and never finishing its execution. According to Velazco and Faure (2007), SEEs in processors may result in several types of errors, depending on the hardware unit affected.

SEEs in the register file may instantaneously impact the program output, resulting in a corrupted output or processor hang (the program execution flow is crashed and the processor needs a soft reset to restart).

SEEs in arithmetic units may lead to incorrect computations.

SEEs in the central interconnect and processor bus, which embed registers to latch data as well as address, may lead to incorrect data/address read or write.

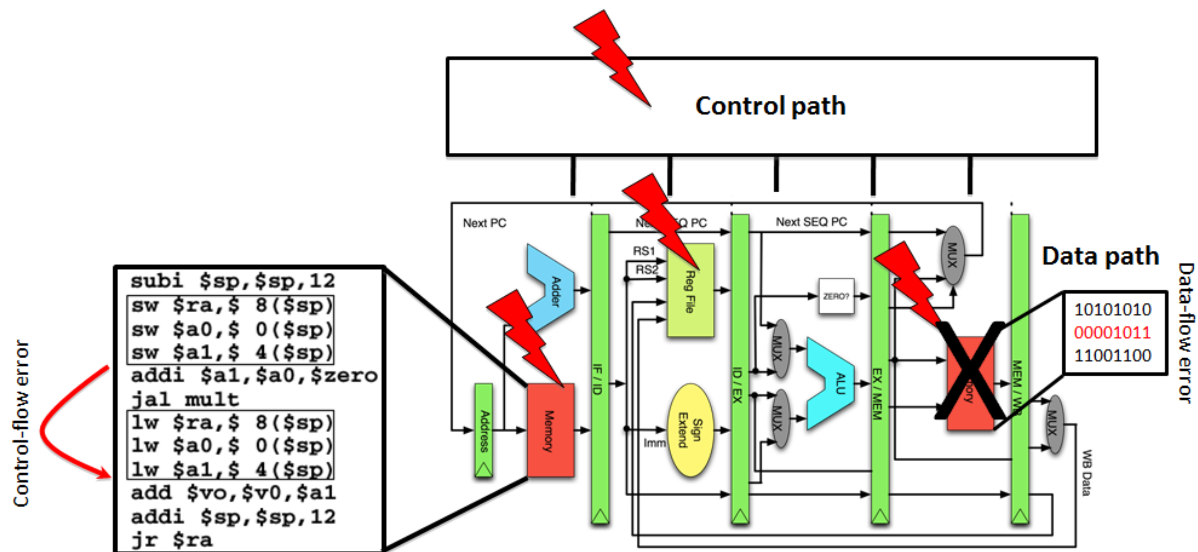
SEEs in the instruction cache are more complex. Instruction caches are usually split in two areas. The largest one is an SRAM array that store fetched instructions. The second one is a tag array, whose purpose is to validate/invalidate fetched code. Upsets in the tag array have mainly two consequences: if an upset invalidates an instruction to be executed, the direct consequence is a delay in the program execution, since this instruction will have to be fetched again (cache miss); if an incorrect instruction is validated, the program flow will be crashed. Consequences of upsets in the instruction array are hard to predict. If the corrupted instruction is not validated by the tag array, no incorrect behavior will be observed. If the code is validated by the tag array, three situations are possible: the corrupted instruction is not anymore in the processor instruction set, so when the control unit tries to decode it, an exception/trap will be generated; the bit-flip changes the instruction; and, the upset changes the operands of the instruction. The final consequence varies from corrupted outputs to processor hangs.

Data caches are built like instruction caches, i.e. with a tag array and a data array. SEEs in the tag array may invalidate data (cache miss) or validate out-of-date data (leading to corrupted output). As expected, SEEs in the data array may lead either to a corrupted output or to no observable effects if the data is out-of-date. SEEs in data cache memories are very unlikely to cause segmentation fault. This situation can occur if the code is manipulated as data by the processor.

Another important point is related to processor peripherals. They are configured through the use of registers. Consequently, as other memory elements, all of these registers are SEE sensitive, which can cause the peripherals to behave erratically. Moreover, as these peripherals control the timers, watchdogs, data input and data output, failures in these peripherals can be disruptive.

At software level, SEEs can affect the control flow and the data flow of a running application, as Fig. 3.13 shows.

Figure 3.13 – Possible effects of SEEs in processors.



Data flow error refers to errors caused by bit-flips in storage devices, such as registers and memories. They affect the program output, but not its execution. When a fault affects the data flow, the application runs normally, but the result in the end is incorrect. In this thesis, errors in the data flow are also called of Silent Data Corruption (SDC). Data flow errors are normally caused by:

- Wrong operation - The bit-flip modifies the instruction, and it performs another operation, which affects a memory element, such as a register or memory cell;
- Incorrect data - The bit-flip affects directly a memory element that contains the data used by an operation. Since the operation input is wrong, it is likely that its output will also be wrong. The error may propagate to the program output.

A control flow error occurs when the program flow is incorrectly followed, i.e., the error changes the program execution. When a fault affects the control flow, an erroneous execution flow occurs. A control flow error may cause a SEFI by crashing the program and hanging the processor. The possible outcomes caused by the fault are:

- Branch creation - A bit-flip converts a non-branch instruction into a branch, leading this illegal branch to change the program flow to a wrong address;
- Branch deletion - A branch instruction is converted into another instruction. Thus, a branch is not taken when it should be;

- Incorrect branch decision - It happens when a branch that should go in one direction, based on a comparison, goes in the other direction, i.e., a branch is not taken when it should be, or when it is taken when it should not be;
- Incorrect target address - The bit-flip modifies the register that contains the target address of a branch instruction (for example, the one used to return from a subroutine). It will change the program execution to an incorrect address;
- Bit-flip in the Program Counter (PC) register - It changes the next instruction to be executed. It has the same effect as branch creation.

It is hard to determine the root causes of processors errors in COTS devices, such as crashes and hangs. Since it takes a few clock cycles to observe the SEFI, it is not simple to catch the exact transition state to determine the reason why proper execution stopped. For example, it is possible that radiation affects portions of the hardware that caused the processor to fault or that SEUs affect critical code causing the software to fault.

3.5 Summary

In summary, based on the information presented in this chapter, one can see that state-of-the-art complex devices such as APSoCs have created many challenges to the radiation effects field. That is because radiation-induced failures in such devices and architectures may result in a complex chain of effects due to their heterogeneous nature. Consequently, additional methodologies and metrics become necessary for estimating the reliability of such devices, as the next chapter shows.

4 METHODS AND METRICS FOR EVALUATING APSOCS UNDER RADIATION

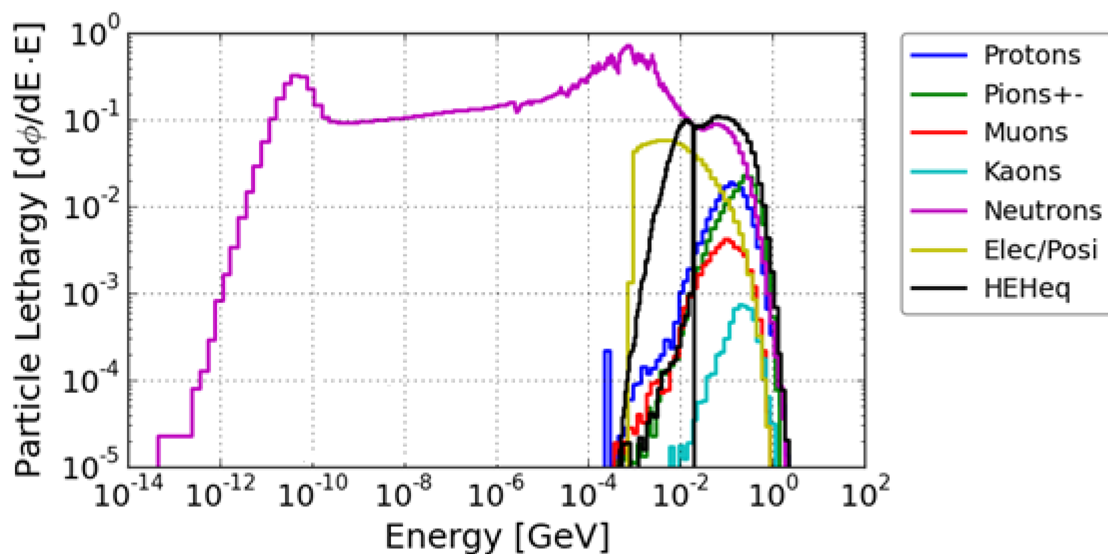
There are several methods to qualify integrated circuits for SEEs. Testing a device in its real application environment (space, high altitude, particle accelerator, etc.) is the most realistic way of evaluating its sensitivity with respect to SEEs. However, this solution has some practical disadvantages that are related to cost and time-to-market. Due to the low error probability, weeks or months are generally required for obtaining valid measures, and even years for gathering enough data to have reliable statistics. Another disadvantage is the unknown relationship between failures and the energy of the particles striking the samples.

4.1 Accelerated radiation tests

The most common way to qualify integrated circuits for SEEs is by means of accelerated radiation testing (JEDEC, 2006). In accelerated radiation tests, the devices are exposed to a specific radiation source whose intensity is much higher than the ambient levels of radiation that the device would normally experience. This induces the occurrence of SEUs, allowing useful data to be obtained in a fraction of the time, such as hours or days, instead of weeks, months, or even years, in case of real-time tests. Accelerated tests are performed at accelerator facilities, which accelerate specific particle species, such as neutrons, protons, and heavy ions. Neutron facilities are generally used for testing parts destined for terrestrial and avionic applications, where neutrons are the main result product of the interaction of cosmic rays with the Earth’s atmosphere. In such facilities, SEEs recorded will be due primarily to high energy neutrons (higher than 10 MeV and in the average of 14 MeV). Two of the main neutrons facilities are the Los Alamos National Science Center (LANSCE) in the United States and the Rutherford Appleton Laboratory (RAL/ISIS) in the United Kingdom. SEEs can also be induced by protons. Proton facilities are very useful because they easily reach very higher energies (usually between 50 MeV and 200 MeV) than neutron facilities. Furthermore, they are capable of generating protons with sufficient energy to simulate solar flares and Earth’s proton belt conditions. One of the main proton facilities is located at the Paul Scherrer Institut (PSI) in Switzerland. Heavy ions facilities are generally used for testing parts destined for space orbit, where primary cosmic rays can cause significant damage to electronic devices. The most important difference among heavy ion experiments and both neutron and proton

experiments is related to the dosimetry. The energy measurement unit of heavy ion experiments is the Linear Energy Transfer (LET), which describes the amount of energy lost per unit length of track. In other words, it describes the action of radiation upon matter, or how much energy an ionizing particle transfers to the material transversed per unit distance. Thus, the LET depends on the nature of the radiation as well as on the material traversed. Brazil has a very useful heavy ion facility, which is located at the Universidade de São Paulo (USP), the São Paulo 8UD Pelletron Accelerator. There is also the case in which electronic components are exposed to a very high flux of different particle species. This is the reality of the CERN's accelerators chain, where electronic components can be exposed to high-energy hadrons (protons, neutrons, pions), heavy ions, and other particles, at the same time. Aiming to simulate such complex environment, in 2015 CERN started operating a new and unique mixed-field radiation test facility, the CERN High Energy Accelerator Mixed-field (CHARM), located at CERN, Switzerland. At CHARM, it is possible to have particles with energies near 10 GeV, as shown in the graph of Fig. 4.1. Particle lethargy is a dimensionless logarithm of the ratio of the energy of source particles to the energy of particles after a collision (GLASSTONE, EDLUND, 1952). Thus, the graph in Fig. 4.1 shows an exponential decay of energy per unit collision showing that the greatest delta E's of energy result from the early collisions.

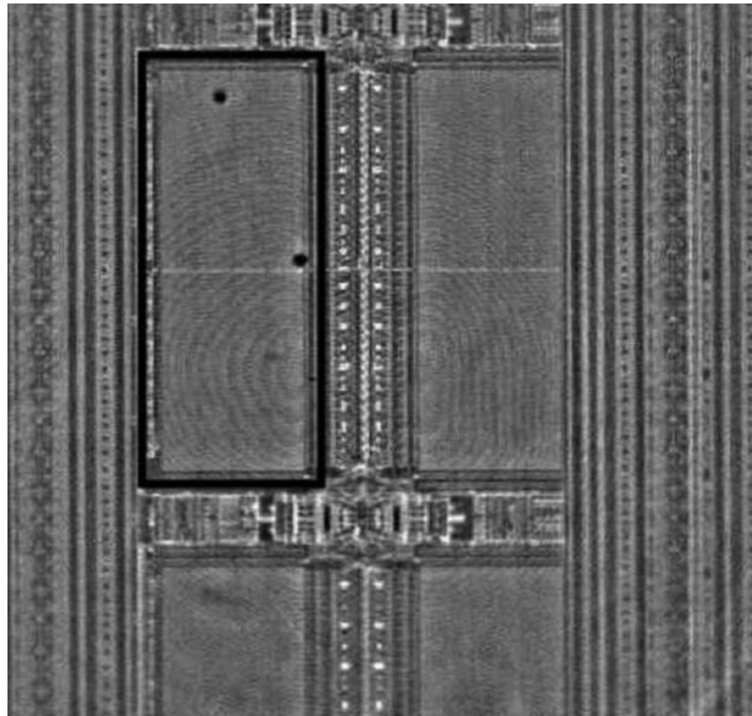
Figure 4.1 – CERN's CHARM particle spectra. Adapted from (ALÍA, 2016).



SEE characterization using particle beams (heavy ions, protons, or neutrons) is a global approach, since the entire device is irradiated. Such test provides a number of events

for a particular fluency, without any information about the detected faults location and the time they happened. In this context, laser beams can be used as an efficient complementary tool for accelerated tests in order to evaluate the sensitivity of complex electronic components exposed to radiation and to distinguish different effects. Laser tests provide a high level of accessibility to locate the circuit elements where faults are injected. The small laser spot and precise beam localization characteristics allow sensitive device nodes to be pinpointed with submicron accuracy. Therefore, taking into account the complexity of modern devices, laser testing is especially useful since it can provoke charges with spatial localization and temporal precision that is mandatory for analyzing faults that can be easily masked in particle accelerator beam tests. Laser has the disadvantage of having its beam reflected by metallization layers, thus complex circuits must be irradiated from the backside. Fig. 4.2 exemplifies the accuracy of laser tests by showing one of the scanned areas during a test campaign performed in (KASTENSMIDT et al., 2014). In this figure, the rectangular block is a BRAM of a Xilinx Virtex-5 FPGA. The National Research Nuclear University MEPhI in Russia has several dedicated laser facilities for testing electronics components.

Figure 4.2 – BRAM of a Xilinx Virtex-5 FPGA scanned during a laser test campaign (KASTENSMIDT et al., 2014).



4.2 Fault injection by emulation

Fault injection can also be performed in the laboratory by means of emulation. It represents the less costly fault injection alternative among all the mentioned ones, besides it being very flexible. Moreover, fault emulation is an attractive technique to predict the susceptibility of a system under SEUs during the early design stages and before submit the target device to an accelerated test, for example.

Basically, the emulation of SEUs and MBUs consists in flipping bits of memory elements of FPGAs and processors through the use of an embedded circuit, a program, or a computer. SEUs can be emulated in random locations, sequentially (every configuration bit or configuration control register is flipped in a sequential order), or user-defined. Then, the output of the Design Under Test (DUT), the system, is constantly monitored for analyzing the effect of the injected fault into it. This scheme provides a superior control over the fault injection when compared to accelerated radiation tests, since the time, the location of flipped bit, and the direct effect of the fault are known. A fault injection campaign can last from a few hours to several days depending on the amount of bits that will be flipped and other factors, such as the connection between the fault injector and its monitor, which can be a computer connected to the fault injector through a serial interface, for example. Although this scheme is easily applicable to both FPGAs and processors, the injection mechanism is different for each one.

There are several fault injection platforms to inject SEUs in the PL part of APSoCs (SRAM-based FPGAs) available in the literature as described in (ALEXANDRESCU, STERPONE, LOPEZ-ONGIL, 2014).

In a Xilinx SRAM-based FPGA, a fault injection can be performed through the Internal Configuration Access Port (ICAP) (XILINX, 2015a), which enables the user to access the configuration memory and to reconfigure it frame by frame internally. The ICAP can be controlled by several ways, such as through the Soft Error Mitigation (SEM) Core from Xilinx, which is an IP core that performs SEU detection, correction, classification, and emulation in the configuration memory (XILINX, 2015c); an embedded soft-core processor, such as a Microblaze or a Picoblaze; or a user-defined control circuit (TARRILLO et al., 2015). The ICAP interface is capable of achieving a maximum data throughput of 400 MB/s is configured to operate at 100 MHz. One of the first ICAP-based fault injection platform described in the literature is the one presented in (STERPONE, VIOLANTE, REZGUI, 2006).

In addition to the ICAP interface, Xilinx APSoCs such as Zynq-7000 have a new feature embedded into the PS part called Processor Configuration Access Port (PCAP). It provides to the PS part the ability to access the configuration memory of the PL part through a high bandwidth DMA channel capable of achieving up to 400 MB/s of data throughput (XILINX, 2016f). There were not found PCAP-based fault injector platforms in the literature up to the closure of this thesis.

Faults can also be injected externally to the device by using the Xilinx's SelectMAP interface, which is a faster option compared to the ICAP interface, since it is a programmable parallel interface capable of achieving up to 3.2 Gbps of data throughput (XILINX, 2012c). However, Zynq-7000 devices do not have an external configuration interface such as the SelectMAP, with the exception of the JTAG port. An example of very powerful and flexible fault injection platform that uses the SelectMAP interface for injecting faults is the FT-UNSHADES2, presented by Mogollon et al. in (MOGOLLON et al., 2011).

In the PS, fault injection can be performed at different abstraction levels such as RTL or software. However, since the RTL descriptions of COTS devices are not publicly available, faults can only be injected in the user-accessible resources, like registers and embedded memories. This approach is commonly called Software Implemented Fault Injection (SWIFI) and it can be performed during compilation time or execution time (HSUEH, TSAI, IYER, 1997). At execution time, typical approaches use timers such as the FERRARI tool (KANAWATI, KANAWATI, ABRAHAM, 1995) or interruption routines such as the XCEPTION tool (CARREIRA, MADEIRA, SILVA, 1998).

4.2.1 Hardware fault injection platform used

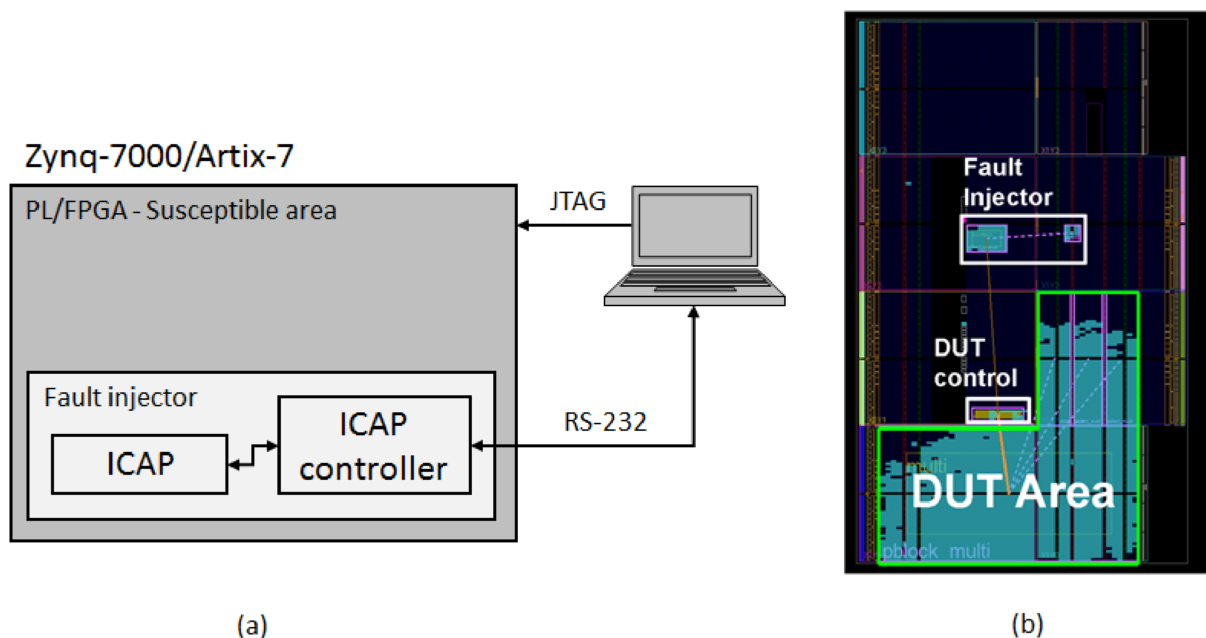
Part of this thesis uses the fault injection platform first presented in (TARRILLO et al., 2015). However, since the original Fault Injector (FI) core was built to handle frames from Xilinx Virtex-5 devices, modifications were made for enabling the core to handle frames from Xilinx Artix-7 (TONFAT et al., 2016) and Zynq-7000, the two main devices used in this thesis.

The fault injection platform is mainly composed of an ICAP controller circuit and a monitor computer, as Fig. 4.3 illustrates. The ICAP controller manages all the commands to read and write frames from the configuration memory using the ICAP interface. Therefore, with the information about the organization of the configuration memory (described in section 2.1.1.2) and the commands to manipulate the frames, it is possible to flip any bit of the

configuration memory, emulating the effect of an SEU. It is worth remembering that the smallest segment of the configuration memory is a frame, thus the ICAP controller always reads an entire frame of the configuration memory, store it in a memory buffer, flips its bits one at a time, and write it back to the configuration memory at each fault injection.

In the fault injection area (DUT area), faults are only injected in the configuration bits related to CLBs (LUTs, user FFs, and interconnections) and clock distribution interconnections. Fault are not injected in the BRAM configuration bits, so the inputs and outputs of the DUTs are not affected. For the versions which include DSP resources (DSP48E), the correspondent DSP configuration bits are added to the injection area. As can be seen in Fig. 4.3, the fault injector is placed in a different area of the FPGA to avoid fault injections that can disrupt its functionality. It is also shown a DUT control block that is also outside the injection area. This block analyzes the correctness of the DUT output for checking if the fault injected provoked an error or not. This result is sent to the fault injector and then to the monitor computer together with the fault position. Errors are classified as SDC or SEFI errors.

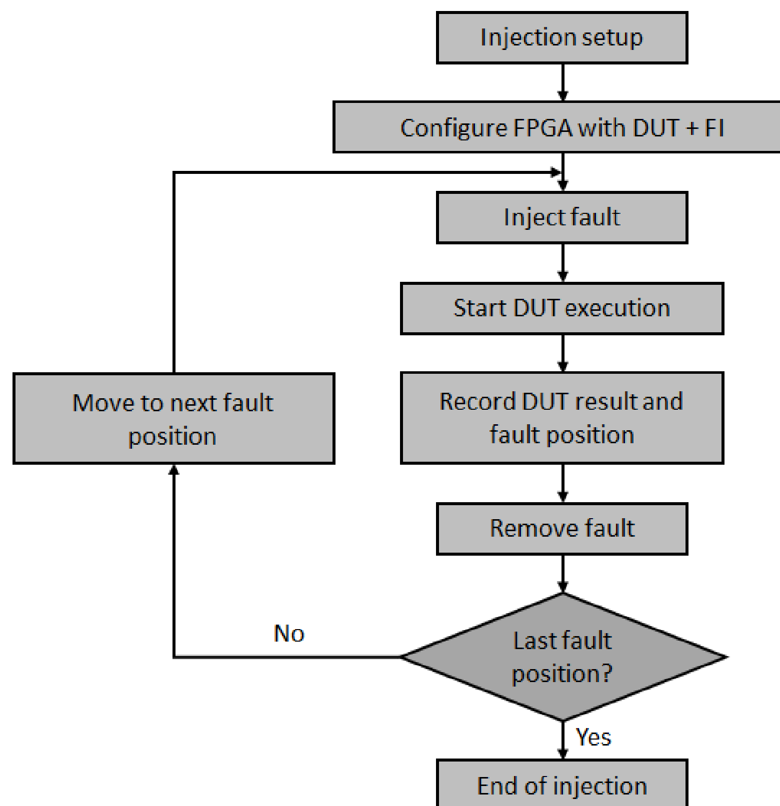
Figure 4.3 – Block diagram of the fault injection platform used in this thesis in (a) and an example of the FPGA floorplanning with the fault injector and the DUT placed in (b).



A fault injection campaign is defined by the flow diagram illustrated in Fig. 4.4. The first step is to setup the injection campaign, which consists of defining the injection area and the type of fault injection. Both configurations are setup in a Python script which runs in the

monitor computer and communicates with the FI. In this thesis, the main objective of using fault injection is to estimate the amount of critical bits of a design. Thereby, a campaign consists in an exhaustive and sequential bit-by-bit fault injection in all the configuration bits of the DUT, aimed to identify the ones that cause functional failures. The second step is to configure the FPGA with the DUT and the FI. After that, faults start being injected one at a time and always before the DUT execution. At this point, it is important to synchronize the FI with the DUT, so that after a fault is injected the DUT starts executing. Once the DUT finishes its execution, the output result is analyzed and the results (including the fault position) are saved. Finally, the fault is removed and the DUT is taken to its initial fault-free condition, prepared for the next fault injection. This process is repeated until all the configuration bits of the DUT area are evaluated. Therefore, one can observe that the required time to complete a fault injection campaign depends on the DUT area and the time to inject and remove a fault. The time to inject a fault is constant and is in the order of a few microseconds for Zynq-7000 and Artix-7 devices. With regard to remove a fault, two approaches are used.

Figure 4.4 – Flow diagram illustrating the hardware fault injection procedure in the Zynq-7000’s PL and Artix-7.



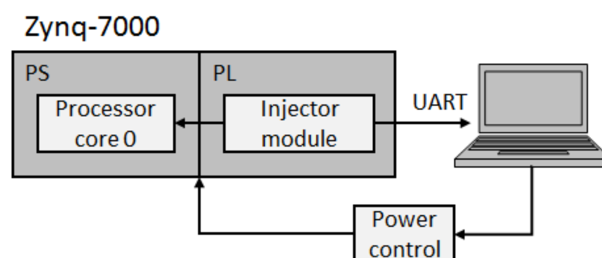
The first one consists on inserting again a fault in the same bit, restoring its original value. Then, the DUT runs again to verify if the fault was successfully removed. If the fault is not removed, a second approach is used. The monitor computer reconfigures the entire FPGA to ensure that all configuration bits are restored. The latter approach is the main reason why some of the fault injection campaigns can last more than one day. This approach can be improved by using dynamic partial reconfiguration on the DUT, for example.

4.2.2 Software fault injection platform used

Part of this thesis uses the fault injection platform first presented in (LINS et al., 2016) and improved in (DE OLIVEIRA, TAMBARA, KASTENSMIDT, 2017). The platform was developed to work on the ARM Cortex-A9 processor cores of the Zynq-7000's PS. It modifies values stored in the processor's internal registers or memories by injecting bit-flips through the use of interruptions and aiming to be the less intrusive as possible.

The platform setup is composed by the following modules, as illustrated in Fig. 4.5: the *power control*, which is an electrical device responsible for powering on and off the board; a software running on a host computer that manages the *power control* and stores the fault injection logs received through an UART interface; and, the *injector module*, which is hardware IP that performs the fault injection procedure.

Figure 4.5 – Platform setup of the software fault injector.

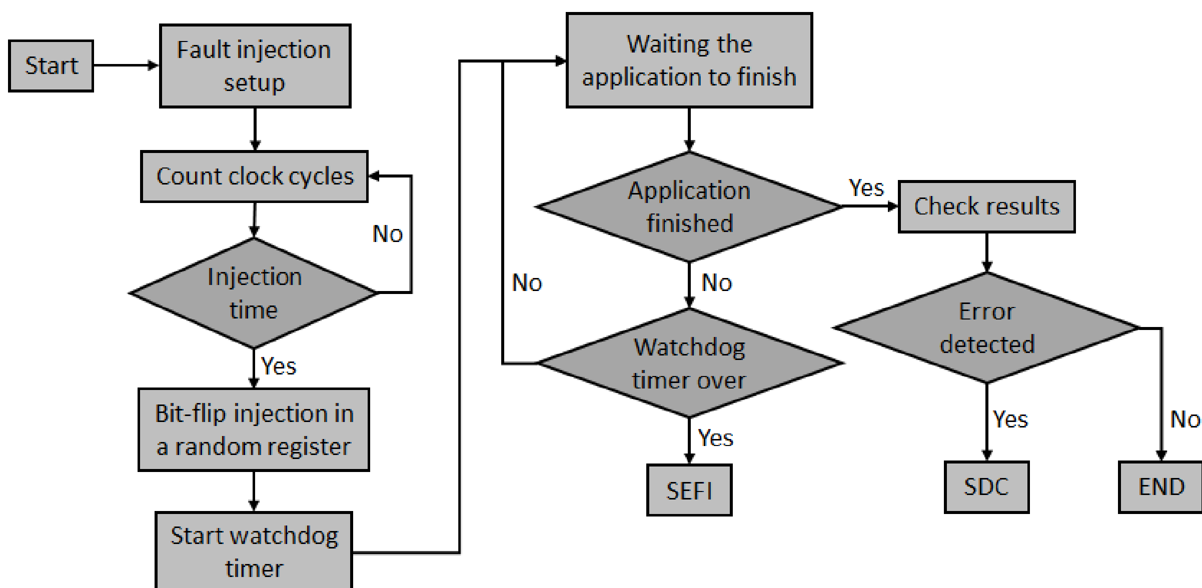


The fault injector is capable of injecting bit-flips in the following ARM registers: general-purpose (R0 to R12), SP (stack pointer), LR (link register), and PC (program counter). The fault injection procedure is illustrated by flow diagram in Fig. 4.6. In the first step, the *injector module* is configured with the injection data, which contains the injection time and the fault target location (the register in which the fault will be injected, besides the specific bit to be flipped). Due to the easiness of generating random numbers in the processor

compared to in hardware (PL), the injection configuration is generated by the processor before the application start and then it is read by the *injector module*. It is worth noting that the injection time is defined based on the execution time of the application, which means that a fault could be inserted at any moment during the execution of the application, as in real scenarios. Once the *injector module* has been configured, it starts counting clock cycles until it reaches the injection time. Then, the *injector module* launches an interruption to the target processor core. In the *injection interrupt routine*, the target register is read, a XOR mask with the appropriated bit to flip is applied to its value and, then, the register is overwritten.

After the fault injection, the *injector module* starts a watchdog timer with twice the value of the application execution time and it remains waiting for the end of the application. If the application does not end before the watchdog timer is over, it is considered the occurrence of an SEFI. In case of the application finished on time, the *injector module* compares the results generated by the processor with the gold ones. If there is any mismatch, it is indicated that an SDC occurred. Otherwise, the application ends normally, meaning that the bit-flip was effectless.

Figure 4.6 – Flow diagram illustrating the software fault injection procedure in the Zynq-7000's PS.



4.3 Test methods and metrics

Based on (IROM, 2008), (QUINN, 2014a), and (QUINN et al., 2015b), it is possible to notice that the methodology for testing FPGAs (Zynq-7000's PL part) and processors

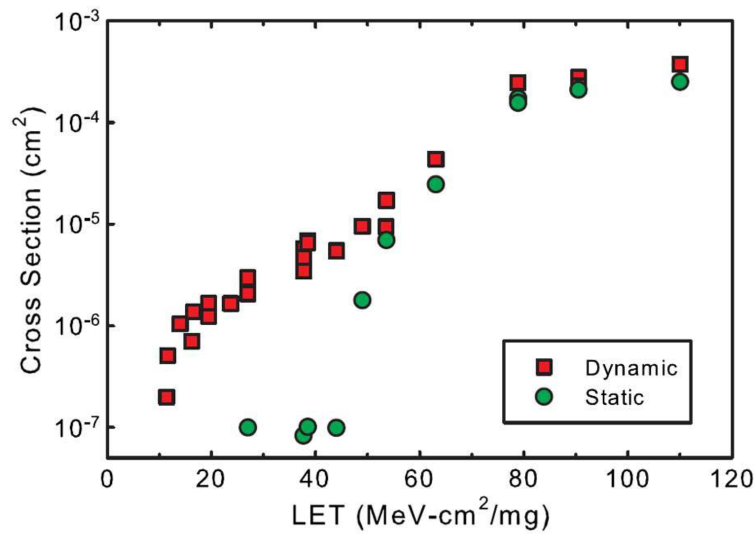
(Zynq-7000's PS part) are similar. SEE tests are event based, where the event is the occurrence of one of the SEEs types. It is necessary to measure the amount of radiation and count the number of events to calculate the cross sections and related metrics. Biasing, clock speed, temperature, and the angle of the radiation can be used to determine worst-case SEE influence. Tests might also be designed to highlight specific issues with functional/application conditions or mitigation methods.

This thesis is focused on the functionality conditions of the Device and/or Design Under Test (DUT) aiming to reveal different aspects of it. In this context, there are two main types of tests to be performed:

- Static - It consists in loading specific values into memory elements of the DUT and continually examine the status and content of them during the time that they are irradiated. This technique is used to measure the device static cross section. Through the static cross section, it is possible to quantify the sensitivity of the device technology to a specific radiation source.
- Dynamic - The DUT is initialized with a user-defined design and/or program. Then, the device is submitted to a set of user-defined stimuli and the outputs are constantly read and compared with the expected ones. This technique is used to measure the device dynamic cross section and related metrics. Through the dynamic cross section, it is possible to quantify the sensitivity of a running design and/or program in a specific device to a specific radiation source.

As a matter of comparison, Fig. 4.7 shows the SEU response of a SRAM memory during static and dynamic tests (SCHWANK, SHANEYFELT, DODD, 2013). As one can notice, while there is some similarity between the two cross sections in the saturation region, there is a significant difference around the threshold LET values. Threshold LET refers to the minimum LET to cause an effect on the component. There can be two reasons for these differences. One is that the dynamic operation of the component exercised a part of the component that was sensitive to radiation that is not possible to capture in a static test. The second explanation is that the dynamic operation could affect the noise margin that causes the operation to subtly change. Because of issues like that, a proper device characterization must always consider both static and dynamic characterization of the component under irradiation.

Figure 4.7 – SEU cross section for static and dynamic tests in a memory (SCHWANK, SHANEYFELT, DODD, 2013).



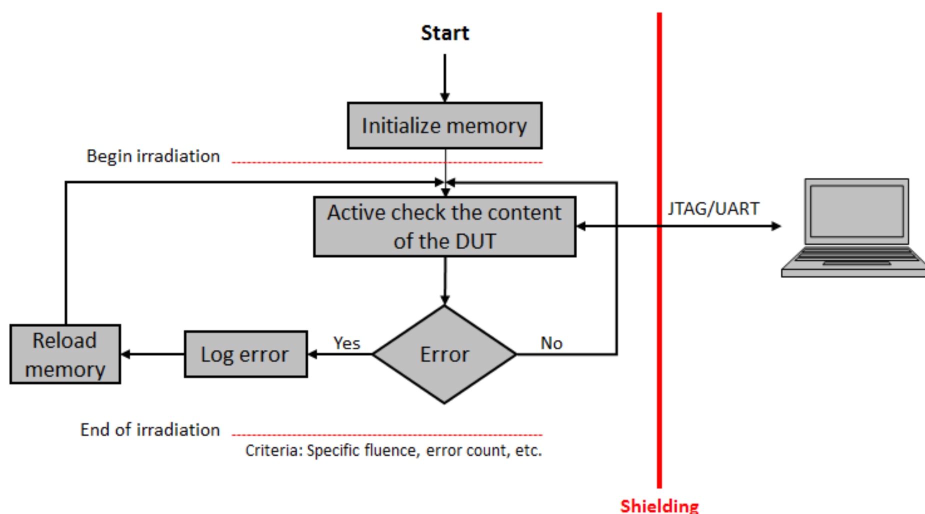
4.3.1 Static test method and metrics

Static tests have simple procedures, such as:

1. Load the memory(ies) of the device with a known pattern;
2. Irradiate the component;
3. Read the memory(ies).

A flow diagram illustrating this approach is shown in Fig. 4.8.

Figure 4.8 – Static test flow diagram. Adapted from (IROM, 2008).



With regard to FPGAs, a static test experiment consists of configuring the FPGA configuration memory with a known bitstream (the *golden* copy) containing the test-design. Then, the FPGA configuration memory is constantly read back (the *readback* copy) during the irradiation by using the Xilinx iMPACT or Xilinx Vivado tools through a JTAG interface. In the experiment control computer, the golden bitstream is compared against the readback bitstream. If differences are found (bit-flips), the FPGA is reconfigured with the golden bitstream and the differences are logged in the computer, along with the elapsed time during the irradiation when the error occurred. These procedures can also be applied for statically testing the FPGA embedded BRAMs.

The static test experiment procedures for processors are similar to the ones for FPGAs. In fact, static tests in processors are semi-static, because the tests typically consist of a small program that runs on the processor. Such program initially loads specific values (the *golden* values) into the processor registers and/or embedded memories (caches, OCM, etc.) and then continually examines their status and content during the time that the DUT is irradiated. As the embedded memories of processors like the ARM Cortex-A9 are relatively small, the time needed for checking the memories is of about a few hundreds of milliseconds, which can provide a nearly continuous evaluation of the memories if the program runs periodically. If an error is detected during a program loop, it is sent to the monitor computer though a serial interface (UART) to be logged, along with the elapsed time during the irradiation when the error occurred. Then, the error is corrected and the active test loop continues executing until a next error is detected or the test is stopped. It is worth noting that this test method assumes that the processor works properly nearly all of the time during the test. However, once the test is not completely static, errors may cause deviations from the expected behavior, which can result in the occurrence of SEFIs. SEFIs must not be taken into account in statistic of a static test. Another important point to highlight is that some embedded memories, such as cache memories, have some error detection and/or correction technique enabled, like parity-bit checking. Consequently, it is mandatory to consider if such feature is enabled or not during the tests.

As can be seen, a static test is far from the real device application, because it will provide the worst-case device sensitivity estimation to SEUs. In a real device application context, part of the device (memory in question) stays unmodified during the whole system operation, while other parts are used more dynamically. A way to determine a sensitivity closer to one of the final application consists in evaluating the average number of used memory bits and estimating the sensitivity as the product of this number by the per-bit

sensitivity (derived from the memory sensitivity issued from radiation testing divided by the number of bits) (VELAZCO, FOUCARD, PERONNARD, 2011).

During a static test, careful must be taken with the SEE rates and time-dependent effects, such as MBUs or SETs (IROM, 2008). Parameters such as the particle flux must be carefully adjusted to avoid the accumulation of too many SEUs in a short period of time because, for example, later it can be difficult to distinguish the difference between an MBU from one particle or a "constructed” MBU from multiple particles (QUINN, 2014a). Moreover, interactions with the device need to be minimized, otherwise it is possible that the test design could overwrite events before being recorded, what will neglect part of the fluence of the test. If events are overwritten, then the cross section based on the total fluence might be unrealistically low. Thus, it is necessary to adjust the fluence to determine how much of it corresponds to the observed events. To determine how much fluence should be used, the amount of time spent in each operation and the amount of time lost from each operation need to be determined. Examples of operations are the verification of a processor's memory and the read back of an FPGA configuration memory.

The static cross section (σ_{static}) is the fundamental metric to evaluate the sensitivity to radiation of a device. By definition, the static cross section is an intrinsic parameter of the device usually expressed in terms of area (cm^2/device or cm^2/bit), and it represents the minimum susceptible area of the device to a particle species (e.g. neutron, proton, heavy ion, etc.) and particle energy (LET) (JEDEC, 2006). In general, it is also a function of the operating conditions of the irradiated device (e.g., applied voltage, temperature, etc.). The static cross section of a device can be experimentally obtained by dividing the number of observed errors (N_{errors}) by the total particles fluence (i.e. the number of particles hitting the device per unit area, Eq. 4.1), as shown in Eq 4.2.

(Equation 4.1) $$\phi_{particles} = \left(particle\ flux\ \left[\frac{particles}{cm^2.\ s}\right]\right).(time\ [s])\quad [\frac{particles}{cm^2}]$$

(Equation 4.2) $$\sigma_{static-device} = \frac{N_{errors}}{\phi_{particles}}\quad [cm^2]$$

In addition, the static cross section per bit is expressed as shown in Eq. 4.3, where N_{bit} is the number of bits of the device.

(Equation 4.3) $$\sigma_{static-bit} = \frac{N_{errors}}{\phi_{particles} \cdot N_{bit}} \quad [\frac{cm^2}{bit}]$$

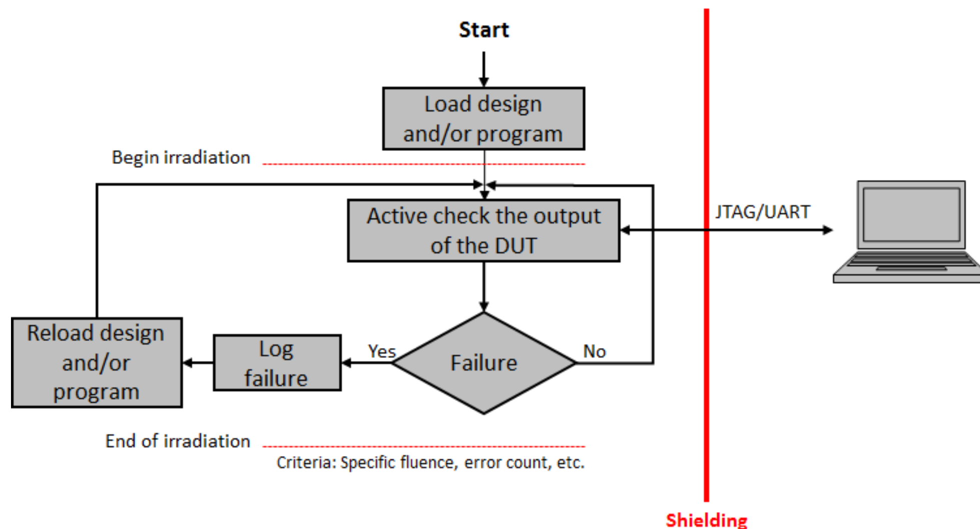
4.3.2 Dynamic test method and metrics

Dynamic tests aim to analyze a system under its functional operation. The procedures are similar to ones of static tests, but now a functional design must be programmed into the device, as follows:

1. Load the DUT with a design and/or program;
2. Irradiate the component;
3. Check the DUT output.

A flow diagram illustrating this approach is shown in Fig. 4.9.

Figure 4.9 – Dynamic test flow diagram. Adapted from (IROM, 2008).



As can be seen, a dynamic test is basically done in the same way as the static one for both FPGAs and processors. It has the same steps, but there are two main differences. The first difference is the fact that during the radiation exposure, the DUT (FPGA and/or processor) is running the target design instead of being idle. The second difference is that the design output is monitored, which means that the occurrence of failures (SDCs, SEFIs) is monitored, not errors as is done in static tests. In case of FPGAs and similar to a static test, the configuration memory can also be read at each failure for correlating the failures with the number of accumulated errors needed to provoke them.

Similar to a static test, careful must be taken with the SEE rates and time-dependent effects, such as MBUs or SETs. Parameters such as the particle flux must be carefully adjusted to avoid the occurrence of too many SEUs in a short period of time because, for example, in this scenario can even be impossible to run the application correctly without experience a failure. As previously explained, interactions with the device need to be minimized to avoid overwriting errors.

The dynamic cross section ($\sigma_{dynamic}$) is the basic metric to evaluate the sensitivity to radiation of a design. It is defined as the ratio between the number of errors observed at the output of a system (a design implemented into an FPGA or a processor running a program) divided by the fluence of hitting particles, as stated in Eq. 4.4. Thus, the dynamic cross section quantifies the sensitivity of a system to a specific particle specie.

(Equation 4.4) $$\sigma_{dynamic} = \frac{N_{errors}}{\Phi_{particles}} \quad [cm^2]$$

The rate at which soft errors occurs is called of Soft Error Rate (SER). The SER of a design is expressed in Failure in Time (FIT), which is the number of errors in one billion (10^9) device-hours operation. It is worth noticing that the SER is proportional to both dynamic cross section and particle flux, as shown in Eq. 4.5. Experimentally, the SER can be also calculated dividing the number of observed errors by the time interval analyzed, as presented in Eq. 4.6.

(Equation 4.5) $$SER = \sigma_{dynamic} \,.\, (particle\ flux) \quad [FIT]$$

(Equation 4.6) $$SER = \frac{N_{errors}}{time} \quad [FIT]$$

In the fault injection by emulation context and mainly at higher abstraction levels, such as at software level, the cross section becomes impossible of being estimated, since it is a metric expressed in terms of area. A possible solution for comparing results from radiation experiments and fault injections by emulation is to estimate the Architectural Vulnerability Factor (AVF) (MUKHERJEE et al., 2003) of the system. The AVF represents the probability that a visible error will occur at the output of a system given a bit-flip in a hardware structure such as a register or memory cell. In this thesis, the AVF is represented as the number of

errors detected (N_{errors}) divided by the number of faults injected (N_{faults}) into the design, as shown in Eq. 4.7.

(Equation 4.7) $$AVF = \frac{N_{errors}}{N_{faults}}$$

As can be seen, these metrics take into account only the sensitivity of a resource, which is the most common approach found in the literature. However, at system level, directly comparing the resources sensitivity of two systems is valid only when the other parameters of the systems do not vary between them, such as the execution time. Otherwise, such comparison becomes inaccurate. In this context, to compare the reliability of systems implemented in an APSoC and with different approaches, it is essential to take into account at least the cross section (σ), execution time (t), and workload of the system (w). Rech et al. (2014) introduced the Mean Workload Between Failures (MWBF) metric for Graphics Processing Units (GPUs). Then, in (TAMBARA et al., 2016) authors successfully adopted it for APSoCs. First, it is considered the Mean Time Between Failures (MTBF) of a system, defined as the average time between two radiation-induced failures on a system continuously executing a given task. By definition, the MTBF is evaluated with Eq 4.8, where *flux* is the particle fluence per unit time.

(Equation 4.8) $$MTBF = \frac{1}{\sigma_{dynamic} \cdot flux} \quad [h]$$

A higher MTBF simply attests that the system could work for a longer period of time before experiencing a radiation-induced failure. Nevertheless, no information on the workload computed during that period of time is given.

To evaluate how many executions has been correctly computed by a system during the MTBF window, a metric called Mean Execution Between Failures (MEBF) is defined. The MEBF is the number of correct executions of an application that are completed between two radiation-induced failures. The MEBF can be evaluated by the division between the MTBF and the execution time, as stated in Eq. 4.9.

(Equation 4.9) $$MEBF = \frac{MTBF}{t} \quad [executions]$$

Finally, each system is characterized by a workload (w), i.e. the amount of data that needs to be processed in one execution. The MEBF can be further generalized to take the workload computed correctly into account, leading to the concept of Mean Workload Between Failures (MWBF) as defined in Eq. 4.10.

(Equation 4.10) $$MWBF = MEBF \, . w = \frac{w}{\sigma_{dynamic} \cdot flux \, . \, t} \quad [data]$$

A higher MWBF actually means that a higher workload was computed correctly before experiencing a failure and that the operational reliability of a design is higher. Therefore, the MWBF metric provides a more consistent comparison in terms of reliability across system architectures that may be very different, since it considers that the reliability of a system is inversely proportional not only to the device sensitivity (σ), but also to the total exposure time (t).

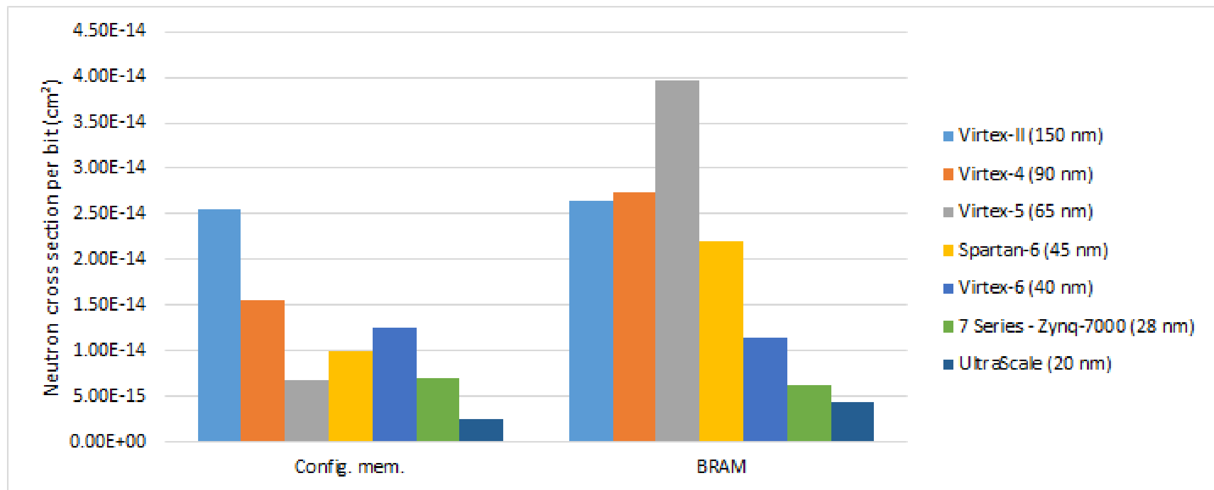
4.4 Related works about APSoCs under radiation

As shown in Section 2.4, several works have analyzed APSoCs, but initially most of them had only taken into account performance measurements. Then, in the last years, the first works concerning radiation effects in APSoCs have started to be published, including the ones derived from this thesis, which are the first ones that analyzed topics such as the trade-offs between performance and reliability in APSoCs.

Technology scaling is one of the main factors that increases the sensitivity of electronic devices to radiation-induced errors. However, in the case of Zynq-7000’s PL, and Xilinx SRAM-based FPGAs in general, the main reason for the error rate increase seems to be the increasing device density and not the sensitivity of its SRAM memory cells (WIRTHLIN et al., 2014). In fact, Xilinx has achieved to reduce the sensitivity of the SRAM cells in their new generations of FPGAs. Fig. 4.10 depicts the neutron cross section per bit of configuration memory bits and BRAM bits for different Xilinx devices (XILINX, 2016c). Xilinx also accomplished to reduce the sensitivity of BRAM bits to the same level of configuration bits. As it is possible to notice, in the case of Virtex-5 devices, BRAM SRAM cells are almost ten times more sensitive than configuration SRAM cells. For the 7-Series family, the sensitivity of both memories is practically the same. As mentioned in (XILINX, 2015b) and (CURD, CRABILL, 2015), Xilinx uses circuit design and layout techniques to improve the tolerance

of SRAM cells to soft errors, such as bit interleaving memory to avoid MBUs and MCUs (WIRTHLIN et al., 2014).

Figure 4.10 – Neutron cross section per bit for different Xilinx devices (XILINX, 2016c).



In (ALLEN, IROM, AMRBAR, 2015) and (AMRBAR et al., 2015), authors performed static heavy ion experiments in the PL part of the Zynq-7000. During the heavy ion experiments, Amrbar et al. (2015) monitored the currents of the different power management buses of the Zynq-7000. Authors observed high current events in the PL's VCC_{aux} line. The current kept increasing in the form of steps, as shown in Fig. 4.11. Authors stated that power cycling of the DUT was needed to recover from these events and that such events were increased at elevated temperature. They concluded that these high current events were due to SEL.

Allen, Irom, and Amrbar (2015), and Amrbar et al. (2015) also estimated the heavy ion static cross section of the Zynq-7000's PL. The obtained results are shown in Fig. 4.12. The graph shows the cross sections for the configuration memory bits (CFG) and BRAMs filled with one (1) and zero (0) values. Their results match with the ones obtained in this thesis, as will be shown later.

No related works were found about the proton static cross section of the Zynq-7000's PL, except the ones that will be presented later in this thesis.

With regard to processors, although Xilinx does not provide official cross section measurements for the Zynq-7000's PS, it states in (XILINX, 2015b) that from 65 nm technology and beyond, processors exhibit significant soft error rates. At 28 nm, the upset rate of a processor is dominated by SETs that propagate through the logic. Furthermore, Xilinx

also mentions that the upset rate has also steadily increased as the voltage drops and dimensions shrink, as already mentioned in this thesis.

Figure 4.11 – SEL occurrence in the PL part of Zynq-7000 during heavy ion experiments (AMRBAR et al., 2015).

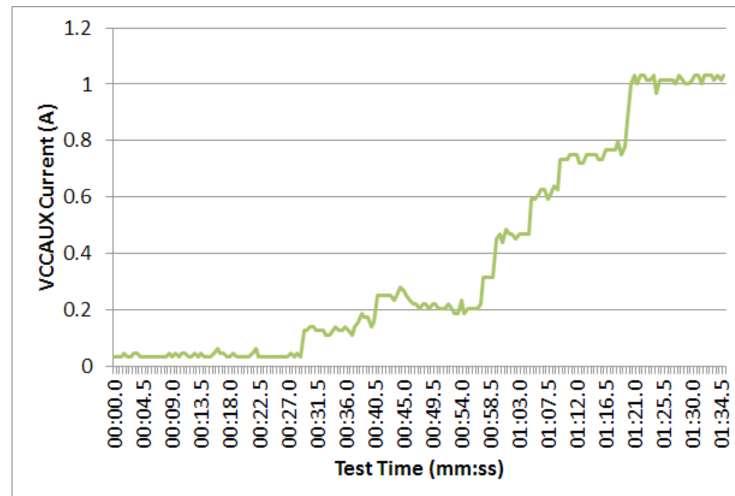
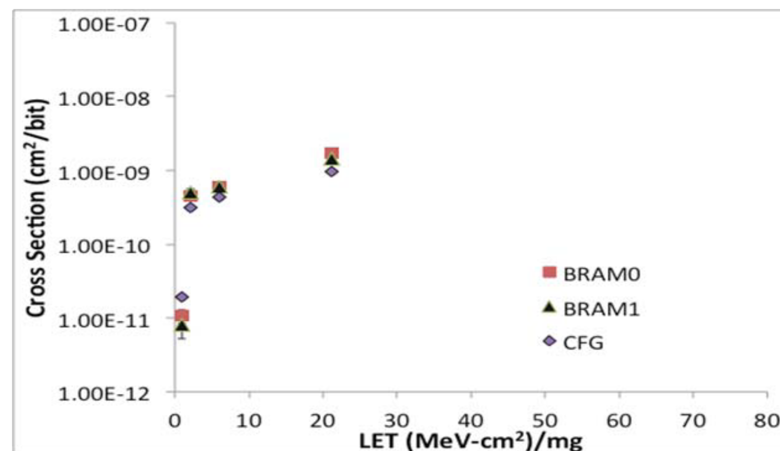


Figure 4.12 – Heavy ion SEU static cross section for the PL part of the Zynq-7000 configuration memory and BRAMs (ALLEN, IROM, AMRBAR, 2015; AMRBAR et al., 2015).

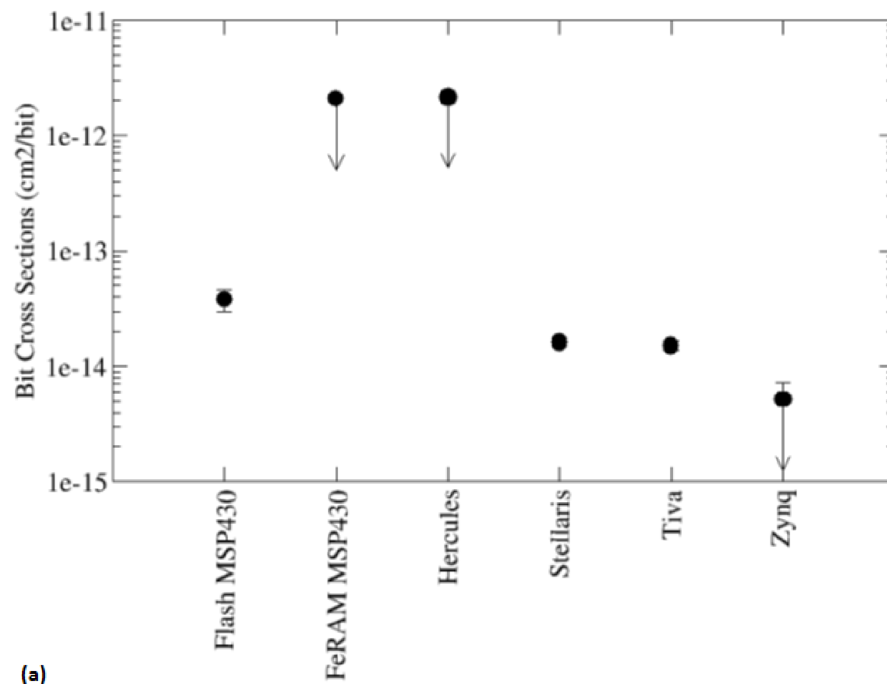


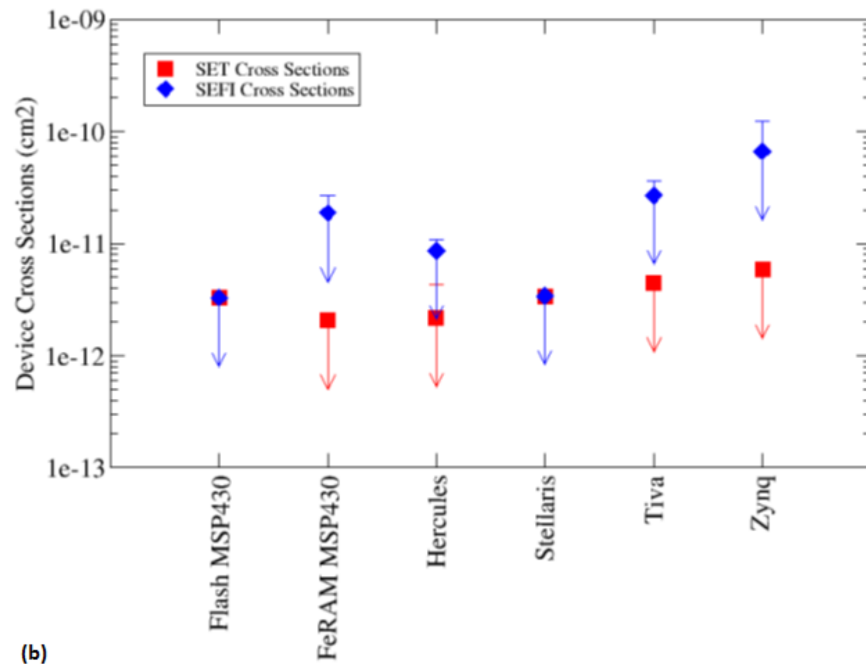
One of the first works that evaluated embedded processors in commercial APSoCs devices was (QUINN et al., 2014b). In this work, which was focused on ARM-based processors, one of the analyzed parts was the PS of the Zynq-7000. Authors tested it under neutrons by running basic benchmark applications. Despite the preliminary results, authors observed that its obtained cross section is similar to many other SRAM technologies, as can be seen in Fig. 4.13. Moreover, they also observed that the Zynq-7000's PS part is very sensitive to functional interrupts caused by radiation-induced errors.

Amrbar et al. (2015) performed proton experiments in the Zynq-7000's PS. The measurements were done for "0" to "1" and "1" to "0" transitions for both OCM and L1 Cache. They obtained an SEU cross section for the OCM of about 7.0×10^{-15} cm^2/bit for both logic directions. Authors did not observe SEUs in the L1 Caches, most probably due some setup error.

In (QUINN et al., 2015a), authors evaluated the effectiveness of the Triple Modular Redundancy (TMR) addition at software level for mitigating the radiation effects in microprocessors, such as the one embedded in the PS part of the Xilinx Zynq-7000. Results show that their approach is effective in masking the effects of SEUs and SETs in microprocessors systems. They achieved a decrease in the corrupted computations by one order of magnitude. However, authors did not take into account the drawbacks of adding a TMR scheme to the programs, such as the code size and execution time increase.

Figure 4.13 – (a) SEU cross section and (b) SET and SEFI cross section in neutrons for different SoCs (QUINN et al., 2014b).



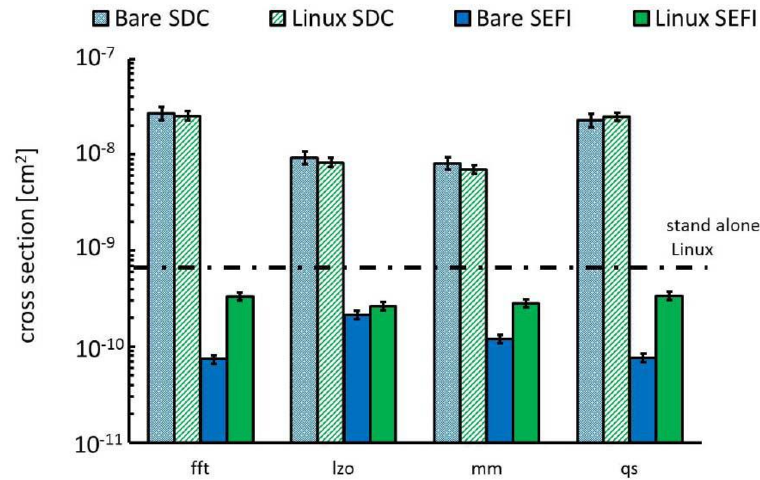


In (CHIELLE et al., 2015), authors propose a new control flow software-only technique that uses assertions to detect errors affecting the program control flow in ARM-based processors. Results show an improvement in cross section of one magnitude order in average with smaller penalty in code size (less than two times) and execution time (less than one and a half times) when compared to standard techniques such as TMR, whose the increase in code size is usually always around three times.

In (ZHAO et al., 2016) and (SANTINI et al., 2016), authors investigate the reliability of operating systems (Linux) running in the PS part of APSoCs (Microsemi SmartFusion2 and Xilinx Zynq-7000, respectively) and under radiation-induced errors. In general, results show that the presence of an operating system barely affects the data error rate, but it greatly increases the functional interrupt rate (up to 3.85 times), as Fig. 4.14 shows. This is not surprising as an application executes almost the same code when running bare to the metal or on the top of Linux. The operating system does not interfere with the amount of resources required for computation and, thus, does not modify the application SDC cross section. With regard to the SEFI rate increase, results indicate that the operating system is more likely to experience SEFIs than the applications running bare to the metal.

In (LINS et al., 2016), authors investigate the criticality of register file and compiler optimizations on ARM-based SoCs reliability. They chose to investigate the processor's register file criticality because of most of the works investigate only embedded memories, such as caches, and registers are among the most critical resources of embedded processors.

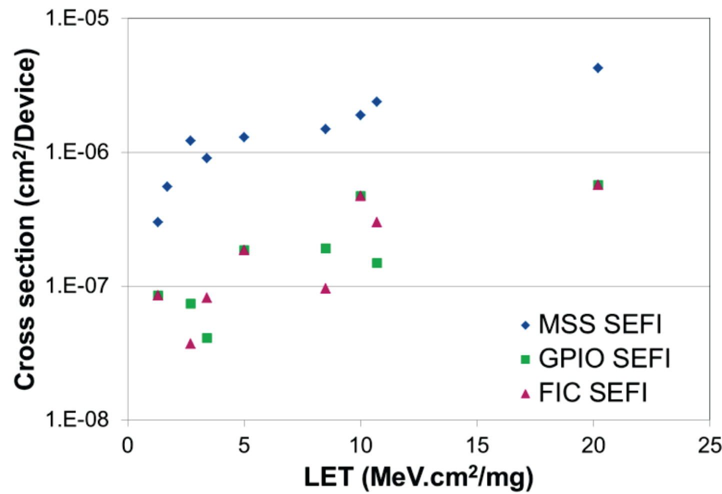
Figure 4.14 – Measured Zynq-7000 SDC and SEFI cross sections of applications running bare to the metal (Bare SDC and Bare SEFI) and on the top of Linux (Linux SDC and Linux SEFI) compared to the expected standalone Linux SEFI cross section (dashed line). (SANTINI et al., 2016).



Moreover, there are not detailed works that investigate the register file reliability under soft errors. Results show that the number of registers used by an application has a direct impact on its sensitivity to errors, as expected. Results show that, although compiler optimizations increase the application sensitivity to soft errors such as SDCs and SEFIs, and the dynamic cross section of the device (Zynq-7000), they are beneficial as they reduce the codes execution times. Consequently, while the probability for one impinging particle to generate an observable error increases with optimizations, the execution time is reduced, reducing the exposure time of the device. As it will be possible to notice in the next chapters, this work is strongly influenced by this thesis.

In (REZZAK et al., 2015) and (DSILVA et al., 2015), authors presented results about the characterization of the Microsemi SmartFusion2 APSoC under neutrons and heavy ions. They reported the static cross section of the different memories embedded into the device. Fig. 4.15 shows the heavy ion SEFI cross section for the Microcontroller Subsystem (MSS), General Purpose I/O (GPIO), and Fabric Interface Controller (FIC). From their results, it is important to notice the significant difference between the MSS (worst) and FIC results.

Figure 4.15 – Heavy ion SEFI cross sections for the MSS, GPIO, and FIC blocks of the SmartFusion2 (REZZAK et al., 2015).



4.5 Summary

This thesis has shown so far that there are several options of architectures and resources to be chosen when implementing a system in an APSoC. As consequence, the adoption of additional metrics beyond cross section becomes necessary for estimating the reliability of a device, system, or design. As previously shown in theory and in the next chapters in practice, today it is mandatory to take into account each design option available and all the parameters of the system involved, such as the amount of resources used, execution time, workload, etc. Furthermore, it is also important to investigate the general benefits that each option brings to the system by comparing them, for example.

With this objective, the next chapters first evaluate the susceptibility to SEEs of specific parts of Zynq-7000, the case-study device of this thesis, aiming to investigate if its reliability levels are as heterogeneous as its hardware. Then, based on the results presented in such chapters, the later ones analyze the reliability and performance trade-offs at system level in Zynq-7000. As final result, an analysis flow based on fault injection for estimating the reliability trend of hardware-only designs, software-only designs, and hardware and software co-designs, is proposed.

5 ANALYSING SINGLE EVENT EFFECTS ON THE PS PART OF ZYNQ-7000

This chapter presents static and dynamic radiation tests performed in the PS part of Zynq-7000 for measuring the sensitivity of its embedded memories under SEUs.

Static tests investigated the main memory storage groups of Zynq-7000 under heavy ions and protons such as OCM; L2 Cache, and the BRAMs of the PL part, which is embedded into the PL but is also an important memory group for the PS. It was not possible to statically test the L1 Cache under a particle beam due to its very dynamic behavior and the absence of a L1 Cache controller. In addition, some of the experiments also considered variations in the nominal supply voltage and temperature according to the ranges specified in the datasheet of the device. Such analysis is necessary because due to the technology scaling, the amount of charge used to store information in the memory nodes is continuously decreasing. Thus, less charge from an energetic particle is needed to change the logical state of a node. Moreover, the analysis of the temperature influence on the cross section behavior of devices like Zynq-7000 is important because according to (BAGATIN et al., 2011), the cross section of SRAM memories is strongly affected by temperature. However, temperature also induces changes in other parameters, such as charge collection efficiency and electron-holes mobility. These parameters can also be responsible for changing the cross section of modern devices like Zynq-7000.

Since it was not possible to statically test the L1 Cache, dynamic tests consisting of different cache schemes (L1 and L2 caches) were performed aiming to evaluate the impacts of the cache scheme on the sensitiveness of the PS part of the Zynq-7000 under heavy ions. The justification for this analysis relies on the fact that cache memories are traditionally disabled in safety-critical applications since it is believed that the sensitive area they introduce compromises the system reliability and the performance, consequently.

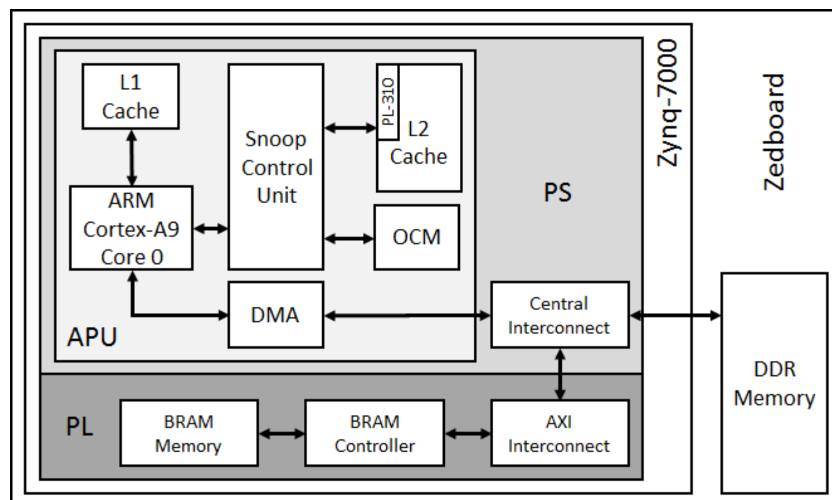
5.1 Static tests

5.1.1 Tests procedures

The static tests of the OCM, L2 Cache, and BRAMs, consisted of the same steps described in Section 4.3.1. Although the static tests procedures were practically the same for

all the tested memories, each one was evaluated separately to achieve more precise results. This means that during an irradiation, only the memory under test was being used by the processor, while the others were disabled (i.e., OCM or L2 Cache) or even not instantiated (i.e., BRAMs). Fig. 5.1 shows an integrated block diagram illustrating the architectures of the static tests. Concerning the size of the memories, three quarters of the OCM (192 KB), half of the L2 Cache (256 KB), and all the available BRAMs (4.9 Mb), were evaluated. It was left free space in OCM and L2 Cache for safety reasons, such as to not have any chance of having a segmentation fault or for the case in which an unknown instruction could start to be executed by the processor. All the obtained cross sections are per bit. It is also worth noting that the parity support for all the memories was disabled during the experiments and none of them were implementing any ECC technique. Finally, the test programs were running from the board's DDR memory and the known pattern adopted (reference data) for all three memories was "*AAAAAAAA'h*".

Figure 5.1 – Integrated block diagram of the static tests architectures.



Differently from OCM and BRAMs, in which the user can freely write and read data and as well as L1 Cache, L2 Cache is a very dynamic memory and additional procedures are required to get a deterministic response from it and test it statically. Aiming to achieve this with the L2 Cache of the PS part of Zynq-7000, it was necessary to manage the L2 Cache controller (PL-310), since it provides a feature to lock data or code into the cache. Locking of data or code in cache is an indication to cache replacement algorithm to prevent these entries from being evicted. Thus, the solution found to perform the static test of the L2 Cache was to preload data into the cache, lock them, and then constantly read the memory content searching

for errors. The procedures to lock data into the L2 Cache are executed in the beginning of the application. They are the following:

1. Write the exact amount of data (*AAAAAAAA'h*) to be preloaded into the L2 Cache in the DDR memory;
2. Invalidate L1 and L2 Data caches;
3. Disable both L1 Instruction as well as Data caches;
4. Preload the data from the DDR into the L2 Cache;
5. Lock all the 8 ways of L2 Cache;
6. Start analyzing the memory content as shown in Section 4.3.1.

In the computer that controlled the experiments, a script monitored the serial interface for incoming errors sent by the running program (described in Section 4.3.1) on the processor for monitoring the OCM and L2 Cache. Once the script received an error, the error was time-stamped and then logged for posterior analysis. Regarding the BRAMs, another script handled the Xilinx iMPACT tool to write and read the content of the BRAMs, which are included in the PL's bitstream. The script constantly compared the fault-free bitstream (the *golden*) against the last read. If differences were found, which means errors, the script time-stamped and logged the errors for then reconfigure the PL with the golden bitstream.

With regard to variations in the nominal supply voltage, they were performed by directly accessing the Zedboard power supply lines after the embedded input power regulator. With regard to the temperature variations, they were performed by heating the device with an air heater.

5.1.2 Tests setups

Heavy ion experiments were performed at two different facilities. For lower energies, a specially experimental setup at the Laboratório Aberto de Física Nuclear (LAFN) of the Universidade de São Paulo (USP), Brazil (AGUIAR et al., 2014) was mounted. Aiming to achieve very low particle fluxes in the range from 10^2 to 10^5 particles.cm^{-2}.s^{-1}, as recommended by the European Space Agency (ESA) for SEU tests (ESA, 2005), a standard Rutherford scattering setup using a gold foil was used. The experiment was performed in both vacuum and air. An approximate pressure of 10^{-6} Torr was used in the chamber for the tests in vacuum. A silicon barrier detector was mounted inside the vacuum chamber at an angle of 45º to monitor the beam intensity. In front of this detector, it was mounted a collimator with a diameter of 4 mm, defining a solid angle of about 0.085 msr. The ion beams were produced

and accelerated by the São Paulo 8UD Pelletron Accelerator. The SEU events were observed irradiating ^{12}C, ^{16}O, ^{19}F, ^{28}Si and ^{35}Cl beams, scattered by a 184 μg/cm² gold target, with energies that provide effective LETs in the border of the active layer ranging from 2.6 to 17 MeV/mg/cm², and penetration in Si ranging from 16.6 to 47.6 μm. To achieve the desired particle flow, the DUT was positioned at a scattering angle of 15º. Table 5.1 summarizes the experiment parameters and Fig. 5.2 shows the experiment setup mounted at the LAFN-USP. For higher energies, experiments were performed at the Russian Federal Space Agency (ROSCOSMOS) tests facilities, Russia. The heavy ion test facility is based on a Cyclotron U-400M. The characteristics of the heavy ion facility are shown in Table 5.2 and the characteristics of the ions used in the experiments are shown in Table 5.3.

Table 5.1 – Characteristics if the heavy ion beams used at LAFN-USP.

Beam type	Vacuum	Incident energy (MeV)	Effective LET (MeV/mg/cm²)	Penetration in Si (μm)
^{12}C	Yes	40.9	2.60	47.6
^{12}C	No	38.0	2.70	43.8
^{12}C	No	29.4	3.13	31.0
^{16}O	No	41.0	5.00	28.5
^{16}O	Yes	41.0	5.00	27.5
^{16}O	Yes	34.0	5.50	22.0
^{16}O	Yes	30.9	5.70	19.0
^{19}F	Yes	48.2	6.20	26.7
^{19}F	No	43.0	6.40	25.0
^{28}Si	Yes	68.4	12.55	20.9
^{28}Si	No	54.6	13.02	18.8
^{35}Cl	Yes	80.8	16.60	19.5
^{35}Cl	No	62.6	17.00	16.6

Figure 5.2 – Heavy ion experiment setup mounted outside and inside the vacuum chamber at the LAFN-USP.

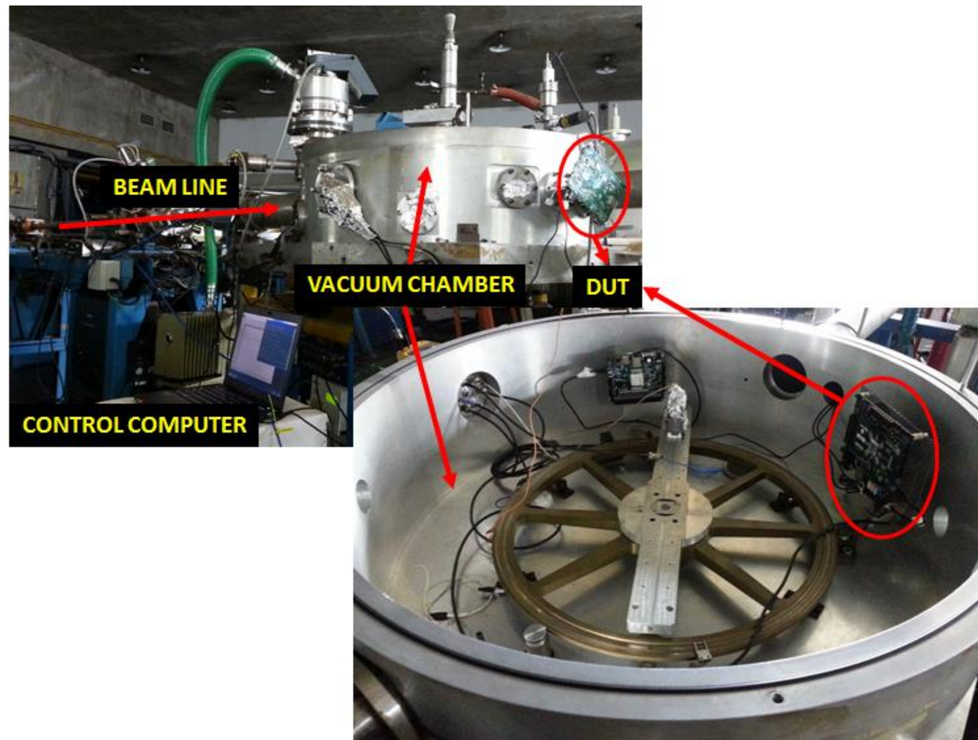


Table 5.2 – Characteristics of the ROSCOSMOS heavy ion facility.

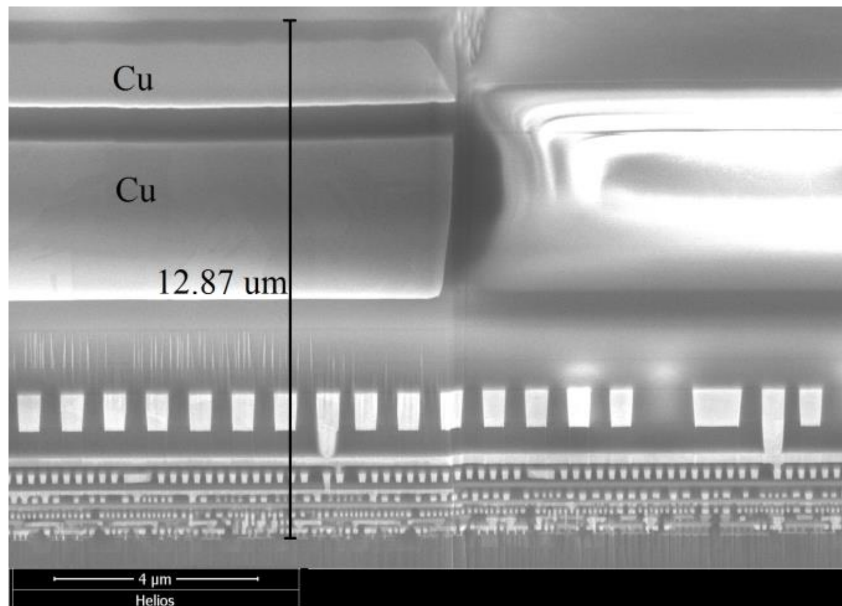
Source name	IS OI-A (400 M) based on U-400M
Accelerator type	Cyclotron
Ion species and energy ranges	O, Ne, Ar, Fe, Kr, Xe, Bi
Effective LET	4.5 - 99 MeV/mg/cm²
Range in Si	> 30 µm
Min - Max fluxes	10 – 105 particles.cm^{-2}.s^{-1}
User flux control	Yes
Spot size / Uniformity	60 x 60 mm /≤ 10% 60 x 90 mm /≤ 15% 60 x 180 mm /≤ 30%
Beam counting and monitoring system	5 proportional counters - active track detectors - passive
Test chamber	Vaccum
Device positioning system	Yes

Table 5.3 – Characteristics if the heavy ion beams used at ROSCOSMOS.

Beam type	Vaccum	Incident energy (MeV)	Effective LET (MeV/mg/cm²)	Penetration in Si (μm)
^{20}Ne	Yes	78	8.40	38
^{40}Ar	Yes	144	18.50	37
^{84}Kr	Yes	253	21.90	42
^{132}Xe	Yes	404	32.30	45

For the heavy ion experiments, the package of a Zynq-7000 device, part XC7Z020-CLG484, was thinned to allow that irradiated particles penetrate the active region of the silicon. Fig. 3.10 shows the surface of the chip without its package, where it is possible to distinguish between the PS part on the top of the left side and the PL. Fig. 5.3 shows the microscopic section of the chip performed for evaluating the energy loss of the heavy ions after passing the passive layers. The passive layers consist of eleven copper metallization layers separated by dielectric layers. The estimated total thickness of the passive layers is 12.87 µm. For estimating the energy loss of the heavy ions, it was assumed a total thickness of copper metallization layers of 7.87 µm and a total thickness of dielectric layers of 5.0 µm. The energy losses estimation and the effective LET values in the border of the active layer were obtained with the SRIM software (SRIM, 2013) together with chip structure data (XILINX, 2015d).

Figure 5.3 – Microscopic section of a Zynq-7000 device, part XC7Z020-CLG484.



Proton experiments were also performed. They were carried out on a compact synchrotron built by *JSC Proton* and located at the medical center of Protvino, Russia. The characteristics of the synchrotron proton used in the experiments are shown in Table 5.4.

Table 5.4 – Characteristics of the Russian proton facility.

Proton energy	60…330 MeV
Bunch duration	10…1000 ms
Proton in the bunch	10^7...10^9 pcs
Proton beam diameter	2…4 mm
Beam bending HWD	70 x 700 mm
Beam non-uniformity	5%

Table 5.5 summarizes the test schemes performed in the embedded memories of the PS part of Zynq-7000.

Table 5.5 – Heavy ion and proton test schemes performed in the embedded memories of the PS part of Zynq-7000.

Memory	Core voltage 1 (V)	Core voltage 2 (V)	Core voltage 3 (V)	Temperature (ºC)
	Heavy ion tests			
BRAM	0.95	*	1.05	52.5
OCM	0.95	*	1.05	52.5
	Proton tests			
L2 Cache	*	1.0	*	36.0

* Configuration unable to test due to beam limitations.

5.1.3 Tests results

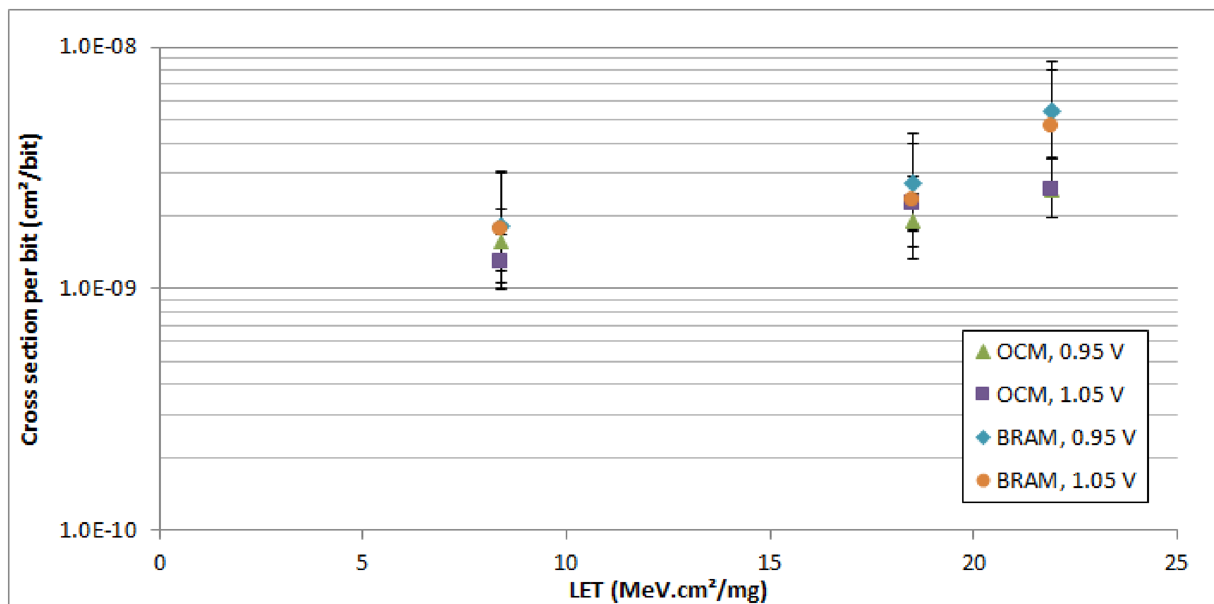
Fig. 5.4 shows the obtained results from heavy ion irradiations, which are shown in terms of cross section versus LET. Considering each memory type, the biggest cross section variation with supply voltage for each one was of 19% for OCM (1.05V/0.95V, LET = 8.4 MeV.cm²/mg) and 15% for BRAM (1.05V/0.95V, LET = 18.5 MeV.cm²/mg). However, if we consider the same supply voltage, like 1.05V, and the same LET, like 21.9 MeV.cm²/mg, the difference between the OCM cross section and BRAM cross section can reach 46%.

Regarding proton irradiations, it was obtained a cross section per bit for the L2 Cache of 1.0×10^{-16} cm²/bit for an energy of 250 MeV. Based on the neutron cross sections informed

by Xilinx in (XILINX, 2016c), such result is in accordance with the expected ones. However, additional data are needed for performing a more accurate analysis.

The obtained results are fundamental, once they can guide designers to choose the most reliable memory for implementing a shared memory between PS and PL parts, for example.

Figure 4.4 – Cross section results from the heavy ion irradiations in the embedded memories OCM and BRAM of the Zynq-7000.



5.2 Dynamic tests

5.2.1 Tests procedures

Dynamic tests were performed to investigate the impacts of the cache organization on the sensitiveness of the PS part of Zynq-7000. Considering a processor with L1 and L2 caches, it is reasonable to assume that there is a given configuration, such as using no caches, only L1, or both levels, that optimizes reliability (in terms of exposure time and sensitive area). As already mentioned, in general, cache memories are traditionally disabled in safety-critical applications since it is believed that the sensitive area they introduce compromises the system reliability. However, the choice of the more reliable configuration is not straightforward.

Tests were based on a bare-metal application running on one core of the ARM Cortex-A9 processor (here referred just as ARM), always at its maximum frequency (667 MHz). The application performs a sequence of multiple matrix multiplication operations in order to increase caches' utilization and exercise all cache levels. Matrix multiplication was chosen as the basic application because it is a common task in critical systems, such as in control and filter operations. The sizes were chosen empirically, resulting in a set of 140 matrices of 25x25 integers, which totalizes 350 KB of data. The choice of performing multiple multiplications of small matrices instead of a single matrix multiplication relies on the fact that it would be necessary very large matrices to fill L1 Data (L1D) and L2 caches and it would not exercise the L1 Instruction (L1I) Cache. Moreover, as it is considered a worst-case scenario, the parity of the L2 Cache was disabled. However, it was not possible to disable the parity of L1 Cache in these experiments.

The dynamic tests procedures consisted of the same steps described in Section 4.3.2. The detection of incorrect outputs was achieved by comparing the computed results *C* of the multiplication of matrices *A* and *B* with a golden copy *G*, which is the expected correct results calculated at compilation time. A monitor computer monitored the processor through a serial interface. At each execution set, if there were not differences in the results, the processor sent a "*PASSED*" message to the computer. Otherwise, if differences were found, the program sent an "*NOT PASSED*" message to the computer and then the processor was reset. In both cases the messages were time-stamped for future analysis.

Since the ARM core has three cache memories (L1I, L1D, L2), eight possible cache configurations were implemented by enabling or disabling each cache level. Table 5.6 lists the possible configuration (*Enabled* – E or *Disabled* – D) together with the respective program size and execution time of each one. When all the caches are disabled (D/D/D configuration), the processor uses the OCM to store instructions and data. If needed, the processor also can use the DDR memory of the board. From the execution time data, it is possible to observe a significant improvement of 92% when all cache levels are enabled (E/E/E configuration). When comparing configuration D/D/D with E/E/D and E/E/E, it is easy to observe by configuration E/E/D that L1 Cache plays a significant role in the performance improvement, providing an enhancement of 91% in performance.

Results are presented in terms of Silent Data Corruption (SDC) cross section, which is related to errors detected by the application without program interruption (data flow errors), and Single Event Interrupt (SEFI) cross section, which is related to program crashes (control flow errors). Errors in L1D cache are more prone to provoke SDC errors, because the L1D

cache memory stores mainly data of the program running in the ARM core. Consequently, errors in the program data may result in unexpected program outputs. Errors in L1I cache are more prone to provoke SEFIs errors, because the L1I cache memory stores the instructions of the program running in one ARM core. Consequently, errors in the program instructions may result in program crashes.

Table 5.6 – Application information running on Zynq-7000's ARM Cortex-A9 Core 0 with different cache schemes (D = Disabled, E = Enabled).

Configuration (L1I/L1D/L2)	**Program size (bytes)**	**Execution time (s)**
D/D/D	8036	2.4966
D/D/E	8036	0.9809
D/E/D	8036	1.9040
D/E/E	8036	0.5722
E/D/D	8036	2.2220
E/D/E	8036	0.9610
E/E/D	8084	0.2012
E/E/E	8080	0.1988

5.2.2 Tests setup

Experiments were also conducted through heavy ion tests performed at LAFN-USP, Brazil, where the ion beams are produced and accelerated by the São Paulo 8UD Pelletron Accelerator. The setup configuration of the experiments is quite similar to the one of the static tests. However, in this case, the experiments were performed in air.

The SEU events were observed using a ^{16}O beam, scattered by a 184 μg/cm² gold target, with an energy of 59 MeV (effective energy at the active region of 11.6 MeV), which provides an effective LET at the active region of 7.3 MeV/mg/cm² and penetration in Si of 28 μm. To achieve the desired particle flow, the DUT was positioned at a scattering angle of 15°, resulting in an average flux of 5.84×10^2 particles.cm^{-2}.s^{-1}. Such configuration was chosen based on several trials and it was the most suitable in terms of particle flux and number of errors for these experiments. Finally, the beam was focused on the PS part.

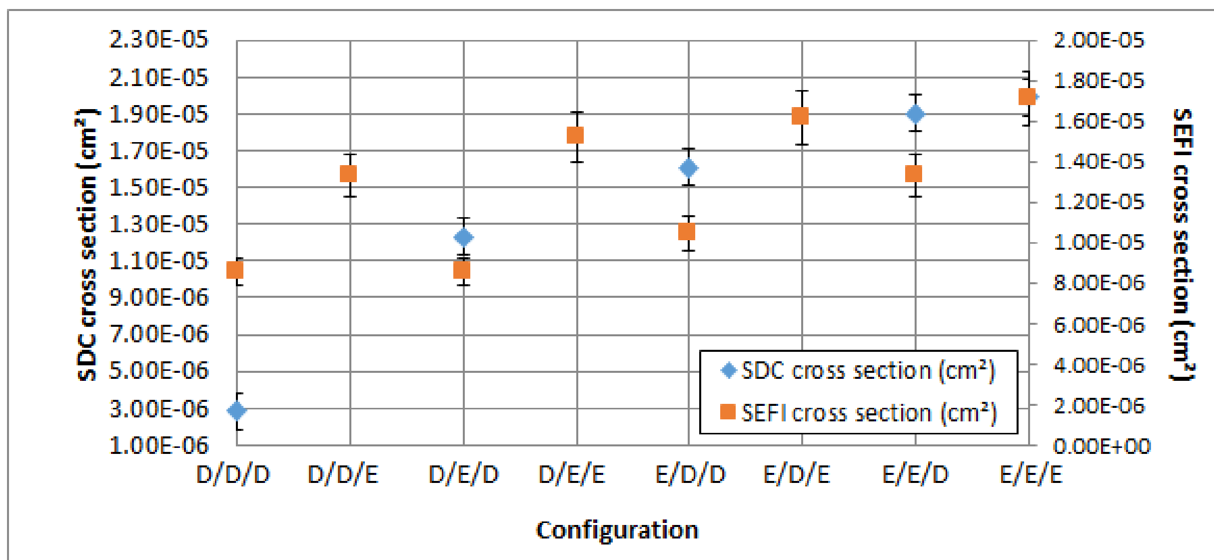
5.2.3 Tests results

Experimental results from the heavy ion irradiation campaigns are listed in Table 5.7 and shown in Fig. 5.5. Errors in which their sources are unknown are considered in the error bars. Data show that the SDC cross section increases with the addition of caches to the memory hierarchy, since radiation can affect cache array and circuitry. Experimental results also show that the addition of L2 Cache to the memory hierarchy is critical, making it even impossible to estimate the SDC cross section due to the prevalence of SEFI over SDC errors. In fact, the addition of any cache memory to the memory hierarchy affects significantly the cross section of the ARM processor.

Table 5.7 – Obtained SDC and SEFI cross sections from the heavy ion irradiations.

Configuration (L1I/L1D/L2)	SDC cross section (cm^2)	SEFI cross section (cm^2)
D/D/D	2.85×10^{-6}	8.56×10^{-6}
D/D/E	-	1.33×10^{-5}
D/E/D	1.23×10^{-5}	8.56×10^{-6}
D/E/E	-	1.52×10^{-5}
E/D/D	1.61×10^{-5}	1.05×10^{-5}
E/D/E	-	1.62×10^{-5}
E/E/D	1.90×10^{-5}	1.33×10^{-5}
E/E/E	1.99×10^{-5}	1.71×10^{-5}

Figure 5.5 – Obtained SDC and SEFI cross sections from the heavy ion irradiations.



The SEFI data in Table 5.7 and Fig. 5.5, which are related to control flow errors, show that the SEFI cross section also increases with the addition of caches to the memory

hierarchy, but mainly with the addition of L2 cache. However, it is possible noticing an increase in cross section when caches L1I and/or L2 are enabled. This is particularly interesting because it confirms that control flow errors are the main source of SEFIs in processors and they are more related to the program instructions than the program data. The adoption of Software Implemented Hardware Fault Tolerance (SIHFT) or Algorithm-based Fault Tolerance (ABFT) techniques are possible solutions to improve the SEFI results, for example.

Table 5.8 shows the MWBF values for all the tested caches configurations. In this case, the workload *w* of the application, i.e. the amount of data processed by the application at each execution, is $2.80x10^6$ bits. Table 5.7 shows that the most reliable configuration under heavy ions is the one with all caches disabled. Regardless the smaller cross section imposed by disabling all caches, its execution time is so high not to be compensated by the benefit in terms of performance. For the other configurations, despite the increase in the complexity and sensitive area, the smaller the execution time, the bigger the MWBF. In this case, although the small improvement in the MWBF values when comparing to the execution time values of the corresponding configurations, it is worth maintaining at least L1I and L1D caches enabled to not compromise the system performance. It is worth mentioning that all the data obtained are strongly dependent of the application and the cache policy.

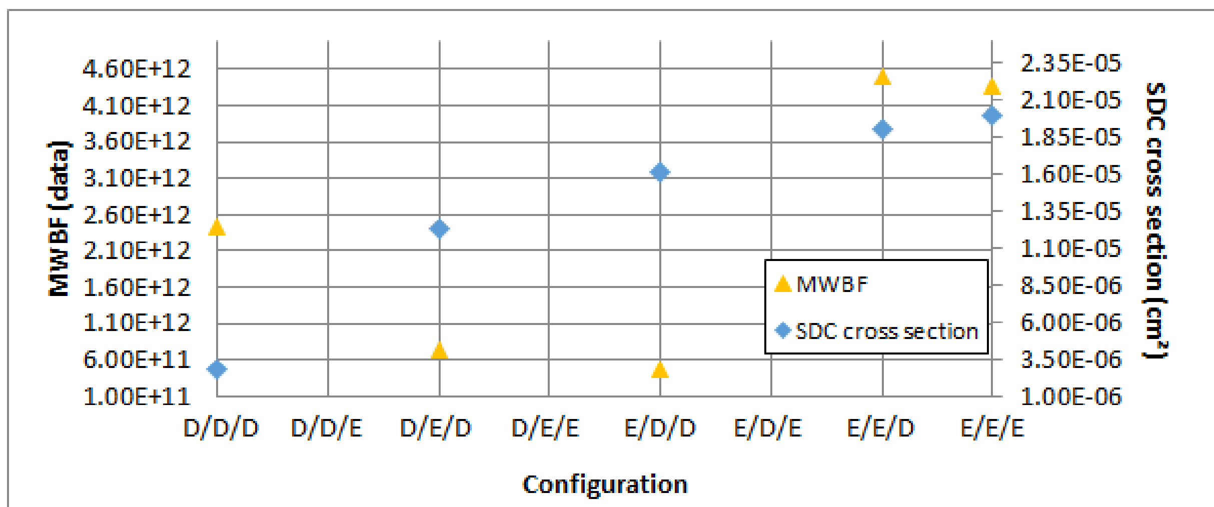
Table 5.8 – Obtained MTBF, MEBF, and MWBF from the heavy ion irradiations.

Configuration (L1I/L1D/L2)	MTBF (hours)	MEBF (executions)	MWBF (data)
D/D/D	$6.00x10^2$	$8.66x10^5$	$2.42x10^{12}$
D/D/E	-	-	-
D/E/D	$1.39x10^2$	$2.63x10^5$	$7.37x10^{11}$
D/E/E	-	-	-
E/D/D	$1.06x10^2$	$1.72x10^5$	$4.82x10^{11}$
E/D/E	-	-	-
E/E/D	$9.01x10^1$	$1.61x10^6$	$4.52x10^{12}$
E/E/E	$8.60x10^1$	$1.56x10^6$	$4.36x10^{12}$

Fig. 5.6 compares the experimentally measured SDC cross section from heavy ion experiments and the evaluated MWBF. The graph shows that whenever the speed-up is higher than the increase in cross section, the faster configuration computes a large workload before experiencing a failure. Thus, enabling caches not only improves performance but also increases reliability to radiation-induced errors.

In processors-based systems running on Zynq-7000, it is possible to achieve an even higher speed-up than just the one provided by enabling cache memories. In this case, co-processors in the PL are used as hardware accelerators, such as dedicated IPs or soft-core processors. Using the PL, tasks are offloaded from the PS to the PL part, which accelerates the tasks and reclaims processor bandwidth for additional tasks. Therefore, the next chapter investigates the Zynq-7000's PL part and the overall trade-offs between performance and reliability that different designs running on the PL provide to a system.

Figure 5.6 – Comparison between the obtained SDC cross section and MWBF values from the heavy ion irradiations.



6 ANALYSING SINGLE EVENT EFFECTS ON THE PL PART OF ZYNQ-7000

This chapter presents static and dynamic radiation tests performed in the PL part of Zynq-7000 for measuring the sensitivity of its configuration memory and the trade-offs between performance and reliability of hardware designs implemented in it.

Static tests investigated the Configuration Memory (CRAM) of the PL part of Zynq-7000 under heavy ions and protons. Similarly to the experiments performed in the Zynq-7000's PS part, some of the experiments also considered variations in the nominal supply voltage and temperature according to the ranges specified in the device's datasheet. Such analysis is necessary because due to the technology scaling, the amount of charge used to store information in the memory nodes is continuously decreasing, and less charge from an energetic particle is needed to change the logical state of a node. Moreover, the analysis of the temperature influence on the cross section behavior of devices like Zynq-7000 is important because according to (BAGATIN et al., 2011), the cross section of SRAM memories is strongly affected by temperature. However, temperature also induces changes in other parameters, such as charge collection efficiency and electron-holes mobility. These parameters can also be responsible for changing the cross section of modern devices like Zynq-7000.

Dynamic tests investigated the trade-offs of different designs implemented into an Artix-7 FPGA (equivalent to the Zynq-7000's PL) in terms of not only resource utilization and performance, but also reliability, by analyzing their behaviors under SEUs and comparing them to a standard processor-based implementation. The designs were implemented by HLS, since HLS tools provide a design methodology able to generate optimized digital designs for different application needs, as already shown in Section 2.2. Moreover, HLS tools have significantly evolved in the last years, providing very optimized results in area and performance with a very short development time. Such analysis is also important because HLS-based designs have been employed in several safety-critical applications that require high performance and high reliability level, such as the ADAS (XILINX, 2016d), medical systems (XILINX, 2016b), satellites (ITURBE et al., 2015), and particle accelerators (HUSEJKO, EVANS, DA SILVA, 2015). The experiments considered different HLS optimization strategies such as *default*, *pipeline* and *loop unroll* in different places, *array partition* with different configurations, and *inline* functions. The case-study designs consisted

of several design architectures based on three benchmark algorithms: Matrix Multiplication (MxM), Advanced Encryption Standard (AES), and Adaptive Differential Pulse-Code Modulation (ADPCM). The designs were evaluated by several radiation experiments with heavy ions and FPGA-based Fault Injection (FI) campaigns. As a result, this chapter proposes a reliability analysis for HLS-based designs and even traditional hardware designs that want to investigate the trade-offs between reliability and performance.

6.1 Static tests

6.1.1 Tests procedures

As well as the static tests of the PS part of Zynq-7000, the tests of the PL part consisted of the same steps described in Section 4.3.1. First, the configuration memory is configured with a fault-free bitstream (the *golden*) containing most of its bits in "*0*". Then, a script running on a control computer was constantly reading back the PL's configuration memory with the Xilinx iMPACT tool through a JTAG interface and comparing the bitstream read against the golden one. If differences were found, which means errors, the script time-stamped and logged the errors for then reconfigure the PL with the golden bitstream. The entire PL's configuration memory (32.3 Mb) was evaluated and all the obtained cross sections are per bit.

Concerning variations in the nominal supply voltage, they were performed by directly accessing the Zedboard power supply lines after the embedded input power regulator. With regard to the temperature variations, they were performed by heating the device with an air heater.

6.1.2 Tests setups

The characteristics of the heavy ion and proton experiments are the same of the static tests performed on the PS part, which were already described in Section 5.1.2. Table 6.1 summarizes the test schemes performed in the configuration memory of the PL part of Zynq-7000.

Table 6.1 – Heavy ion and proton test schemes performed in the configuration memory of the PL part of Zynq-7000.

Memory	Core Voltage 1 (V)	Core Voltage 2 (V)	Core Voltage 3 (V)	Temperature (ºC)
	Heavy ion tests			
CRAM	0.95	*	1.05	51.0
	Proton tests			
CRAM	0.95**	1.0	1.05	36.0

* Configuration unable to test due to beam limitations.

** Configuration also tested at 92 ºC.

6.1.3 Tests results

Fig. 6.1 shows the obtained results from heavy ion irradiations. Experiments performed at lower LETs were performed at the LAFN-USP facility, while the ones performed at higher LETs were performed at the ROSCOSMOS facility. The obtained results are in accordance to what was expected. Regarding supply voltage variation, the biggest cross section difference was only 4% (1.05V/0.95V, LET = 8.4 MeV.cm²/mg), what is smaller than the error bars. It is important to highlight that for LETs higher than 6.5 MeV/mg/cm², it was possible to observe precisely the occurrence of MCUs, when an SEU affects multiple bits, and MBUs, when an SEU affects multiple bits in the same word. From the total number of events observed, 33% were SEUs, 16% were MBUs together with MCUs and 51% were MCUs. No isolated MBUs were observed. Such behavior is quite critical, since MBUs and MCUs reduce the efficacy of ECCs, such as Single-Error Correct/Double-Error Detect (SECDED) codes.

Fig. 6.2 shows the obtained results from proton irradiations. From the results, one can observe that there are not significant differences in cross section according to the supply voltage applied to the PL part (CRAM). Moreover, uncertainties are such that the behavior is approximately constant. It is interesting to note that the biggest cross section variation was of about 20% (0.95V,92ºC/0.95V,36ºC, 250 MeV), and it was achieved when the device was heated and not with the supply voltage variation.

Figure 6.1 – Cross section results from the heavy ion irradiations in the configuration memory of the PL part of Zynq-7000.

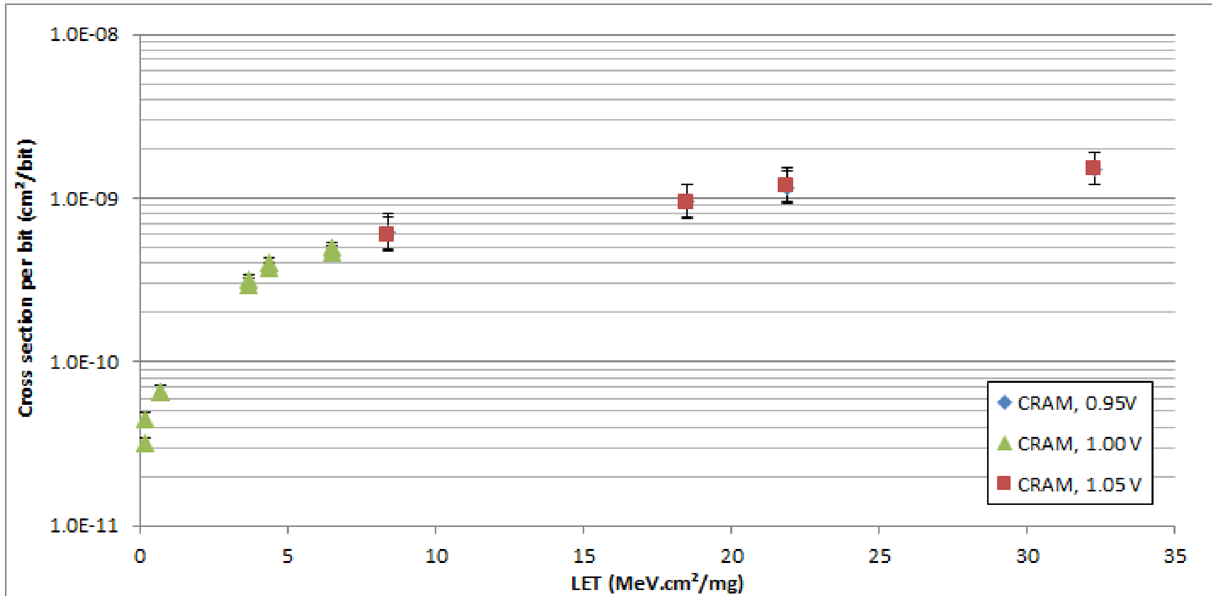
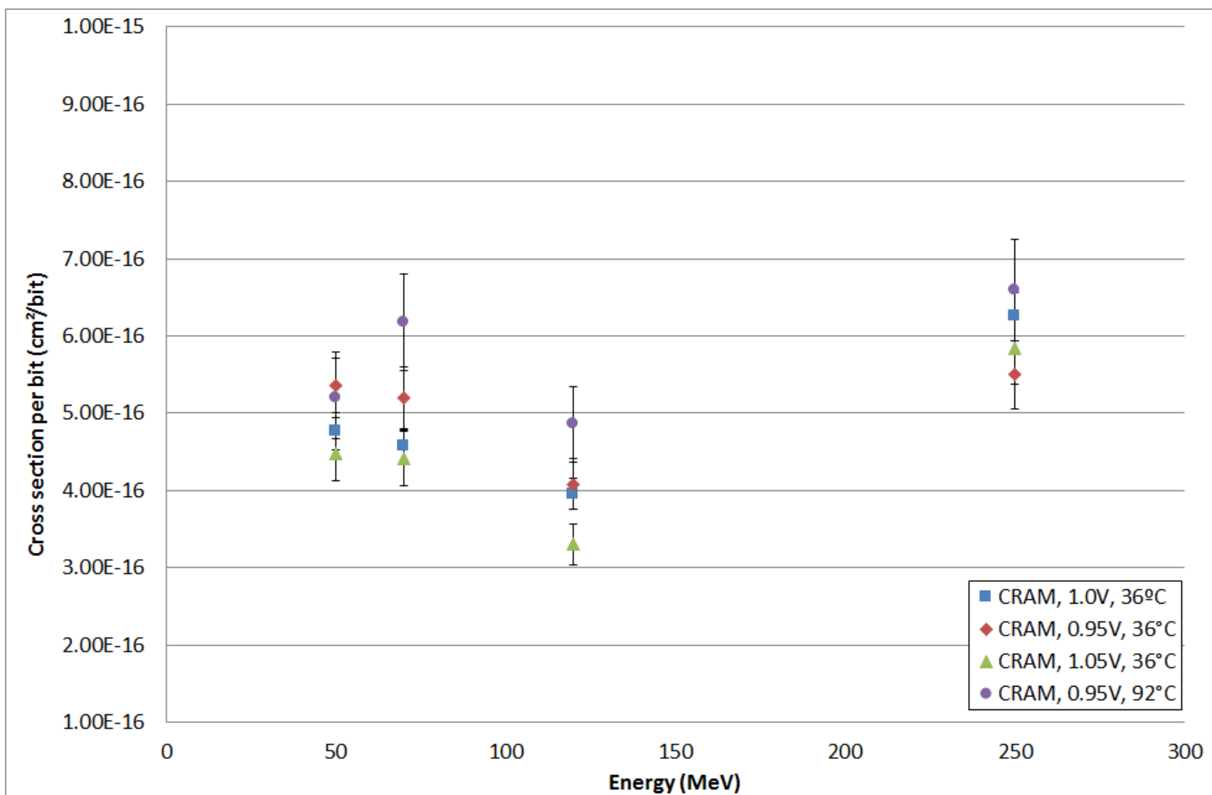


Figure 6.2 – Cross section results from the proton irradiations in the configuration memory of the PL part of Zynq-7000.



6.2 Dynamic tests and proposed reliability analysis for hardware-only designs

The dynamic tests performed focused on investigating the trade-offs of different hardware designs in terms of not only resource utilization and performance but also reliability. Thus, an analysis of their behavior under soft errors was performed and compared to a standard processor-based implementation running on a soft-core processor embedded in an SRAM-based FPGA. Although such investigation is not new, the novelty of it is that the hardware designs were generated by HLS targeting future applications in APSoCs, as addressed in the next chapter in Zynq-7000.

Different HLS optimization strategies were considered, such as *default*, *pipeline* and *loop unroll* in different places, *array partition* with different configurations, and *inline* functions. The goal was to observe that depending on the coding style of the high-level software programming language and optimization directives applied in the HLS tool, different amounts of FPGA resources and configuration bits, and distinct execution times are achieved by the generated hardware. Consequently, the cross section and MWBF may present significantly different results. However, it is not only the amount of FPGA resources and configuration bits that determine the sensitivity of a hardware. The error masking effect of the application algorithm implemented using HLS plays an important role, directly affecting the reliability of a design implemented in an FPGA, as shown in this section.

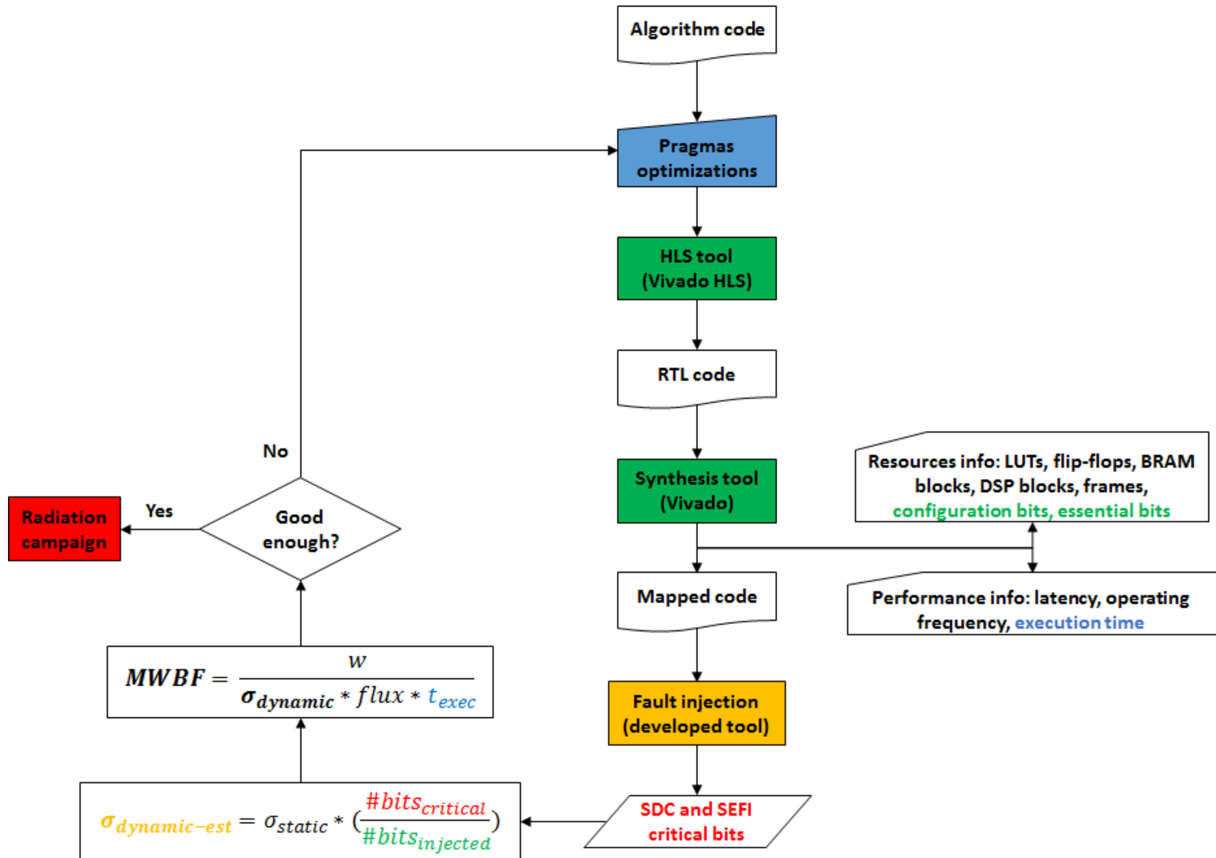
The designs were evaluated by several radiation experiments with heavy ions and FPGA-based Fault Injection (FI) campaigns, from which the dynamic cross section and MWBF are estimated for each design version. As a result, a reliability analysis for HLS-based designs is proposed. Results show that, in general, the estimation of the dynamic cross section and MWBF values of HLS-based designs through the proposed flow based on fault injection is a suitable method for predicting their trend before radiation experiments. Results also reveal that there are important trade-offs between the use of resources, optimization strategies, and execution time in order to achieve higher MWBF values.

6.2.1 Proposed reliability analysis for hardware-only designs

The reliability analysis of a design implemented in an FPGA depends on the characteristics of the design and susceptibility of the underlying FPGA platform. This section presents the proposed methodology flow to estimate the reliability of HLS-based designs based on fault injection campaigns. Nevertheless, the proposed methodology is capable to be generic and extendable to any type of design if slight adjustments are performed. It aims to accelerate the search for the design with the best trade-off between performance and

reliability, i.e. the design that provides a performance enhancement higher than the cross section increase. The methodology takes into account four parameter groups: A) Resources and performance in terms of execution time; B) Errors and critical bits; C) Radiation measurements such as static and dynamic cross sections; and D) Mean Workload Between Failures. Fig. 6.3 summarizes the proposed reliability analysis in a flow diagram.

Figure 6.3 – Proposed reliability analysis flow for hardware-only designs.



6.2.1.1 Resources and performance

The area of an implemented design can be expressed in terms of the number of used resources, such as LUTs, flip-flops, BRAM blocks, DSP blocks, etc. It is also possible to express the area in terms of configuration frames and configuration bits. A configuration frame is a group of configuration bits. It is the smallest addressable memory segment of the configuration memory (bitstream) of an SRAM-based FPGA. Since each frame is related to a specific resource and position in the floorplanning of the FPGA, the number of configuration frames (and configuration bits) used by a design can be calculated. In case of Xilinx 7-Series devices, a configuration frame is composed of 101 32-bit words (XILINX, 2014). In terms of

reliability, the resource information is important since it is used to determine how much the design is physically exposed to radiation.

The performance of a design can be expressed in terms of execution time, operational frequency, and processed workload. The execution time can be defined by the number of clock cycles needed to perform an operation. According to the FPGA and embedded design architecture, a maximum clock frequency is achieved. Another important parameter is the workload processed by the design, which is the amount of data computed at each design execution. In terms of reliability, performance information is important to determine how much time the design is exposed to soft errors during the execution of the implemented function.

6.2.1.2 Errors and critical bits

An error is defined as any deviation from the expected behavior of a design. As already mentioned, an error can be classified as SDC error (erroneous data in the design output) or SEFI error (absence of data in the design output after a given time). With regard to critical bits, Xilinx defines critical bits in (XILINX, 2012b) as the amount of configuration bits that once flipped, they cause an error in the expected design behavior (SDC or SEFI). Critical bits are obtained by means of fault injection in the DUT. In this thesis, the critical bits of each design version are obtained by an exhaustive and sequential fault injection campaign, in which all the configuration bits of the injection area are flipped one at a time. A bit is considered critical if it causes a deviation from the expected design output (SDC or SEFI). Thus, the process of generating a complete list of critical bits for a specific design is a time-consuming task that involves validating the correct design behavior while moving a single upset through all the configuration memory bits in the design region.

6.2.1.3 Radiation measurements

The methodology takes into account the cross section parameters already introduced in Section 4.3.2.

6.2.1.4 Mean Workload Between Failures

The methodology also takes into account the MWBF metric, which was also already introduced in Section 4.3.2.

6.2.1.5 Xilinx analysis tools

Xilinx Vivado design tool provides the area of an implemented design in terms of resource utilization after the design is placed and routed, and Xilinx Vivado HLS tool provides the performance of the design in terms of clock cycles. The execution time of an HLS-based design is obtained by multiplying the number of clock cycles by the clock period. The execution time of the processor-based design is obtained at execution time.

6.2.1.6 Fault injection method and analysis

The fault injector platform used in this thesis and the method for obtaining the critical bits were already presented in Section 4.2.1.

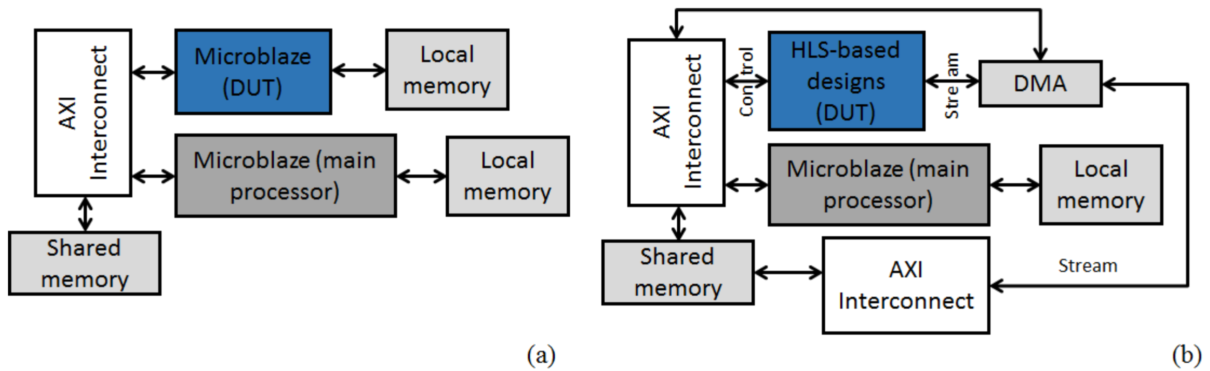
With regard to the analysis of the fault injection results, the method proposed by Velazco, Foucard, and Peronnard (2010) was used to estimate the dynamic cross section and MWBF, consequently, of hardware designs before radiation test campaigns. The method supposes that an approach allowing injecting bit-flips can be implemented for the considered DUT, which in this case is the fault injection platform already presented. In addition, the method requires a cross section derived from a static test for providing the average number of particles of a given type which is necessary to provoke a bit-flip of one of the memory cells included in the DUT. The static cross section for a specific particle and for a specific device can be obtained from previous experiments or reports, such as (XILINX, 2016c). The masking upset probability can be considered by using the information about the critical bits, which are first evaluated by means of fault injection. The dynamic cross section is then calculated by multiplying the static cross section by the masking upset probability of the design, as shown in Eq. 6.1.

(Equation 6.1) $$\sigma_{dynamic-estimated} = \sigma_{static} \cdot \left(\frac{\#bits_{critical}}{\#bits_{injected}}\right) \quad [cm^2]$$

6.2.2 Case-study designs and resources and performance results

Three C language-based benchmark algorithms were evaluated: i) 32x32 floating-point matrix multiplication (MxM) (XILINX, 2016a), ii) 128-bit Advanced Encryption Standard (AES) (HARA et al., 2008), and iii) Adaptive Differential Pulse-Code Modulation (ADPCM) (HARA et al., 2008). These algorithms were chosen because they cover a variety of domains, such as arithmetic (MxM), security (AES), and media processing (ADPCM). Moreover, they range from a data flow oriented algorithm (MM) to a control flow oriented algorithm (ADPCM). The processor-based design has a Microblaze (MB) soft-core processor, which is a 32-bit 5-state pipeline Reduced Instruction Set Computer (RISC) soft processor. The architecture of both HLS-based designs and processor-based design are shown in Fig. 6.4.

Figure 6.4 – Architecture of the (a) processor-based design and the (b) HLS-based design.



Three HLS-based designs for each benchmark application were generated in Xilinx Vivado HLS, each one with different optimization strategies. The optimizations applied are summarized in Table 6.2. The optimizations were chosen focusing on speeding up the critical path of the algorithms. In the MxM benchmark, the main optimization effect was the increase in pipeline depth, which resulted in a 21-stage pipeline in HLS1, 220-stage pipeline in HLS2, and a 3235-stage pipeline in HLS3. It is worth noting that in MxM-HLS3, the *array partition* directive was applied aiming to increase the throughput of the input data, which significantly increased its pipeline depth and performance. In the AES and ADPCM benchmarks, the main optimizations applied were also related to the data parallelism (unroll loop and pipeline), although at different levels of granularity. This was done since they present a lot of data dependencies among their several internal functions and to not modify the original benchmark programs. The inputs and outputs of the three benchmarks use the *resource* directive to force the variables to be synthesized as Advanced eXtensible Interface Stream (AXI-S) channels in order that all the HLS-based designs have the same interface architecture, as shown in Fig.

6.4(b). Moreover, the input data of the three benchmarks were fixed so that they were known. All the input and output elements were implemented with BRAM memories protected with ECC for the designs tested under radiation. All designs were synthesized into a Xilinx Artix-7 FPGA, part XC7A100TCSG324-1, embedded into a Digilent Nexys 4 board, and with an input clock of 100 MHz.

Table 6.2 – Optimization strategies applied in each HLS-based design for each benchmark program.

Benchmark	Optimization strategy		
	HLS1	HLS2	HLS3
MxM	- None	- Pipeline in the middle loop	- Pipeline in the middle loop - Array partition in the inputs - Function inline
AES	- None	- Unroll the loops of the main - Pipeline the encryption function - Pipeline the decryption function	- Unroll the loops of the main - Pipeline the encryption function - Pipeline the decryption function - Pipeline the key function
ADPCM	- None	- Pipeline fine grain	- Pipeline coarse grain

Resource usage, performance results in terms of clock cycles, and number of critical bits obtained by fault injection for each case-study design are presented in Table 6.3. The workload considered for each benchmark was 32768 bits (1024 values of 32-bit each) for the MxM, 1024 bits (32 values of 32-bit each) for the AES, and 3200 bits (100 values of 32-bit each) for the ADPCM. Results show that as the optimization level of the HLS-based designs is increased, the number of resources used (15.6 times for the MxM in the worst case) and the number of critical bits increases as well (4.19 times for the AES in the worst case). However, the highest increase is related to the performance (73.5 times for the MxM in the worst resource increase). This means that increasing the hardware parallelism using more resources, greatly increases the workload capability of the system. Moreover, the use of the *array partition* directive also plays an important role by decreasing memory bottlenecks. By comparing the use of a processor-based approach versus an HLS-based design, one can see that, in general, HLS-based designs use more area. However, the performance improvement they provide is even higher (31.0 times when comparing ADPCM-MB versus ADPCM-HLS1 in the worst case).

Table 6.3 – Resource usage and performance results of each case-study design.

Design version	Resources and performance						
	# LUT	# FF	# DSP	# BRAM	# Config. bits	# Critical bits (% of config. bits)	t_{exec} (clock cycles)
MxM							
MB	1159	1452	0	0	524342	70089 (13.37)	41611263
HLS1	755	944	5	3	435879	35199 (8.08)	366354
HLS2	2397	3374	10	3	1000132	47164 (4.72)	19613
HLS3	11787	13924	160	33	4606244	94692 (2.06)	4982
AES							
MB	1159	1452	0	0	524342	70089 (13.37)	104905
HLS1	5780	2380	0	17	896694	79205 (8.83)	3188
HLS2	25243	6511	0	19	2023681	198661 (9.82)	2269
HLS3	34725	12343	0	15	3823592	331971 (8.68)	1317
ADPCM							
MB	1159	1452	0	0	524342	70089 (13.37)	1091250
HLS1	5435	6385	103	8	1898542	230420 (12.14)	35144
HLS2	4911	6157	103	14	1806674	232686 (12.88)	17316
HLS3	14959	18681	240	14	6023727	817025 (13.56)	8462

The number of critical bits that cause SDC or SEFI errors for each case-study design is also shown in Table 6.3. Comparing the obtained results from fault injection with the number of configuration bits in the DUT area, one can observe that a small fraction of the configuration bits (from 2% to 13%) is, in fact, critical. Moreover, a slight variation in the percentage of critical bits over the total number of configuration bits among the different HLS-based designs of each benchmark circuit can also be observed. In some cases (all MxM-HLS designs and AES-HLS1 vs. AES-HLS3), the percentage of critical bits decreased even with the increase in the resource utilization. This behavior is similar to one observed in (HILL, LIPASTI, 2010), in which authors stated that deeper pipelines are in many cases more resilient that their shallower counterparts if proper metrics are taken into account for considering effects such as timing window masking, such as MWBF.

6.2.3 Cross section and MWBF results

Radiation experiments were also carried out with heavy ions at LAFN-USP, Brazil (AGUIAR et al., 2014), where the ion beams were produced and accelerated by the São Paulo 8UD Pelletron Accelerator. The setup configuration of the experiments is quite similar to the

one of the previous experiments. However, this time the experiments were performed in vacuum and the SEU events were observed using a ^{16}O beam scattered by a 184 µg/cm² gold target, with an energy of 56 MeV, which provided an effective LET on the active region of 5 MeV/mg/cm² and penetration in Si of 28.5 µm. To achieve the desired particle flux, the DUT was positioned at a scattering angle of 90º, which resulted in an average flux between 2.0×10^{2} and 2.5×10^{2} particles·cm^{-2}·s^{-1}. This configuration was chosen based on several trials and was the most suitable in terms of particle flux and number of errors. Each experiment run was set by the total number of errors observed (60) in the DUT output. Errors in the soft-core processor that sends data to the DUT are not considered in the count. All other uncertain errors observed are considered in the errors bars. For the heavy ion experiments, the package of an XC7A100TCSG324-1 device was thinned to allow irradiated particles to penetrate the active region of the silicon.

A small FSM (one-hot encoding) implemented with TMR performed the detection of incorrect outputs by comparing the computed results of the DUTs with their expected reference values stored in embedded BRAMs protected with ECC. A host computer monitored the FSM through a serial interface. If there were no differences in the results, the FSM sent an alive signal to the host computer at each execution set. Otherwise, if differences were found, the FSM sent the number of mismatches to the host computer and then the FPGA was reset. Timeouts in the DUT were monitored through a watchdog circuit in the FSM. The host computer also had a watchdog circuit to monitor timeout occurrences in the FSM. In case of timeout, the FPGA was reset.

The obtained SDC, SEFI, and TOTAL (sum of SDCs and SEFIs) dynamic cross section results from both fault injections and radiation experiments are presented in Fig. 6.5, 6.6, and 6.7. With regard to the fault injection results, a reference static cross section of 1.11×10^{-9} cm² (effective LET = 5.25 MeV/mv/cm^{2} and flux = 1.32×10^{2} particles·cm^{-2}·s^{-1}) was adopted, which was obtained from previous heavy ion experiments with the same device. Notably, one can observe that there are differences in the obtained values from fault injections and radiation experiments (3.18 times in average and 13.07 times for the worst case - ADPCM-HLS3-TOTAL). In general, data from fault injection are more pessimistic than the ones from the radiation experiments. Such behavior was already expected, since the fault injection methodology is more deterministic and gives a very fine fault granularity, enabling to estimate the worst-case cross section of a design without considering side-effects, such as beam variations and fault masking by the design. Therefore, it is possible to determine precisely the exact number of bits that are critical and consider them in the estimated dynamic

cross section. It is also worth noting that radiation experiments cover the whole architecture, while the fault injection campaigns cover only the DUT (shown in Fig. 6.4). However, for a same benchmark application, the HLS-based design (DUT) is the only part of the system that changes among the different versions. This somehow normalizes the sensitivity of the processor among the designs, since the soft-core processor and peripherals never change among the designs. It is also important to reinforce that the proposed reliability analysis is capable of estimating the dynamic cross section trend for different HLS-based designs of a same benchmark application, even with the mentioned differences.

By analyzing the obtained dynamic cross section trends from both radiation experiments and fault injections for all HLS-based designs, one can notice interesting behaviors. The higher the optimization level, the more resources a design uses. However, in general, this does not necessarily imply in higher values of critical bits and SDC and SEFI cross sections. Data flow oriented algorithms are characterized by a significant amount of latches (high number of FFs compared to LUTs) and arithmetic operations with few control dependencies, while control flow oriented algorithms contain many combinational logic (high number of LUTs compared to FFs) and relational operations. Thus, in the most data flow oriented algorithm, the MxM, one can generalize that their cross sections followed the trend of the amount of configuration bits, most probably due to the increase in the pipeline depth and consequent latch count, which could have masked SEUs. On the contrary, in the most control flow-oriented algorithms, the AES and ADPCM, one can generalize that their cross sections followed the trend of the amount of critical bits as expected, most probably due to the lower probability of masking effects in their combinational logic. With regard to the comparison between SDC and SEFI cross section results, the most evident one is that, in general, HSL-based designs presented a higher SDC cross section than the processor, which presented higher a SEFI cross section than the HLS-based designs. This happened most likely because the processor is more prone to experience SEFIs due to its control flow logic, which is more complex than in HLS-based designs. On the contrary, in HLS-based designs predominate data structures, making them more prone to experience SDCs. One can also note that is important to consider both SDC and SEFI cross section separately, since the TOTAL cross section may mask important data about the designs, such as the SEFI cross section decrease of the AES HLS-based designs with the optimization level increase.

Figure 6.5 – SDC, SEFI, and TOTAL dynamic cross section results obtained for the MxM designs from both fault injections and radiation experiments.

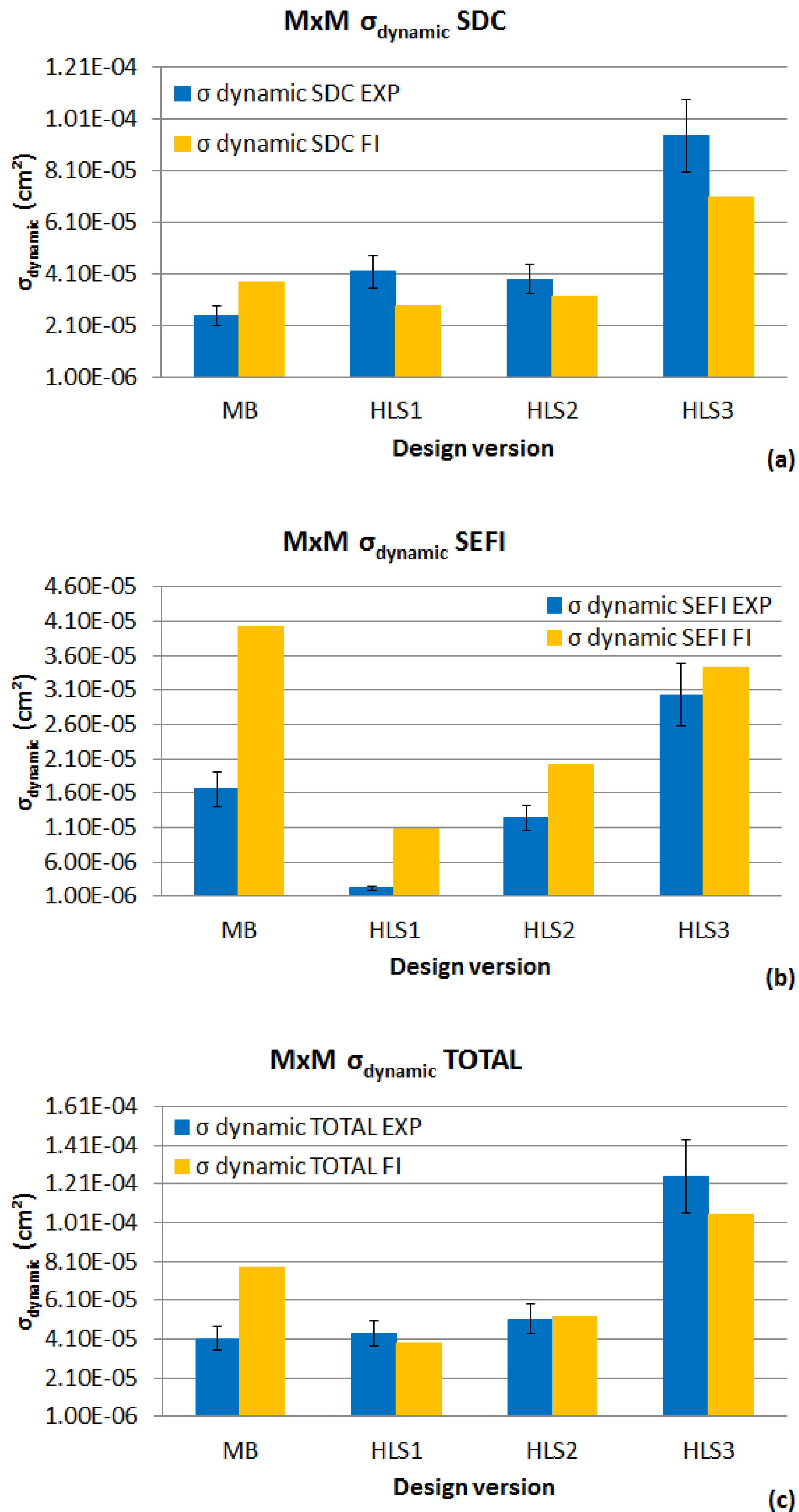


Figure 6.6 – SDC, SEFI, and TOTAL dynamic cross section results obtained for the AES designs both fault injections and radiation experiments.

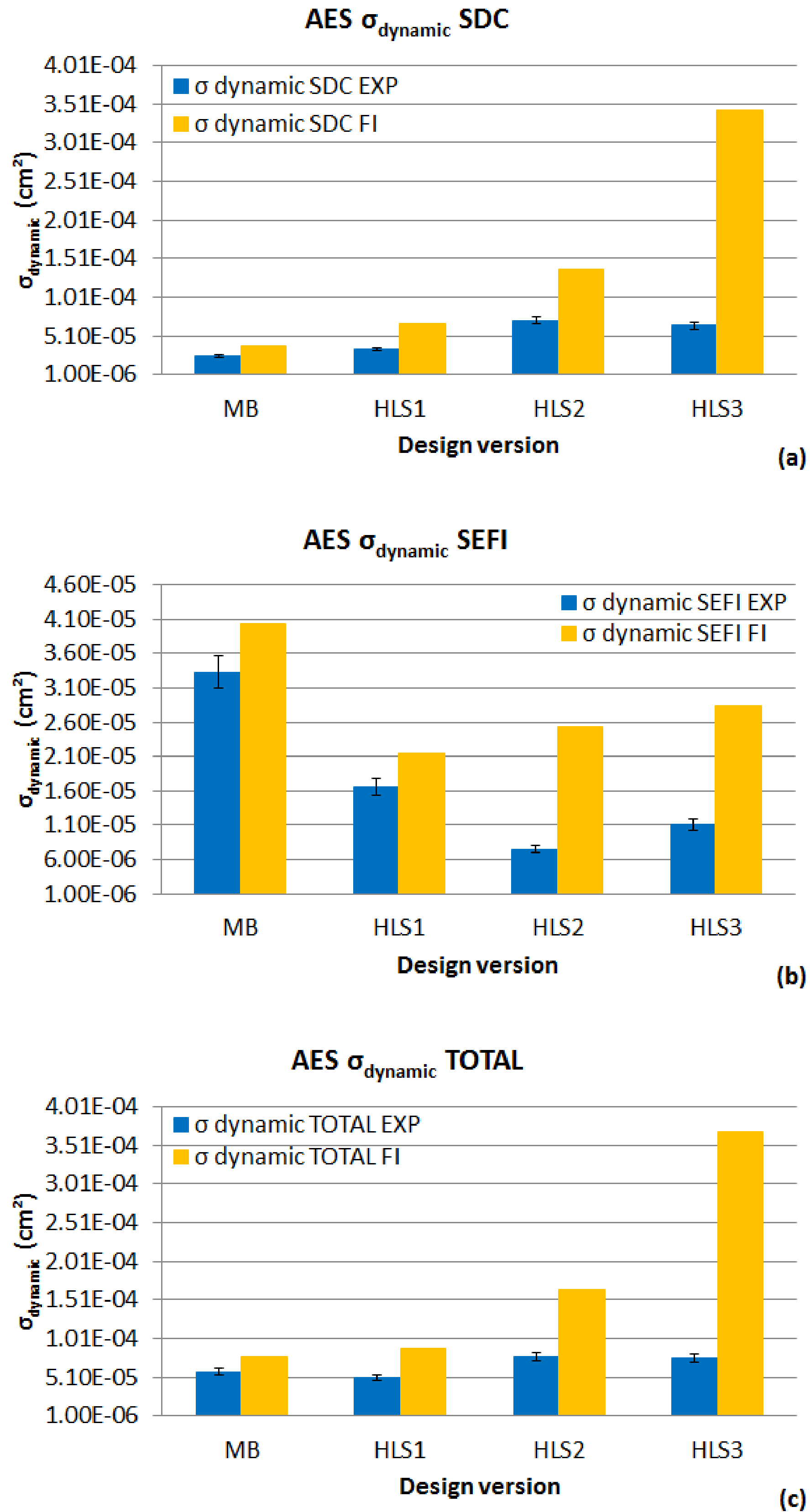
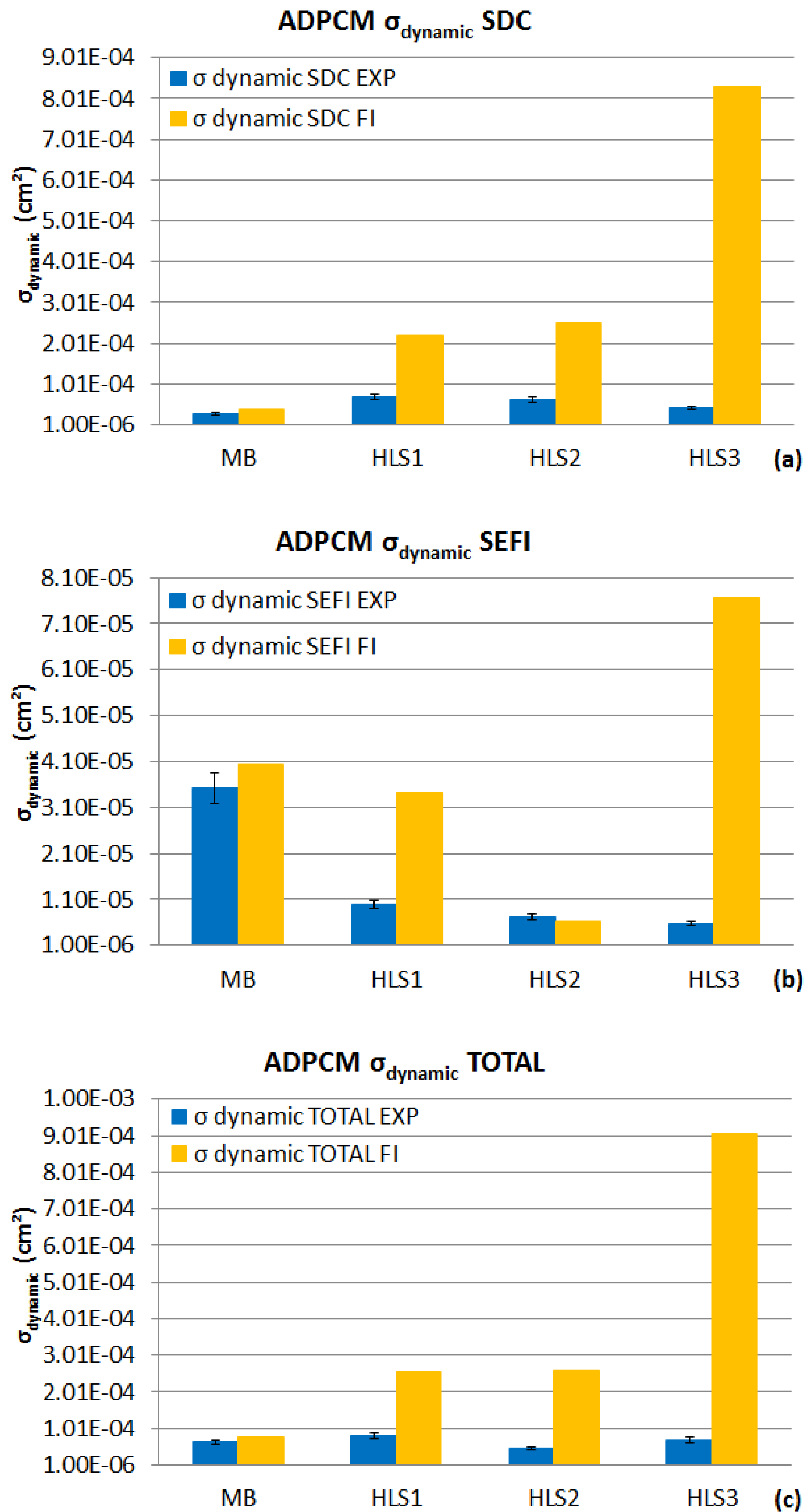


Figure 6.7 – SDC, SEFI, and TOTAL dynamic cross section results obtained for the ADPCM designs both fault injections and radiation experiments.



The obtained MWBF results from both fault injections and radiation experiments are depicted in Fig. 6.8, 6.9, and 6.10. A comparison between the fault injections and radiation experiments results shows that there are slight differences in the obtained values (2.13 times in average and 7.21 times for the worst case - ADPCM-HLS3-SDC). In addition, they show that the proposed reliability analysis is capable of estimating the MWBF trend of different HLS-based designs for the same benchmark application with a higher precision than for the cross section. This reinforces the importance of taking into account additional parameters, such as the execution time, and not only the sensitivity of the device. By analyzing the obtained MWBF values from both fault injections and radiation experiments, it is possible to observe that the use of different optimization strategies has a direct impact in the final MWBF values of the different HLS-based. Such behavior is mainly guided by the execution time improvement achieved with the different optimization strategies chosen and cannot be observed if only the dynamic cross section measurements are taken into account. The only performance decrease observed after applying some optimization strategy is related to the AES benchmark. This is related to how the benchmark is encoded. Moreover, we chose not to modify the algorithm. In this case, the main AES function has the following structure: *for* loop, *encrypt* function call, *for* loop, *decrypt* function call, *for* loop. In addition, inside the *encrypt* and *decrypt* functions, there are calls to a *key* function, which calculates the key of the algorithm. With such organization, whatever the optimization strategy applied in the *for* loops or the internal functions, there will always be a data bottleneck among such blocks. This behavior cannot be observed in the MxM benchmark because the algorithm is quite simple and very data flow oriented, which facilitates data parallelization. Concerning the ADPCM benchmark, even though the algorithm is control flow oriented, it is more modularized than the AES and with less data exchange among them. The lower MWBF values for the processor-based design were already expected due to their higher execution times.

Figure 6.8 – SDC, SEFI, and TOTAL MWBF results obtained for the MxM designs both fault injections and radiation experiments.

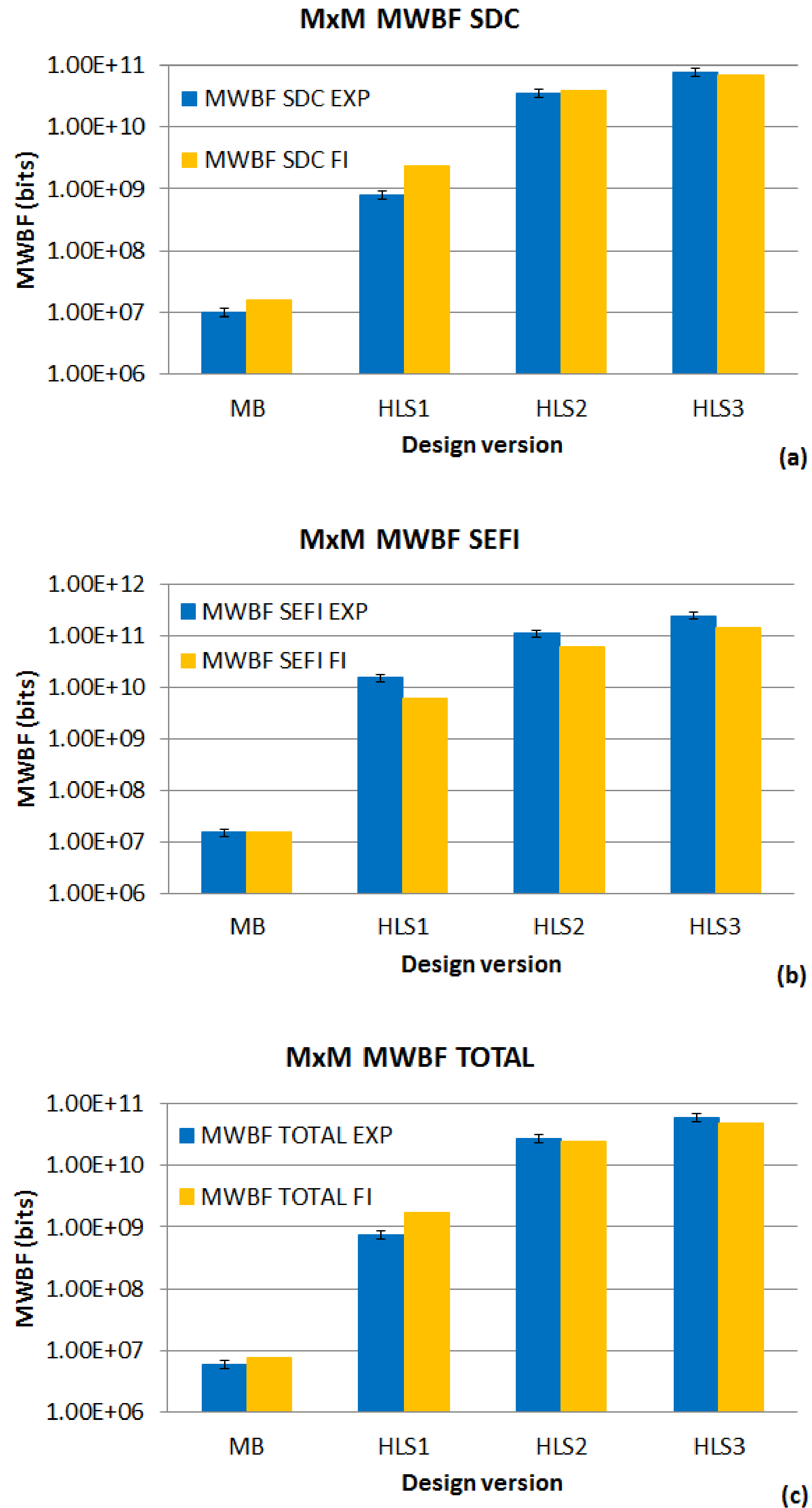


Figure 6.9 – SDC, SEFI, and TOTAL MWBF results obtained for the AES designs both fault injections and radiation experiments.

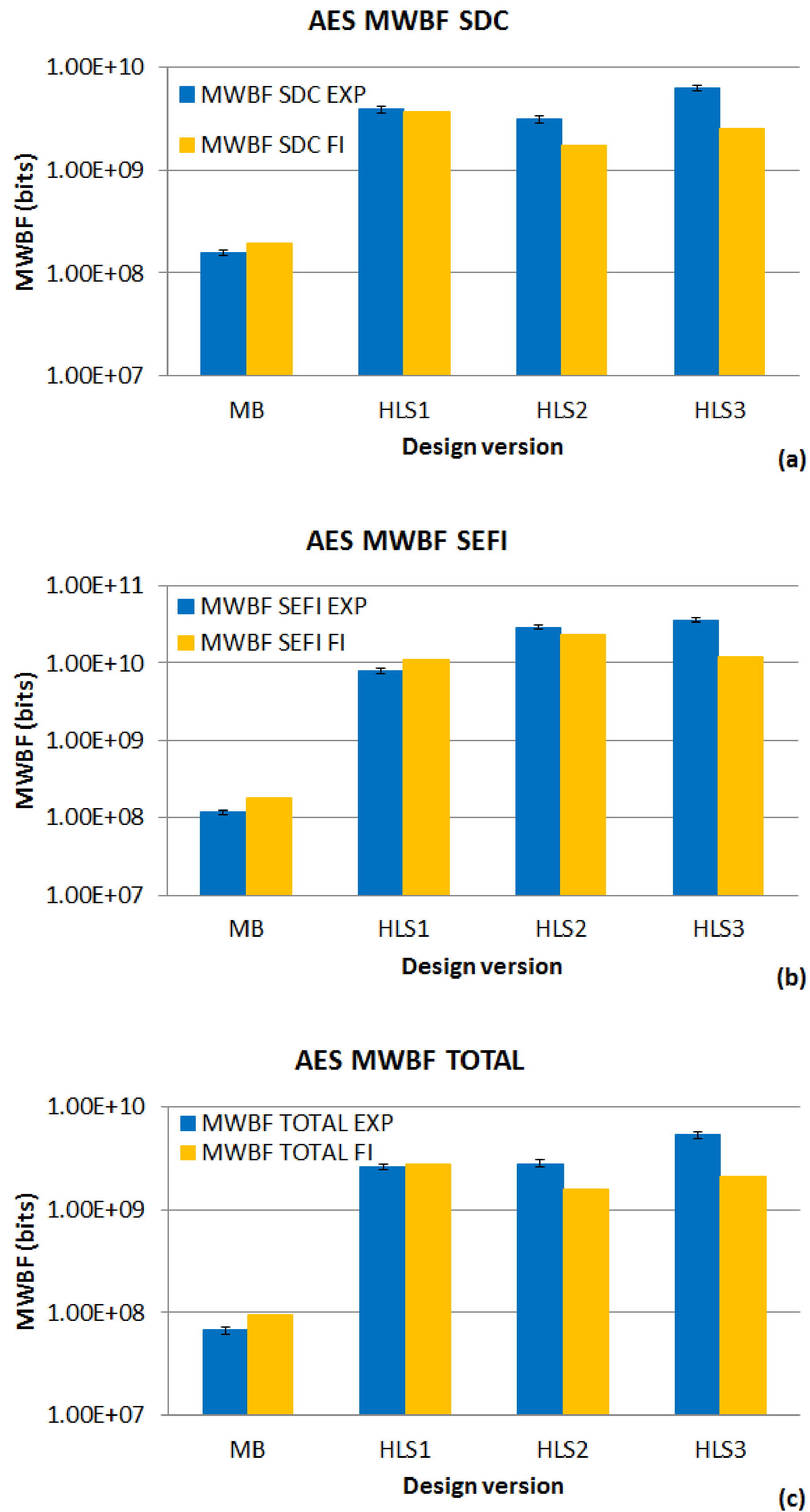
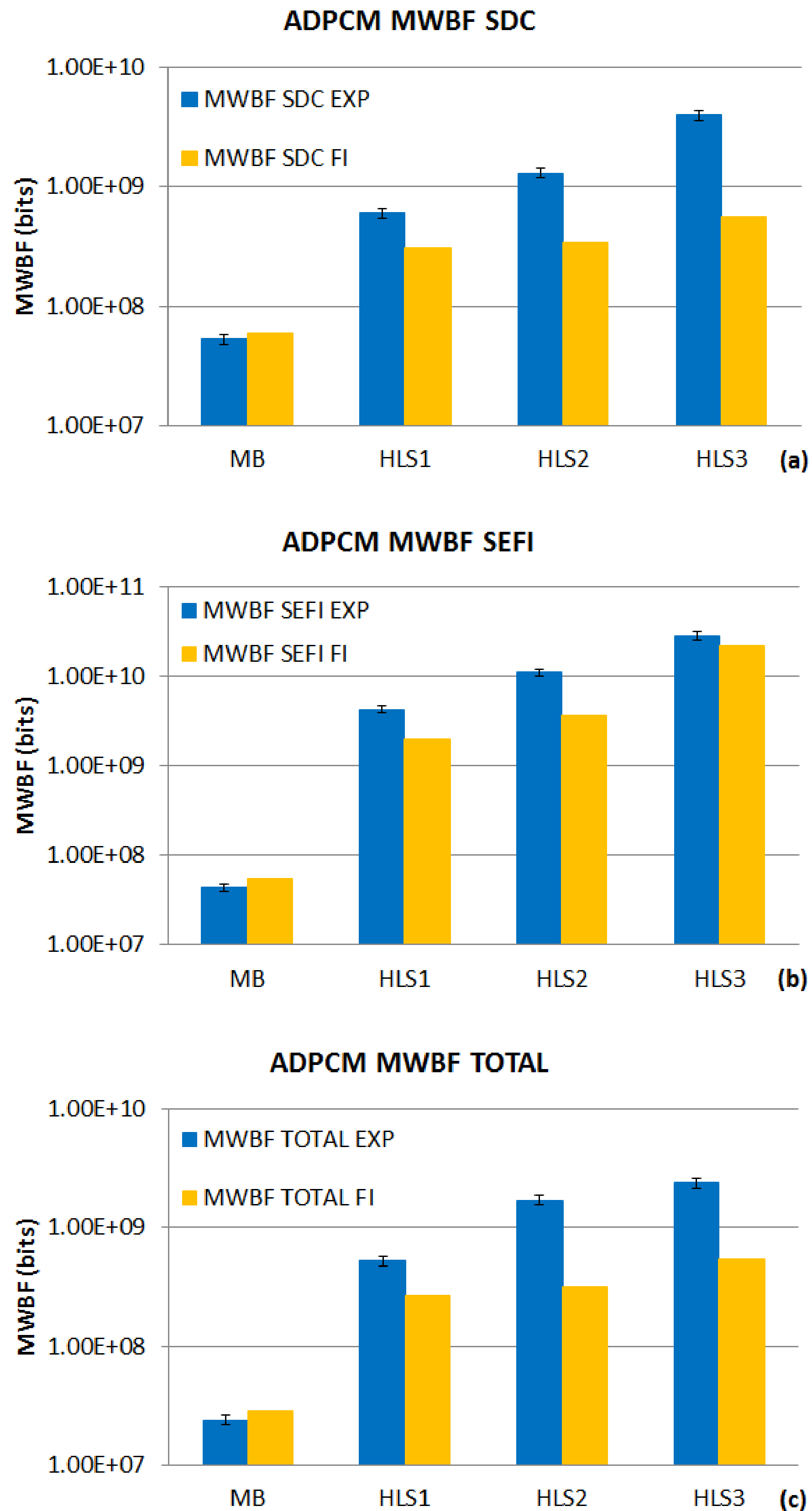


Figure 6.10 – SDC, SEFI, and TOTAL MWBF results obtained for the ADPCM designs both fault injections and radiation experiments.

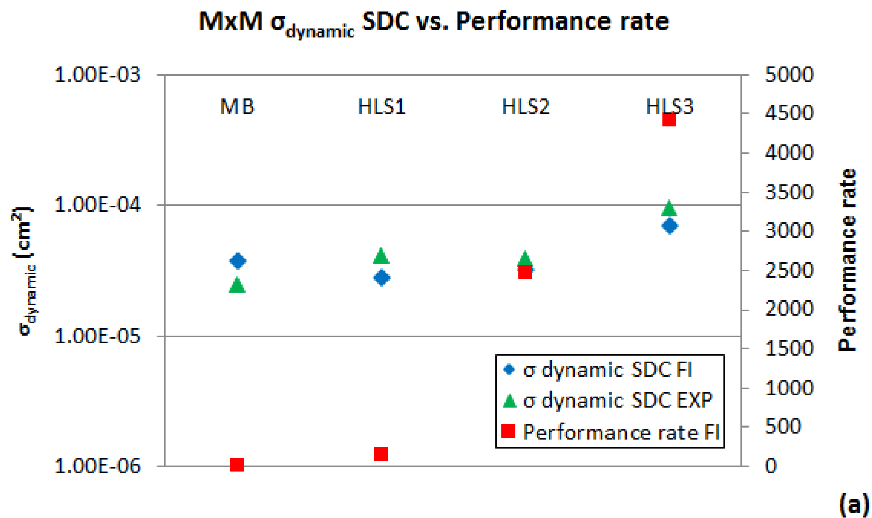


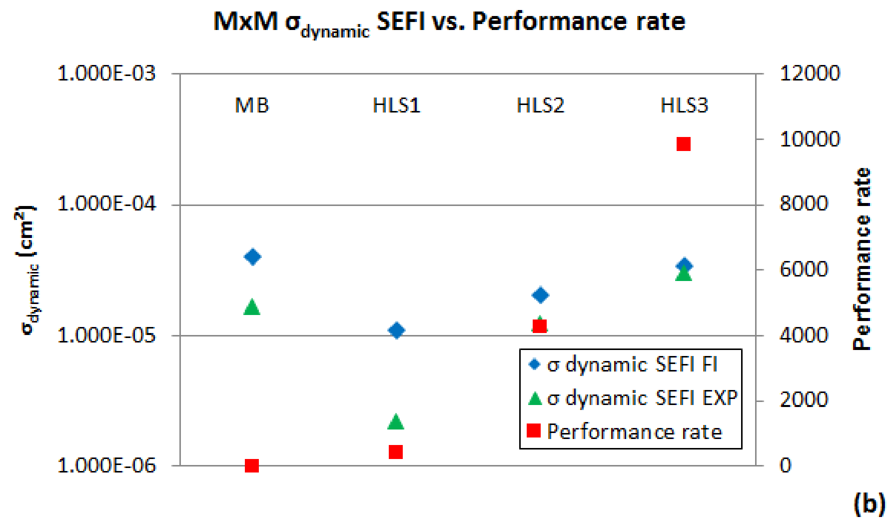
A comparison between the obtained dynamic cross sections by fault injection and radiation experiments and the performance rate improvement for each HLS-based design compared to its corresponding processor-based version is presented in Fig. 6.11, 6.12, and 6.13. The performance rate is calculated as shown in Eq. 6.2.

(Equation 6.2) $$Performance\ rate = \frac{t_{processor} \cdot \sigma_{dynamic-processor}}{t_{HLS} \cdot \sigma_{dynamic-HLS}}$$

The performance rate relation states that, whenever the speed-up is higher than the increase in cross section ($\frac{t_{processor}}{t_{HLS}} > \frac{\sigma\ dynamic_{HLS}}{\sigma\ dynamic_{processor}}$), the faster configuration computes a larger workload before experiencing a failure and, thus, the operational reliability of the system is higher. Thus, one can be concluded that, in this case, the designs with the best trade-off between reliability and performance for each benchmark application are MxM-HLS3, AES-HLS2, and ADPCM-HLS2. These result reveals that designs with very different architectures may present similar results due to a smaller cross section or execution time, such as in the AES HLS-based designs.

Figure 6.11 – Comparison between SDC, SEFI, and TOTAL dynamic cross sections and their respective performance rates for the MxM designs from fault injection and radiation experiment results.





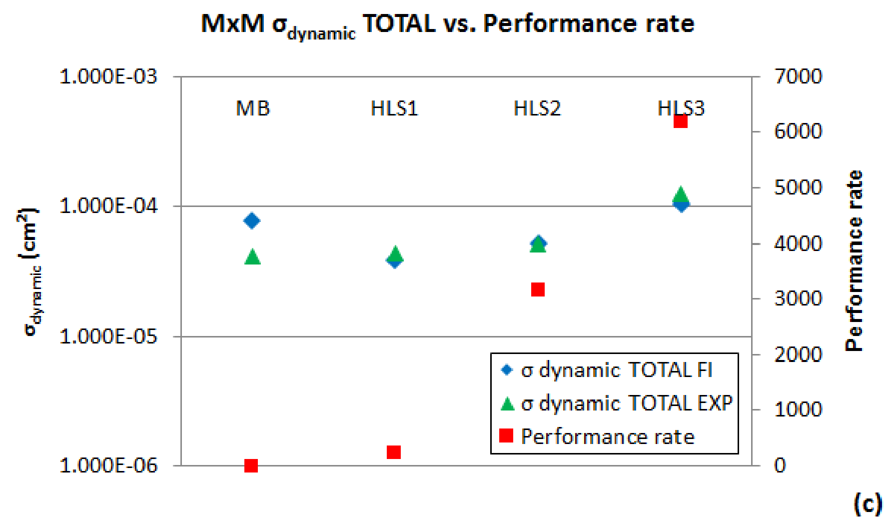
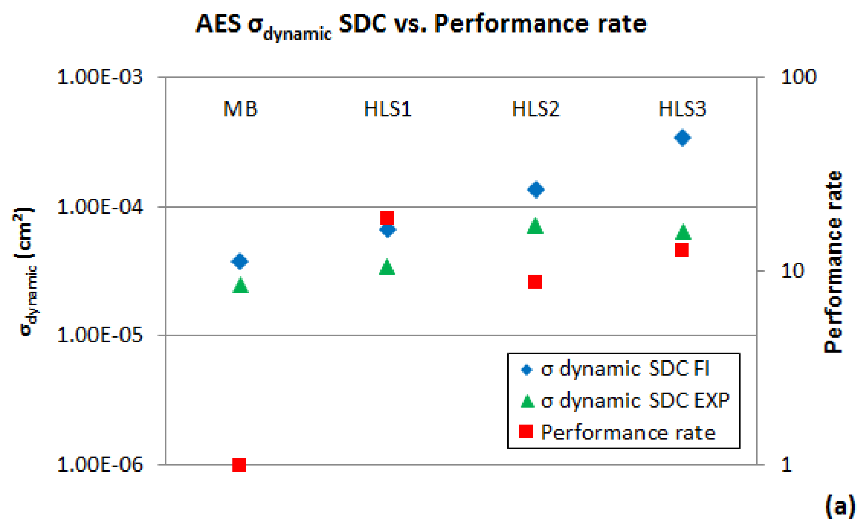
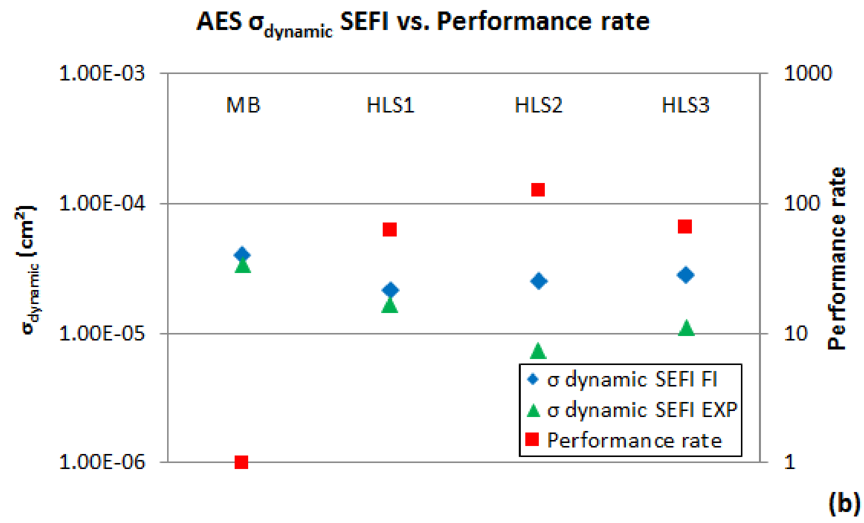


Figure 6.12 – Comparison between SDC, SEFI, and TOTAL dynamic cross sections and their respective performance rates for the AES designs from fault injection and radiation experiment results.





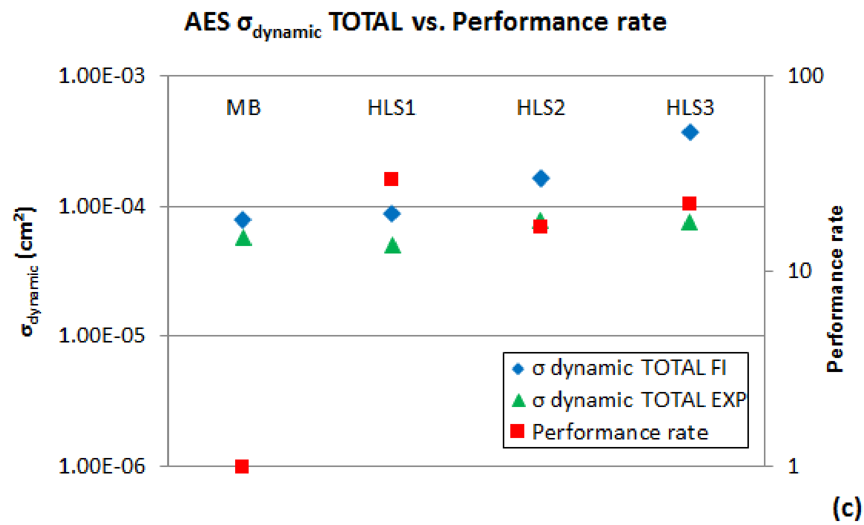
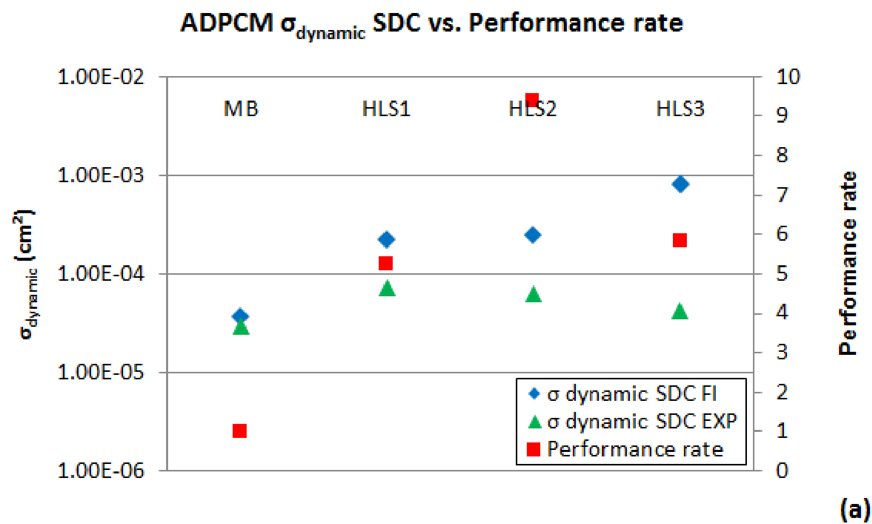
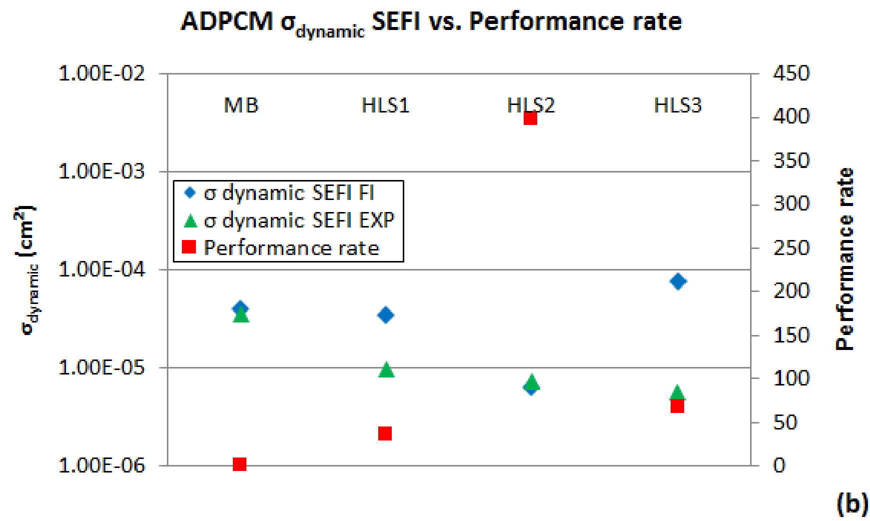
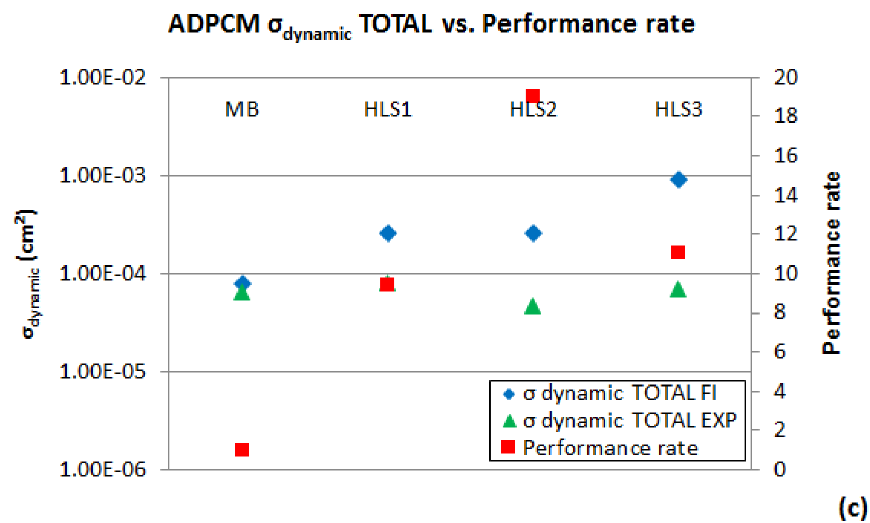


Figure 6.13 – Comparison between SDC, SEFI, and TOTAL dynamic cross sections and their respective performance rates for the ADPCM designs from fault injection and radiation experiment results.







6.3 Summary

Results show that, in general, the estimation of the dynamic cross section and MWBF of an HLS-based design through the proposed flow is a suitable method for predicting their trend before radiation experiments. Results also show that it is mandatory to take into account each design option available and all the parameters of the system involved in a dynamic test, such as the cross section, execution time, and workload of the application. Moreover, the proposed methodology is capable to be generic and extendable to any type of design if slight adjustments are performed.

With regard to the effects of optimization strategies in the final HLS-based designs, results reveal that HLS tools have evolved in the last decade by providing very structured and

optimized RTL codes. The influence of HLS optimizations in the dynamic cross section of the designs is lower (increase up to 2.85, 1.49, and 0.86 times for the MxM, AES, and ADPCM benchmarks, respectively) when compared to their performance enhancement (increase up to 73.54, 2.42, and 4.15 times for the MxM, AES, and ADPCM benchmarks, respectively), which contributes significantly to increase the MWBF and the performance rate of them.

7 EXPLORING BOTH PS AND PL PARTS OF ZYNQ-7000 UNDER SINGLE EVENT EFFECTS

This chapter presents an analysis of the impact of using both PS and PL parts of Xilinx Zynq-7000 APSoC in the overall system failure rate. Different memory organizations, communication schemes, and computing modes, were considered. Such investigation is fundamental since there are various possibilities to implement a system in an APSoC and each one imposes a different amount of resources and a different resource utilization efficiency, which impacts the reliability of the system. Moreover, defining the correlation between hardware and software resource sensitivity and the benefits brought to the system efficiency is essential to evaluate the system sensitivity, besides the balance between hardware and software in terms of reliability has not yet been investigated.

7.1 Reliability of hard- and soft-cores heterogeneous processing in the Zynq-7000

This section presents a reliability analysis of a multiprocessor-based heterogeneous system aiming to evaluate the sensitivity of different system architectures implemented in an APSoC to radiation-induced errors. Different configurations were implemented, each one using the PS and PL parts of the Zynq-7000 in distinctly, as described below. The processors were configured in an asymmetric multi-processing scheme (please refer to Secion 2.1.1.1).

The case-study architecture is a heterogeneous computing system consisting of one ARM Cortex-A9 (ARM) core of the PS part and one Microblaze soft-core processor (MB) embedded in the PL part, as shown in Figure 7.1. Both processors are 32-bits and have their caches enabled. The design uses the OCM memory as data memory and to exchange data among the processors. Four different architectures were implemented. The first two configurations were developed to investigate the reliability of sharing memory among the ARM and Microblaze processors while in the latter two configurations were developed to investigate the co-processing behavior by using both ARM and Microblaze processors in tandem. The resources usage of each configuration is shown in Table 7.1 and each configuration is described as follows:

- *Case A*: parallel executions of ARM and Microblaze processors without shared data. ARM uses OCM memory space from address *FFFC0020'h* to

FFFCEA60'h and Microblaze uses OCM memory space from address *FFFCEAE0'h* to *FFFDD540'h*.

- *Case B*: parallel executions of ARM and Microblaze processors with shared data. ARM uses OCM memory space from address *FFFC9CA0'h* to *FFFCEAC0'h* and Microblaze uses OCM memory space from address *FFFD8760'h* to *FFFDD580'h*. The operands are shared and they are stored in a shared memory space that ranges from address *FFFC0020'h* to *FFFC9C60'h*. When the system runs for the first time, ARM processor generates the operands. In any other case, like when one of the processors crashes, the operational processor refreshes the operands generating new values.
- *Case C*: heterogeneous co-processing with ARM processor acting as main processor and Microblaze processor as a support processor. In this case, Microblaze processor performs only the matrix multiplication operation and ARM processor performs all other operations. Both processors share the same memory space, which ranges from address *FFFC0020'h* to *FFFCEA80'h*.
- *Case D*: heterogeneous co-processing with Microblaze processor acting as main processor and ARM processor as a support processor. In this case, ARM processor performs only the matrix multiplication operation and Microblaze processor performs all other operations. As in Case C, both processors share the same memory space.

Both processors, in all configurations, execute the benchmark application in bare-metal, which is a 25x25 integer matrix multiplication operation implemented in standard C programming language. The detection of incorrect answers is achieved by calculating and comparing the Cyclic Redundancy Check (CRC) of the results. The code executed by both processors is represented in Figure 7.2.

Figure 7.1 – Block diagram of the case-study multiprocessor-based heterogeneous system.

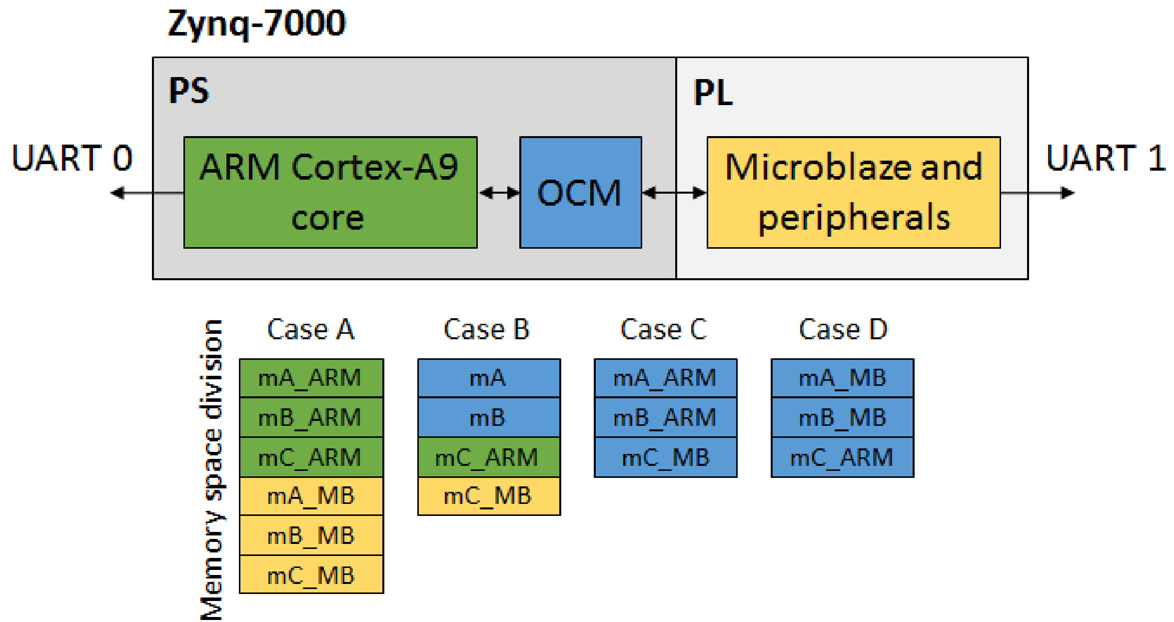


Table 7.1 – Resource usage and performance of each case-study architecture implemented.

<table>
<tr><th rowspan="2"></th><th colspan="2">Case A</th><th colspan="2">Case B</th><th>Case C</th><th>Case D</th></tr>
<tr><th>ARM</th><th>MB</th><th>ARM</th><th>MB</th><th>ARM</th><th>MB</th></tr>
<tr><td>Area of the processors</td><td>1</td><td>1640 LUTs, 6 BRAMs, 3 DSPs</td><td>1</td><td>1640 LUTs, 6 BRAMs, 3 DSPs</td><td>1 + (1640 LUTs, 6 BRAMs, 3 DSPs)</td><td>1 + (1640 LUTs, 6 BRAMs, 3 DSPs)</td></tr>
<tr><td rowspan="2">Address space</td><td rowspan="2">60 Kb in OCM and L1 and L2 caches</td><td rowspan="2">60 Kb in OCM and 8 KB of instructio n and data caches in BRAMs</td><td>20 Kb in OCM and L1 and L2 caches</td><td>20 Kb in OCM and 8 KB of instructio n and data caches in BRAMs</td><td rowspan="2">60 Kb shared in OCM, L1 and L2 caches of ARM and 8 KB of instruction and data caches in BRAMs for MB</td><td rowspan="2">60 Kb shared in OCM, L1 and L2 caches of ARM and 8 KB of instruction and data caches in BRAMs for MB</td></tr>
<tr><td colspan="2">40 Kb shared in OCM</td></tr>
<tr><td>Peripheral resources</td><td colspan="6">2 AXI interconnect buses, 1 AXI bus split, 1 AXI timer, 1 clock generator, 1 reset system, 2 high performance ports, 1 general purpose port</td></tr>
<tr><td>Area of the peripherals resources</td><td colspan="6">3337 LUTs</td></tr>
<tr><td>Execution time</td><td>3.06 ms</td><td>18.2 ms</td><td>3.06 ms</td><td>18.2 ms</td><td>15.5 ms</td><td>5.09 ms</td></tr>
</table>

Figure 7.2 – Matrix multiplication algorithm.

```
setup_system()
loop
  A = generate_random_matrix()
  B = generate_random_matrix()
  C_golden = A * B
  CRC_golden = calculate_crc(C_golden)
  run = 0
  loop
    C = A * B // main operation
    CRC = calculate_crc(C)
    if CRC = CRC_golden then
      if run <500 then
        run = run + 1
      else
        run = 0
        print(success)
      end if
    else
      print(fail)
    end if
  end loop
end loop
```

Experiments were carried out at Los Alamos National Laboratory's (LANL) Los Alamos Neutron Science Center (LANSCE) Irradiation of Chips and Electronics House II, Los Alamos, USA. As mentioned in (VIOLANTE et al., 2011), LANSCE provides a white neutron source that emulates the energy spectrum of the atmospheric neutron flux. The neutron flux was approximately $1x10^6$ $n.cm^{-2}.s^{-1}$) for energies above 10 MeV. The beam was focused on a spot with a diameter of 2 inches plus 1 inch of penumbra, which provided uniform irradiation of the APSoC chip without directly affecting nearby board power control circuitry. Irradiation was performed at room temperature with normal incidence and nominal voltages. Each approach was executed for more than 6 hours under the beam, each receiving a fluence of at least $3x10^8$ $n.cm^{-2}$.

A test monitor application running on a monitor computer was responsible for collecting and time-stamping incoming logs from the DUT through UART connections. In case of error in the ARM processor, the processor is reset. In case of error in the MB processor, first a readback of the configuration memory is performed and then the processor is reset. The test monitor application also served as a watchdog for irrecoverable situations, such as SEFIs, detecting when the DUT exceeded a timeout larger than the application execution

time without sending executions logs. In such cases, the timeout was time-stamped and logged, and the board was rebooted.

Table 7.2 summarizes the obtained experimental results during the neutrons radiation test campaign. One can notice that the presented cross section values refer to SDC and SEFI errors, and not to permanent errors in the PL. As already mentioned and shown, results show that for APSoCs systems, the cross section measurement itself is also not enough to characterize the reliability of the system. It is also fundamental to take into account also the system execution time and data workload to have a better picture of the sensitivity of the system under upsets.

Table 7.2 – Experimental results from the neutron radiation tests for the four case-studied system in Zynq-7000.

	Case A		**Case B**		**Case C**	**Case D**
	ARM	MB	ARM	MB	ARM+MB	MB+ARM
Cross section (cm²)	$1.01x10^{-9}$	$4.60x10^{-8}$	$9.00x10^{-10}$	$1.50x10^{-9}$	$3.43x10^{-10}$	$4.23x10^{-8}$
MTBF (hours)	$7.62x10^{7}$	$1.67x10^{6}$	$8.55x10^{7}$	$5.13x10^{7}$	$2.24x10^{8}$	$1.82x10^{6}$
MEBF (executions)	$1.78x10^{11}$	$6.59x10^{8}$	$2.00x10^{11}$	$2.02x10^{10}$	$1.04x10^{11}$	$2.22x10^{9}$
MWBF (data)	$1.11x10^{14}$	$4.12x10^{11}$	$1.25x10^{14}$	$1.26x10^{13}$	$6.50x10^{13}$	$1.39x10^{12}$

As a first analysis it is investigated the impact of sharing memory between the ARM and MB processors, looking at implementations in *Case A* and in *Case B*. Notice that the measured cross section of ARM in *Case A* is $1.01x10^{-9}$ cm² compared to $9.00x10^{-10}$ cm² of *Case B*. ARM has shown a small improvement in cross section when the memory space of the systems is reduced (which means that the sensitive area and, thus, its cross section, is also reduced). However, comparing the metrics of MEBF and MWBF, both implementation cases *A* and *B* of ARM present very similar results, probably due to the same execution times of the applications.

With regards to the Microblaze implementations in cases *A* and *B*, one can notice that the measured cross section of Microblaze in *Case A* is $4.60x10^{-8}$ cm² compared to $1.50x10^{-9}$ cm² of *Case B*. Microblaze also presented an improvement in cross section when the memory space of the systems is reduced (i.e. sensitive area). Now comparing the metric MEBF and MWBF, *Case B* also present a significant improvement in two orders of magnitude due to its implementation. Such improvement probably happened because the first part of the matrix

multiplication algorithm, the generation of the matrices, is not performed on the Microblaze, but on the ARM processor. Thus, beyond the reduction of the sensitive area by sharing part of the memory, sharing part of the memory also reduced the exposure time of the algorithm running in the Microblaze.

When comparing soft- and hard-core processors, ARM processor shows a smaller cross section compared to Microblaze in both cases *A* and *B*. This result was already expected, since hard-core processors are known to be less sensitive to radiation than soft-core processors embedded in FPGAs (KASTENSMIDT, CARRO, REIS, 2006). Interesting results were obtained by analyzing the MEBF and MWBF values of the processors of *Case B*. For sake of comparison, in *Case A*, the difference between processors in terms of MWBF was of about three orders of magnitude. However, in *Case B*, this difference was reduced to one order of magnitude even with both cases having the same execution time. This result was also achieved by reducing the exposure time of the algorithm running in the Microblaze processor, which resulted in less failures and more workload correctly processed, consequently.

Comparing single-core to multi-core, one can observe that *Case C*, where Microblaze processor performs only the matrix multiplication operation and ARM processor performs all other operations, has a lower cross section to Case A-ARM and a similar one to *Case B-ARM*, where the hard-core ARM processor performs all the matrix multiplication operation, even *Case C* presenting very different implementation areas than cases *A-ARM* and *B-ARM*. Concerning MWBF, all implementations have similar values, even with their very distinct execution times. It is also interesting to notice that there were practically no changes in the MEBF values in all three cases. These results show the advantages of using heterogeneous systems where the ARM processor is the main processor and a hardware embedded in the FPGA as a secondary processor, because even with a significant overhead in area and time, the obtained results are similar or even better than using a single-core processor. Additionally, having a longer execution time in *Case C* may bring benefits to the system, because it enables the master processor to perform other tasks while waiting for the slave processor.

When comparing *Case D* with cases *A-MB* and *B-MB*, where ARM processor performs only the matrix multiplication operation and Microblaze processor performs all other operations, *Case D* presented intermediate values for both cross section, MEBF, and MWBF. These results are similar to what were obtained for the ARM processor cases.

Now, comparing both heterogeneous cases *C* and *D*, it is evident that ARM plays a major role in terms of reliability, because even with an execution time three times worse than *Case D*, *Case C* presented better results in all metrics. This result is of extreme importance,

because if the soft-core processor is changed by a dedicated hardware in the FPGA with a better execution time than the ARM processor, the results should be even better.

7.2 Reliability of heterogeneous processing through hardware and software co-designs in the Zynq-7000

This section aims at complementing the previous one by performing a reliability and performance analysis of heterogeneous processing through hardware and software co-designs with different workload distributions between hardware and software. The novelty of this section is that now the PL part has embedded dedicated hardware cores instead of a soft-core processor, which implies in smaller execution times compared to the PS part.

7.2.1 Case-study designs and resources and performance evaluations

Based on Section 6.2, two benchmark algorithms were selected for this study: a data flow oriented 32x32 floating-point Matrix Multiplication (MxM) (XILINX, 2016a) and a control flow oriented 128-bit Advanced Encryption Standard (AES) (HARA et al., 2008). The selected output size (workload) is 32,768 bits (1024 values of 32-bit each) for the MxM and 1024 bits (32 values of 32-bit each) for the AES.

Aiming at generating different hardware and software co-designs, both algorithms were partitioned having portions running on the PS, i.e., one ARM Cortex-A9 core (hereafter shortened to only "processor") and portions implemented in the PL, used in the form of a hardware accelerator. MxM algorithm was partitioned in two sets (*set1* and *set2*) of matrix multiplication operations, while AES algorithm was partitioned with respect to its encode (*enc*) and decode (*dec*) functions. The AES algorithm is particularly interesting because the *enc* function takes 10 times more time than the *dec* function. Fig. 7.3 shows a generic representation of the case-study benchmark algorithms.

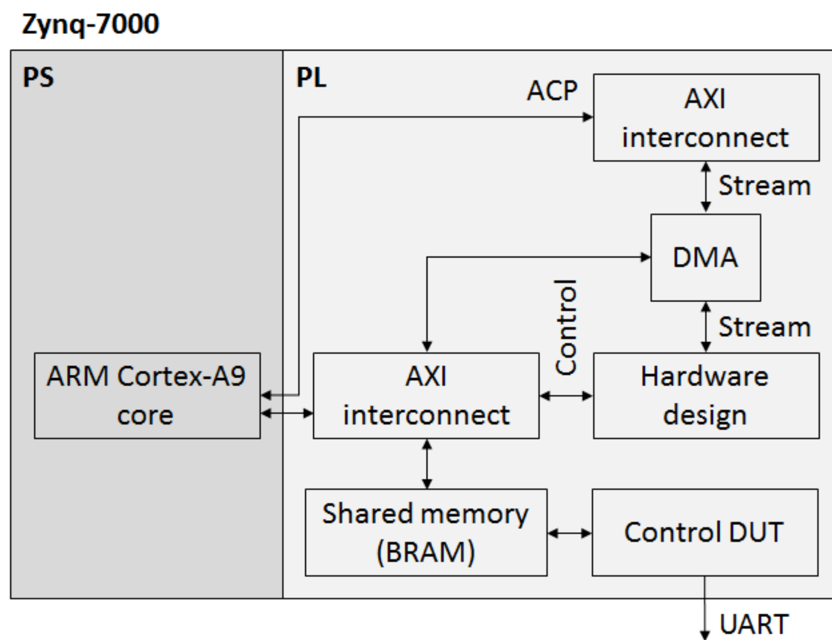
Figure 7.4 shows the architecture developed for evaluating the case-study designs. The data transfer between the processor (PS) and the hardware design implemented in the PL part is performed through the processor's Accelerator Coherency Port (ACP) and a BRAM protected with ECC. Besides the processor, the hardware design, the shared memory, and the peripherals needed to connect them, the architecture also has embedded a hardware block

(*Control DUT*) protected with TMR that monitors the system during the radiation experiments and sends diagnostic data to a computer.

Figure 7.3 – Generic representation of the case-study benchmark algorithms.

// SOFTWARE-ONLY VERSION	*// HARDWARE-ONLY VERSION*
setup_system()	*setup_system()*
generate_golden_values_set1()	*generate_golden_values_set1()*
generate_golden_values_set2()	*generate_golden_values_set2()*
loop	***loop***
wait_start_from_control_dut()	*wait_start_from_control_dut()*
run_set1()	*send_input_a_to_hardware()*
run_set2()	*send input_b_to_hardware()*
send_done_to_control_dut()	*run_hardware()*
end loop	**wait until** *hardware_is_busy()*
	send_done_to_control_dut()
	end loop

Figure 7.4 – Block diagram of the architecture developed for evaluating the case-study hardware and software co-designs in Zynq-7000.



The use of the processor's L1 cache is also considered. L2 cache was left disabled since its addition to the memory hierarchy severely affects the cross section of the processor, as observed in Section 5.2. The hardware designs in the PL were generated by High-Level Synthesis (HLS) directly from the original C language code with the Xilinx Vivado HLS tool.

Two different versions were generated for each set, one without any performance optimization and other highly optimized for performance, as investigated in Section 6.2.

All the case-study designs are presented in Table 7.3. They are classified into workload distribution (how much of the algorithm is implemented in software and hardware), design configuration of the PS (L1 cache disabled or not), and the HLS-based hardware design (with or without optimization). The resources are given in terms of number of processors, LUTs, flip-flops, DSPs, BRAMs, and essential bits (PL configurations bits associated with the circuitry of the design). The performance is presented as number clock cycles. All hardware and software co-designs operate in the PS part at 667 MHz and in the PL part at 100 MHz.

Data resources show that the more performance-optimized a hardware accelerator is, the more resources it uses (2.31 times for the MxM and 3.15 times for the AES in the best cases). However, the number of clock cycles are significantly reduced (83.18 times for MxM and 94.69 times for AES). This means that offloading the entire algorithm for the PL aiming to increase its parallelism at hardware level greatly increases the workload capability of the system. The best hardware and software co-designs in terms of acceleration for the MxM and AES are 10 and 14, when considering using cache L1, otherwise are versions 09 and 13, respectively. However, comparing the designs with distributed workload with the ones that run entirely in the PL, one can observe that the inherent sequential processing of the processor severely affects the execution time, mostly in the MxM algorithm, which is very data flow oriented. In addition, comparing the essential bits data with the obtained execution times, one can also observe that it is worth offloading the entire algorithm to the PL instead of distribute it between PS and PL.

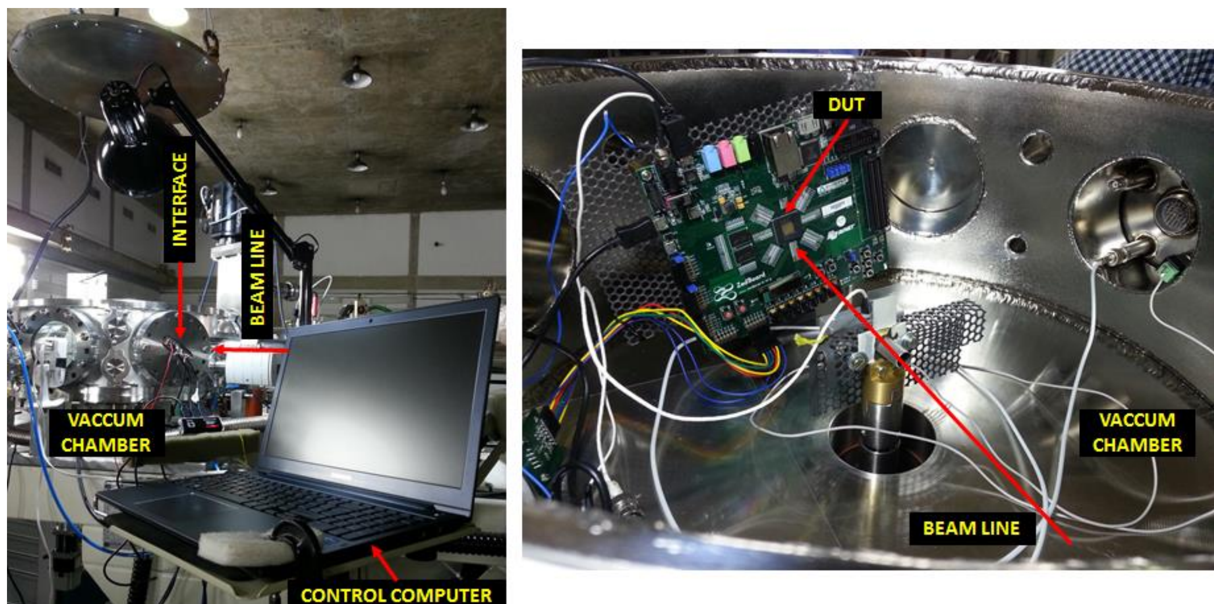
Table 7.3 – Case-study hardware and software co-designs and their respective configurations, resource usage, and performance results.

Design version	Workload distribution		Design configuration		Resources						Performance
	PS	PL	PS	PL	# Processor	# LUT	# FF	# DSP	# BRAM	# Essential bits	# Clock cycles
MxM (set1, set2)											
01	set1+set2	-	L1 off	-	1	6,017	7,531	0	20	1,490,755	5,846,209
02	set1+set2	-	L1 on	-	1 + L1 cache	6,017	7,531	0	20	1,490,755	2,412,012
03	-	set1+set2	L1 off	No opt.	1	9,687	12,397	6	27	2,472,279	616,650
04	-	set1+set2	L1 on	No opt.	1 + L1 cache	9,687	12,397	6	27	2,472,279	615,138
05	-	set1+set2	L1 off	Max. opt.	1	10,920	15,236	84	27	3,453,515	71,700
06	-	set1+set2	L1 on	Max. opt.	1 + L1 cache	10,920	15,236	84	27	3,453,515	70,280
07	set1	set2	L1 off	No opt.	1	9,521	12,034	3	26	2,427,779	3,225,653
08	set1	set2	L1 on	No opt.	1 + L1 cache	9,521	12,034	3	26	2,427,779	1,485,882
09	set1	set2	L1 off	Max. opt.	1	10,468	13,829	96	56	3,017,275	2,965,633
10	set1	set2	L1 on	Max. opt.	1 + L1 cache	10,468	13,829	96	56	3,017,275	1,189,736
AES (enc, dec)											
01	enc+dec	-	L1 off	-	1	6,010	7,572	0	20	1,505,479	216,393
02	enc+dec	-	L1 on	-	1 + L1 cache	6,010	7,572	0	20	1,505,479	16,642
03	-	enc+dec	L1 off	No opt.	1	11,178	14,154	0	31.5	2,745,437	4,556
04	-	enc+dec	L1 on	No opt.	1 + L1 cache	11,178	14,154	0	31.5	2,745,437	4,181
05	-	enc+dec	L1 off	Max. opt.	1	19,791	23,460	0	30.5	4,755,736	2,663
06	-	enc+dec	L1 on	Max. opt.	1 + L1 cache	19,791	23,460	0	30.5	4,755,736	2,291
07	dec	enc	L1 off	No opt.	1	10,364	13,097	0	27	2,558,713	134,430
08	dec	enc	L1 on	No opt.	1 + L1 cache	10,364	13,097	0	27	2,558,713	12,481
09	dec	enc	L1 off	Max. opt.	1	16,813	19,718	0	27	3,154,647	132,939
10	dec	enc	L1 on	Max. opt.	1 + L1 cache	16,813	19,718	0	27	3,154,647	11,877
11	enc	dec	L1 off	No opt.	1	10,491	13,174	0	27	2,554,304	77,844
12	enc	dec	L1 on	No opt.	1 + L1 cache	10,491	13,174	0	27	2,554,304	8,892
13	enc	dec	L1 off	Max. opt.	1	18,013	21,125	0	27	4,263,214	76,493
14	enc	dec	L1 on	Max. opt.	1 + L1 cache	18,013	21,125	0	27	4,263,214	7,553

7.2.2 Cross section, MWBF, and performance evaluation

Radiation experiments were also carried out with heavy ions at Laboratório Aberto de Física Nuclear at Universidade de São Paulo (LAFN-USP), Brazil. The ion beams were produced and accelerated by the São Paulo 8UD Pelletron Accelerator, but now in a new beamline (AGUIAR et al., 2017). This new beamline is used to transport very low-flux, high uniformity, and large area heavy ion beams up to ^{28}Si by defocusing and multiple scattering techniques using two gold foils and from ^{35}Cl to ^{107}Ag by a defocusing technique. These characteristics are required by the European standards (CARLIN et al., 2005) for radiation testing of electronic devices and to test the radiation hardness of devices to SEEs. The SEU events were observed using a ^{16}O beam, with a primary energy of 39 MeV, which provides an effective LET on the active region of about 5.5 MeV/mg/cm² and penetration in Si of about 24 µm. The average beam flux was maintained between 1.0×10^2 and 1.0×10^3 particles.cm^{-2}.s^{-1} in a 4.0 cm² beam area with an uniformity of about 93%.

Figure 7.5 - Heavy ion experiment setup mounted at the new beam line of the LAFN-USP.



A host computer monitored the experiments through a serial interface. If there were no differences in the results, the *Control DUT* sent an alive signal to the computer at each execution set. Otherwise, if differences are found, it sent the number of mismatches to the computer and then the Zynq-7000 is reset. Timeouts in the system were monitored through a watchdog circuit in the *Control DUT*. The host computer also acts as a watchdog to monitor

timeout occurrences in the *Control DUT*. In case of timeout, the Zynq-7000 is reset. Each experiment run was set by the total number of errors observed (50) at the system output. All other uncertain errors observed are considered in the errors bars. For the experiments, the package of an XC7Z020-CLG484 device was thinned to allow irradiated particles to penetrate the active region of the silicon. Results are presented in terms of total errors, which consider SDC errors (data errors detected by the application without system interruption) and SEFI errors (system crashes).

The obtained SDC, SEFI, and TOTAL (sum of SDCs and SEFIs) dynamic cross sections for the MxM and AES algorithms are presented in Fig. 7.6 and 7.7. With regard to the PS-only designs (design versions 01 and 02), one can observe that enabling the L1 cache barely affects the cross section (difference of 28% and 2% for the MxM and AES, respectively). The nature of the algorithm (data or control flow oriented) and its execution time seems to directly affect the cross section of the system. With regard to the designs in which the entire algorithm is offloaded to the PL (design versions 03, 04, 05, and 06), one can observe that, for both algorithms, the cross section of the system is directly proportional to the PL's resources usage. When there is hardware and software co-designs, one can observe that the lowest cross sections were achieved by enabling the L1 cache in the PS and using a hardware design without optimization in the PL (design version 08). This configuration is particularly interesting because it helps reducing the exposure time by enabling the L1 cache and the sensitive area by using a hardware design that occupies a smaller area. Moreover, one can observe that in the AES, the best cross section results were achieved by offloading the most time-consuming set (*enc*) to the PL (design versions 07, 08, 09, and 10). Note that for hardware and software co-designs, the interface and the amount of data exchange between PS and PL (and vice versa) play an important hole in the susceptibility too, explaining the average increase in cross section.

The obtained MWBFs for the MxM and AES algorithms are presented in Fig. 7.8 and 7.9. For both algorithms, the MWBF increase follows the execution time improvement, which in turn is obtained by offloading the workload to the PL and enabling the processor's L1 cache.

Figure 7.6 – SDC, SEFI, and TOTAL dynamic cross section results obtained for the MxM designs from radiation experiments.

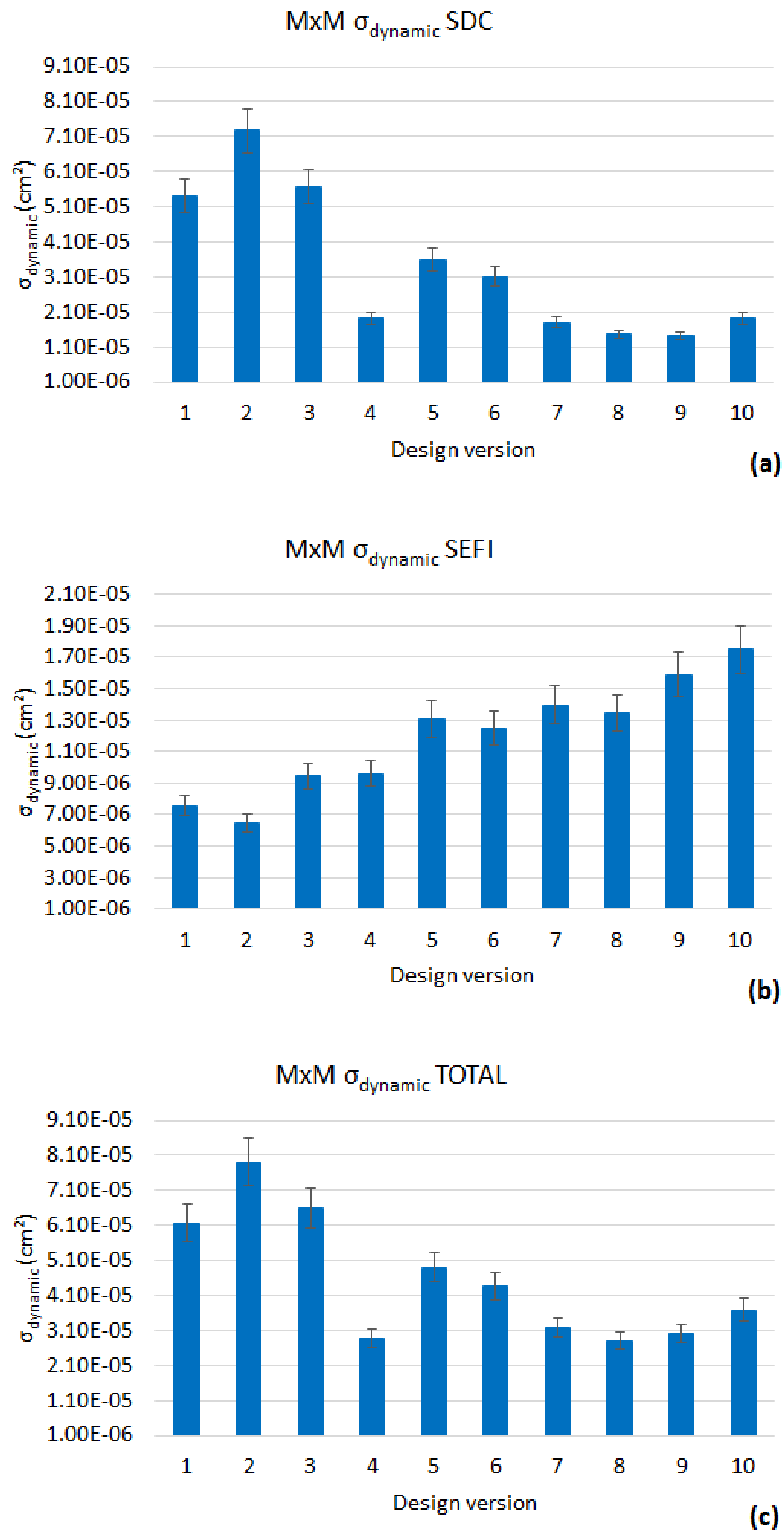


Figure 7.7 – SDC, SEFI, and TOTAL dynamic cross section results obtained for the AES designs from radiation experiments.

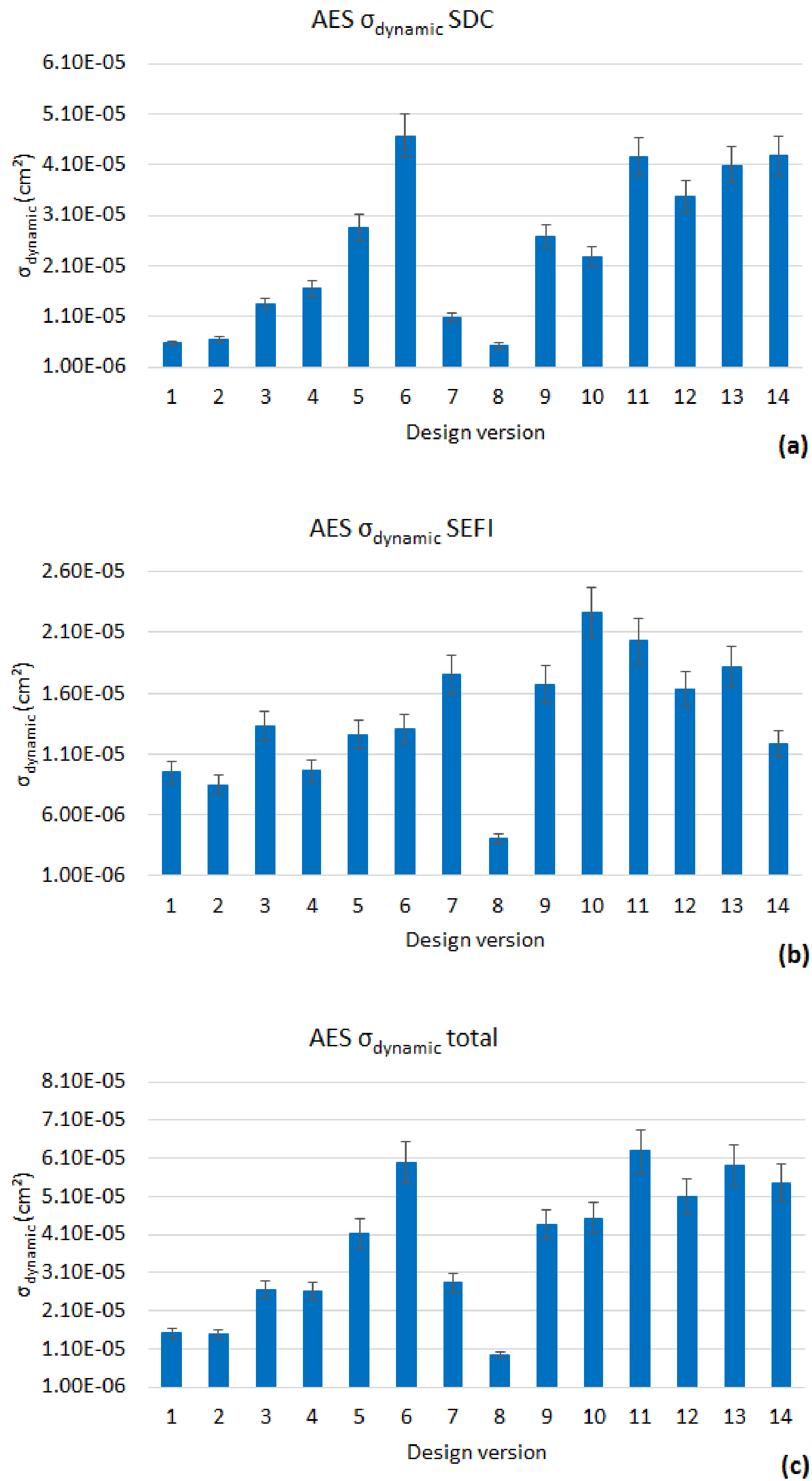


Figure 7.8 – SDC, SEFI, and TOTAL MWBF results obtained for the MxM designs from radiation experiments.

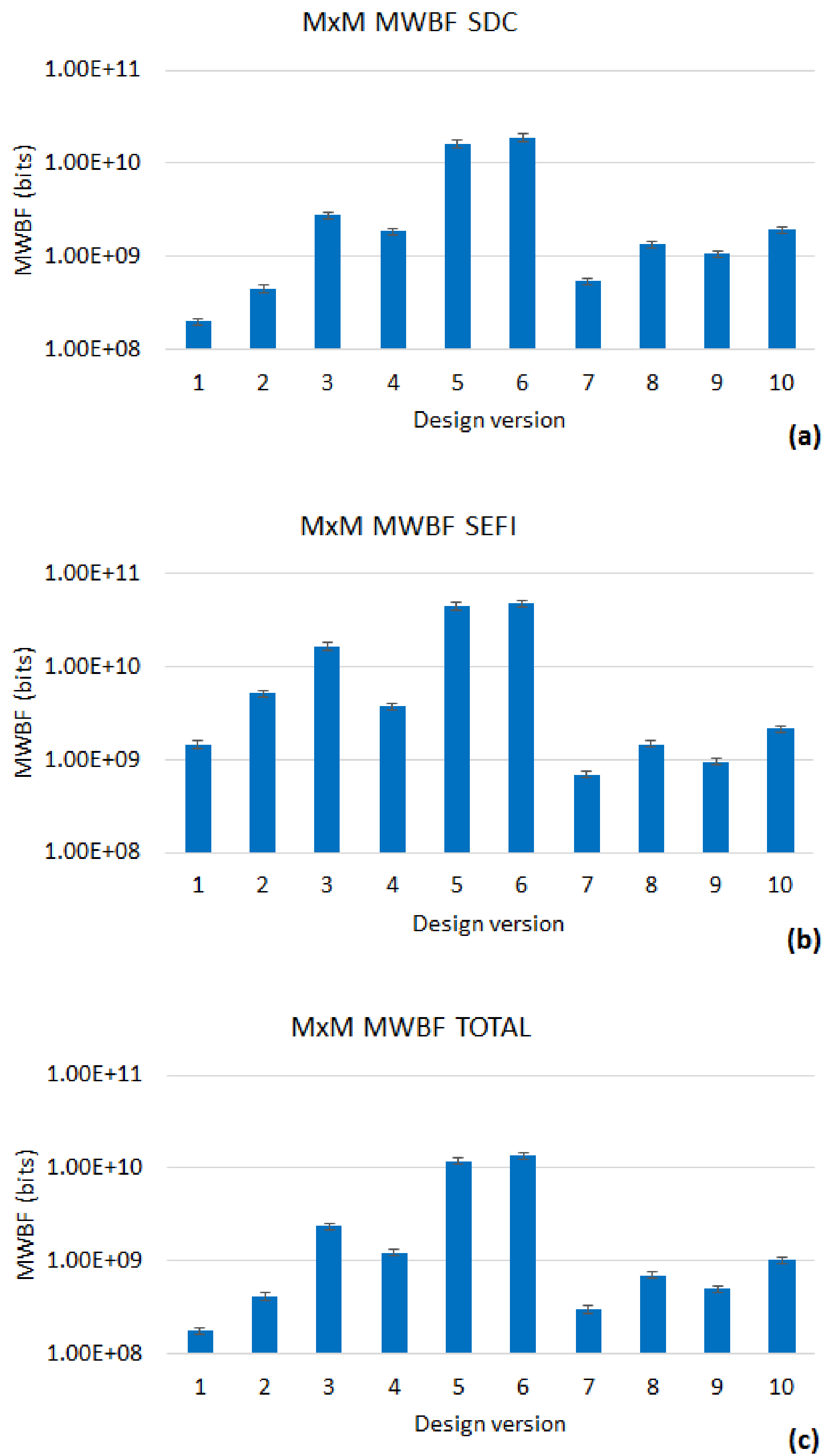
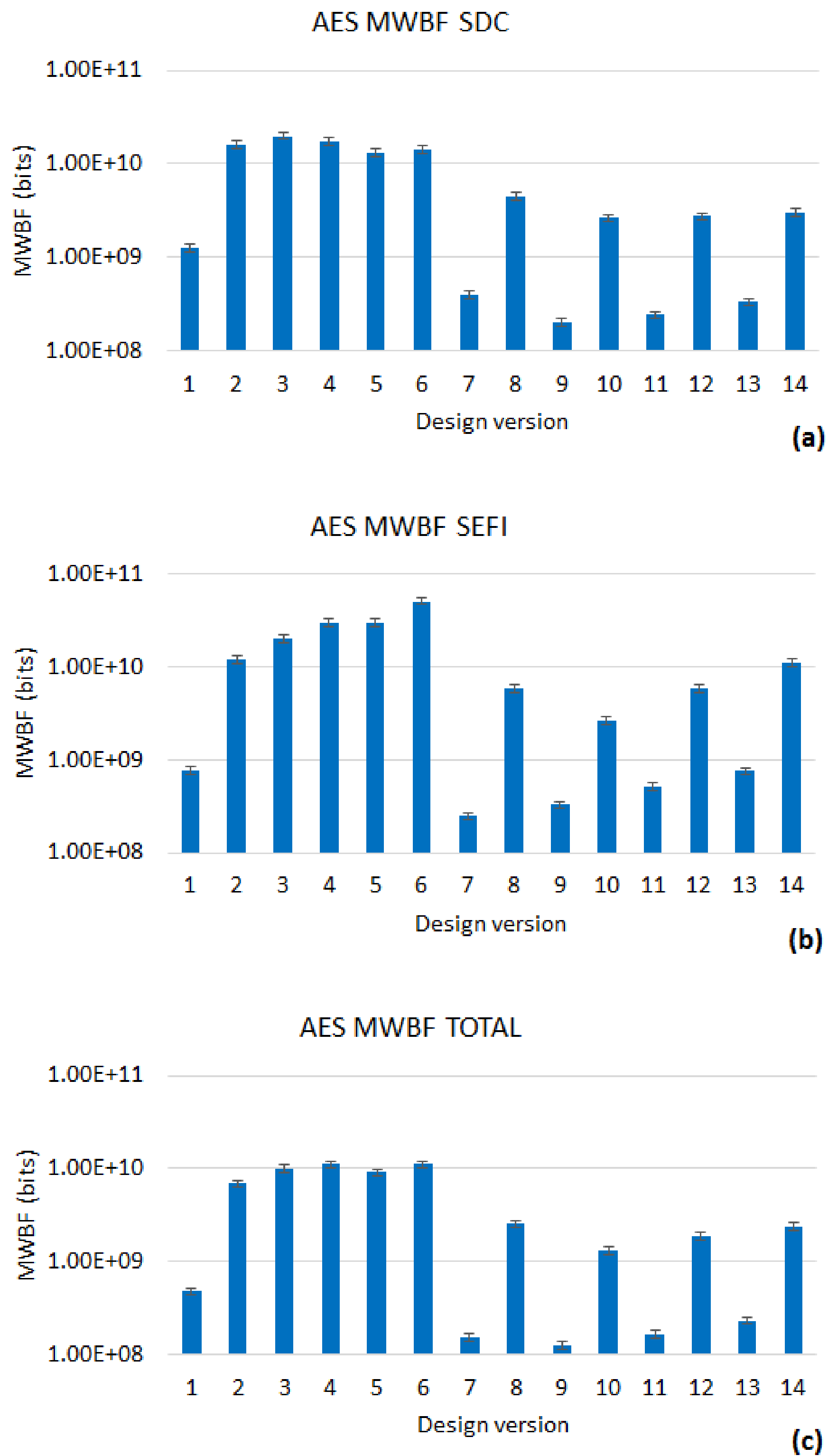


Figure 7.9 – SDC, SEFI, and TOTAL MWBF results obtained for the AES designs from radiation experiments.

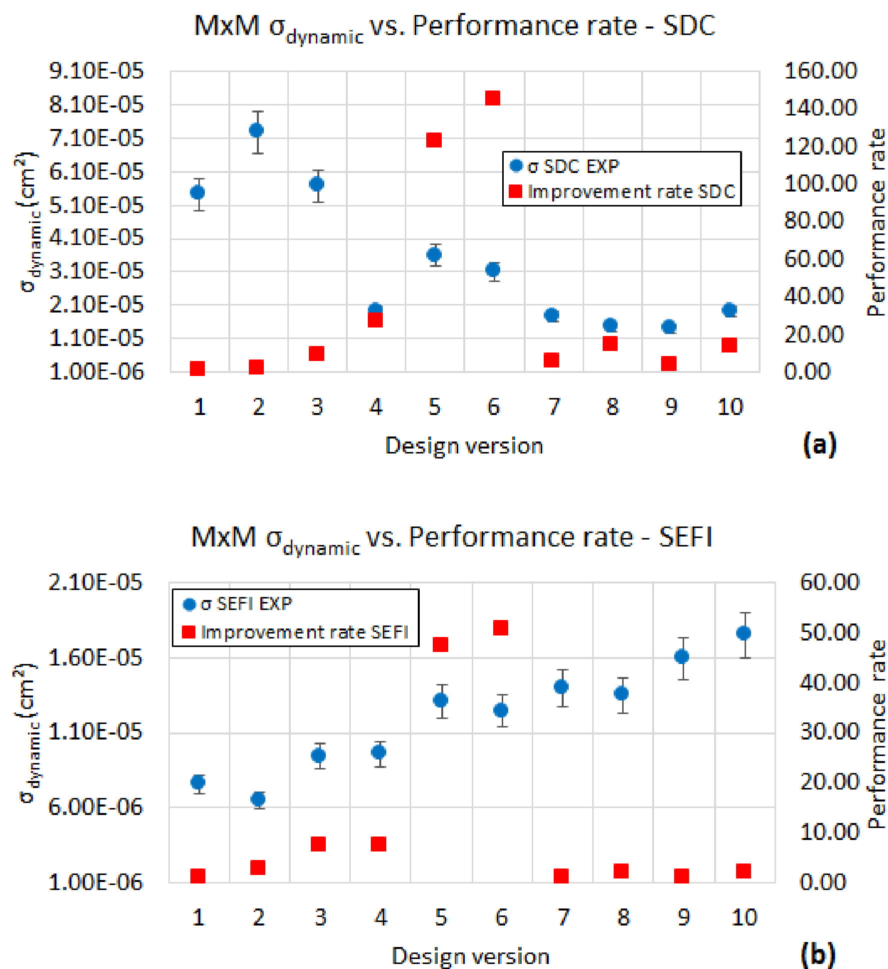


A more direct comparison between the obtained dynamic cross sections and the performance of the designs can be performed through Eq. 6.2. As stated, the performance rate

relation states that whenever the speed-up is higher than the increase in cross section, the faster configuration computes a larger workload before experiencing a failure and, thus, the operational reliability of the system is higher.

Fig. 7.10 and 7.11 compare the obtained dynamic cross section and performance rate for the MxM and AES, respectively. With regard to the MxM algorithm, one can observe that the designs with the best performance rates are versions 05 and 06 due to their significant improvement in execution time. With regard to the AES algorithm, one can observe that the designs with the best performance rates are versions 02, 03, 04, 05, and 08. Such result reveals that designs with very different architectures may present similar results due to a smaller cross section (design version 08) or execution time (design version 05).

Figure 7.10 – Comparison between SDC, SEFI, and TOTAL dynamic cross sections and their respective performance rates for the MxM designs from radiation experiment results.



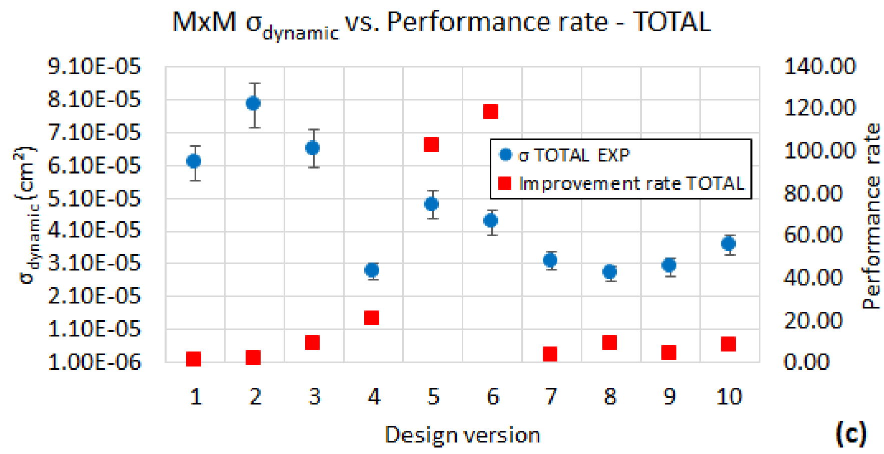
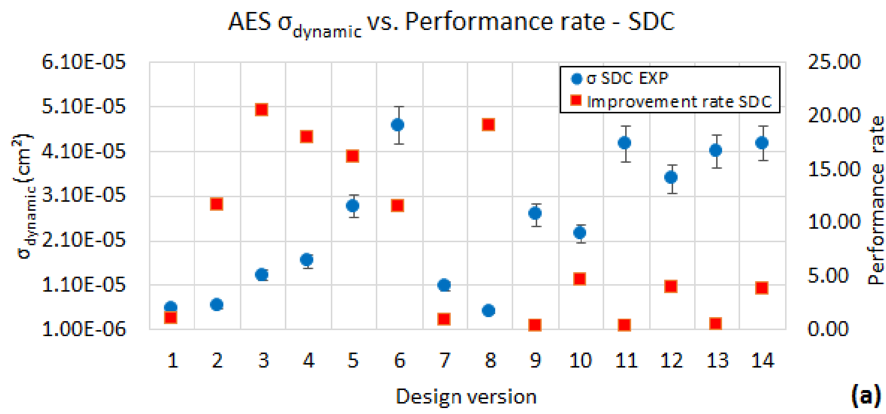
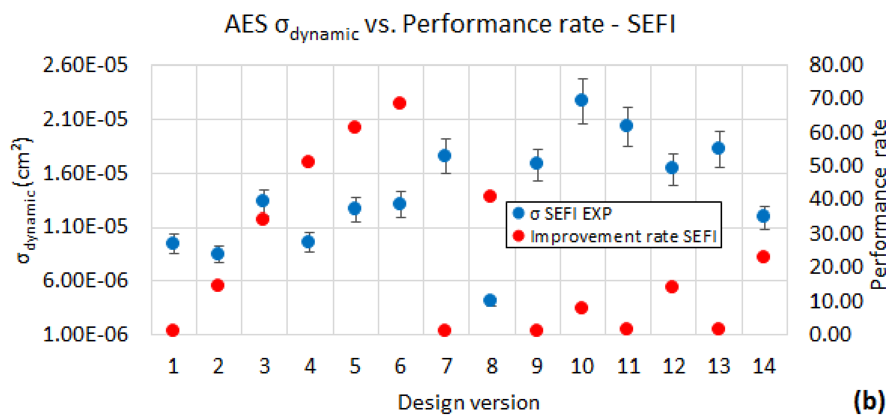
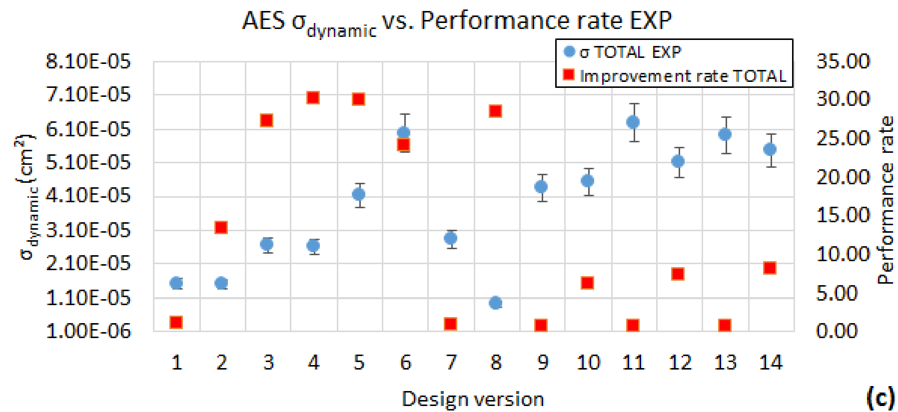


Figure 7.11 – Comparison between SDC, SEFI, and TOTAL dynamic cross sections and their respective performance rates for the AES designs from radiation experiment results.







7.3 Summary

The reliability of systems based on hardware and software co-design seems to be inversely proportional not only to the device sensitivity, but also to the total exposure time. Thus, the next chapter aims confirming such statement and the obtained results through several fault injection campaigns in both PS and PL parts of Zynq-7000. As final a result of this thesis, the next chapter also expands the proposed flow in Section 6.2.1 for trying to estimate the reliability trend prior to radiation experiments of not only hardware-only designs, but also software-only designs and hardware and software co-designs.

8 PROPOSED RELIABILITY ANALYSIS FOR HARDWARE AND SOFTWARE CO-DESIGNS

The reliability prediction of system designs through fault injection by emulation is a complex task due to several factors. In FPGAs (PL parts), in general, data from fault injection are more pessimistic than the ones from the radiation experiments, as presented in Section 6.2.3. Such behavior is mainly guided by the fact that fault injection methodologies are more deterministic and give a very fine fault granularity, enabling to estimate the worst-case cross section of a design without considering side-effects, such as particle beam variations and fault masking by the design. Therefore, it is possible to determine precisely the exact number of bits that are critical and consider them in the estimated dynamic cross section. It is also worth noting that, in radiation experiments, the particle flux covers the whole architecture, while fault injection campaigns may be constrained to only the DUT. However, in COTS processors (PS parts), the fault injection behavior is generally the opposite in comparison with radiation experiments. This happens mainly because the end-users have no access to all resources of a COTS processor, which limits fault injection platforms to only a subset of available resources, such as the general-purpose registers and embedded memories, while ionizing particles are capable of hit on any resource of the device. Another issue is related to the fact that, in COTS processors, faults are injected more at software level than at hardware level, which makes unfeasible the estimation of the dynamic cross section of a software, since the cross section metric is expressed in are units (cm^2).

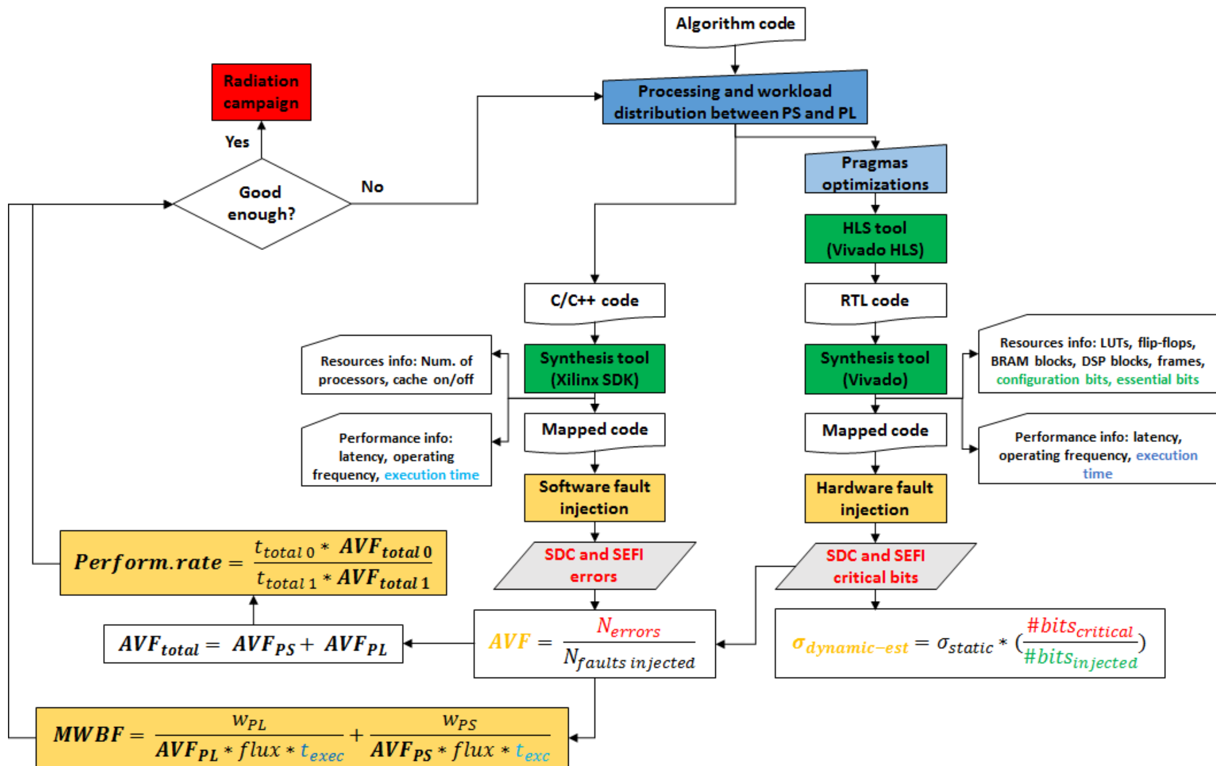
In this context, since is very complex to precisely estimate in numbers the behavior of a system running on a COTS device under radiation, this chapter aims to expand the analysis flow proposed in Section 6.2.1 for estimating also the reliability trend prior to radiation experiments of not only hardware-only designs, but also hardware and software co-designs and software-only designs.

8.1 Analysis flow

Fig. 8.1 summarizes the proposed reliability analysis in a flow diagram. The hardware flow concepts are similar to the ones presented in Section 6.2.1. However, a parallel software flow was added to the main one for also considering software-only designs and hardware and

software co-designs. Moreover, additional metrics were adopted to make possible the merge between the hardware and software fault injection results.

Figure 8.1 – Proposed reliability analysis flow for hardware and software co-designs.



The input of the flow must be an algorithm written in C or C++ programming languages so that they can be synthesized by the synthesis tools adopted. The first step is the division of the algorithm source code for distributing the processing and the workload between PS and PL parts.

The second step is the setup of the hardware and software parts. The portion of the code that will run on the PL has to be configured with pragmas optimizations to guide the HLS tool (Xilinx Vivado HLS) during the generation of the RTL code. The remaining portion of the code, the one that will run on the PS part, must be adapted so that the software can interface with the hardware design in the PL.

The third step is the synthesis of each portion by the synthesis tools. Xilinx Vivado Design Suite and Xilinx SDK are used for generating the hardware and the software, respectively. Information about the usage of resources and performance are collected at this stage.

The fourth step consists in carrying out the fault injection campaigns in both hardware and software (or only in one of them, in case of standalone hardware or software designs).

The fault injection platforms adopted are the ones previously described in the sections 4.2.1 and 4.2.2. It is worth mentioning that the platforms are not integrated, thus fault injection campaigns in hardware and software must be performed independently, which can affect the final result. In addition, the software fault injection injects faults only in the processor's register file. However, this limitation is minimized by the fact that every data must be loaded in registers to be processed. Concerning the size of a fault injection campaign, the hardware fault injection is limited by the number of configuration bits inside the DUT area, while the software fault injection is limited by the number of faults to be injected, which is defined by the user. This means that the higher the number of faults to be injected in software, the higher the confidence of the results. The results of this step are the numbers of SDC and SEFI errors in each part of the design.

The fifth step consists in calculating the chosen reliability metrics. With regard to the hardware flow, the estimation of the dynamic cross section is still possible, since the methodology of analysis is the same of Section 6.2.1. However, this method is not applicable for software designs for two main reasons. First, it is practically impossible for an end-user obtaining the static cross section of a COTS processor, since it does not have access to the entire low-level processor hardware. Moreover, COTS processor tests are done with specific softwares, which means that they are not static and thus the resulting cross sections differ according to the software in use. Second, since the static cross section of the processor is not available and the faults are injected in a software context in the software designs (PS part), it is more meaningful to evaluate the probability that a fault in the system will result in an error in its final output than in terms of sensitive area. Therefore, AVF was chosen as the main resulting metric from the fault injection campaigns of the proposed reliability analysis.

The total AVF (AVF_{total}) of a hardware and software co-design is obtained by adding the hardware AVF (AVF_{PL}) and the software AVF (AVF_{PS}) resulting from the hardware and software fault injection campaigns, respectively. However, as well as previously observed for the cross section, the AVF also does not give information about the performance efficiency and operational reliability of the system. Therefore, MWBF was chosen as the main resulting metric of the proposed reliability analysis.

The total MWBF ($MWBF_{total}$) of a hardware and software co-design is obtained by adding the hardware MWBF ($MWBF_{PL}$) and the software MWBF ($MWBF_{PS}$). It is worth mentioning that, since the AVF was chosen as the main resulting metric from the fault injections in the proposed reliability analysis, it replaced the cross section in the MWBF equation (Eq. 4.10), as shown in Fig. 8.1. As consequence, the MWBF values from fault

injections and radiation experiments will present differences of magnitudes when compared. However, the proposed reliability analysis flow aims to estimate the reliability trend of systems designs prior to radiation experiments and not the exact values to be obtained from them.

The proposed reliability analysis flow also considers the performance rate metric previously introduced in Section 6.2.3 for comparing directly two designs in terms of execution time and reliability (AVF). As in the MWBF, the cross section was replaced by the AVF in the performance rate equation (Eq. 6.2), as shown in Fig. 8.1. The performance rate relation states that, whenever the speed-up is higher than the increase in cross section ($\frac{t_{total\,0}}{t_{total\,1}} > \frac{AVF_{total\,1}}{AVF_{total\,0}}$), the faster configuration computes a larger workload before experiencing a failure and, thus, the operational reliability of the system is higher.

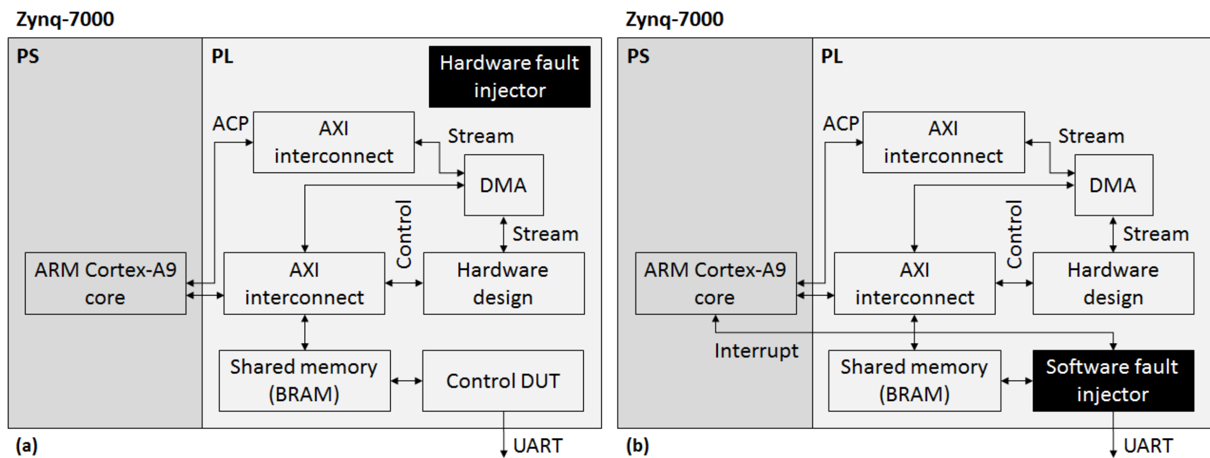
8.2 Case-study designs

The case-study designs chosen for evaluating the proposed reliability analysis flow are the same of Section 7.2.1. The benchmark algorithms chosen are a data flow oriented 32x32 floating-point Matrix Multiplication (MxM) (XILINX, 2016a) and a control flow oriented 128-bit Advanced Encryption Standard (AES) (HARA et al., 2008). The selected output size (workload) is 32,768 bits (1024 values of 32-bit each) for the MxM and 1024 bits (32 values of 32-bit each) for the AES.

Aiming at generating different hardware and software co-designs, both algorithms were partitioned having portions running on the PS, i.e., one ARM Cortex-A9 core (hereafter shortened to only "processor") and portions implemented in the PL, used in the form of a hardware accelerator. MxM algorithm was partitioned in two sets (*set1* and *set2*) of matrix multiplication operations, resulting in ten different design versions. AES algorithm was partitioned with respect to its encode (*enc*) and decode (*dec*) functions, resulting in fourteen different design versions. As in the other chapters, the target device is the Xilinx Zynq-7000 APSoC.

With regard to the fault injection campaigns, the respective fault injection hardware blocks were added for injecting faults in the PS and PL parts, as illustrated in Fig. 8.2. However, they are not considered in any measurement, such as resource usage and performance.

Figure 8.1 – Block diagrams of the architectures developed with the hardware (a) and software (b) fault injectors embedded.



8.3 Obtained results

Table 8.1 presents the number of critical bits and AVF of the hardware part of each case-study design obtained from the hardware fault injection campaigns. Each hardware fault injection campaign was divided into two steps. The first step consisted of injecting faults only in the DUT area. The second step consisted of injecting faults in the peripheral hardware blocks, such as the AXI interconnects and DMA. It is worth noting that all the designs have the same peripheral blocks, which in somehow normalizes them. Moreover, since the design versions 01 and 02 do not have a hardware accelerator, the critical bits and AVF values presented in Table 8.1 are related to the peripheral blocks. These values are already included in the results of the other designs, which have a hardware accelerator beyond the peripheral blocks.

Comparing the obtained results from fault injection with the number of configuration bits used in the PL, one can observe that a small fraction of the configuration bits (from 0.98% to 3.32%) is, in fact, critical. The number of critical bits and, consequently, the AVF (obtained from Eq. 4.7), followed the resource usage increase, as already observed in Section 6.2.2.

Table 8.2 presents the AVF result of the software part of each case-study design obtained from the software fault injection campaigns in the processor’s register file. Aiming to modify as little as possible the algorithms, the software fault injection campaigns focused on injecting faults within their main loop (Fig. 7.3), since it is the most executed part of them

and, thus, the most exposed in a harsh environment. Each fault injection campaign injected 25,000 bit-flips randomly on time and space.

Table 8.1 – Number of critical bits and AVF of the hardware part of each hardware and software co-design obtained from the hardware fault injection campaigns.

Design version	Critical bits			AVF		
	# SDC	# SEFI	# TOTAL (% of config. bits)	SDC	SEFI	TOTAL
MxM						
01	17820	6096	23916 (0.98)	0.01195	0.00409	0.01604
02	17820	6096	23916 (0.98)	0.01195	0.00409	0.01604
03	27615	60395	88010 (2.11)	0.06348	0.18834	0.03560
04	27615	60395	88010 (2.11)	0.00655	0.00995	0.03560
05	52332	74308	126640 (2.28)	0.01117	0.02443	0.03667
06	52332	74308	126640 (2.28)	0.06348	0.18834	0.03667
07	20586	50700	71286 (1.76)	0.00655	0.00995	0.02936
08	20586	50700	71286 (1.76)	0.01117	0.02443	0.02936
09	48451	50359	98810 (1.98)	0.03169	0.04375	0.03275
10	48451	50359	98810 (1.98)	0.00655	0.00995	0.03275
AES						
01	17820	6096	23916 (0.98)	0.01184	0.00405	0.01589
02	17820	6096	23916 (0.98)	0.01184	0.00405	0.01589
03	66726	39343	106069 (2.22)	0.02430	0.01433	0.03863
04	66726	39343	106069 (2.22)	0.02430	0.01433	0.03863
05	193045	73876	266921 (3.32)	0.04059	0.01553	0.05613
06	193045	73876	266921 (3.32)	0.04059	0.01553	0.05613
07	46293	32746	79039 (1.84)	0.01809	0.01280	0.03089
08	46293	32746	79039 (1.84)	0.01809	0.01280	0.03089
09	78210	40830	119040 (2.27)	0.02479	0.01294	0.03773
10	78210	40830	119040 (2.27)	0.02479	0.01294	0.03773
11	45810	32591	78401 (1.82)	0.01793	0.01276	0.03069
12	45810	32591	78401 (1.82)	0.01793	0.01276	0.03069
13	157720	63710	221430 (3.07)	0.03700	0.01494	0.05194
14	157720	63710	221430 (3.07)	0.03700	0.01494	0.05194

Software-only designs (design versions 01 and 02) presented the higher AVF SDC values on both algorithms tested. This behavior is mainly guided by the fact that the entire algorithm is processed on the processor, which increases the susceptibility of its data to SDCs. On the contrary, the software-only versions presented the lower AVF SEFI values on both algorithms tested, probably because the code (Fig. 7.3) for interfacing with the hardware part (hardware accelerator and DMA interface codes), which is quite substantial, is more susceptible to crashes than the original algorithm.

With regard to the designs in which the entire algorithm is offloaded to the hardware part (design versions 03, 04, 05, and 06), results show that the SDC AVF values decreased 11 times in average when compared with the software-only designs. This happens mainly because all the data processing is performed in hardware, which decreases the software susceptibility to SDCs. On the contrary, the AVF SEFI values of the designs increased 7.5

Table 8.2 – AVF of the software part of each hardware and software co-design obtained from the software fault injection campaigns.

Design version	AVF		
	SDC	SEFI	TOTAL
MxM			
01	0.16316	0.04506	0.20822
02	0.17160	0.04503	0.21663
03	0.01140	0.55511	0.56651
04	0.01156	0.55646	0.56802
05	0.01044	0.55586	0.56630
06	0.01036	0.55646	0.56682
07	0.01480	0.61166	0.62646
08	0.01380	0.61174	0.62554
09	0.01420	0.61166	0.62586
10	0.01400	0.61110	0.62510
AES			
01	0.13957	0.07697	0.21654
02	0.13915	0.08093	0.22008
03	0.01740	0.22214	0.23954
04	0.01773	0.23307	0.25081
05	0.01788	0.22250	0.24038
06	0.01656	0.20454	0.22110
07	0.02512	0.24482	0.26995
08	0.02368	0.23366	0.25735
09	0.02400	0.24458	0.26859
10	0.02407	0.26683	0.29090
11	0.02796	0.24746	0.27543
12	0.02727	0.24605	0.27333
13	0.02648	0.24718	0.27367
14	0.02604	0.24722	0.27327

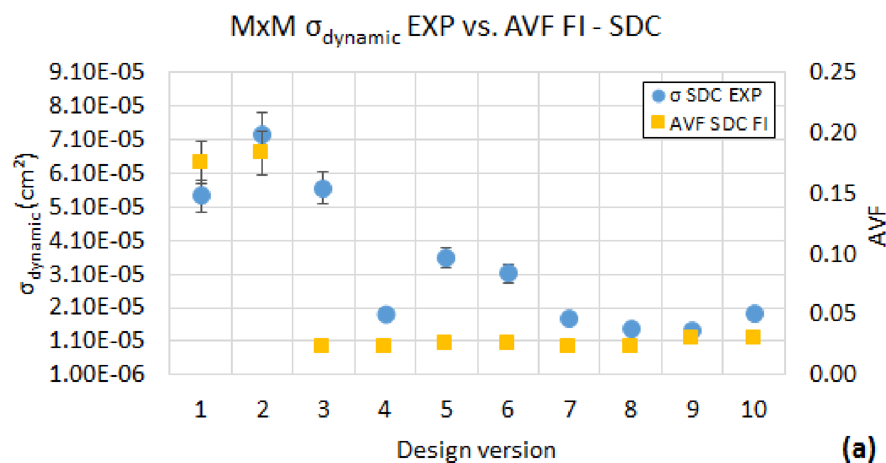
times in average when compared with the software-only designs. This result reinforces the last statement of the previous paragraph, which mentions that the code for interfacing with the hardware part is more susceptible to crashes than the original algorithms.

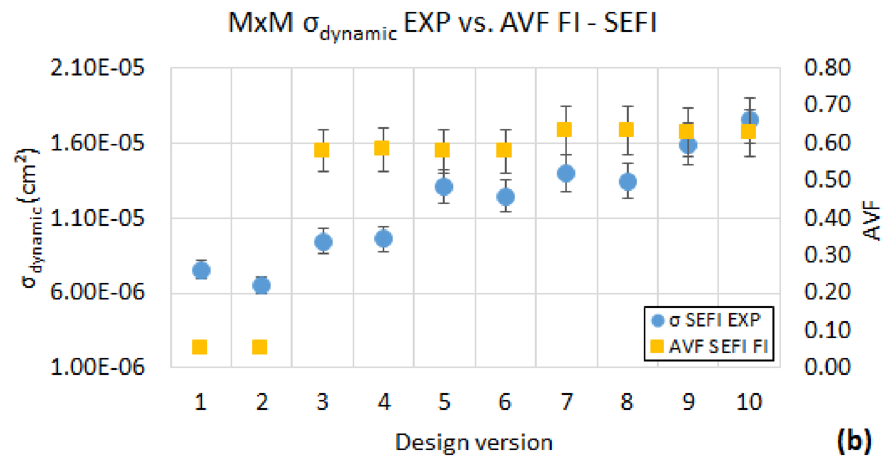
With regard to the designs with distributed workload between software and hardware (design versions from 07 to 14), the obtained AVF SDC values are in accordance with the ones from the other design versions, since half of the algorithms are executed on the processor. Thus, it is understandable that they are lower than the ones of the software-only designs, but higher than the ones of the designs with the entire algorithm offloaded to the hardware. Concerning the AVF SEFI values, one can notice that they are slight higher than the ones of the designs with the entire algorithm offloaded to the hardware. This happens because the softwares of these versions are the most complex of all, since they consist of part of the original algorithm plus the code for interfacing with the hardware part embedded, while the other ones have one part or another. It is worth mentioning that the interface code with a hardware part is the same for any design that has a hardware accelerator in the PL,

independently if it implements one set of the algorithm or both. The interface consists always of one input stream and one output stream.

Fig. 8.2 and 8.3 compare the total SDC, SEFI, and TOTAL (SDC + SEFI) AVF values of the MxM and AES benchmark designs obtained from both hardware and software fault injection campaigns with the respective dynamic cross sections obtained from radiation experiments. By comparing the obtained AVF and dynamic cross section trends, one can clearly observe that there are differences in the obtained values from fault injections and radiation experiments, as also observed in Section 6.2.3 for hardware-only designs. The reasons can be quite diverse, such as: in the PL part, faults are injected only in configuration bits related to CLBs (LUTs, user FFs, and interconnections) and clock distribution interconnections, and not in the BRAMs; in the PS part, faults are only injected into the processor's register file, and not in the other functional blocks and memories of the processor; and, in case of the radiation experiments, the particle beam and flux may present variations and the probability of a particle hits the PS or PL parts is different, since the PS occupies almost only one quarter of the Zynq-7000's die. However, in most of the cases, mainly in the SDC and SEFI ones, it is still possible to observe the reliability trend of the designs. Results also reveals that it is mandatory to consider both SDC and SEFI values separately, since the TOTAL values may mask important data about the designs, such as the behavior of the SEFI AVF and dynamic cross section in the MxM benchmark.

Figure 8.2 – Comparison between the total SDC, SEFI, and TOTAL (SDC + SEFI) AVF values obtained from both hardware and software fault injection campaigns with the respective dynamic cross sections obtained from radiation experiments for the MxM benchmark.





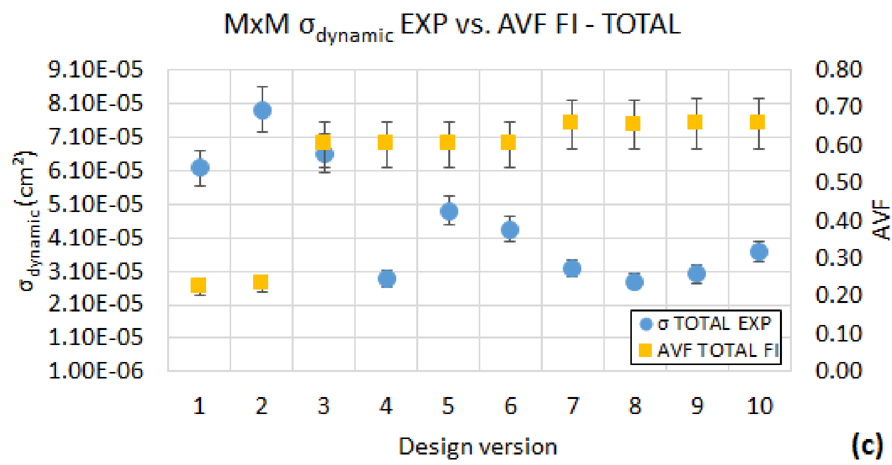
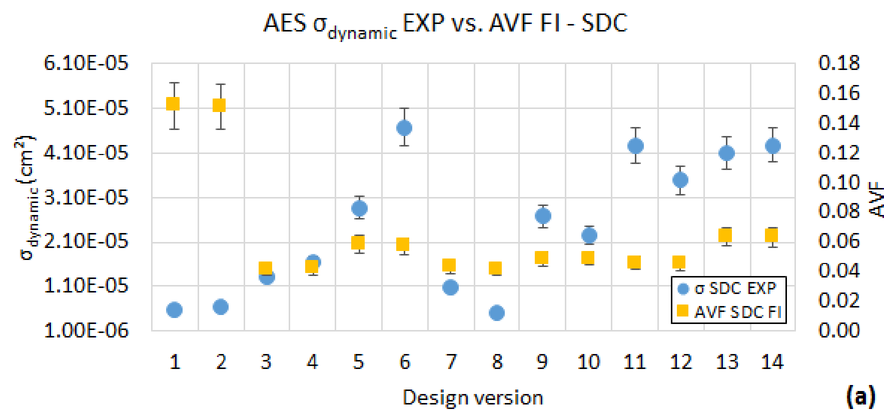


Figure 8.3 – Comparison between the total SDC, SEFI, and TOTAL (SDC + SEFI) AVF values obtained from both hardware and software fault injection campaigns with the respective dynamic cross sections obtained from radiation experiments for the AES benchmark.



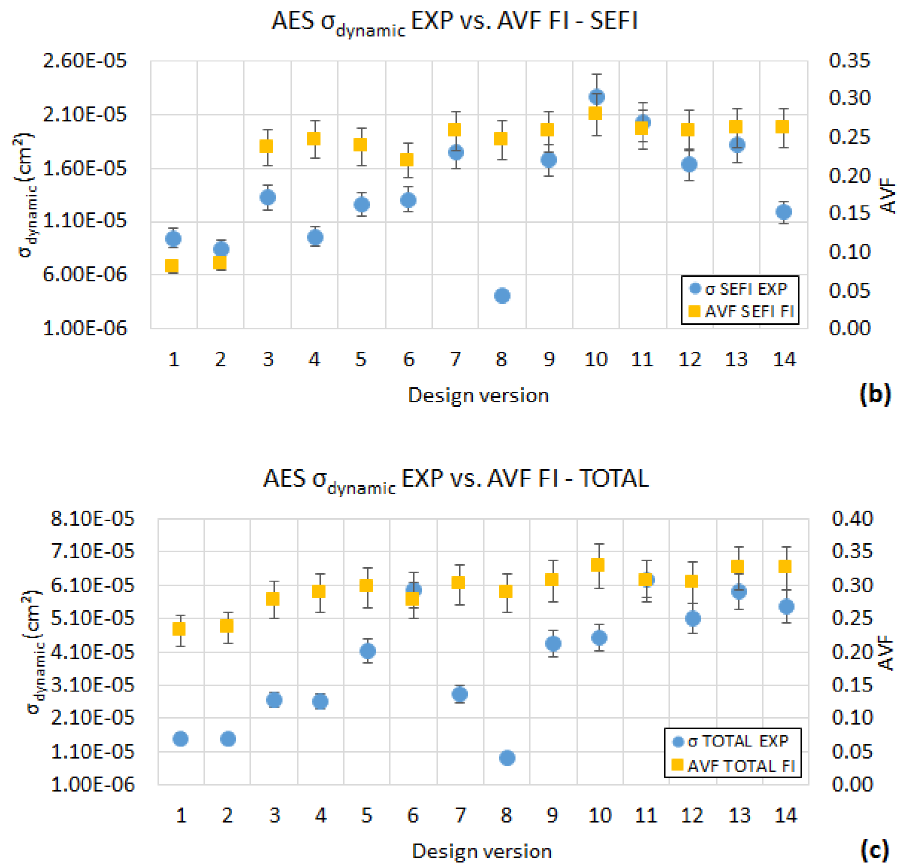


Fig. 8.4 and 8.5 compare the total SDC, SEFI, and TOTAL (SDC + SEFI) MWBF values of the MxM and AES benchmark designs obtained from both hardware and software fault injection campaigns with the respective MWBF values obtained from radiation experiments. The MWBF values obtained from fault injections considered the same particle flux experienced by the designs during the radiation experiments of Section 7.2. In those experiments, the particle flux varied between $1.0x10^2$ and $1.0x10^3$ particles.$cm^{-2}.s^{-1}$. The execution time of each design was already specified in Table 7.3.

The comparison between the fault injections and radiation experiments results shows clearly that there are differences in the obtained values, as expected and mentioned in Section 8.1. However, results also show that the proposed reliability analysis is capable of estimating the MWBF trend of different hardware and software co-designs for the same benchmark application with a considerably precision. This verifies the effectiveness of the proposed analysis, since the MWBF metric was chosen as the main resulting metric of it.

Figure 8.4 – Comparison between the total SDC, SEFI, and TOTAL (SDC + SEFI) MWBF values obtained from both hardware and software fault injection campaigns with the respective MWBF values obtained from radiation experiments for the MxM benchmark.

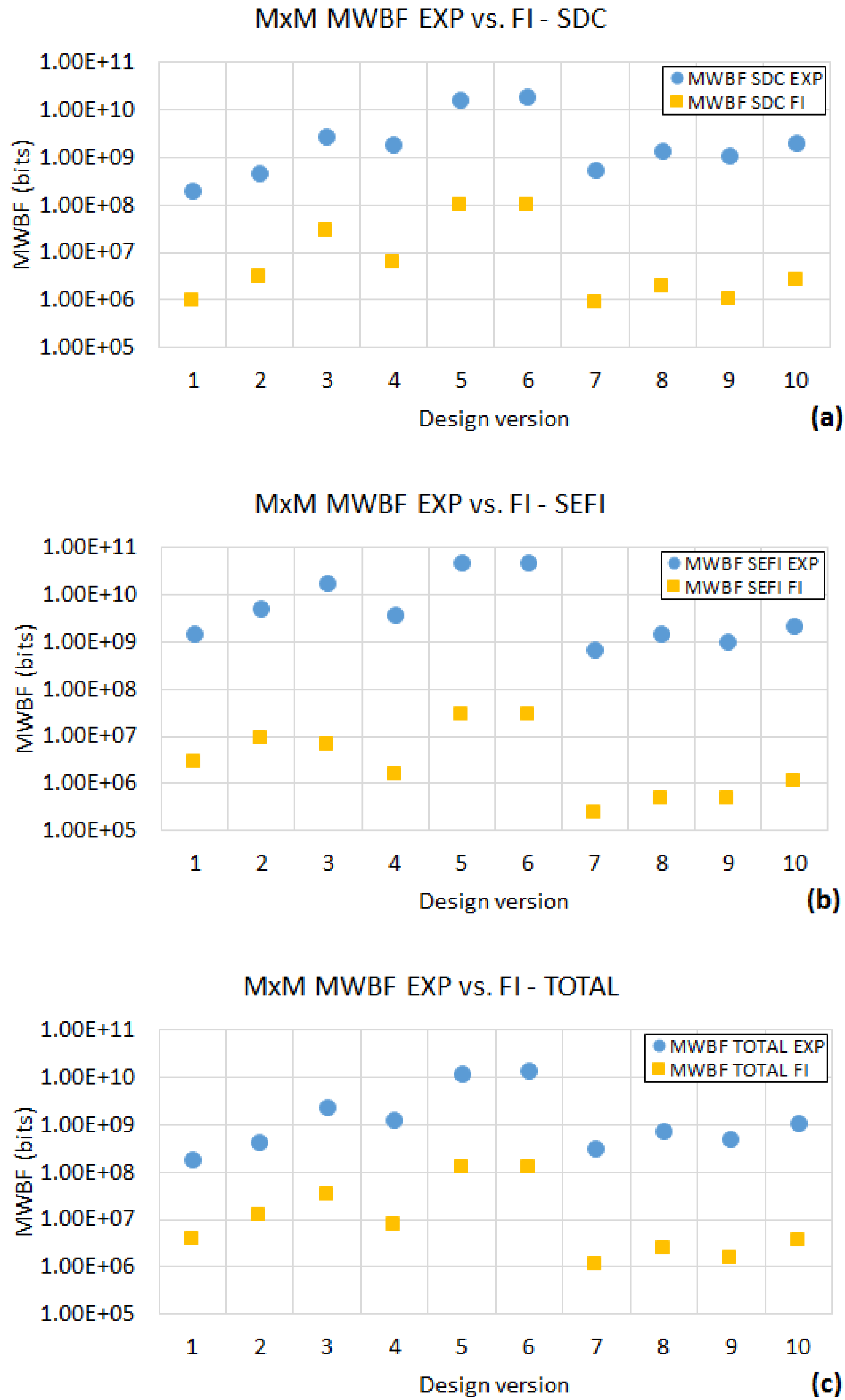


Figure 8.5 – Comparison between the total SDC, SEFI, and TOTAL (SDC + SEFI) MWBF values obtained from both hardware and software fault injection campaigns with the respective MWBF values obtained from radiation experiments for the AES benchmark.

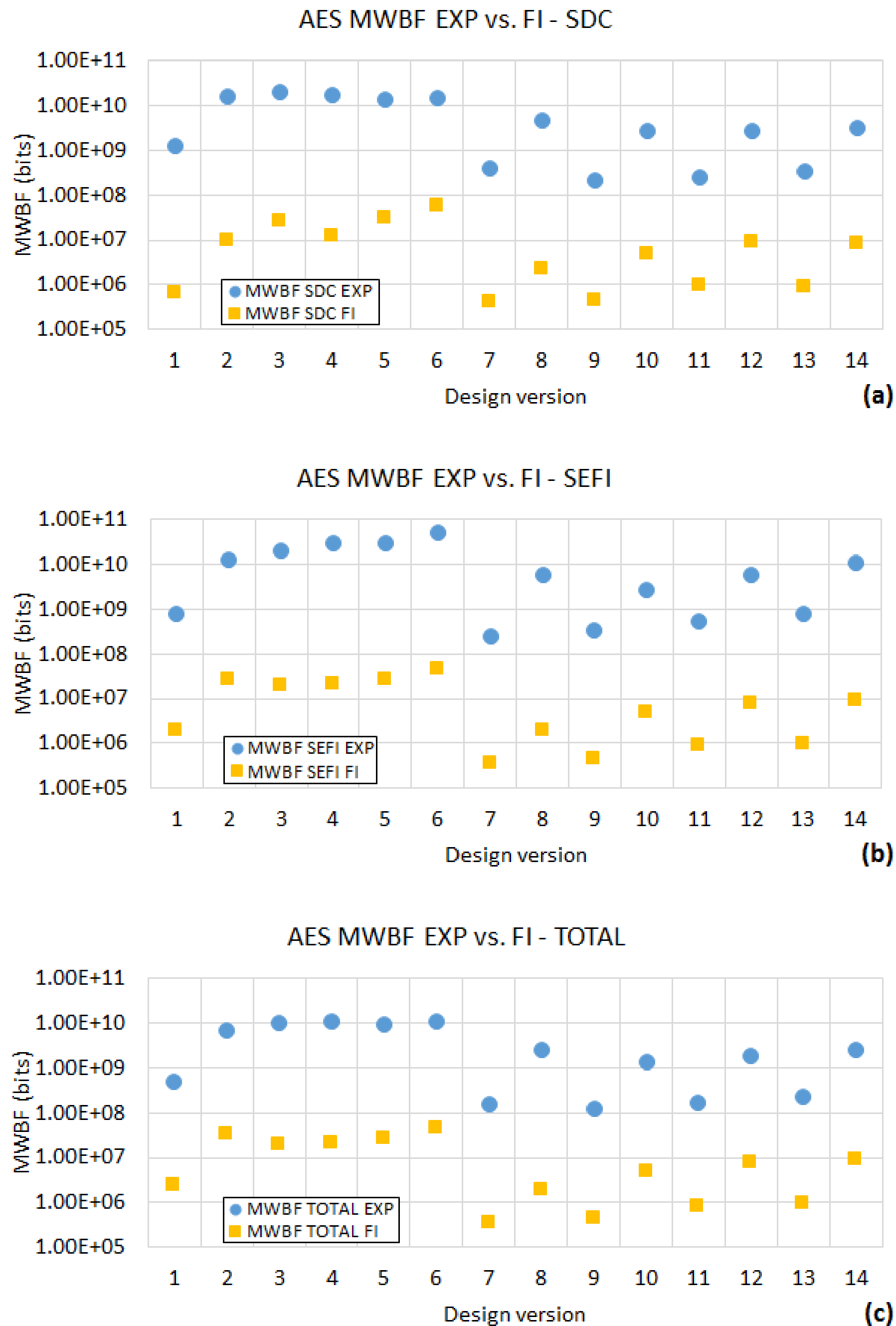


Fig. 8.6 and 8.7 compare the total SDC, SEFI, and TOTAL (SDC + SEFI) Performance rate values of the MxM and AES benchmark designs obtained from both hardware and software fault injection campaigns with the respective Performance rate values obtained from radiation experiments. In general, results from fault injections and radiation

experiments show a good agreement, reinforcing the effectiveness of the proposed reliability analysis.

Figure 8.6 – Comparison between the total SDC, SEFI, and TOTAL (SDC + SEFI) Performance rate values obtained from both hardware and software fault injection campaigns with the respective MWBF values obtained from radiation experiments for the MxM benchmark.

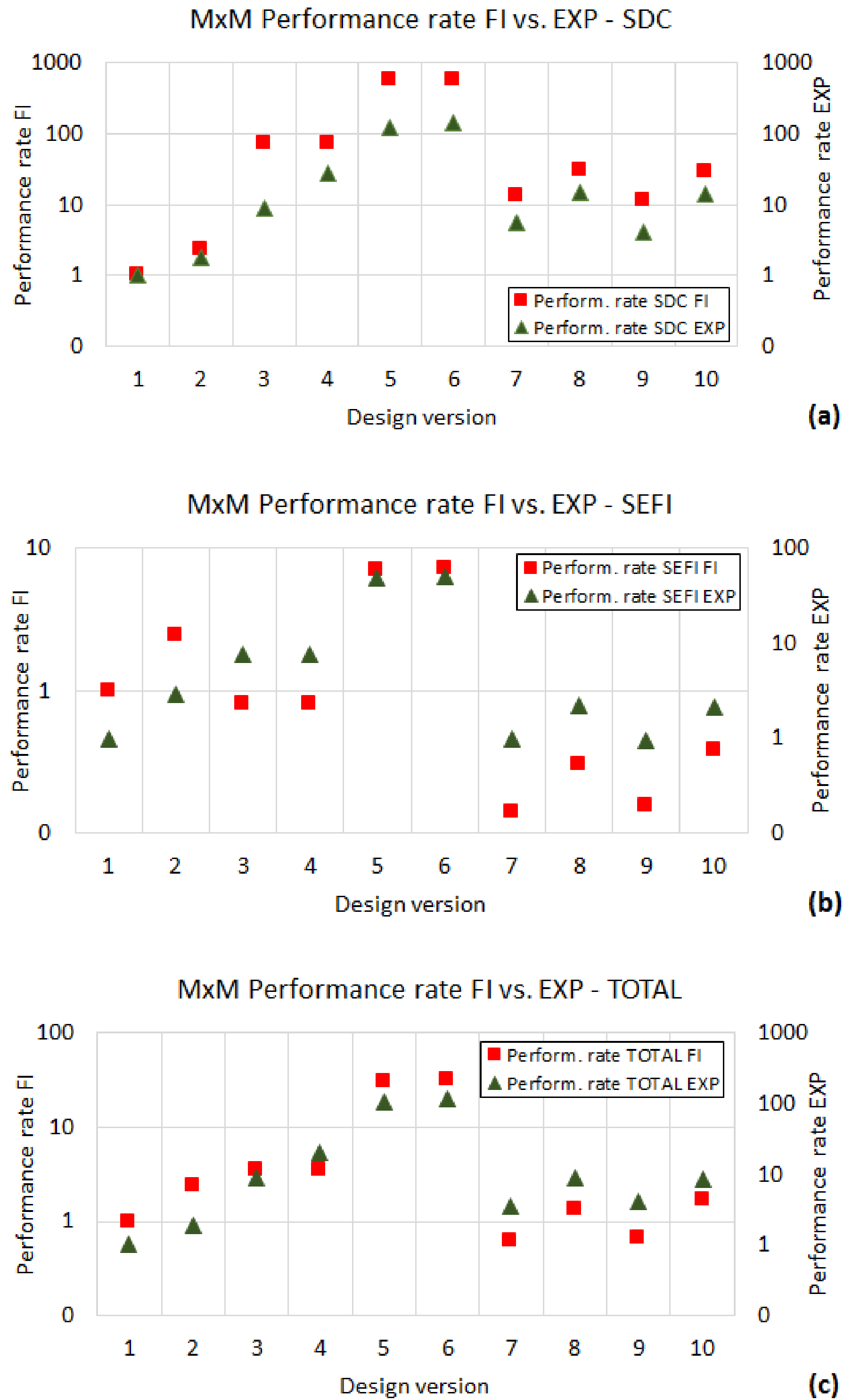
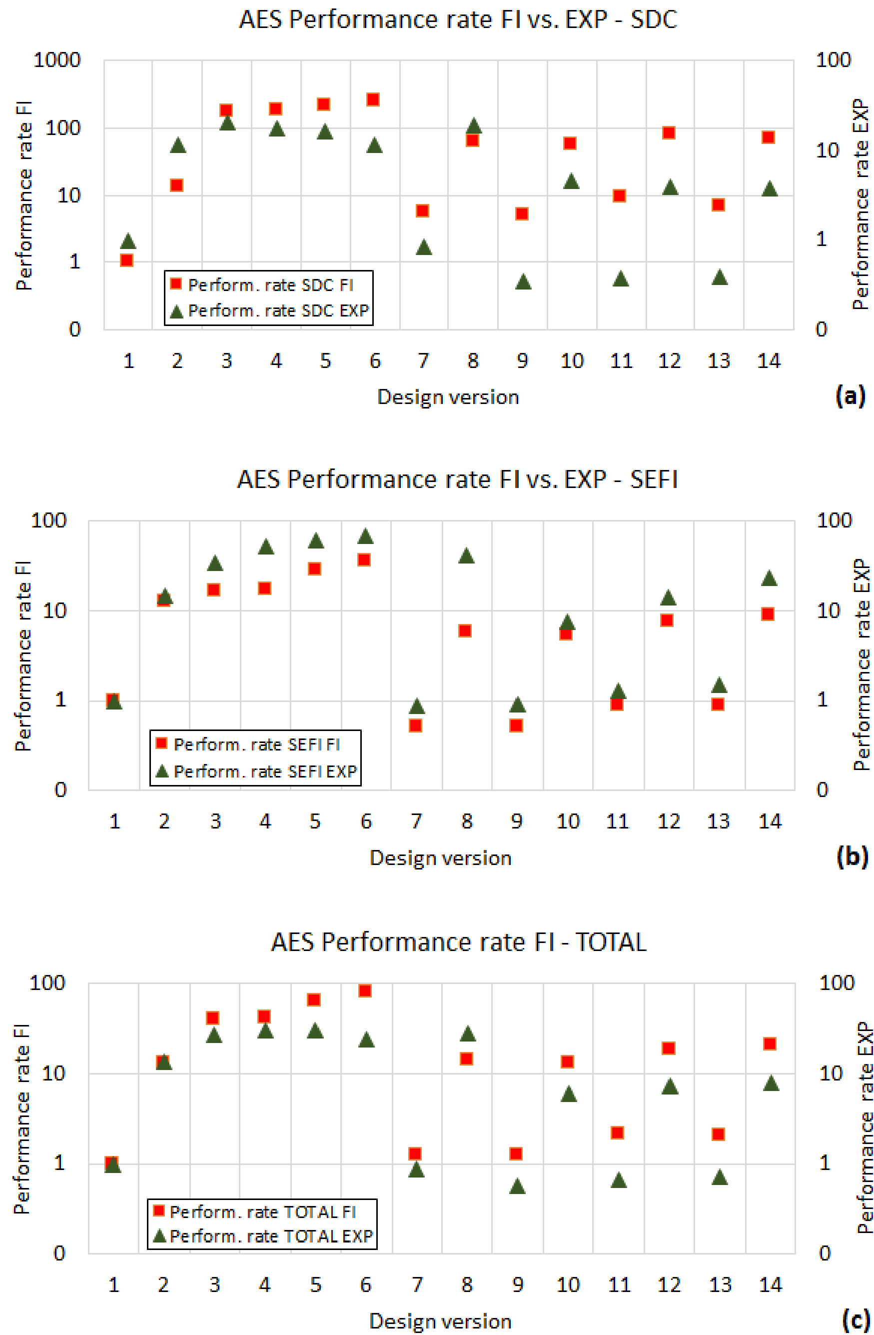


Figure 8.7 – Comparison between the total SDC, SEFI, and TOTAL (SDC + SEFI) Performance rate values obtained from both hardware and software fault injection campaigns with the respective MWBF values obtained from radiation experiments for the AES benchmark.



8.4 Summary

Results show that, in general, the estimation of the reliability trend of hardware and software co-designs, hardware-only designs, and software-only designs, through the proposed

flow is a suitable method for estimating their behavior before radiation experiments. Moreover, fault injection results also reinforce the need of taking into account each design option available and all the parameters of the system involved, such as the cross section/AVF, execution time, and workload.

Several improvements can be performed aiming to increase the confidence of the fault injection platforms and the accuracy of the flow. In the PL (hardware) side, faults can also be injected in the BRAMs. In the PS (software) side, the software fault injector can be updated to consider injecting faults in the processor's configuration registers and embedded memories. In addition, the integration of both fault injectors would certainly improve the precision of the results.

9 CONCLUDING REMARKS

As observed, modern commercial APSoCs offer a plethora of advantages and are very attractive for safety-critical markets. However, their high complexity and density increase the susceptibility of systems implemented in them to noises that are present in the environment, such as the ones caused by the radiation. In this thesis, a deep investigation of the radiation effects on APSoCs was performed together with an investigation about the correlation between hardware and software resources sensitivity in the overall system performance. As a final result, a reliability analysis flow was proposed aiming to estimate the reliability trend of hardware and software co-designs, hardware-only designs, and software-only designs implemented in APSoCs.

The next sections summarize the main contributions of this thesis, present future works, and list the publications of the author during his Ph.D.

9.1 Main contributions

9.1.1 Extensive review about APSoCs, possible radiation effects on them, and the methods and metrics for evaluating them under radiation

This thesis presented that programmable devices have evolved very rapidly in the last decade, mainly because of performance pressure in the high-volume commercial marketplace. As a consequence, several APSoC devices were introduced in the market providing higher programmable flexibility and overall system performance at lower costs than standalone processors and FPGAs. However, the high complexity and density of these devices increase the system's susceptibility to noises that are present in the environment, such as the ones caused by radiation.

With regards to radiation effects, this thesis made it clear that state-of-the-art complex devices such as APSoCs have created many challenges for the radiation effects field. That is because radiation-induced failures in such devices and architectures may result in a complex chain of effects due to their heterogeneous nature. Consequently, additional methodologies and metrics become necessary for estimating the reliability of such devices, such as the MWBF. The MWBF metric identifies the workload that can be correctly computed by the system before experiencing a failure. Thus, this thesis supports that the MWBF must always

be considered in the analysis of APSoC-based systems, since it considers that the capability of a system to provide correct data depends on several factors, such as the execution time and workload of the system, and not only of the sensitive area (cross section). It is also worth highlighting that, as far as it is known, this is the first time that the possible radiation effects on APSoCs are listed together in one single document.

9.1.2 Original static data about Xilinx Zynq-7000 under radiation

This thesis presented original static data about several hardware parts of Xilinx Zynq-7000 under heavy ion and proton irradiations. The CRAM, BRAM, OCM, and L2 cache were tested. Some of them were also tested under different conditions. Although results revealed that there are not significant differences among their cross sections, the obtained results are important for guiding designers during the implementation of a shared memory between the PS and PL parts of Zynq-7000, for example).

9.1.3 Original dynamic analysis and data about Xilinx Zynq-7000 under radiation

Dynamic experiments showed that there are several choices of architectures and resources to be chosen when implementing a system on an APSoC. Moreover, results also showed that there are logic resources that can increase or decrease the vulnerability of an entire system to failures.

In the PS part, dynamic tests consisted of different cache schemes (L1 and L2 caches) aiming to evaluate the impacts of the cache scheme on the sensitiveness of the processor under heavy ions. Cross section results showed that the addition of any cache memory to the memory hierarchy affects the sensitivity of the processor significantly. However, regardless the smaller cross section imposed by disabling all caches, its execution time is so high not to be compensated by the benefit in terms of performance. Thus, for the other configurations, despite the increase in the complexity and sensitive area, the smaller the execution time, the bigger the MWBF.

In the PL part, for the first time, dynamic tests investigated the trade-offs of different HLS-based designs implemented into an Artix-7 FPGA (equivalent to the Zynq-7000's PL) in terms of not only resource utilization and performance, but also reliability, by analyzing their behaviors under SEUs and comparing them to a standard processor-based implementation. Results showed that the influence of HLS optimizations in the dynamic cross section of the

designs is low when compared to their performance enhancement, which contributed significantly to increase the MWBF and the performance rate of them.

This thesis also investigated for the first time the impact of using both PS and PL parts of Xilinx Zynq-7000 APSoC in the overall system failure rate. Different memory organizations, communication schemes, and computing modes were considered for building hardware and software co-designs. Results showed that the reliability of systems based on hardware and software co-design seems to be inversely proportional not only to the device sensitivity but also to the system execution time.

9.1.4 Reliability analysis flow for hardware-only designs, software-only designs, and hardware and software co-designs

The final result of this thesis was a methodology flow to estimate the reliability trend of software-only, hardware-only, and hardware and software co-designs based on fault injection campaigns. The main objective was to accelerate the search for the design with the best trade-off between performance and reliability, i.e. the design that provides a performance enhancement higher than the sensitivity increase. Results showed that, in general, the estimation of the reliability trend of software-only, hardware-only, and hardware and software co-designs through the proposed flow is a suitable method for estimating their behavior before radiation experiments.

9.2 Future works

9.2.1 Completing the static measurements of Zynq-7000

Additional heavy ion and proton experiments can be performed to complete the static data results of the PS part and for help refining the proposed reliability analysis flow. However, obtaining the static cross section of hardware blocks of a processor, such as the processor's register file and L1 cache, is a complex task, since the entire device is irradiated and the static test of these blocks are, in fact, semi-static, which certainly affects the final results. An interesting approach to work around this problem is to perform laser test campaigns for evaluating the static cross section of each memory block separately, since laser tests provide a high level of accessibility to locate the circuit elements where faults are

injected. The small laser spot and precise beam localization characteristics allow sensitive device nodes to be pinpointed with submicron accuracy without affecting the entire device.

9.2.2 Improving the reliability analysis flow

Several improvements can be performed aiming to increase the confidence of the fault injection platforms and the accuracy of the flow. In the PL side, faults can also be injected in the BRAMs. In the PS side, the software fault injector can be updated to consider injecting faults in the processor's configuration registers and embedded memories. In addition, the integration of both fault injectors would certainly improve the precision of the results.

9.2.3 Analyzing the use of fault-tolerant techniques in APSoCs

The use of fault-tolerant techniques can also be evaluated. In fact, this thesis motivated the beginning of three ongoing works, which are the following:

- The exploration of the use of dual-core lockstep as a fault tolerance solution to increase the dependability in hard-core processors embedded in COTS APSoCs. As a case study, it was designed and implemented an approach based on lockstep to protect the dual-core ARM Cortex-A9 processor embedded into the Xilinx Zynq-7000. Experimental results show the effectiveness of the proposed approach in mitigating around 91% of the bit-flips injected in the processor's registers. It was also observed that the performance overhead depends on the application size, the number of checkpoints performed, and the checkpoint and rollback routines.
- The use of TMR at processor's instruction level. The VAR3Ra is a software-only technique capable of recovering from errors, specially designed for the ARM-v7 architecture (ARM Cortex-A9). The technique is based on the detection technique VAR3 (CHIELLE et al., 2016). Each operation, data, or register has two replicas, which are independent of the original, providing both spatial and temporal redundancy/protection. Due to the lack of enough available registers to apply software triplication in the ARM-v7 architecture, part of an embedded memory is used as the third register (second replica).
- The use of TMR in hardware accelerators designs described in C programming language and synthesized by HLS. A setup composed of a soft-

core processor and a matrix multiplication design protected by TMR and embedded into an SRAM-based FPGA was analyzed under accumulated bit-flips in its configuration memory bits. Different configurations using single and multiple inputs and output workload data streams were tested. Results show that by using a coarse grain TMR with triplicated inputs, voters, and outputs, it is possible to reach 95% of reliability by accumulating up to 61 bit-flips and 99% of reliability by accumulating up to 17 bit-flips in the configuration memory bits. These numbers imply in an MTBF of the coarse grain TMR at ground level from 50% to 70% higher than the MTBF of the unhardened version for the same reliability confidence.

Another fault-tolerant technique that could be interesting to evaluate in an APSoC context is the use of scrubbing in the configuration memory of the PL and configuration registers of the PS. Scrubbing would avoid the accumulation of bit-flips and could reduce the occurrence of SDCs and SEFIs drastically.

9.2.4 Evaluation of other APSoCs

Similar APSoCs to Xilinx Zynq-7000, such as Microsemi SmartFusion and SmartFusion2 and Altera Cyclone V can be easily evaluated by using the evaluation methodology adopted in this thesis. In fact, Microsemi SmartFusion2 has already started being evaluated and a work presenting the first results was already approved for publication.

Finally, it is also worth mentioning that the proposed reliability analysis flow is capable of being generic and extendable to other APSoCs if slight adjustments are performed.

9.3 Publications

9.3.1 Book chapters

First author:

1. **Neutron-Induced Single Event Effect in Mixed-Signal Flash-Based FPGA.** L. A. Tambara, M. S. Lubaszewski, T. R. Balen, P. Rech, F. L. Kastensmidt, C. Frost. FPGAs and Parallel Architectures for Aerospace Applications. 1ed.:

Springer International Publishing, 2016, pp. 201-216. DOI: 10.1007/978-3-319-14352-1_14.

2. **Fault-Tolerant Manager Core for Dynamic Partial Reconfiguration in FPGAs.** L. A. Tambara, J. Tarrillo, F. L. Kastensmidt, L. Sterpone. FPGAs and Parallel Architectures for Aerospace Applications. 1ed.: Springer International Publishing, 2016, pp. 121-133. DOI: 10.1007/978-3-319-14352-1_9.
3. **Measuring Failure Probability of Coarse and Fine Grain TMR Schemes in SRAM-based FPGAs Under Neutron-Induced Effects.** L. A. Tambara, F. Almeida, P. Rech, F. L. Kastensmidt, G. Bruni, C. Frost. Lecture Notes in Computer Science. 1ed.: Springer International Publishing, 2015, v. 9040, pp. 331-338. DOI: 10.1007/978-3-319-16214-0_28.

Co-author:

1. **Applying TMR in Hardware Accelerators Generated by High-Level Synthesis Design Flow for Mitigating Multiple Bit Upsets in SRAM-Based FPGAs.** A. F. dos Santos, L. A. Tambara, F. Benevenuti, J. Tonfat, F. L. Kastensmidt. Lecture Notes in Computer Science. 1ed.: Springer International Publishing, 2017, v. 10216, pp. 202-213. DOI: 10.1007/978-3-319-56258-2_18.
2. **Exploring Performance Overhead Versus Soft Error Detection in Lockstep Dual-Core ARM Cortex-A9 Processor Embedded into Xilinx Zynq APSoC.** Á. B. de Oliveira, L. A. Tambara, F. L. Kastensmidt. Lecture Notes in Computer Science. 1ed.: Springer International Publishing, 2017, v. 10216, pp. 189-201. DOI: 10.1007/978-3-319-56258-2_17.
3. **Method to Analyze the Susceptibility of HLS Designs in SRAM-Based FPGAs Under Soft Errors.** J. Tonfat, L. A. Tambara, A. F. dos Santos, F. L. Kastensmidt. Lecture Notes in Computer Science. 1ed.: Springer International Publishing, 2016, v. 9625, pp. 132-143. DOI: 10.1007/978-3-319-30481-6_11.
4. **Multiple Fault Injection Platform for SRAM-Based FPGA Based on Ground-Level Radiation Experiments.** J. Tonfat, J. Tarrillo, L. A. Tambara, F. L. Kastensmidt, R. Reis. FPGAs and Parallel Architectures for Aerospace Applications. 1ed.: Springer International Publishing, 2016, pp. 135-151. DOI: 10.1007/978-3-319-14352-1_10.

9.3.2 Journals

First author:

1. **Analyzing the Impact of Radiation-induced Failures in Flash-based APSoC with and without Fault Tolerance Techniques at CERN Environment.** L. A. Tambara, E. Chielle, F. L. Kastensmidt, G. Tsiligiannis, S. Danzeca, M. Brugger, A. Masi. Microelectronics Reliability, 2017. (approved for publication)
2. **Analyzing Reliability and Performance Trade-offs of HLS-based Designs in SRAM-based FPGAs under Soft Errors.** L. A. Tambara, J. Tonfat, A. Santos, F. L. Kastensmidt, N. H. Medina, N. Added, V. A. P. Aguiar, F. Aguirre, M. A. G. Silveira. IEEE Transactions on Nuclear Science, v. 64, n. 2, pp. 874-881, Feb. 2017. DOI: 10.1109/TNS.2017.2648978.
3. **Analyzing the Impact of Radiation-Induced Failures in Programmable SoCs.** L. A. Tambara, P. Rech, E. Chielle, J. Tonfat, F. L. Kastensmidt. IEEE Transactions on Nuclear Science, v. 63, n. 4, pp. 2217-2224, Aug. 2016. DOI: 10.1109/TNS.2016.2522508.

Co-author:

1. **Register File Criticality and Compiler Optimization Effects on Embedded Microprocessors Reliability.** F. M. Lins, L. A. Tambara, F. L. Kastensmidt, P. Rech. IEEE Transactions on Nuclear Science, 2016. (pending publication)
2. **Soft Error Susceptibility Analysis Methodology of HLS Designs in SRAM-based FPGAs.** J. Tonfat, L. A. Tambara, A. F. dos Santos, F. L. Kastensmidt. Microprocessors and Microsystems, v. 51, pp. 209-2019, Jun. 2017. DOI: 10.1016/J.MICPRO.2017.04.016.
3. **Reliability on ARM Processors Against Soft Errors Through SIHFT Techniques.** E. Chielle, F. Rosa, G. S. Rodrigues, L. A. Tambara, J. Tonfat, E. Macchione, F. Aguirre, N. Added, N. Medina, V. Aguiar, M. A. G. Silveira, L. Ost, R. Reis, S. Cuenca-Asensi, F. L. Kastensmidt. IEEE Transactions on Nuclear Science, v. 63, n. 4, pp. 2208-2216, Aug. 2016. DOI: 10.1109/TNS.2016.2525735.

4. **S-SETA: Selective Software-Only Error-Detection Technique Using Assertions.** E. Chielle, G. S. Rodrigues, L. A. Tambara, P. Rech, F. L. Kastensmidt, S. Cuenca-Asensi, H. Quinn. IEEE Transactions on Nuclear Science, v. 62, n. 6, pp. 3088-3095, Dec. 2015. DOI: 10.1109/TNS.2015.2484842.
5. **Exploring Design Diversity Redundancy to Improve Resilience in Mixed-Signal Systems.** C. P. Chenet, L. A. Tambara, G. M. de Borges, F. L. Kastensmidt, M. S. Lubaszewski, T. R. Balen. Microelectronics and Reliability, v. 55, n. 12, pp. 2833-2844, Dec. 2015. DOI: 10.1016/J.MICROREL.2015.08.011.
6. **Laser Testing Methodology for Diagnosing Diverse Soft Errors in a Nanoscale SRAM-Based FPGA.** F. L. Kastensmidt, L. A. Tambara, D. V. Bobrovsky, A. A. Pechenkin, A. Y. Nikiforov. IEEE Transactions on Nuclear Science, v. 61, n. 6, pp. 3130-3137, Dec. 2014. DOI: 10.1109/TNS.2014.2369008.

9.3.3 Conferences and workshops

First author:

1. **Reliability-Performance Analysis of Hardware and Software Co-Designs in SRAM-based APSoCs.** L. A. Tambara, F. Lins, P. Rech, F. L. Kastensmidt, N. H. Medina, N. Added, V. A. P. Aguiar, M. A. G. Silveira. European Conference on Radiation and Its Effects on Components and Systems (RADECS), 2017, Geneva, Switzerland. (submitted)
2. **Evaluating the Use of APSoCs for CERN Applications.** L. A. Tambara, E. Chielle, F. L. Kastensmidt, G. Tsiligiannis, S. Danzeca, M. Brugger, A. Masi. European Conference on Radiation and Its Effects on Components and Systems (RADECS), 2016, Bremen, Germany. (pending publication)
3. **Analyzing Reliability and Performance Trade-offs of HLS-based Designs in SRAM-based FPGAs under Soft Errors.** L. A. Tambara, J. Tonfat, A. Santos, F. L. Kastensmidt, N. H. Medina, N. Added, V. A. P. Aguiar, F. Aguirre, M. A. G. Silveira. IEEE Nuclear and Space Radiation Effects Conference (NSREC), 2016, Portland, USA.

4. **On the Reliability of Coarse and Fine Grain TMR Schemes in SRAM-based FPGAs under Single Event Upsets.** L. A. Tambara, P. Rech, F. L. Kastensmidt. South Symposium on Microelectronics (SIM), 2015, Santa Maria, Brazil.
5. **Heavy Ions Induced Single Event Upsets Testing of the 28 nm Xilinx Zynq-7000 All Programmable SoC.** L. A. Tambara, F. L. Kastensmidt, N. H. Medina, N. Added, V. A. P. Aguiar, F. Aguirre, E. Macchione, M. A. G. Silveira. IEEE Radiation Effects Data Workshop (REDW), 2015, Boston, USA. DOI: 10.1109/REDW.2015.7336716.
6. **Analyzing the Failure Impact of Using Hard- and Soft-Cores in All Programmable SoC under Neutron-Induced Upsets.** L. A. Tambara, P. Rech, E. Chielle, F. L. Kastensmidt. Conference on Radiation and Its Effects on Components and Systems (RADECS), 2015, Moscow, Russia. DOI: 10.1109/RADECS.2015.7365586.
7. **On the Characterization of Embedded Memories of Zynq-7000 All Programmable SoC under Single Event Upsets Induced by Heavy Ions and Protons.** L. A. Tambara, A. Akhmetov, D. V. Bobrovsky, F. L. Kastensmidt. European Conference on Radiation and Its Effects on Components and Systems (RADECS), 2015, Moscow, Russia. DOI: 10.1109/RADECS.2015.7365643.
8. **Soft Error Rate in SRAM-based FPGAs Under Neutron-induced and TID Effects.** L. A. Tambara, J. Tonfat, R. Reis, F. L. Kastensmidt, E. C. F. Pereira, R. G. Vaz, O. L. Goncalez. IEEE Latin American Test Workshop (LATW), 2014, Fortaleza, Brazil. DOI: 10.1109/LATW.2014.6841920.
9. **Evaluating the Robustness of TMR Schemes with Different Levels of Granularity in SRAM-based FPGAs under Neutron-induced Effects.** L. A. Tambara, P. Rech, F. L. Kastensmidt, G. Bruni, C. Frost, H. Quinn. IEEE Nuclear and Space Radiation Effects Conference (NSREC), 2014, Paris, France.
10. **Decreasing FIT with Diverse Triple Modular Redundancy in SRAM-based FPGAs.** L. A. Tambara, F. L. Kastensmidt, P. Rech, C. Frost. IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2014, Amsterdam, The Netherlands. DOI: 10.1109/DFT.2014.6962070.

11. **Neutron-induced Single Event Effects Analysis in a SAR-ADC Architecture Embedded in a Mixed-signal SoC.** L. A. Tambara, F. L. Kastensmidt, P. Rech, T. R. Balen, M. S. Lubaszewski. IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2013, Natal, Brazil. DOI: 10.1109/ISVLSI.2013.6654657.
12. **Evaluating the Effectiveness of a Diversity TMR Scheme Under Neutrons.** L. A. Tambara, F. L. Kastensmidt, J. R. Azambuja, E. Chielle, F. Almeida, G. Nazar, P. Rech, C. Frost, M. S. Lubaszewski. European Conference on Radiation and Its Effects on Components and Systems (RADECS), 2013, Oxford, United Kingdom. DOI: 10.1109/RADECS.2013.6937382.
13. **Neutron-induced Single Event Effects in a SRAM-based FPGA by Using 241Am-Be Source at IEAv.** L. A. Tambara, J. Tonfat, F. L. Kastensmidt, E. C. F. P. Junior, R. G. Vaz, O. L. Goncalez. Workshop Sobre os Efeitos das Radiações Ionizantes em Componentes Eletrônicos e Fotônicos de Uso Aeroespacial (WERICE), 2013, São José dos Campos, Brazil.

Co-author:

1. **Evaluation of Feed-Forward Artificial Neural Networks Reliability in FPGAs.** F. Libano, P. Rech, L. A. Tambara, J. L. Tonfat, N. H. Medina, N. Added, V. A. P. Aguiar, F. Aguirre, M. A. G. Silveira, F. L. Kastensmidt. IEEE Nuclear and Space Radiation Effects Conference (NSREC), 2017, New Orleans, USA. (approved for publication)
2. **Evaluating the Efficiency of using TMR in the High-Level Synthesis Design Flow of SRAM-based FPGA.** A. F. dos Santos, L. A. Tambara, F. L. Kastensmidt. IEEE Latin American Symposium on Circuits and Systems (LASCAS), 2017, Bariloche, Agentina. (pending publication)
3. **Applying Lockstep in Dual-Core ARM Cortex-A9 to Mitigate Radiation-induced Soft Errors.** A. B. de Oliveira, L. A. Tambara, F. L. Kastensmidt. IEEE Latin American Symposium on Circuits and Systems (LASCAS), 2017, Bariloche, Agentina. (pending publication)
4. **Register File Criticality on Embedded Microprocessor Reliability.** F. M. Lins, L. A. Tambara, F. L. Kastensmidt, P. Rech. European Conference on Radiation and Its Effects on Components and Systems (RADECS), 2016, Bremen, Germany. (pending publication)

5. **Applying Lockstep in Dual-Core ARM Cortex-A9 to Mitigate Radiation-induced Soft Errors.** A. B. de Oliveira, L. A. Tambara, F. L. Kastensmidt. South Symposium on Microelectronics (SIM), 2017, Rio Grande, Brazil.
6. **Reliability Analysis of Feed-Forward Artificial Neural Networks in System on Chips.** F. Libano, P. Rech, L. A. Tambara, J. L. Tonfat, N. H. Medina, N. Added, V. A. P. Aguiar, F. Aguirre, M. A. G. Silveira, F. L. Kastensmidt. Workshop on Silicon Errors in Logic - System Effects (SELSE), 2017, Boston, USA.
7. **Using Programmable System-on-Chip for Aerospace Applications.** F. L. Kastensmidt, L. A. Tambara, E. Chielle, J. Tonfat, A. Santos. Single Event Effects Symposium and Military and Aerospace Programmable Logic Devices Workshop (SEE/MAPLD), 2016, San Diego, USA.
8. **Multiple Fault Injection Platform for SRAM-based FPGA Based on Ground-level Radiation Experiments.** J. Tarrillo, J. Tonfat, L. A. Tambara, F. L. Kastensmidt, R. Reis. IEEE Latin American Test Symposium (LATS), 2015, Puerto Vallarta, Mexico. DOI: 10.1109/LATW.2015.7102494.
9. **Selective Software Techniques to Detect Neutron-induced Soft Errors in Processors with Minimum Overhead.** E. Chielle, G. S. Rodrigues, F. L. Kastensmidt, S. Cuenca-Asensi, L. A. Tambara, P. Rech, H. Quinn. IEEE Nuclear and Space Radiation Effects Conference (NSREC), 2015, Boston, USA.
10. **Reliability on ARM Processors Against Soft Errors by a Purely Software Approach.** E. Chielle, F. Rosa, G. S. Rodrigues, L. A. Tambara, F. L. Kastensmidt, R. Reis, S. Cuenca-Asensi. European Conference on Radiation and Its Effects on Components and Systems (RADECS), 2015, Moscow, Russia. DOI: 10.1109/RADECS.2015.7365660.
11. **Laser Testing for Diagnosing SEU and SET in Virtex-5.** F. L. Kastensmidt, L. A. Tambara, D. Bobrovsky, A. Pechenkin, A. Nikiforov. IEEE Nuclear and Space Radiation Effects Conference (NSREC), 2014, Paris, France.
12. **Power Dissipation Effects on 28nm FPGA-based System-on-Chip Neutron Sensitivity.** G. Bruni, P. Rech, L. A. Tambara, G. L. Nazar, F. L. Kastensmidt, R. Reis, A. Paccagnella. International Conference on Very Large Scale Integration (VLSI-SoC), 2014, Playa del Carmen, Mexico. DOI: 10.1109/VLSI-SoC.2014.7004195.

REFERENCES

AGUIAR, V. A. P.; ADDED, N.; MEDINA, N. H.; MACCHIONE, E. L. A.; TABACNIKS, M. H.; AGUIRRE, F. R.; SILVEIRA, M. A. G; SANTOS, R. B. B.; SEIXAS, L. E. Experimental Setup for Single Event Effects at the São Paulo 8UD Pelletron Accelerator. **Nuclear Instruments & Methods in Physics Research**, vol. 332, 2014, pp. 397-400.

AGUIAR, V.A.P.; MEDINA, N. H.; ADDED, N.; MACCHIONE, E. L. A.; AGUIRRE, F. R.; RIBAS, R. V.; NASCIMENTO, S. G.; ESCUDEIRO, R.; ALLEGRO, P. R. P.; PEREGO, C. C.; FAGUNDES, L. M.; DUARTE, J. G.; SCARDUELLI, V. B.; MORAIS, O. B.; ALMEIDA, E. A.; JOAQUIM, P. M.; SOUZA, M. S.; CECOTTE, A. F. M.; MARTINS, R.; BRAGE, J. A. P.; LEISTENSCHNEIDER, E.; OLIVEIRA, R. A. N.; ASSIS, R. F.; LEITE, A. R.; TERASSI, J. C.; ABREU, J. C.; SIMOES, R. F.; JOAQUIM, A. S.; SERVELO, W. A.; SILVA, S. C.; MINAS, J. H. P.; SILVA, M. T.; SILVA, V. E. S.; KSHINSKIY, D. A.; RODRIGUES, C. L. Safiira Facility At LAFN-USP: Beam Characteristics and Scientific Program. **XL Workshop on Nuclear Physics**, 2017. (to be published)

ALEXANDRESCU, D.; STERPONE, L.; LOPEZ-ONGIL, C. Fault Injection and Fault Tolerance Methodologies for Assessing Device Robustness and Mitigating Against Ionizing Radiation. **IEEE European Test Symposium**, 2014, pp. 1-6.

ALÍA, R. G. Facilities for Radiation Testing and Technologies for Radiation Monitoring. **CERN/SSC Technology Transfer Day**, Jun. 2016.

ALLEN, G.; IROM, F.; AMRBAR, M. Zynq SoC Radiation Test Results and Plans for the Altera MAX10. **NEPP Electronics Technology Workshop**, Jun. 2015.

ALTERA. Adding Hardware Accelerators to Reduce Power in Embedded Systems, WP-01112-1.0, 2009.

ALTERA. Cyclone V FPGAs & SoCs, 2015. Available at: <https://www.altera.com/products/fpga/cyclone-series/cyclone-v/overview.html>. Accessed on: August 2015.

AMDAHL, G. M. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. **AFIPS 1967**, 1967, pp. 483–485.

AMRBAR, M.; IROM, F.; GUERTIN, S. M.; ALLEN, G. Heavy Ion Single Event Effects Measurements of Xilinx Zynq-7000 FPGA. **IEEE Radiation Effects Data Workshop**, 2015, pp. 1-4.

ARM. Cortex-A9 – Technical Reference Manual. Revision r4p1, 2012.

ARM. AMBA Specifications, 2015. Available at: <http://www.arm.com/products/system-ip/amba-specifications.php>. Accessed on: April 2016.

ARM. NEON, 2015. Available at: < http://www.arm.com/products/processors/technologies/neon.php>. Accessed on: April 2016.

ATLAS. ATLAS, 2016. Available at: <http://home.cern/about/experiments/atlas>. Accessed on: May 2016.

AVIZIENIS, A.; LAPRIE, J. C.; RANDELL, B.; LANDWEHR, C. Basic Concepts and Taxonomy of Dependable and Secure Computing. **IEEE Transactions on Dependable and Secure Computing**, vol. 1, n. 1, Mar. 2004, pp. 11-33.

BAGATIN, M.; GERARDIN, S.; PACCAGNELLA, A.; ANDREANI, C.; GORINI, G.; FROST, C.D. Temperature Dependence of Neutron-induced Soft Errors in SRAMs. **Microelectronics Reliability**, vol. 52, n. 1, Jan. 2011, pp. 289-293

BOUDENOT, J. C. Radiation Space Environment. In. VELAZCO, R.; FOUILLAT, P.; REIS, R. (Ed.). **Radiation Effects on Embedded Systems**. Dordrecht: Springer, 2007. p. 1-9.

CANIS, A.; CHOI, J.; ALDHAM, M.; ZHANG, V.; KAMMOONA, A.; ANDERSON, J. H.; BROWN, S.; CZAJKOWSKI, T. LegUp: High-Level Synthesis for FPGA-based Processor/Accelerator Systems. **ACM/SIGDA International Symposium on Field Programmable Gate Arrays**, Feb. 2011, pp. 33-36.

CARLIN, N.; DE SOUZA, J. C.; SZANTO, E. M.; ACQUADRO, J. C.; OKUNO, E.; TAKAHASHI, J.; UMISEDO, N. K.; DE OLIVEIRA FILHO, F.J.; VASCONCELOS, J. A. C. Irradiation Facility for Radiobiology and Molecular Biophysics Studies at the University of São Paulo Pelletron Accelerator Laboratory. **Nuclear Instruments and Methods in Physics**, vol. 540, n. 2-3, Mar. 2005, pp. 215-221.

CARREIRA, J.; MADEIRA, H.; SILVA, J. G. Xception: A Technique for the Experimental Evaluation of Dependability in Modern Computers. **IEEE Transactions on Software Engineering**, vol. 24, n. 2, Feb. 1998, pp. 125-136.

CHEN, L.; EBRAHIMI, M.; TAHOORI, M. B. Reliability-Aware Resource Allocation and Binding in High-Level Synthesis. **ACM Transactions on Design Automation of Electronic Systems**, vol. 21, n. 2, Jan. 2016, pp. 1-27.

CHEN, X.; YANG, W.; ZHAO, M.; WANG, J. HLS-based Sensitivity-Inductive Soft Error Mitigation for Satellite Communications Systems. **IEEE International Symposium on On-Line Testing and Robust System Design**, Jul. 2016, pp. 143-148.

CHIELLE, E.; RODRIGUES, G. S.; KASTENSMIDT, F. L.; CUENCA-ASENSI, S.; TAMBARA, L. A.; RECH, P.; QUINN, H. S-SETA: Selective Software-Only Error-Detection Technique Using Assertions. **IEEE Transactions on Nuclear Science**, vol. 62, n. 6, Dec. 2015, pp. 3088-3095.

CHIELLE, E.; ROSA, F.; RODRIGUES, G. S.; TAMBARA, L. A.; TONFAT, J.; MACCHIONE, E.; AGUIRRE, F.; ADDED, N.; MEDINA, N.; AGUIAR, V.; SILVEIRA, M. A. G.; OST, L.; REIS, R.; CUENCA-ASENSI, S.; KASTENSMIDT, F. L. **IEEE Transactions on Nuclear Science**, v. 63, n. 4, Aug. 2016, pp. 2208-2216.

CHOI, J.; NAM, K.; CANIS, A.; ANDERSON, J.; BROWN, S.; CZAJKOWSKI, T. Impact of Cache Architecture and Interface on Performance and Area of FPGA-Based

Processor/Parallel-Accelerator Systems. **IEEE Annual International Symposium on Field-Programmable Custom Computing Machines**, 2012, pp. 17-24.

CLAEYS, C.; SIMOEN, E. **Radiation Effects in Advanced Semiconductor Materials and Devices**. Berlin: Springer, 2002.

CROCKETT, L. H.; ELLIOT, R. A.; ENDERWITZ, M. A.; STEWART, R. W. **The Zynq Book**. 1st ed. Glasgow: University of Strathclyde, 2014.

CURD, D.; CRABILL, E. UltraScale Devices Maximize Design Integrity with Industry-Leading SEU Resilience and Mitigation, 2015. Available at: <http://www.xilinx.com/support/documentation/white_papers/wp462-ultrascale-SEU.pdf>. Accessed on: July 2016.

DALLY, W. J.; BALFOUR, J.; BLACK-SHAFFER, D.; CHEN, J.; HARTING, R. C.; PARIKH, V.; PARK, J.; SHEFFIELD, D. Efficient Embedded Computing. **Computer**, vol. 41, n. 7, Jul. 2008, pp. 27-32.

DE MICHELI, G. **Synthesis and Optimization of Digital Circuits**. 1. ed. McGraw-Hill Higher Education, 1994.

DE OLIVEIRA, A. B.; TAMBARA, L. A.; KASTENSMIDT, F. L. Exploring Performance Overhead Versus Soft Error Detection in Lockstep Dual-Core ARM Cortex-A9 Processor Embedded into Xilinx Zynq APSoC. **Lecture Notes in Computer Science**, vol. 10216, Mar. 2017, pp. 189-201.

DOOD, P. E.; MASSENGILL, L.W. Basic Mechanisms and Modeling of Single Event Upset in digital Microelectronics. **IEEE Transactions on Nuclear Science**, vol. 53, n. 6, Dec. 2006, pp. 1747-1763.

DODD, P. E.; SHANEYFELT, M. R.; SCHWANK, J. R.; FELIX, J. A. Current and Future Challenges in Radiation Effects on CMOS Electronics. **IEEE Transactions on Nuclear Science**, vol. 57, n. 4, Aug. 2010, pp. 1747-1763.

DOS SANTOS, A. F.; TAMBARA, L. A.; BENEVENUTI, F.; TONFAT, J.; KASTENSMIDT, F. L. Applying TMR in Hardware Accelerators Generated by High-Level Synthesis Design Flow for Mitigating Multiple Bit Upsets in SRAM-Based FPGAs. **Lecture Notes in Computer Science**, vol. 10216, Mar. 2017, pp. 202-213.

DSILVA, D.; WANG, J-J.; REZZAK, N.; JAT, N. Neutron SEE Testing of the 65nm SmartFusion2 Flash-Based FPGA. **IEEE Radiation Effects Data Workshop**, Jul. 2015, pp. 1-5.

DUZELLIER, S.; BERGER, G. Test Facilities for SEE and Dose Testing. In: VELAZCO, R.; FOUILLAT, P.; REIS, R. (Eds.). **Radiation Effects on Embedded Systems**. 1. ed. Berlin: Springer-Verlag Berlin Heidelberg, 2007. pp. 201-232.

ESA. **The Radiation Design Handbook**. ESA, Noordwijk, Netherlands, 1993.

ESA. **ESA/SCC Basic Specification No. 25100: Single Event Effects Test Methods and Guidelines**. ESA, Noordwijk, Netherlands, 2005.

FAJARDO, C. F.; FANG, Z.; IYER, R.; GARCIA, G. F.; LEE, S. E.; ZHAO, L. Buffer-Integrated-Cache: A Cost-Effective SRAM Architecture for Handheld and Embedded Platforms. **ACM/EDAC/IEEE Design Automation Conference**, 2011, pp. 966-971.

FLEMING, S. T.; THOMAS, D. B. StitchUp: Automatic Control Flow Protection for High Level Synthesis Circuits. **ACM/EDAC/IEEE Design Automation Conference**, Jun. 2016, pp. 1-6.

GLASSTONE, S.; EDLUND, M. C. **The Elements of Nuclear Reactor Theory**. Van Nostrand, 1952, pp. 146.

GOLDHAGEN, P. Cosmic-ray Neutrons on the Ground and in the Atmosphere. **MRS Bulletin**, vol. 28, n. 2, 2003, pp. 131-135.

GOODACRE, J.; SLOSS, A. N. Parallelism and the ARM Instruction Set Architecture. **Computer**, vol. 38, n. 7, Jul. 2005, pp. 42-50.

GRASSI, T. FPGA Use Within the Detector Volume. **ECFA High Luminosity LHC Experiments Workshop**, 2014.

GRIEDER, P. **Cosmic Rays at Earth: Researcher's Reference Manual and Data Book**. Elsevier Science Limited, 2001.

GUSSENHOVER, M. S.; MULLEN, E. G.; BRAUTIGAM, D. H. Improved Understanding of the Earth's Radiation Belts from the CRRES Satellite. **IEEE Transactions on Nuclear Science**, vol. 43, n. 2, Aug. 1996, pp. 353-368.

HARA, Y.; TOMIYAMA, H.; HONDA, S.; TAKADA, H.; ISHII, K. CHStone: A Benchmark Program Suite for Practical C-based High-Level Synthesis. **IEEE International Symposium on Circuits and Systems**, May 2008, pp. 1192-1195.

HILL, E. L.; LIPASTI, M. H. The Effect of Pipeline Depth on Logic Soft Errors. **Workshop on Silicon Errors in Logic – System Effects**, Mar. 2010, pp. 1-6.

HILL, M. D.; MARTY, M. R. Amdahl's Law in the Multicore Era. **Computer**, vol. 41, n. 7, Jul. 2008, pp. 33-38.

HSUEH, M. C.; TSAI, T. K.; IYER, R. K. Fault Injection Techniques and Tools. **IEEE Computer**, vol. 30, n. 4, Apr. 1997, pp. 75-82

HUANG, Q.; LIAN, R.; CANIS, A.; CHOI, J.; XI, R.; BROWN, S.; ANDERSON J. The Effect of Compiler Optimizations on High-Level Synthesis for FPGAs. **IEEE Annual International Symposium on Field-Programmable Custom Computing Machines**, Apr. 2013, pp. 89-96.

HUSEJKO, M.; EVANS, J.; DA SILVA, J. C. R. Investigation of High-Level Synthesis Tools' Applicability to Data Acquisition Systems Design Based on the CMS ECAL Data

Concentrator Card Example. **Journal of Physics: Conference Series**, 664 (2015) 082019, pp. 1-8.

IROM, F. Guideline for Ground Radiation Testing of Microprocessors in the Space Radiation Environment. **NASA Electronic Parts and Packaging (NEPP) Program**, JPL Publication 08-13 4/08, 2008.

ITRS. International Technology Roadmap for Semiconductors: 2013 Edition, Chapter Emerging Research Devices (ERD), 2013, pp. 1-5.

ITURBE, X.; KEYMEULEN, D.; YIU, P.; BERISFORD, D.; HAND, K.; CARLSON, R.; OZER, E. Towards a Generic and Adaptive System-on-Chip Controller for Space Exploration Instrumentation. **NASA/ESA Conference on Adaptive Hardware and Systems (AHS)**, 2015, pp. 1-8.

JEDEC. Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices, JESD89A (Revision of JESD89, Aug. 2001), Oct. 2006.

KANAWATI, G.; KANAWATI, N. A.; ABRAHAM, J. A. FERRARI: A Flexible Software-Based Fault and Error Injection System. **IEEE Transactions on Computers**, vol. 44, n. 2, Feb. 1995, pp. 248-260.

KASTENSMIDT, F. L.; CARRO, L.; REIS. R. **Fault-Tolerant Techiniques for SRAM-based FPGAs**. 1. ed. Berlin: Springer-Verlag Berlin Heidelberg, 2006.

KASTENSMIDT, F. L.; TAMBARA, L.; BOBROVSKY, D. V.; PECHENKIN, A. A.; NIKIFOROV, A. Y. Laser Testing Methodology for Diagnosing Diverse Soft Errors in a Nanoscale SRAM-Based FPGA. **IEEE Transactions on Nuclear Science**, vol. 61, n. 6, Dec. 2014, pp. 3130-3137.

KATZ, R.; BARTO, R.; MCKERRACHER, P.; CARKHUFF, B.; KOGA, R. SEU Hardening of Field Programmable Gate Arrays (FPGAs) for Space Applications and Device Characterization. **IEEE Transactions on Nuclear Science**, vol. 41, n. 6, Aug. 1994, pp. 2197-2186.

KUON, I.; ROSE, J. Measuring the Gap Between FPGAs and ASICs. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, vol. 26, n. 2, Jan. 2007, pp. 203-215.

LAFOND, S.; LILIUS, J. Interrupt Costs in Embedded System with Short Latency Hardware Accelerators. **IEEE International Conference and Workshop on the Engineering of Computer Based Systems**, 2008, pp. 317-325.

LEE, V. W.; KIM, C.; CHUGANI, J.; DEISHER, M.; KIM, D.; NGUYEN, A. D.; SATISH, N.; SMELYANSKIY, M.; CHENNUPATY, S.; HAMMARLUND, P.; SINGHAL, R.; DUBEY, P. Debunking the 100X GPU vs. CPU Myth: An Evaluation of Throughput Computing on CPU and GPU. **International Symposium on Computer Architecture**, 2010, pp. 451-460.

LINS, F. M.; TAMBARA, L. A.; KASTENSMIDT, F. L.; RECH, P. Register File Criticality on Embedded Microprocessor Reliability. **European Conference on Radiation and Its Effects on Components and Systems**, 2016. (to be published)

LLVM. The LLVM Compiler Infrastructure, 2016. Available at: <http://llvm.org/>. Accessed on: July 2016.

MANOOCHEHRI, M.; ANNAVARAM, M.; DUBOIS, M. CPPC: Correctable Parity Protected Cache. **International Symposium On Computer Architecture**, 2011, pp. 223-234.

MEKKI, J.; BRUGGER, M.; ALIA, R. G.; THORNTON, A.; DOS SANTOS MOTA, N. C.; DANZECA, S. CHARM: A Mixed Field Facility at CERN for Radiation Tests in Ground, Atmospheric, Space and Accelerator Representative Environments. **IEEE Transactions on Nuclear Science**, v. 63, n. 4, Aug. 2016, p. 2106-2114.

MICROSEMI. SmartFusion SoC FPGAs, 2015. Available at: <http://www.microsemi.com/products/fpga-soc/soc-fpga/smartfusion>. Accessed on: August 2015.

MICROSEMI. SmartFusion2 SoC FPGAs, 2015. Available at: <http://www.microsemi.com/products/fpga-soc/soc-fpga/smartfusion2>. Accessed on: August 2015.

MOGOLLON, J. M.; GUZMÁN-MIRANDA, H.; NÁPOLES, J.; BARRIENTOS, J.; AGUIRRE, M. A. FTUNSHADES2: A Novel Platform for Early Evaluation of Robustness Against SEE. **European Conference on Radiation and Its Effects on Components and Systems**, pp. 169-174, 2011.

MONSON, J.; WIRTHLIN, M.; HUTCHINGS, B. L. Implementing high-performance, low-power FPGA-based optical flow accelerators in C. **IEEE International Conference on Application-Specific Systems, Architectures and Processors**, Jun. 2013, pp. 363-369.

MOORE, G. E. Cramming More Components Onto Integrated Circuits. **Electronics**, vol. 38, n. 8, Apr. 1965, pp. 114-117.

MUKHERJEE, S. S.; WEAVER, C. T.; EMER, J.; REINHARDT, S. K.; AUSTIN, T. Measuring Architectural Vulnerability Factors. **IEEE Micro**, vol. 23, n. 6, Nov. 2003, pp. 70-75.

MUSA, L. FPGAs in High Energy Physics Experiments at CERN. **International Conference on Field Programmable Logic and Applications (FPL)**, pp. 2, 2008.

MUSHTAQ, H.; AL-ARS, Z.; BERTELS, K. Survey of Fault Tolerance Techniques for Shared Memory Multicore/Multiprocessor Systems. **IEEE International Design and Test Workshop**, pp. 12-17, 2011.

NANE, R.; SIMA, V.-M.; PILATO, C.; CHOI, J.; FORT, B.; CANIS, A.; CHEN, Y. T.; HSIAO, H.; BROWN, S.; FERRANDI, F.; ANDERSON, J.; BERTELS, K. A Survey and

Evaluation of FPGA High-Level Synthesis Tools. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, vol. PP, n. 99, Dec. 2015, pp. 1.

NASA. Solar Cycle Primer, 2011. Available at: <https://www.nasa.gov/mission_pages/sunearth/news/solarcycle-primer.html>. Accessed on: September 2016.

PATTERSON, D. A.; HENNESSY, J. L. **Computer Organization and Design – The Hardware/Software Interface**. 3. Ed. San Francisco: Elsevier, 2005.

POLLACK, F. J. New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies. **ACM/IEEE International Symposium on Microarchitecture**, 1999, pp. 2.

QUINN, H.; GRAHAM, P. Terrestrial-based Radiation Upsets: A Cautionary Tale. **IEEE Symposium on Field-Programmable Custom Computing Machines**, 2005, pp. 193-202.

QUINN, H. Challenges in Testing Complex Systems. **IEEE Transactions on Nuclear Science**, vol. 61, n. 2, Apr. 2014, pp. 766-786.

QUINN, H.; FAIRBANKS, T.; TRIPP, J. L.; DURAN, G.; LOPEZ, B. Single-Event Effects in Low-Cost, Low-Power Microprocessors. **IEEE Radiation Effects Data Workshop**, Jul. 2014, pp. 1-9.

QUINN, H.; BAKER, Z.; FAIRBANKS, T.; TRIPP, J. L.; DURAN, G. Software Resilience and the Effectiveness of Software Mitigation in Microcontrollers. **IEEE Transactions on Nuclear Science**, vol. 62, n. 6, Dec. 2015, pp. 2532-2538.

QUINN, H.; ROBINSON, W. H. RECH, R.; AGUIRRE, M.; BARNARD, A.; DESOGUS, M.; ENTRENA, L.; GARCIA-VALDERAS, M.; GUERTIN, S. M.; KAELI, D.; KASTENSMIDT, F. L.; KIDDIE, B. T.; SANCHEZ-CLEMENTE, A.; REORDA, M. S.; STERPONE, L.; WIRTHLIN, M. Using Benchmarks for Radiation Testing of Microprocessors and FPGAs. **IEEE Transactions on Nuclear Science**, vol. 62, n. 6, Dec. 2015, pp. 2547-2554.

RECH, P.; PILLA, L. L.; NAVAUX, P. O. A.; CARRO, L. Impact of GPUs Parallelism Management on Safety-Critical and HPC Applications Reliability. **IEEE/IFIP International Conference on Dependable Systems and Networks**, Jun. 2014, pp. 455-466.

REZZAK, N.; DSILVA, D.; WANG, J-J.; JAT, N. SET and SEFI Characterization of the 65 nm SmartFusion2 Flash-Based FPGA under Heavy Ion Irradiation. **IEEE Radiation Effects Data Workshop**, Jul. 2015, pp. 1-4.

RUDOLPH, D.; WILSON, C.; STEWART, J.; GAUVIN, P; GEORGE, A.; LAM, H; CRUM, G; WIRTHLIN, M; WILSON, A.; STODDARD, A. CSP: A Multifaceted Hybrid Architecture for Space Computing. **AIAA/USU Conference on Small Satellites**, SSC14-III-3, 2014, pp. 1-7.

SADRI, M.; WEIS, C.; WEHN, N.; BENINI, L. Energy and Performance Exploration of Accelerator Coherency Port Using Xilinx ZYNQ. **FPGAworld Conference**, 2013, pp. 1-8.

SANTINI, T.; CARRO, L.; WAGNER, F. R.; RECH, P. Reliability Analysis of Operating Systems and Software Stack for Embedded Systems. **IEEE Transactions on Nuclear Science**, vol. PP, n. 99, Mar. 2016, pp. 1-8.

SCHWANK, J. R.; SHANEYFELT, M. R.; DODD, P. E. Radiation Hardness Assurance Testing of Microelectronic Devices and Integrated Circuits: Radiation Environments, Physical Mechanisms, and Foundations for Hardness Assurance. **IEEE Transactions on Nuclear Science**, vol. 60, n. 3, Jun. 2013, pp. 2074-2100.

SHEN, H.; PÉTROT, F. Using Amdahl's Law for Performance Analysis of Many-Core SoC Architectures Based on Functionally Asymmetric Processors. In: BEREKOVIC, M.; FORNACIARI, W.; BRINKSCHULTE, U.; SILVANO, C. (Eds.). **Architecture of Computing Systems – ARCS 2011**. 1. ed. Berlin: Springer-Verlag Berlin Heidelberg, 2011. pp. 38-49.

SILVA, J.; SKLYAROV, V.; SKLIAROVA, I. Comparison of On-chip Communications in Zynq-7000 All Programmable Systems-on-Chip. **IEEE Embedded Systems Letters**, vol. 7, n. 1, Feb. 2015, pp. 31-34.

SOOS, C. SEU Effects in FPGAs. **CERN R2E Radiation School**, 2009.

SRIM. Particle Interactions with Matter, 2013. Available at: <http://www.srim.org/>. Accessed on: March 2015.

STEFAN. Institut Jožef Stefan. Understanding the Muon Lifetime Experiment, 2001. Available at: <http://www-f9.ijs.si/~rok/sola/praktikum4/mioni/muonexp.html>. Accessed on: May, 2016.

STERPONE, L.; VIOLANTE, M.; REZGUI, S. An Analysis Based on Fault Injection of Hardening Techniques for SRAM-Based FPGAs. **IEEE Transactions on Nuclear Science**, vol. 53, n. 4, Aug. 2006, pp. 2054-2059.

TAMBARA, L. A.; RECH, P.; CHIELLE, E.; TONFAT, J.; KASTENSMIDT, F. L. Analyzing the Impact of Radiation-induced Failures in Programmable SoCs. **IEEE Transactions on Nuclear Science**, vol. 63, n. 4, Aug. 2016, pp. 1-8.

TARRILLO, J. F. Exploring the Use of Multiple Modular Redundancies for Masking Accumulated Faults in SRAM-based FPGAs. 112 p. Thesis (PhD) – UFRGS, 2014.

TARRILLO, J.; TONFAT, J.; TAMBARA, L.; KASTENSMIDT, F. L.; REIS, R. Multiple Fault Injection Platform for SRAM-based FPGA Based on Ground-level Radiation Experiments. **IEEE Latin-American Test Symposium**, Mar. 2015, pp. 1-4.

TONFAT, J. L. Frame-Level Redundancy Scrubbing Technique for SRAM-based FPGAs. 111 p. Thesis (PhD) - UFRGS, 2015.

TONFAT, J. L.; TAMBARA, L. A.; SANTOS, A.; KASTENSMIDT, F. L. Method to Analyze the Susceptibility of HLS Designs in SRAM-Based FPGAs Under Soft Errors. **Lecture Notes in Computer Science**, vol. 9625, Mar. 2016, pp. 132-143.

TSAO, C. H.; SILBERBERG, R.; LETAW, J. R. Cosmic-ray Heavy Ions at and Above 40,000 Feet. **IEEE Transactions on Nuclear Science**, vol. 31, n. 6, Dec. 1984, pp. 1066-1068.

VELAZCO, R.; FAURE, F. Error Rate Prediction of Digital Architectures: Test Methodology and Tools. In: VELAZCO, R.; FOUILLAT, P.; REIS, R. (Eds.). **Radiation Effects on Embedded Systems**. 1. ed. Berlin: Springer-Verlag Berlin Heidelberg, 2007. pp. 233-258.

VELAZCO, R.; FOUCARD, G.; PERONNARD, P. Combining Results of Accelerated Radiation Tests and Fault Injections to Predict the Error Rate of an Application Implemented in SRAM-Based FPGAs. **IEEE Transactions on Nuclear Science**, vol. 57, n. 6, Dec. 2010, pp. 3500-3505.

VELAZCO, R.; FOUCARD, G.; PERONNARD, P. Integrated Circuit Qualification for Space and Ground-Level Applications: Accelerated Tests and Error-Rate Predictions. In: NICOLAIDIS, M. (Ed;). **Soft Errors in Modern Electronic Systems**. 1. ed. Berlin: Springer-Verlag Berlin Heidelberg, 2011. pp. 167-201.

VIOLANTE, M.; STERPONE, L.; MANUZZATO, A.; GERARDIN, S.; RECH, P.; BAGATIN, M.; PACCAGNELLA, A.; ANDREANI, C.; GORINI, G.; PIETROPAOLO, A.; CARDARILLI, G.; PONTARELLI, C.; FROST, C. A New Hardware/Software Platform and a New 1/E Neutron Source for Soft Error Studies: Testing FPGAs at the ISIS Facility. **IEEE Transactions on Nuclear Science**, vol. 54, n. 4, Aug. 2007, pp. 1184-1189.

VIPIN, K.; FAHMY, S. A. ZyCAP: Efficient Partial Reconfiguration Management on the Xilinx Zynq. **IEEE Embedded Systems Letters**, vol. 6, n. 3, Sept. 2014, pp. 41-44.

XILINX. Considerations Surrounding Single Event Effects in FPGAs, ASICs, and Processors. WP402 (v.1.0.1) Mar. 7, 2012.

XILINX. Soft Error Mitigation Using Prioritized Essential Bits. XAPP (v1.0) April 4, 2012.

XILINX. Virtex-5 FPGA Configuration User Guide. UG191 (v3.11) October 19, 2012.

XILINX. Introduction to High-Level Synthesis with Vivado HLS. 2013.

XILINX. 7 Series FPGAs Configurable Logic Block – User Guide. UG474 (v.17) November 17, 2014.

XILINX. 7 Series FPGAs Configuration – User Guide. UG470 (v1.10) June 24, 2015.

XILINX. Mitigating Single-Event Upsets. WP395 (v1.1) May 19, 2015.

XILINX. Soft Error Mitigation Controller v4.1. PG036 September 30, 2015.

XILINX. Zynq-7000 All Programmable SoC, 2015. Available at: <http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html>. Accessed on: August 2015.

XILINX. A Zynq Accelerator for Floating Point Matrix Multiplication Designed with Vivado HLS. XAPP1170 (v2.0), 2016.

XILINX. Applications, 2016. Available at: <http://www.xilinx.com/>. Accessed on: February 2016.

XILINX. Device Reliability Report – Second Half 2015. UG116 (v10.4) April 1, 2016.

XILINX. Multi-Camera Driver Assistance Platform, 2016. Available at: <http://www.xilinx.com/applications/automotive/multi-camera-adas.html>. Accessed on: February 2016.

XILINX. Vivado Design User Guide – High-Level Synthesis. UG902 (v2016.1), Apr. 6, 2016.

XILINX. Zynq-7000 All Programmable SoC – Technical Reference Manual. UG585 (v1.11), Sept. 27, 2016.

WANG, F.; AGRAWAL, V. D. Single Event Upset: An Embedded Tutorial. **International Conference on VLSI Design**, Jan. 2008, pp. 429-434.

WINDH, S.; XIAOYIN, M.; HALSTEAD, R. J.; BUDHKAR, P.; LUNA, Z.; HUSSAINI O.; NAJJAR, W. A. High-Level Language Tools for Reconfigurable Computing. **Proceedings of IEEE**, vol. 103, n. 3, Mar. 2015, pp. 390-408.

WINTERSTEIN, F.; BAYLISS, S.; CONSTANTINIDES, G. A. High-level Synthesis of Dynamic Data Structures: A Case Study Using Vivado HLS. **International Conference on Field-Programmable Technology**, Dec. 2013, pp. 362-365.

WIRTHLIN, M.; LEE, D.; SWIFT, G.; QUINN, H. A Method and Case Study on Identifying Physically Adjacent Multiple-Cell Upsets Using 28-nm, Interleaved and SECDED-Protected Arrays. **IEEE Transactions on Nuclear Science**, vol. 61, n. 6, Dec. 2014, pp. 3080-3087.

WIRTHLIN, M. High-Reliability FPGA-Based Systems: Space, High-Energy Physics, and Beyond. **Proceedings of the IEEE**, vol. 103, n. 3, Apr. 2015, pp. 379-389.

WU, W.; BEGEL, M.; CHEN, H.; CHEN, K.; LANNI, F.; TAKAI, H. The Development of the Global Feature Extractor for the LHC Run-3 Upgrade of the L1 Calorimeter Trigger System. **IEEE Nuclear and Plasma Sciences Society Real Time Conference**, Jun. 2016, pp. 1-3.

ZHAO, C.; ALME, J.; ALT, T.; APPELSHAUSER, H.; BRATRUD, L.; CASTRO, A.; COSTA, F.; DAVID, E.; GUNJI, T.; KIRSCH, S.; KISS, T.; LANGOY, R.; LIEN, J.; SEKIGUCHI, Y.; STUART, M.; ULLALAND, K.; VELURE, A.; YANG, S.; OSTERMA, L. First Performance Results of the ALICE TPC Readout Control Unit 2, **Journal of Instrumentation**, vol. 11, Jan. 2016, pp. 1-12.