A strategy for soft-error vulnerability estimation using the single-event transient susceptibilities of each gate

Fábio Batagin Armelin

université PARIS-SACLAY

NNT : 2019SACLT035

Thèse de doctorat

Stratégie d'estimation de la vulnérabilité aux erreurs 'soft' basée sur la susceptibilité aux événements transitoires de chaque porte logique

A strategy for soft-error vulnerability estimation using the single-event transient susceptibilities of each gate

Thèse de doctorat de l'Université Paris-Saclay
préparée à Télécom ParisTech

Ecole doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : réseaux, information et communications

Thèse présentée et soutenue à São Paulo, le 27 Août 2019, par

FÁBIO BATAGIN ARMELIN

Composition du Jury :

Osamu SAOTOME *Professor Titular*, Instituto Tecnológico de Aeronáutica	President
Ricardo REIS *Professor Titular*, Universidade Federal do Rio Grande do Sul	Rapporteur
Eduardo DA COSTA *Professor Titular*, Universidade Católica de Pelotas	Rapporteur
Salvador GIMENEZ *Professor Titular*, Fundação Educacional Inaciana	Examinateur
Pietro FERREIRA Maître de Conférences, CentraleSupélec	Examinateur
Lírida NAVINER Professeur, Télécom ParisTech	Directrice de thèse
Roberto D'AMORE *Professor Associado*, Instituto Tecnológico de Aeronáutica	Co-Directeur de thèse

Acknowledgments

I would like to thank Lírida Naviner and Roberto d'Amore, my thesis advisors at Télécom ParisTech and Instituto Tecnológico de Aeronáutica, not only for our scientific discussions and their guidance but also for their patience and trust in me.

To the thesis committee, Ricardo Reis and Eduardo da Costa, the *rapporteurs*, and Salvador Gimenez, Pietro Ferreira and Osamu Saotome, the *examinateurs* of the thesis, for their constructive remarks about this work.

To Mario Selingardi, head of the Divisão de Eletrônica Aeroespacial, and Tatiana Kuplich, head of the Centro Regional Sul de Pesquisas Espaciais, for allowing me to develop this work. Additionally, I would like to thank my colleagues from the Grupo de Supervisão de Bordo, for the discussions about this work, and especially Fabrício Kucinskis, for his detailed revisions.

To Conselho Nacional de Desenvolvimento Científico e Tecnológico and Agência Espacial Brasileira for the financial support (grant number 207364/2015-0/SWE).

To my parents, Olívio (In Memoriam) and Iracema, and my sister, Cláudia, for their love and support even in absent times.

Finally, to my wife, Marina, and our daughter, Clara, that patiently followed all the process of the thesis development, and, in their own way, supported and encouraged its conclusion.

Abstract

The Soft-Error Vulnerability (SEV) is an estimated parameter that, in conjunction with the characteristics of the radiation environment, is used to obtain the Soft-Error Rate (SER), that is a metric used to predict how digital systems will behave in this environment. Currently, the most reliable method for SER estimation is the radiation test, since it has the actual interaction of the radiation with the electronic device. However, this test is expensive and requires the real device, that becomes available late on the design cycle. These restrictions motivated the development of other SER and SEV estimation methods, including analytical, electrical and logic simulations, and emulation approaches. These techniques usually incorporate the logical, electrical and latching-window masking effects into the estimation process. Nevertheless, most of them do not take into account a factor that is intrinsic to the radiation test: the probability of the radiation particle of producing a Soft-Error (SE) at the output of the logic gates of the circuit, referred to as Single-Event Transient (SET) susceptibility. In this context, we propose a strategy for SEV estimation based on these SET susceptibilities, suitable for simulation- and emulation-based frameworks. In a simplified version of this strategy, the SET susceptibilities take into account only the effects of the logic gate internal circuitry, while in a complete version, these susceptibilities consider both the internal circuitry and the operation of the circuit, that affects its input pattern distribution. The proposed strategy was evaluated with a simulation-based framework, estimating the SEV of 38 benchmark circuits. The average estimation error was reduced from 15.27%, ignoring the susceptibilities, to 4.70%, for the simplified version, and to 0.68%, for the complete version. Finally, we discuss the feasibility of adopting the proposed strategy with an emulation-based framework.

List of Figures

FIGURE 2.1 – Trapped Radiation – The Van Allen Belts. 27

FIGURE 2.2 – MOPITT Device Single Events (DSEs). 28

FIGURE 2.3 – SET modelling for a CMOS inverter. 32

FIGURE 2.4 – Illustration of a SEU. 33

FIGURE 2.5 – Logical masking factor. 37

FIGURE 2.6 – Logical masking factor at the transistor level. 38

FIGURE 2.7 – Electrical masking factor. 39

FIGURE 2.8 – Latching-window masking factor. 40

FIGURE 2.9 – Basic elements of an FPGA. 50

FIGURE 2.10 –Configurable Logic Block of the Xilinx XC2000 FPGA family. 51

FIGURE 2.11 –Programmable Logic Block (PLB) of the Lattice Semiconductor iCE40 UltraPlus FPGA family. 51

FIGURE 2.12 –Partial representation of the Configuration Logic Block of the Xilinx Spartan-3 FPGA family. 52

FIGURE 2.13 –Adaptive Logic Module (ALM) of the Logic Array Block (LAB) of the Intel Stratix 10 FPGA family. 53

FIGURE 2.14 –Adaptive Logic Module (ALM) of the Logic Array Block (LAB) of the Intel Agilex FPGA family. 53

FIGURE 2.15 –Configuration Logic Block used in the Microsemi IGLOO, ProASIC3, and Fusion FPGA families and their variants. 54

FIGURE 3.1 – Proposed framework for SEV estimation. 56

FIGURE 3.2 – Simplified proposed strategy for SEV estimation. 57

FIGURE 3.3 – Complete proposed strategy for SEV estimation. 59

FIGURE 3.4 – Comparison of VHDL files before and after adding the saboteurs. . . 62

FIGURE 3.5 – Example circuit for input patterns extraction. 63

FIGURE 3.6 – Graphical visualisation of the example VCD file. 63

FIGURE 4.1 – Adopted circuits for the internal logic gates of the Versatile CLB. . 72

FIGURE 4.2 – VersaTile internal structure with the adopted IDs. 72

FIGURE 4.3 – VersaTile configured as an AND3 macro. 78

FIGURE 4.4 – VersaTile configured as an DFN1C0 macro. 78

FIGURE 4.5 – SET susceptibilities of the 2-input combinational functions. 83

FIGURE 4.6 – SET susceptibilities of some 3-input flip-flop macros. 83

FIGURE 4.7 – SET susceptibilities of some 3-input latch macros. 83

FIGURE 4.8 – Percentage of generated SEUs for some 3-input flip-flop macros. . . 83

FIGURE 4.9 – Percentage of generated SEUs for some 3-input latch macros. 84

FIGURE 5.1 – Errors of the SEV estimation processes. 97

FIGURE 5.2 – Simulation time of the SEV estimation processes. 99

FIGURE 6.1 – Proposed emulation-based platform. 102

FIGURE 6.2 – Illustration of the serial message reporting a SET injection. 105

FIGURE 6.3 – Illustration of the serial message reporting an observed SE. 105

FIGURE 6.4 – Time vs. place uniform SET distribution for the *c8* benchmark. . . 111

FIGURE 6.5 – Time vs. place weighted SET distribution for the *c8* benchmark. . . 111

FIGURE 6.6 – Time vs. place uniform SET distribution for the *comp* benchmark. . 111

FIGURE 6.7 – Time vs. place weighted SET distribution for the *comp* benchmark. 112

FIGURE 6.8 – Uniform SET distribution for the *c8* benchmark. 114

FIGURE 6.9 – Weighted SET distribution for the *c8* benchmark. 114

FIGURE 6.10 –Uniform SET distribution for the *comp* benchmark. 114

FIGURE 6.11 –Weighted SET distribution for the *comp* benchmark. 115

FIGURE 6.12 –Comparison between SEV estimation using logic simulation and emulation. 117

FIGURE C.1 – SET quantised delay model. 151

FIGURE C.2 – SET emulation considering electrical masking effects. 151

FIGURE C.3 – Electrical pulse generation circuit. 153

FIGURE C.4 – Saboteur Candidates. 154

FIGURE C.5 – Transient electrical pulses observed at an output of the FPGA. . . . 156

FIGURE C.6 – Dissociation between pulses and system clock. 156

FIGURE C.7 – Comparison between local and global net for pulse distribution. . . . 158

FIGURE D.1 – Cadre proposé pour l'estimation SEV. 169

List of Tables

TABLE 3.1 – Intermediate table of the input patterns extraction process. 65

TABLE 4.1 – ProASIC3E A3PE1500-PQ208 main features. 70

TABLE 4.2 – VersaTile transistors. 73

TABLE 5.1 – Combinational benchmark circuits. 87

TABLE 5.2 – SEV estimation at the transistor level 90

TABLE 5.3 – SEV estimation adopting uniform distribution 92

TABLE 5.4 – SEV estimation adopting the simplified strategy 94

TABLE 5.5 – SEV estimation adopting the complete strategy 96

TABLE 6.1 – Benchmark circuits used in emulation. 107

TABLE 6.2 – Resources consumption of the emulation platform with the *cm152a* benchmark. 108

TABLE 6.3 – Resources consumption of the emulation platform with the *cmb* benchmark. 108

TABLE 6.4 – Resources consumption of the emulation platform with the *c8* benchmark. 109

TABLE 6.5 – Resources consumption of the emulation platform with the *comp* benchmark. 109

TABLE 6.6 – Estimated SEV using the emulation platform. 116

TABLE A.1 – Configuration vectors for the combinational macros. 133

TABLE A.2 – Configuration vectors for the flip-flops. 137

TABLE A.3 – Configuration vectors for the latches. 139

TABLE B.1 – SET susceptibilities for the combinational macros. 140

TABLE B.2 – SET susceptibilities for the flip-flops. 144

TABLE B.3 – SEU susceptibilities for the flip-flops. 146

TABLE B.4 – SET susceptibilities for the latches. 148

TABLE B.5 – SEU susceptibilities for the latches. 149

TABLE C.1 – Resulting pulse widths for SET emulation. 157

TABLE D.1 – Circuits combinatoire de référence – Caractéristiques et résultats de l'estimation SEV. 178

List of Abbreviations and Acronyms

2D	Two-Dimensional
3D	Three-Dimensional
ALM	Adaptive Logic Module
AMUSE	Autonomous Multilevel emulation system for Soft Error evaluation
ASEE	Analogue Single-Event Effects
ASET	Analogue Single-Event Transient
ASEU	Analogue Single-Event Upset
ASIC	Application-Specific Integrated Circuit
CCC	Clock Conditioning Circuits
CCD	Charged-Coupled Device
CLB	Configuration Logic Block
CME	Coronal Mass Ejection
CMOS	Complementary Metal Oxide Semiconductor
CSV	Comma-Separated Value
DD	Displacement Damage
DRAM	Dynamic Random-Access Memory
DSE	Device Single Event
DSET	Digital Single-Event Transient
DUT	Device Under Test
EPP	Error Propagation Probability
FERRARI	Fault and Error Automatic Real-Time Injection
FET	Field Effect Transistor
FIFA	Fault Injection and Fault masking Analysis
FIFO	First-In First-Out
FIT	Failures In Time
FPGA	Field Programmable Gate Array

GCR	Galactic Cosmic Radiation
HDL	Hardware Description Language
HZE	High Z and Energy
IC	Integrated Circuit
IOC	Input and Output Cell
LAB	Logic Array Block
LC	Logic Cell
LED	Light-Emitting Diode
LEO	Low Earth Orbit
LET	Linear Energy Transfer
LFSR	Linear-Feedback Shift Registers
LP	Latching Probability
LUT	LookUp Table
LVTTL	Low-Voltage Transistor Transistor Logic
MBU	Multiple-Bit Upset
MCU	Multiple-Cell Upset
MEFISTO	Multi-level Error/Fault Injection Simulation Tool
MOPITT	Measurements Of Pollution In The Troposphere
MOS	Metal Oxide Semiconductor
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MTBF	Mean Time Between Failures
NIEL	Non-Ionising Energy Loss
NLC	Non-Linear Counter
NMOS	N-channel MOSFET
OSVVM	Open Source VHDL Verification Methodology
OTP	One-Time Programmable
PGM	Probabilistic Gate Models
PGP	Pulse Generation Probability
PLB	Programmable Logic Block
PLL	Phase-Locked Loop
PMOS	P-channel MOSFET
PRNG	Pseudo-Random Number Generator
PROM	Programmable Read-Only Memory
PTM	Probabilistic Transfer Matrices
PVW	Probabilistic Vulnerability Window

RAM	Random-Access Memory
ROM	Read-Only Memory
RTL	Register-Transfer Level
SAMA	South-Atlantic Magnetic Anomaly
SCFIT	Shadow Components-based Fault Injection Technique
SCM	Stochastic Computational Model
SCR	Silicon Controlled Rectifier, Solar Cosmic Radiation
SE	Soft-Error
SEB	Single-Event Burnout
SEE	Single-Event Effect
SEFI	Single-Event Functional Interrupt
SEGR	Single-Event Gate Rupture
SEL	Single-Event Latch-up
SER	Soft-Error Rate
SET	Single-Event Transient
SEU	Single-Event Upset
SEV	Soft-Error Vulnerability
SHE	Single-Event Hard Error
SoC	System on Chip
SP	Signal Probability
SPE	Solar Proton Event
SPICE	Simulation Program with Integrated Circuit Emphasis
SRAM	Static Random-Access Memory
TAP	Test Access Port
TCS	Triple Constraint Satisfaction
TID	Total Ionising Dose
TMR	Triple Modular Redundancy
USB	Universal Serial Bus
VCD	Value Change Dump
VERIFY	VHDL-based Evaluation of Reliability by Injecting Faults Efficiently
VFIT	VHDL-based Fault Injection Tool
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuits
VITAL	VHDL Initiative Towards ASIC Libraries

List of Symbols

ϕ	particle flux
φ	particle fluency
σ	cross-section
χ	susceptibilidade
D	absorbed dose
E	particle energy
R	SEE rate
Z	atomic number

Contents

1 Introduction 17

1.1 Motivations 20

1.2 Objectives and Contributions 21

1.3 Text Organisation 23

2 Concepts and Literature Review 25

2.1 Radiation Effects on Electronic Devices 26

2.1.1 Radiation Environment 26

2.1.2 Radiation Interaction with the Matter 29

2.1.3 Classification of the Radiation Effects on the Electronics Devices . . . 31

2.1.4 Soft-Error Rate and Soft-Error Vulnerability 34

2.2 Error-Masking Factors 36

2.2.1 Logical Masking 37

2.2.2 Electrical Masking 39

2.2.3 Latching-Window Masking 39

2.3 Methods for Soft-Error Analysis 40

2.3.1 Analytical 41

2.3.2 Electrical Simulation 42

2.3.3 Logic Simulation 43
2.3.4 Emulation 44
2.3.5 Software Fault-injection 47
2.3.6 Hardware Tests 47
2.3.7 Radiation Tests 48
2.4 SET Susceptibility on SEV Estimation 49
2.5 CLB Architectures 50

3 Proposed Strategy for SEV Estimation 55
3.1 Simplified Version of the Strategy 57
3.2 Complete Version of the Strategy 58
3.3 Adding Saboteurs 61
3.4 Extraction of the Input Patterns Distribution 63
3.5 Calculation of the SET Susceptibilities 66
3.6 Generating the Testbench for SEV Estimation 67

4 Estimation of the SET Susceptibilities 69
4.1 Analysed Technology 70
4.2 CLB Simulation Model 71
4.3 Definition of the Configuration Vectors 76
4.4 CLB Library 79
4.5 Estimation of the SET Susceptibilities 80
4.6 Results 81

5 Evaluation of the Proposed Strategy 85

5.1 Analysed Benchmarks . . . 86

5.2 Operational Scenarios . . . 88

5.3 SEV Estimation at the Transistor Level . . . 88

5.4 SEV Estimation adopting Uniform Distribution . . . 91

5.5 SEV Estimation adopting the Simplified Strategy . . . 93

5.6 SEV Estimation adopting the Complete Strategy . . . 95

5.7 Comparative Analysis . . . 97

6 Using the Proposed Strategy with Emulation . . . 100

6.1 Use with Host-Based Emulation Approach . . . 101

6.2 Use with Autonomous Emulation Approach . . . 101

6.3 Proposed Emulation-Based Framework . . . 102

6.4 SET Distribution Scheme . . . 105

6.5 Evaluation of the Proposed Emulation Platform . . . 106

6.5.1 Analysed Circuits . . . 107

6.5.2 Resources Consumption . . . 107

6.5.3 SET Distribution Analysis . . . 110

6.5.4 Estimated SEV . . . 115

7 Conclusions and Perspectives . . . 118

7.1 Main Contributions . . . 119

7.2 Future Work . . . 121

7.3 Publications . . . 123

7.3.1 Conference Proceedings . . . 123

7.3.2 Journal Article . . . 123

Bibliography 124

Appendix A – Configurations of the VersaTile Functions 133

Appendix B – SET Susceptibilities of the VersaTile Functions 140

Appendix C – Self-Produced Transients for SET Emulation 150

C.1 Context 150

C.2 Evaluation Process 152

C.2.1 Evaluation Environment 152

C.2.2 Analysed Characteristics 154

C.2.3 SET Emulations 155

C.3 Evaluation Test Cycles 155

Appendix D – Résumé Français 159

D.1 Introduction 159

D.2 Concepts et revue de littérature 162

D.3 Stratégie proposée pour l'estimation SEV 168

D.4 Estimation des susceptibilités SET 173

D.5 Évaluation de la stratégie proposée 175

D.6 Utilisation de la stratégie proposée avec émulation 179

D.7 Conclusions 180

1 Introduction

In digital systems, a Soft Error (SE) is a wrong value in a signal or data register that is not permanent, i.e., the system can restore its operational state after the treatment of the SE. Contrarily, a hard error is a permanent effect, usually caused by damage, but can also be a design mistake, for instance.

There are many possible causes for SEs, including voltage scaling, ageing, crosstalk, electromagnetic interference, hazard effects, and radiation. The occurrence of a SE commonly depends on some device characteristics and external factors, as the environment and the device operation.

For example, the SEs produced due to the voltage scaling depends on the operational temperature of the device. The ageing leads to a permanent change in the device characteristics that may result in timing issues for some operational scenarios and environmental conditions, generating SEs. If the error due to the ageing effects occurs independently of the environment and operation, it becomes a hard error.

Similarly, crosstalk depends on the susceptibility of the circuit and the characteristics of the signal that induces the SE between circuit lines. The electromagnetic interference depends on the characteristics of the electromagnetic signal and the circuit susceptibility.

The hazard effects are a significant concern on asynchronous circuits, in which they may occur due to the implementation of the circuit and the sequence of the signal changes, likewise the previous cases.

The radiation effects may induce SEs due to the ionisation of the electronic device. Thus, they also depends on the susceptibility of the circuit and the characteristics of the

radiation environment.

Although the adopted definition of a SE covers any cause, in many works the term Soft Error is associated exclusively to the radiation effects. This work also focuses on the SEs caused by radiation. More specifically, the SEs that are directly produced by the Single-Event Effects (SEEs), that is the class of ionising radiation effect generated by a single ionisation particle. Differently, the Total Ionising Dose (TID) is another class of ionising effect that may resemble the ageing effect and could indirectly cause SEs, but is out of the scope of this work.

The first SEE of interest is Single-Event Transient (SET), that is a temporary change in the logic value of a node, caused by an excursion on its voltage. This voltage excursion results from the transient current pulse produced by the recombination of the ions induced by the ionisation particle.

The SET is generated in the sensitive area of the transistors that compose the logic elements and can affect them in different ways. For the combinational elements (logic gates), the SET can propagate to the output or can be filtered out internally. For the sequential elements (registers – latches and flip-flops), the SET can also propagate to the output or be filtered, but these are minor effects. For them, the primary effect is the Single-Event Upset (SEU), that is the change of the stored value, that holds until the next write cycle. Similarly, for the memory cells, the SEU is also the major concern.

The reliability assessment of digital systems adopts the Soft-Error Rate (SER) as the metric associated with the SEs. This parameter is intrinsically related to the rate of transient electrical pulses, R, generated in the transistors of the circuit. However, the circuit's SER is not the sum, or combination, of the R value of each transistor. The logical, electrical, and latching-window masking effects filter many transient pulses. It is difficult to obtain a mathematical formula for the SER, based on the radiation and circuit characteristics, that also includes these masking effects. This difficulty led to the development of many SER estimation methods.

The SER estimation methods may adopt analytical, simulation, emulation, and radia-

tion tests approaches, among others. However, except for the radiation tests, they do not deal directly with the SER, as it requires the ionising particle flux. Instead, they deal with the probability of occurrence of a SE, adopting terms like SE sensitivity, susceptibility or vulnerability. In this work, we adopt two terms. For the probability of occurrence of SEs at the output of the gate-level components due to the SETs generated in their internal transistors, we use the term SET Susceptibility. For the probability of occurrence of SEs at the output of the circuit, we adopt the term Soft-Error Vulnerability (SEV).

We differentiate these two probabilities because we focus on the influence of the SET susceptibilities when evaluating the SEV at the gate level. This influence is taken into account by some classes of estimation methods (analytical and electrical simulation) but ignored by others (logic simulation and emulation), as briefly introduced below.

In Buard and Anghel (2011), the authors present a logic gate modelling and simulation method at the transistor level using SPICE (Simulation Program with Integrated Circuit Emphasis). Their method intrinsically includes the internal logical masking, with the influence of both the logic gate internal circuitry and the input values (hereafter referred to as input patterns). It also includes the electrical masking, through the electrical modelling of the transistors and the SETs. Their results show that both logical and electrical masking effects affect the SET susceptibility of the analysed logic gate.

The analytical method proposed in Rezaei *et al.* (2014) also includes the internal masking effects of the logic gates. A major concern of the authors is how the input patterns influence the SET generation at the output of the logic gates. Their results also show that the SET susceptibilities are affected by these factors and, additionally, how they influence the circuit SEV.

In opposition, the logic-simulation methods presented in Jenn *et al.* (1994), Baraza *et al.* (2005), and Lopes Filho and D'Amore (2012), and the emulation methods presented in Civera *et al.* (2001), Entrena *et al.* (2012), and Ebrahimi *et al.* (2014) do not take the SET susceptibilities into account. Unlike the previously mentioned methods, these do not deal with transistors. They work directly at the Register Transfer Level (RTL) for

logic simulation, or gate-level for emulation, on the output of its components. As they inject the SEs uniformly, they intrinsically assume that all components have the same SET susceptibility.

In this context, we propose a SEV estimation strategy that takes into account the SET susceptibilities of the logic elements of the circuit. This approach is intended for use with a logic-simulation approach, but may be adapted for use with emulation platforms. The SET injection process follows a weighted distribution, for which the weights are the SET susceptibilities of the logic elements.

Since this probability-aware approach requires accessing the output of the logic elements, the simulation requires the use of the post-synthesis back-annotated file, that is a netlist with the logic elements for the target technology. This approach can be applied to both Application-Specific Integrated Circuits (ASICs) and Field Programmable Gate Arrays (FPGAs). However, in this work, we targeted only a specific FPGA family. For this reason, hereafter, we usually refer to the logic elements directly as the CLBs (Configuration Logic Blocks) of an FPGA.

The adoption of the proposed SEV estimation method requires the SET susceptibilities for the target technology. As this information is not available, we estimated the SET susceptibilities of the CLBs of the selected technology.

To evaluate the proposed approach, we estimated the SEV for a set of benchmark circuits. Finally, we explore the adoption of the proposed strategy with an emulation platform.

1.1 Motivations

The SER is an essential reliability metric to evaluate how the electronics systems behave in the radiation environment. Its estimation is part of the radiation analysis and is useful for supporting design decisions that can affect since the adopted technology to the architectural solution. For this reason, the SER estimation needs to be obtained as

soon as possible in the design cycle.

In critical applications, as some aerospace projects and nuclear plants, it is common to adopt conservative approaches, in which they only use radiation-hardened or tolerant devices. These devices will not produce or produce very few SEs due to the nominal expected radiation. In these cases, a conservative approach for the SER estimation may be enough for the radiation analysis, approving or rejecting the use of the device or design solution.

However, depending on the application, a higher SER may be acceptable, although it is always desirable to minimise it, pursuing an optimal solution. In these cases, a more accurate SER estimation approach would be more suitable than a conservative one, since it enables the comparison between possible solutions. It is especially attractive for FPGAs, in which it is possible to have many alternative solutions using the same device technology.

This scenario motivated us to propose a strategy that improves the SEV and consequently the SER estimation accuracy, while it applies early on the design cycle since it is applicable as soon as there are a circuit netlist and a desired operational scenario, without requiring the real device.

1.2 Objectives and Contributions

The main objective of this work is to incorporate the specific SET susceptibility of each logic gate of a given device into the SEV estimation approaches based on, primarily, logic-simulation, and emulation. With this purpose, this work introduces a probability-aware fault-injection strategy that distributes the SETs into the circuit nodes, at the gate level, following a weighted distribution defined by the SET susceptibilities. The proposed strategy aims the following features:

- be flexible, not restricted to any HDL language, logic-simulator, or synthesiser;
- be of general use, not limited to any circuit architecture;

- can be easily incorporated to the verification flow;
- can be automated by scripts.

In fact, this work brings two versions of this strategy. A simplified version assumes that the SET susceptibilities depends only on the logic gate internal circuitry, while a complete version additionally takes into account how the circuit operation affects the SET susceptibility of each logic gate. The first one requires less information and data processing, and is less accurate, while the second is more accurate, but requires more information and data processing.

The strategy relies on the SET susceptibilities of the logic gates. However, this information is not usually available. For this reason, this work includes the estimation of the SET susceptibilities of the CLBs of a target FPGA family, that was used to evaluate the proposed strategy.

The strategy for SEV estimation and the adopted process for SET susceptibility estimation focus on the logical masking effect, due to the logic gate circuitry and the input patterns generated by the circuit operation. The electrical and the latching-window masking effects are out of the scope of the work, primarily, due to the lack of electrical and timing details from the selected technology, but also because they are not directly related to the input pattern effects.

The application of the proposed strategy concentrate on the logic-simulation approach, adopted for the evaluation of the strategy. However, the work includes a discussion about the adoption of the strategy with emulation approaches, emphasising on the SET distribution mechanism.

In summary, the main contributions of this work are the probability-aware fault-injection strategy, the reference SET susceptibilities for the analysed technology, and the probabilistic fault-distribution approach for use in emulation platforms.

1.3 Text Organisation

Chapter 1, Introduction, contextualise the work and presents its motivations, objectives, a summary of contributions and the organisation of the text.

Chapter 2, Concepts and Literature Review, brings the concepts used in this work along with the bibliographic review that supports the proposed strategy. This chapter describes the effects of the ionising radiation on the electronics devices, that are the source of the SEs analysed in this work, and the error-masking factors that influence the SEV. Additionally, this chapter presents the various methods for SE analysis and briefly describes the examples found in the literature.

Chapter 3, Proposed Strategy for SEV Estimation, details our probability-aware strategy for the estimation of the SEV of a digital circuit based on the SET susceptibilities of its logic gates. We propose two alternatives for this strategy, a simplified and a complete one. The simplified requires less information and data processing, but results in a lesser accurate estimation, while the complete needs more information and data processing, but is more accurate. This chapter presents the estimation process for both alternatives, describing each step of these processes. The simplified strategy is based on a preliminary study presented in Armelin *et al.* (2016), while the complete one is based on Armelin *et al.* (2019).

Chapter 4, Estimation of the SET Susceptibilities, describes how we estimated the SET susceptibilities used in this work. First, this chapter presents and briefly compares various architectures of CLBs from the most commonly used FPGA families. Then, it details the architecture of the CLB that was selected for this work. Finally, it details the CLB modelling at the transistor level, its configuration to the various functions, and the estimation process to obtain the reference SET susceptibilities.

Chapter 5, Evaluation of the Proposed Strategy, presents how we evaluated the proposed strategy, described in Chapter 3, using a simulation-based environment. This chapter presents the analysed circuits, and the four distinct SEV estimation processes used in the analysis. The first estimation considers the CLB model at the transistor level,

detailed in Chapter 4, to obtain a basis of comparison for the others estimations. The second one resembles the approaches found in the literature, that consider the same SET susceptibility for all logic gates. The third and fourth estimations respectively adopt the simplified and completed proposed strategies described in Chapter 3. Finally, by the end of the chapter, we present the results, regarding the estimated SEV and the simulation time.

Chapter 6, Using the Proposed Strategy with Emulation, brings some considerations regarding the use of the proposed strategy with an emulation-based environment. We took into account the variations of emulation-based approaches presented in Chapter 2. Additionally, this chapter details the implementation of an emulation-based environment that uses the proposed strategy, focusing on its ability to distribute the SETs according to the susceptibility of each logic gate. The last part of this chapter is based on Armelin *et al.* (2018a).

Chapter 7, Conclusions and Perspectives, discuss the conclusions of this work and some perspectives of future works.

Appendix A, Configurations of the VersaTile Functions, reports the details of the configuration of the analysed CLB model for each implemented function.

Appendix B, SET Susceptibilities of the VersaTile Functions, lists the SET susceptibility of each function implemented with the analysed CLB model.

Appendix C, Self-Produced Transients for SET Emulation, describes an strategy for SET emulation, that were used with the emulation-based environment presented in Chapter 6. This Appendix is based on Armelin *et al.* (2018b).

2 Concepts and Literature Review

The proposed strategy for SEV estimation of digital circuits taking into account the SET susceptibilities of their logic gates is an incremental contribution to the scientific community efforts to properly analyse and predict the behaviour of electronic devices on a radiation environment.

In this chapter, we present the concepts and the literature review concerning the main topics of interest for this work. In Section 2.1, we introduce the effects of the radiation on the electronics devices, covering the radiation environment, the interaction of the radiation with the matter, and the classification of the radiation effects on electronics devices, characterising the SEs analysed in this work. This section also presents a discussion about the correlation between the terms Soft-Error Rate and Soft-Error Vulnerability.

In Section 2.2, we discuss the error masking factors that are intrinsically correlated to the Soft-Error Rate of digital circuits, i.e., the logical, the electrical, and the latching-window masking.

Section 2.3 presents seven classes of methods used for analysing the effects of the SEs on digital systems. This section covers the radiation tests, the analytical methods, the electrical and logic simulations, and the emulation-based methods. For each method, we present some examples found in the literature.

In Section 2.4, we present our analysis on how the SET susceptibility of the logic gates are taken into account by the various methods and examples described in the previous section. This analysis led to the strategy for SEV estimation proposed in this work, that we applied for an specific FPGA technology.

Finally, in Section 2.5, we present some CLB architectures from the most known manufacturers, covering since the first devices to some of the most advanced ones.

2.1 Radiation Effects on Electronic Devices

The effects of the radiation on electronics devices is a complex knowledge field that comprises the sources of the radiation, how they interact with the semiconductors and the effects of this interaction on the behaviour of these devices.

For this work, the main interest is on the later part, more specifically on how the transient electrical pulses generated in the basic semiconductor elements affect the outputs of a digital circuit. For this reason, this section presents a brief introduction to the radiation environment and their interaction with the matter, followed by a more detailed classification and description of the effects of the radiation on the electronic devices.

A detailed description of the radiation environment and their effects on electronics devices may be found in Holmes-Siedle and Adams (2002), Claeys and Simoen (2002), Okuno and Yoshimura (2010), and Velazco *et al.* (2007).

2.1.1 Radiation Environment

The radiation environment is a heterogeneous mixture of many types of radiations, with a wide range of energy (E) and flux, coming from different sources. For this reason, each intended application, e.g., space, aeronautics, ground-level operation, and radioactive installation, has a specific associated radiation environment.

As discussed in Pereira Junior (2015), the radiation can be divided into five types: electromagnetic (photons – x-rays and gamma-rays); light charged particles (electrons and positrons); heavy charged particles (protons, alpha particles, fission fragments, and other ions); neutral particles (neutrons); and elementary particles (resulting from the interaction of the cosmic radiation with the atmosphere).

Concerning the near-Earth space-environment, there is two sources of radiation: the Galactic Cosmic Radiation (GCR) that comes from outside the Solar System, and the Solar Cosmic Radiation (SCR), coming from the Sun activities.

The GCR is composed of high-energy particles that were probably generated by high energetic phenomena, e.g., pulsars and supernova explosions. It is mainly composed by protons (hydrogen nuclei) and alpha particles, but it also has heavier nuclei (HZE-particles – High atomic number, Z, and Energy) and electrons (REITZ, 2008).

The SCR comprise two components: the nearly continuous solar wind, that is the continuous solar activity that emits low-energy protons and electrons; and a periodic one, in which the Sun releases particles with higher energy in events like Solar Proton Event (SPE), Coronal Mass Ejection (CME), and Solar Flares during the approximately seven years of high activity of its nearly eleven years period (BOUDENOT, 2007; REITZ, 2008).

These cosmic radiations interact with the Earth magnetic field, which may trap the particles, deflect them or let them pass through. The trapped particles form the Van Allen Radiation Belts, illustrated in Fig. 2.1.

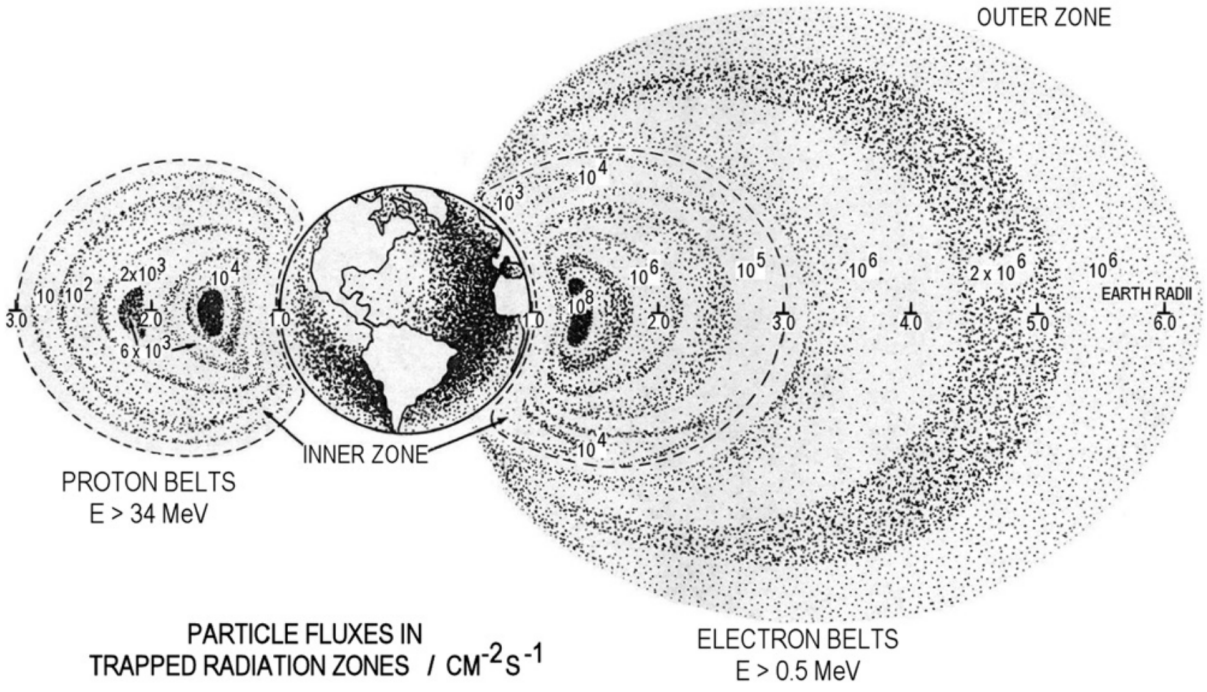

FIGURE 2.1 – Trapped Radiation – The Van Allen Belts. On the left, a representation of the proton flux inside the inner belt. On the right, a representation of the electron flux inside both inner and outer belts. Source: reproduced from Reitz (2008).

These radiation belts consist of electrons, protons, and some heavier ions (REITZ, 2008). Although this trapped radiation results from the interaction of the cosmic radiation with the Earth magnetic field, it is also considered as a radiation source, due to its relevance to the aerospace applications.

The Earth magnetic field presents some anomalies, as the South-Atlantic Magnetic Anomaly (SAMA), that interfere in the local radiation environment. The SAMA is a region where the magnetic field is weaker than in other regions at similar altitude and latitude. This anomaly leads the inner radiation belt to come closest to the surface of Earth, increasing the radiation flux at lower altitudes. Fig. 2.2 illustrates how the SAMA may affect Low Earth Orbit (LEO) satellites, showing the reported events for the MOPITT (Measurements Of Pollution In The Troposphere) mission, concentrated in the SAMA region.

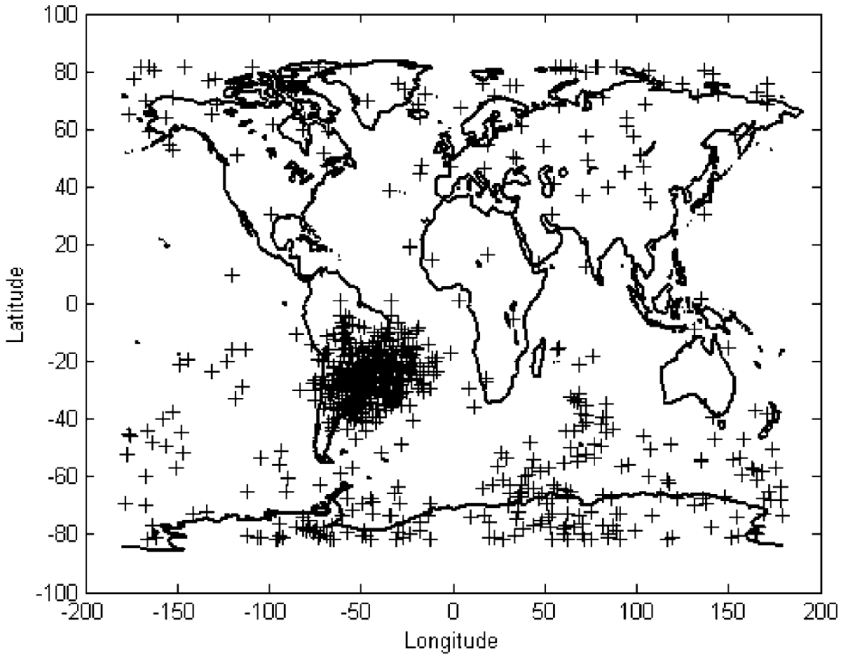

FIGURE 2.2 – MOPITT Device Single Events (DSEs). Geographical distribution of the DSE reported by the Measurements Of Pollution In The Troposphere mission. The SAMA can be identified by the density stop. Source: reproduced from Nichitiu *et al.* (2004).

The cosmic radiation may also interact with the dense matter present in the Earth atmosphere, in a nuclear spallation process, in which the cosmic radiation transfer its energy through elastic and inelastic collisions with the nuclei. This interaction produces a

radiation shower that includes neutrons and elementary particles. Due to this interaction, a small part of the cosmic radiation reaches the Earth surface (both the Earth magnetic field and atmosphere act like radiation shields). Additionally, the resulting atmospheric radiation environment depends on the altitude of interest.

The energy of the GCR particles ranges to above 10 GeV per nucleon (STASSINOPOULOS; RAYMOND, 1988). For the SCR, the solar wind has low-energy particles, in the range from 0.5 to 2 keV per nucleon, while the SPE, CME, and solar flares have high-energy particles, from tens of MeV to hundreds of GeV per nucleon (BOUDENOT, 2007).

In the radiation belts, the electrons may have energy up to 5 MeV in the inner belt and may reach 7 MeV in the outer belt, while the protons of the inner belt may reach tens of MeV. When a high-energy solar event occurs, the energy of the electrons may reach tens of MeV, while the energy of the protons may reach hundreds of MeV (BOUDENOT, 2007).

Finally, as reported by Federico *et al.* (2010), the energy of neutrons induced by cosmic radiation may reach tens of MeV.

2.1.2 Radiation Interaction with the Matter

The interaction of the radiation with the matter can be divided into two classes, according to the characteristics of the radiation: charged and uncharged particles. The charged particles, also known as directly ionising particles, are subdivided into heavy and light particles. The uncharged particles, also known as indirectly ionising particles, are subdivided into electromagnetic (photons – considered as particles with energy and momentum) and neutrons. Based on Okuno and Yoshimura (2010) and on Pereira Junior (2015), in this section we list the different interaction mechanisms for each radiation and the main physical quantities involved.

The charged particles usually interact with the atomic electron cloud, but they may interact with the whole atom or its nucleus. The possible interactions are the soft collision (inelastic atomic collision), hard collision (inelastic collision with electrons of inner shells),

elastic and inelastic collision with the nucleus, nuclear reaction, and positron annihilation. For the heavy particle, the ionisation along its trajectory is denser than those produced by light particles, and the stopping distance of heavy particle is shorter than those of the light ones. The trajectory of the heavy particle is nearly rectilinear, while the trajectory of the light particles may present large deflections from the original direction. The energy transferred to the matter depends on the distance the particle travels in it. The Linear Energy Transfer (LET) is the physical quantity used to express how much energy a particle deposits by each travelled distance unit. It is usually expressed in MeV/cm, but may also be expressed in terms of MeV.cm^2/mg. Lastly, besides the ionisation, the charged particles may also lead to atomic displacement.

The photons may ionise the matter through the photoelectric effect, the Compton effect and the pair production. The interactions of the photons with the matter also produce a cascade of secondary radiation, with photons, electrons and positrons that leads to further ionisation. The LET is smaller than those from the charged particles.

Finally, the free neutrons primarily interact with the nuclei of the atoms because there is no Coulomb interaction with the charged elements. Moreover, since there is no Coulomb repulsion, even low energy neutrons may interact with the nuclei. The possible interactions are the elastic and inelastic scattering, and the absorption nuclear reactions that produce secondary radiations (as beta- and gamma-radiation) that may ionise the matter. Thus, the effects of the neutron radiation are the atomic displacement and the ionisation of the matter.

Besides the LET, there are some other physical quantities related to the radiation. The Absorbed Dose, D, is the amount of transferred energy per unit mass, expressed in gray, Gy = J/kg, or rad = 0.01 Gy. The cross-section, σ, is the area transverse to the relative motion between the particle flow and the target matter, expressed in cm^2. The particle flux, ϕ, is the rate of particle flow per unit area, expressed in [counts]/cm^2.s, while the particle fluency, φ, is the particle flow per unit area, or the integrated flux in time, expressed in [counts]/cm^2.

2.1.3 Classification of the Radiation Effects on the Electronics Devices

There are three classes of radiation effects on the electronics devices: the Displacement Damage (DD), the Total Ionising Dose (TID), and the Single-Event Effect (SEE). The DD and TID are cumulative effects, meaning the effects will be perceptible only after the interaction of many radiation particles with the matter, while the SEE is a singular effect, caused by the interaction of a single radiation particle.

The DD is a non-ionising cumulative effect caused by the charged particles and neutrons that interact with the nuclei of the atoms. This effect affects the structure of the material and leads to long-term degradation in optoelectronics, e.g., increasing the dark current of Charged-Coupled Devices (CCDs), as discussed by Srour *et al.* (2003), and altering the response of Silicon Carbide (SiC) Light-Emitting Diodes (LEDs), as reported by Hinrichsen *et al.* (1998). The Non-Ionising Energy Loss (NIEL) is the physical quantity associated with the particles that may produce DD, expressed in MeV.cm^2/g. Lastly, Srour *et al.* (2003) present a review of the DD effects in silicon devices.

The TID is a cumulative ionising effect induced by any interaction of radiation particles with the matter that leaves trapped charges in the device structure, i.e., ionising interactions that are not fully recombined. These charges trapped inside the semiconductor, or the oxide, change the device characteristics, as the voltage threshold and leakage current of a MOSFET transistor (FACCIO; CERVELLI, 2005). There are many studies devoted to the TID effects on electronics devices. Barnaby (2006) discusses the TID effects on modern CMOS technologies, while Ding *et al.* (2015) report the TID effects in 65 nm CMOS transistors. Snow *et al.* (1967) reported one of the first studies about the charges accumulation on the isolation oxide, and Armani *et al.* (2015) compare the TID response of some FPGAs and memories.

The SEE is an effect induced by a single radiation particle with enough LET to provoke a relevant disturbance in the device behaviour. The SEE is the class of radiation effects of interest in this work, which is the reason why it is discussed in more details in the

remainder of this section.

The primary SEE is the Single-Event Transient, that is a voltage excursion in a circuit node inside the device. When the ionising radiation strikes through a reverse-biased junction, the electric field induces a transient current with the produced free charges. This transient current causes a momentary voltage excursion in the electric node. Fig. 2.3 illustrates a SET modelling for a CMOS inverter.

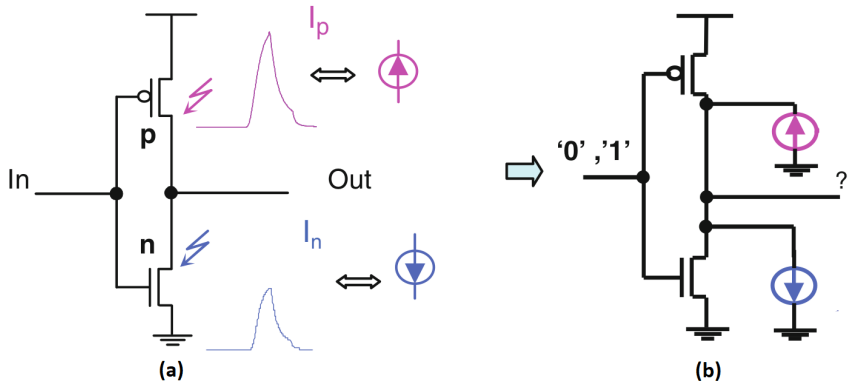

FIGURE 2.3 – SET modelling for a CMOS inverter. The figure illustrates two situations: a particle strike in the PMOS and in the NMOS. In both cases, it will induce a transient current between the struck area and the body. Source: adapted from Buard and Anghel (2011).

The SET effects on analogue devices are different from digital ones. For this reason, some works use specific acronyms to differentiate them: Digital Single-Event Transient (DSET), for digital circuits, and Analogue Single-Event Transient (ASET) for analogue and mixed-signal circuits. Further, some older works related to SEE in analogue circuit adopted the more generic term ASEE (Analogue Single-Event Effects) to refer to the transients, as in Turflinger (1996), or even the term ASEU (Analogue Single-Event Upset), as in Koga *et al.* (1993), that would be the analogue version of the Single-Event Upset (SEU).

The first reports of SEUs were published by Binder *et al.* (1975) and Pickel and Blandford (1978), while the first studies on DSET were published by Diehl *et al.* (1983). Regardless, the nature of the DSETs and ASETs are the same. The main difference is that the analogue circuits are susceptible to any voltage disturbance, while the digital circuits are susceptible to voltage disturbances that exceed the logic threshold. In this

work, we only deal with digital circuits and adopt the term SET. Finally, Ferlet-Cavrois *et al.* (2013) present an extensive review of SETs on digital CMOS.

Another SEE of interest for this work is the already mentioned SEU, which is a change in the stored logic value of a memory element (memory cell, flip-flop, or latch). This effect is also referred to as a bit-flip and sometimes used as a synonymous for Soft-Error. The SEU is produced by the same mechanism that produces the SET when the struck reverse-biased junction is part of a storage element. Fig. 2.4 illustrates the waveform of a storage element signal in two situations: when the voltage excursion results in an SEU and when it does not.

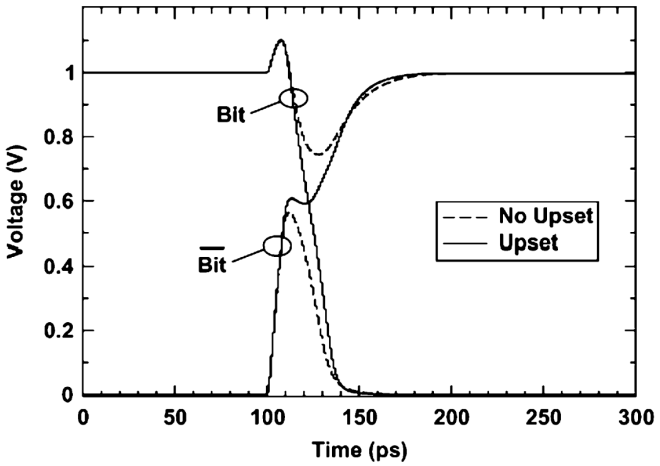

FIGURE 2.4 – Illustration of a SEU. This figure shows the voltage excursion of the storage signals, in a memory element, in two situations: a SEU is produced and no SEU is produced. Source: reproduced from Gaillard (2011).

Depending on the feature size of the adopted technology and the LET of the incident radiation, the resulting SEE may affect more than one circuit node. The Multiple-Cell Upset (MCU) is the occurrence of bit-flips in more than one storage element due to the interaction of a single particle. The Multiple-Bit Upset (MBU) is a particular case of MCU in which the multiple bit-flips occur in the same data word.

However, the SEEs are not restricted to bit-flips and voltage transients. If the ionisation particle hits the parasitic bipolar transistors of a CMOS inverter, acting as a trigger of a Silicon Controlled Rectifier (SCR), it may produce a Single-Event Latch-up (SEL), that is characterised by an induced short-circuit between the power lines. It is potentially

destructive, but can be mitigated by protecting the power lines. Another SEE related to induced high current is the Single-Event Burnout (SEB). Differently from the SEL, the SEB is caused by a localised high current that damages the device.

The SEEs are divided into Soft and Hard Errors. The SEs are the recoverable errors on signals or data registers, while the Hard Error is an irreversible change in the device's operation. The Hard Error is associated with permanent damage or degradation of the device, and sometimes is referred to as Single-Event Hard Error (SHE), to differentiate from damages and degradations not related to SEE. Among the mentioned SEEs, the SEB is the only SHE, while the SET, SEU, MCU, MBU and SEL are SEs (SLAYMAN, 2011).

There are some other SEEs, e.g., SEFI (Single-Event Functional Interrupt) and SEGR (Single-Event Gate Rupture), to describe specific effects caused by single radiation particles. However, since this work focuses on the SEs observed at the gate-level, i.e., SETs and SEUs, we will not further discuss the other SEEs.

2.1.4 Soft-Error Rate and Soft-Error Vulnerability

The SER is an essential parameter to properly assess the reliability of electronic systems that operate under the effects of the radiation particles. It is usually expressed in FIT (Failures In Time – [counts]/10^9h), but also in terms of MTBF (Mean Time Between Failures, in hours). This parameter is intrinsically related to the rate of transient electrical pulses, R, generated in the sensitive internal elements of the device (SLAYMAN, 2011),

$$R = \int_0^\infty \phi(E) \cdot \sigma(E) \cdot dE, \tag{2.1}$$

where $\phi(E)$ is the particle flux, $\sigma(E)$ is the element cross-section, and E is the particle energy.

In digital systems, these sensitive elements are the switching transistors, for which the cross-section is directly related to their drain areas.

However, the device's SER is not the sum, or combination, of the R value of each transistor. The logical, temporal and electrical masking effects, that are described in the next section, filter-out many transient pulses.

Rewriting (2.1), at the transistor level, the rate of transient electrical pulses, R_q, that are induced in a transistor by the ionising radiation particles is a function of the particle flux, $\phi(E)$, the cross-section of the transistor, $\sigma_q(E)$, and the particle energy, E, as shown in (2.2). In other words, the only parameter that is related to the device is $\sigma_q(E)$.

$$R_q = \int_0^\infty \phi(E) \cdot \sigma_q(E) \cdot dE \tag{2.2}$$

At the gate level, the rate of induced pulses inside the logic gate, R_i, depends on the new sensitive cross-section, $\sigma_c(E)$, that is the sum of the $\sigma_q(E)$ of each of its transistors, shown in (2.3). For a circuit with n transistors with the same $\sigma_q(E)$, R_i can be obtained with (2.4).

$$R_i = \int_0^\infty \phi(E) \cdot \sigma_c(E) \cdot dE \tag{2.3}$$

$$R_i = n \cdot \int_0^\infty \phi(E) \cdot \sigma_q(E) \cdot dE \tag{2.4}$$

However, the generated SEs at the logic gate output, R_o, additionally depends on its internal circuit and operation, being subjected to electrical, logical and temporal masking effects (see Section 2.2). At this level, R_o can be correlated to R_i through a susceptibility factor, χ_{gate}, as in (2.5).

$$R_o = \chi_{gate} \cdot R_i \tag{2.5}$$

Similarly, at the Register-Transfer Level (RTL), the same masking effects may filter-out the the SEs generated at the output of the circuit gates (R_{gates}), affecting the SER of the circuit, SER_{RTL}. This may also be correlated through a susceptibility factor, χ_{RTL},

shown in (2.6).

$$SER_{RTL} = \chi_{RTL} \cdot R_{gates} \tag{2.6}$$

Additionally, the SER, at the RTL level, can be correlated to the SEs generated at the transistor level by a parameter that, in this work, we referred to as SEV, as in (2.7).

$$SER_{RTL} = SEV \cdot R_i \tag{2.7}$$

As introduced in Chapter 1, in this work we use the term SET susceptibility (χ_{gate}) for the sensitivity of the logic gates to the SEs generated in their transistors, while we use SEV for the sensitivity of the circuits (RTL) to the SEs generated in their transistors.

Depending on the circuit (and logic gate) architecture and operation, SEV (and χ_{gate}) may be a function of the operational clock frequency, the inputs patterns, and the induced transient pulse width. In the next section we describe the masking factors that affect them.

2.2 Error-Masking Factors

The error-masking factors filter-out some of the SEs generated inside of an electronic device. Three types of error-masking factors are relevant for digital circuits: the logical masking, the electrical masking, and the latching-window masking. Each of them has different mechanisms by which the SE may be filtered.

The effects of these masking factors may appear in different hierarchical levels. For this reason, before describing these factors, we present the hierarchical levels considered in this work, and the representation domain associated with them, adapted from Anghel *et al.* (2007). The hierarchical levels are:

System Level an item composed by functional blocks, as processors, memory blocks, and peripherals.

Register Transfer Level a functional block composed by logic gates, registers, memory cells, etc.

Gate Level the logic gates, registers, memory cells, etc., composed by electronic components, as transisotrs.

Transistor Level electronic components.

Concerning the representation domain, we may have:

Behavioral functional descriptions define the operation of the device, usually based on Hardware Description Language (HDL) codes.

Structural components and interconnections describe the device circuit.

Physical equations describe the operation, or the real interaction in the physical device.

2.2.1 Logical Masking

A SE may be logically masked in combinational circuits when one of the inputs has a value that imposes the output value, regardless of the value of the other inputs. Fig. 2.5 illustrates this masking factor. When a 2-input AND gate has one of its inputs receiving logical value '1', a SE on the other input can propagate to the output, in a non-masking situation. However, if one of the inputs of this logic gate receives logical value '0', a SE on the other input is filtered-out, because the output value will be '0', regardless of the value of this second input.

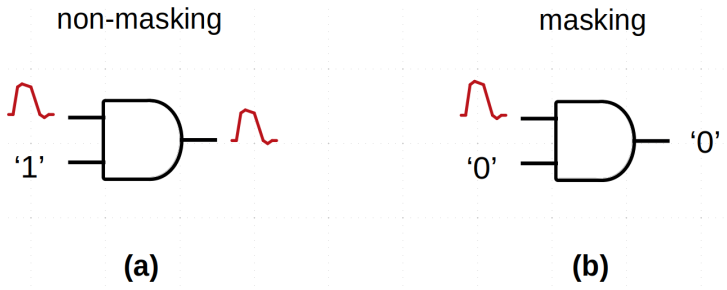

FIGURE 2.5 – Logical masking factor on a 2-input AND gate – **(a)** non-masking situation and **(b)** masking situation.

The logical masking is usually taken into account at the RTL and gate levels and its analysis can be extended to the system level. However, when analysing the SE generation inside the logic gate, it is essential to evaluate its masking effect at the transistor level. Fig. 2.6 illustrates the internal masking effect for a 2-input NAND gate.

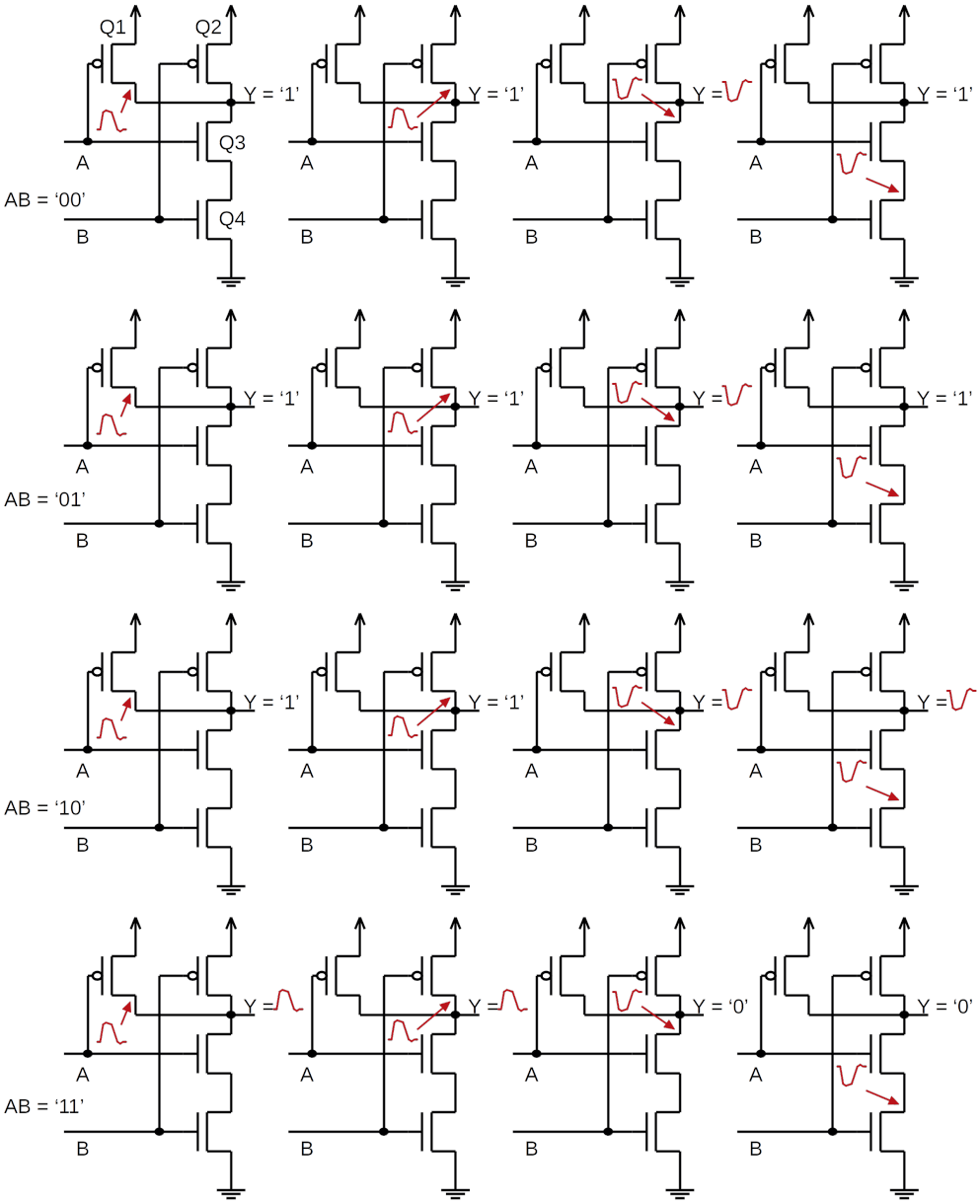

FIGURE 2.6 – Logical masking factor at the transistor level on a 2-input NAND gate. Each row has one input pattern AB = {00, 01, 10, 11}, while each column has one target transistor {Q1, Q2, Q3, Q4}. For the first input pattern, AB = '00', the output Y changes from '1' to '0' only when the SET occurs in the transistor Q3 (its drain area). For AB = '01', the output Y changes from '1' to '0' only when the SET occurs in the transistor Q3. For AB = '10', the output Y changes from '1' to '0' when the SET occurs in the transistor Q3 or Q4. Finally, for AB = '11', the output Y changes from '0' to '1' when the SET occurs in the transistor Q1 or Q2. This analysis ignores possible layout effects. Assuming all transistors have the same cross-section, $\chi_{10} = \chi_{11} = 2 \cdot \chi_{00} = 2 \cdot \chi_{01}$.

This masking factor is usually analysed in the behavioral and structural domain. However, depending on the analysis performed in the physical domain, it can be intrinsically included, as in Buard and Anghel (2011).

2.2.2 Electrical Masking

The electrical masking effect is the successive attenuation of the transient pulse until it is completely quenched. It is caused by the electrical interaction of the transient pulse with the electronic components. Its occurrence depends on the characteristics of the device, as its parasitic capacitances, and of the transient, as its amplitude and width. Fig. 2.7 illustrates this masking factor.

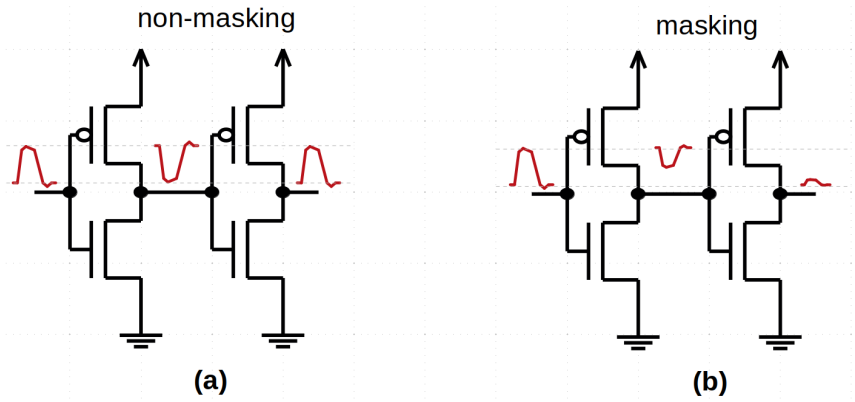

FIGURE 2.7 – Electrical masking factor on two successive inverters – **(a)** non-masking situation and **(b)** masking situation.

This masking factor is analysed in the physical domain, at the transistor and gate levels. Nevertheless, its effects may be emulated at the RTL level, in the behavioral domain, as in García-Valderas *et al.* (2009) and Entrena *et al.* (2009).

2.2.3 Latching-Window Masking

The latching-window masking effect, or temporal masking effect, is the 'loss' of the transient pulse due to its occurrence out of the latching window of the flip-flops and latches. The masking probability depends on the operational frequency of the latching

element and the width of the transient pulse. Fig. 2.8 illustrates this masking factor, that is analysed in the behavioral and structural domain, at the gate, RTL and system levels.

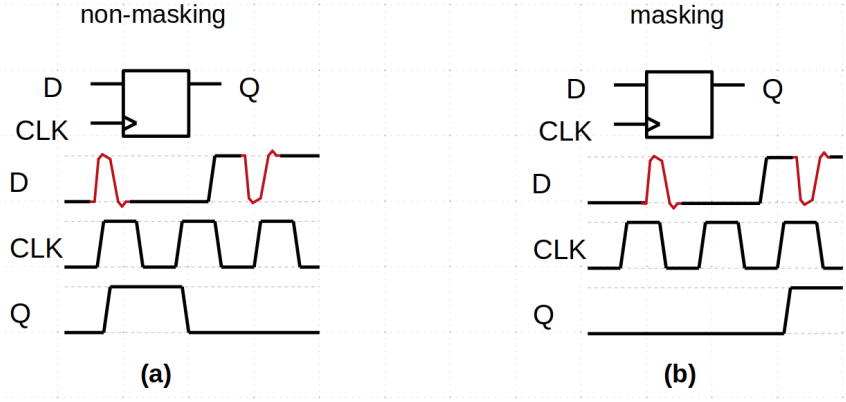

FIGURE 2.8 – Latching-window masking factor on a D flip-flop – **(a)** non-masking situation and **(b)** masking situation.

2.3 Methods for Soft-Error Analysis

The methods for SE analysis are based on many different techniques, from mathematical models to physical tests. In this section, we describe the most common techniques related to SEV estimation, divided into seven categories: analytical, electrical simulation, logic simulation, emulation, software fault-injection, hardware tests, and radiation tests. As the studies about SET propagation is closely related to the SEV estimation ones, and they may also be divided into some of these categories, we include examples of these studies along with the examples about SEV estimation. Additionally, some of the presented examples deal with both SEs and hard-errors.

The literature has some reviews covering the SE analysis, specially the works related to the fault-injection techniques. Ziade *et al.* (2004) present a survey of fault-injection techniques, divided into hardware, software, simulation, emulation and hybrid-based methods. Although it is outdated, they briefly describe the main techniques available at that time, and compare the advantages and disadvantages of each of this five classes. Kooli and Di-Natale (2014) focused their survey on the simulation-based fault-injection techniques, including the tools that emerged in the decade that separate this review from the previ-

ously mentioned. Jeong *et al.* (2016) partially updated the survey from Ziade *et al.* (2004), as they also cover hardware, software, simulation, and emulation-based techniques. However, they focused on those methods suitable for Systems on Chip (SoC). Finally, Natella *et al.* (2016) presents a detailed review of software-based fault-injection techniques, covering software bugs and hardware faults.

2.3.1 Analytical

The analytical methods use some mathematical model to predict the occurrence of SE. These models differ from the semiconductor modelling used in the electrical simulations. They usually deal with the associated probabilities, as the Pulse Generation Probability (PGP), Error Propagation Probability (EPP), Latching Probability (LP), Probabilistic Vulnerability Window (PVW), Transfer Matrices (PTM), Probabilistic Gate Models (PGM), and Signal Probability (SP). These methods work at the system, the RTL, and, in most cases, the gate level, with the device representation in the structural domain.

Fazeli *et al.* (2010) propose an analytical method based on the EPP, from the error generation site to the primary output. The used EPP was derived from another analytical method for SER estimation, proposed in Asadi and Tahoori (2005). Raji *et al.* (2015) propose a metric, Triple Constraint Satisfaction (TCS), based on PVW, that is an inference of the necessary conditions for the occurrence of a SE. In Rezaei *et al.* (2014), the authors present a method for SER estimation and mitigation by characterising the input patterns. They analyse the effects of the input patterns on the PGP, while they also use the EPP and LP to obtain the SER. The effects of the input patterns on the SER were also analysed by Sootkaneung and Saluja (2010), that proposed a input reordering technique to reduce the SER, in a work that also adopts electrical simulations to obtain the probabilities.

The influence of the input values on the reliability of digital circuits is taken into account in other analytical based studies, concerning both SEs and hard-errors. In Franco *et al.* (2008), the authors present an algorithm based on SP and PTM. Han *et al.* (2011)

describe a reliability evaluation method with PGMs, that also deals with signal probabilities. While in Han *et al.* (2014), the authors introduce a stochastic computational approach that uses Stochastic Computational Models (SCMs) and signal probabilities embedded in pseudo-random binary streams.

Finally, Dabiri *et al.* (2008) proposed an analytical method for power optimisation using gate-sizing that takes the SE impact into account.

2.3.2 Electrical Simulation

The electrical simulation methods are divided into those dedicated to analysing the interaction of the radiation particle with the electronic device, based on Three-Dimensional (3D) simulations, and those dedicated to analysing the electrical behaviour of the electronic device, based on circuit simulations similar to SPICE. These methods works at the transistor and gate levels, in the physical domain.

In Dodd (1996), the author discusses the device simulation of charge colection and SEUs, including the drawbacks of adopting Two-Dimensional (2D) and Quasi-3D models instead of 3D ones. Dodd and Massengill (2003) describe the physical mechanisms and 3D modelling of generating SEUs in Dynamic Random-Access Memories (DRAMs) and SRAM Static Random-Access Memories (SRAMs). Dodd *et al.* (2004) discuss the production and propagation of SETs in logic devices using 3D and circuit simulations, respectively. Similarly, Wirth *et al.* (2008) adopt SPICE simulation to analyse the SET pulse broadening due to load and propagation effects.

In a different application, Sajjade *et al.* (2018) use circuit simulation to compare various latches that were radiation hardened by design. This work also present a review of radiation hardened by design techniques.

Finally, as already discussed in the previous section, Sootkaneung and Saluja (2010) use SPICE simulation to obtain the SE probabilities that take into account the input pattern effects. Similarly, Buard and Anghel (2011) showed the influence of the input patterns on the SER of logic gates.

2.3.3 Logic Simulation

The logic simulation methods use HDL models and logic simulator, and can be divided, as discussed in Baraza *et al.* (2005), into two main groups: those based on simulator commands, and those based on code modification. Additionally, they include a minor generic group for other techniques. Both groups can work on behavioral and structural domains, in system, RTL or gate level.

The methods based on the simulator commands relies on the ability of the chosen logic simulator to modify the values of the signals and timing characteristics of the model during the simulation.

On the other hand, the methods based on the code modification relies on inserting *saboteurs* (components that intercept a signal and changes its value) or replacing a component by a *mutant* (a faulty version of the component).

Concerning the code modification methods, mutants are adopted in the studies of both SEs and hard-errors. In Ward and Armstrong (1990) and Ghosh and Chakraborty (1991), the authors use mutants for modelling hard-errors, as stuck-at, in which one or more signals get stuck-at some value, while in Leveugle and Hadjiat (2000), the authors use mutants for modelling SEUs, and that are suitable for use in emulation platforms. The saboteurs are commonly used for SE, as presented in Amendola *et al.* (1996) and Boue *et al.* (1998). For instance, in Lopes Filho and D'Amore (2012), the authors proposed a fault-injection technique that uses saboteurs to inject SETs and SEUs into a RTL description.

There are some logic-simulation tools for SE analysis, as: MEFISTO (Multi-level Error/Fault Injection Simulation Tool), proposed by Jenn *et al.* (1994), and the VFIT (VHDL-based Fault Injection Tool), proposed by Baraza *et al.* (2002), make use of both simulator commands and code modifications; MEFISTO-L, proposed by Boue *et al.* (1998), derives from MEFISTO, focusing on the use of mutants; MEFISTO-C, described by Folkesson *et al.* (1998), also derives from MEFISTO, but focus on commands of a specific simulator; and VERIFY (VHDL-based Evaluation of Reliability by Injecting

Faults Efficiently), proposed by Sieh *et al.* (1997), relies on extending the VHDL language, that fits into the other techniques group.

As examples of logic-simulation fault-injection method at the structural domain, Bottoni *et al.* (2015) present a partial triplication approach, in which they use logic-simulation to inject faults into the circuit netlist, for selecting the flip-flops to apply Triple Modular Redundancy (TMR). Similarly, Srinivasan *et al.* (2005) presented a selective hardening technique for combinational circuits based on netlist logic simulation.

Concerning the studies of SET propagation on combinational circuits, Violante (2003) presents a zero-delay logic-simulation approach that uses circuit expansion to embed timing information, reducing the simulation time.

Finally, a disadvantage of the logic-simulation method is the amount of demanded computational resources. Some works, as Parrotta *et al.* (2000) and Berrojo *et al.* (2002), present optimisation techniques for reducing the required simulation time during fault-injection campaigns.

2.3.4 Emulation

The emulation methods for SE analysis are those in which a hardware platform, usually an FPGA, is used to implement the desired circuit functionality, adding some resources to inject faults, or errors, and to detect the propagated errors. They work in the physical domain, at the system, RTL, or gate level.

These methods can be classified according to some characteristics of the whole solution, from which we highlight the four topics described below:

host-based vs. autonomous: In the host-based architectures, the fault-injection and, in some cases, the error detection are controlled by an external host computer, while in the autonomous ones, these tasks are performed inside the emulation platform, and an external computer is used for data configuration and analysis. The host-based emulation approaches have the advantage of more flexibility to control the

fault-injection, however, they are slower, due to the communication between the host and the emulation platform during the fault-injection process. Contrarily, the autonomous platforms have less flexibility, due to the limited resources, but they are more rapid, since the communication with the external computer is limited to the beginning and the end of the process, for configuration and data collection.

golden-result vs. golden-circuit: This difference is related to the detection of the error occurrence. With the use of the golden-circuit, it can be achieved by comparing, on the run, the behaviour of the circuit, while the faults are injected, with the behaviour of another instance of the same circuit without the injection of faults. Another approach, with the golden-result, is to compare the behaviour of the circuit, under the effects of the faults, with a previously obtained reference behaviour of the circuit. The golden-circuit case has the advantage of flexibility in terms of which test vectors can be applied, since it does not need to be previously tested, but requires more resources, since it needs another copy of the circuit. In opposition, the golden-result approach has the advantage of not needing the other copy of the circuit, but the test vector results needs to be previously obtained and stored. These two approaches also apply for the logic-simulation methods.

saboteurs vs. mutants: A third subdivision of the emulation approaches is related to how the circuit is modified for the fault injection. Similarly to the logic-simulation methods, it is possible to use saboteurs or to modify the CLB configuration, generating a mutant. Although the saboteurs can cover a wide range of fault models for a CLB, depending on the complexity added to them, in some cases the use of mutants are more suitable. The most important situation is when analysing the effects of an SEE in the configuration memory of an FPGA. These effects are generally the change of the CLB configuration, which are the conception of the mutants. However, these effects are restricted to the reconfigurable FPGA technologies that are susceptible to the SEEs, the SRAM (see Section 2.5).

direct-access vs. scan-chain: This difference is a subdivision of the emulation methods

that adopts saboteurs, related to how they are accessed for the fault injection. In the direct-access approaches, the fault-injection controller has direct access to the control signals of the saboteurs, while in the scan-chain ones, the controller has access to the saboteurs trough a scan-chain that interconnects all saboteurs. The direct-access approach provides quick access to the saboteurs, but requires more resources, specially from the routing network and for the address decoding. On the other hand, the scan-chain approach is slower, due to the need of moving the control signals through the chain, but requires less resources. Usually, the scan-chain follows the architecture defined for the Test Access Port (TAP) and boundary-scan architecture (INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, 1993).

In the literature, there are many research works based on emulation approaches for the analysis of the SEs, covering the aspects described above. In what follows, we present some examples.

Civera *et al.* (2001) present a host-based, golden-result platform with scan-chain saboteurs for SEU analysis. In Antoni *et al.* (2002), the authors propose a host-based, golden-result platform based on partial reconfiguration of an FPGA. Lopez-Ongil *et al.* (2007) describe an autonomous, golden-circuit platform with scan-chain saboteurs for SEU analysis.

In Naviner *et al.* (2011), the authors introduce the tool FIFA (Fault Injection and Fault masking Analysis), an autonomous, golden-circuit platform with direct-access saboteurs, for the analysis of SEs and hard-errors. Entrena *et al.* (2012) propose the tool AMUSE (Autonomous Multilevel emulation system for Soft Error evaluation), an autonomous, golden-circuit platform, with scan-chain saboteurs, that emulates the SET propagation, based on the quantised delay model proposed by García-Valderas *et al.* (2009), and the electrical masking model proposed by Entrena *et al.* (2009).

Ebrahimi *et al.* (2014) details the SCFIT (Shadow Components-based Fault Injection Technique), a host-based, golden-result, scan-chain platform based on specific resources provided by the FPGA vendor, that were introduced by Mohammadi *et al.* (2012).

Finally, May and Stechele (2012) and May and Stechele (2013) present an autonomous, golden-circuit probabilistic fault-injection platform, with direct-access saboteurs, in which concomitant and independent faults can occur, for the analysis of voltage over-scaling.

2.3.5 Software Fault-injection

The software fault-injection methods are those implemented by the software that runs in the evaluated microprocessor or makes part of the evaluated system. Thus, it can be divided into the methods dedicated to analyse the effects of the hardware faults and those dedicated to the software bugs. In both cases they work in the physical domain, at the RTL or system level.

This method requires that the hardware have dedicated resources accessible through the software. Some examples of such method are the tools FERRARI (Fault and Error Automatic Real-Time Injection), proposed by Kanawati *et al.* (1992), and XCEPTION, proposed by Carreira *et al.* (1998).

2.3.6 Hardware Tests

The hardware tests for error analysis requires that the hardware has specific resources for fault-injection in the final design. As in the software fault-injection methods, this requirement limits the number of circuit nodes in which a SE can be injected. These methods also work in the physical domain, at the RTL or system level.

While the tools MESSALINE, described by Arlat *et al.* (1990), and RIFLE, proposed by Madeira *et al.* (1994), inject the fault stimulus at the pin-level, the approach described by Portela-Garcia *et al.* (2007) uses the TAP and boundary-scan for injecting the fault inside the circuit.

2.3.7 Radiation Tests

The radiation test methods require the actual device and a radiation source. Since they rely on the interaction of the radiation with the electronic device, they are the closest method to the actual interaction in the real application. These tests can be used to analyse the three classes of radiation effects (TID, DD, and SEE), requiring different radiation sources. For the SEE studies, they generally use heavy-ions and protons, but also use photon sources. These methods work at the transistor level, in the physical domain.

There are many works that adopt the radiation tests to analyse the SEEs on digital circuits. We present some examples in what follows.

Karlsson *et al.* (1994) present an approach to validate fault-handling mechanisms using heavy-ion radiation from a Californium-252 (Cf-252) source. Bottoni *et al.* (2014) report the alpha-particle test results of a 65 nm LEON3 microprocessor, and Cai *et al.* (2019) report the heavy-ion test results of a 65 nm radiation hardened FPGA, using the HI-13 Tandem accelerator.

Similarly, in Tambara *et al.* (2015b), the authors report the heavy-ion test results for a 28 nm all-programmable SoC, using the 8UD Pelletron accelerator, while in Tambara *et al.* (2015a), the authors present the characterisation of the embedded memories of the same 28 nm all-programmable SoC family, based on heavy-ion and proton radiation tests.

Concerning the transient characterisation, Rezgui *et al.* (2007) present a study of SET propagation and mitigation in 130 nm flash-based FPGAs, based on heavy-ion radiation tests, while Evans *et al.* (2014) also present a study, with new techniques, for characterising SET propagation in the same 130 nm flash-based FPGA family, based on heavy-ions. Lastly, Ferlet-Cavrois *et al.* (2008) investigate the SET propagation and broadening in 130 nm technology, based on both laser and heavy-ions radiation tests.

2.4 SET Susceptibility on SEV Estimation

From the discussed methods for SE analysis, the radiation tests are the only one that intrinsically takes into account the SET susceptibilities of the circuit gates, because, during these tests, the radiation particle interacts with the circuit transistors while the circuit is running the desired operational scenario. Thus, all error masking effects are intrinsically considered by this method.

However, there are some analytical and electrical simulation methods that also take the SET susceptibilities into account. In Buard and Anghel (2011), the authors present a logic gate modelling and electrical simulation method at the transistor level using SPICE. Their method intrinsically includes the internal logical masking, with the influence of both the logic gate internal circuitry and the input values. It also includes the electrical masking, through the electrical modelling of the transistors and the SETs. Their results show that both logical and electrical masking effects affect the SET susceptibility of the analysed NAND2 gate, that presented an increase of approximately 63% from the lowest to the highest failure rate as a function of the input patterns variation.

Likewise, the analytical method proposed in Rezaei *et al.* (2014) also includes the internal masking effects of the logic gates. A major concern of the authors is how the input patterns influence the SET generation at the output of the logic gates. Their results also show that the SET susceptibilities are affected by these factors and, additionally, how they influence the circuit SEV. For instance, they show an increase of approximately 900% from the lowest to the highest vulnerability of a NAND2 gate as a function of the input patterns.

On the other hand, logic-simulation and emulation-based methods do not consider the SET susceptibilities of the circuit gates. In fact, from all the literature research, the only method we found that adopts a probabilistic fault distribution, that would resemble considering the SET susceptibilities, is the emulation approach described in May and Stechele (2012) and May and Stechele (2013). However, their work focus on stochastic computing and voltage over-scaling, in which is essential to analyse the effects of independent and

concomitant multiple faults, in a different scenario from the SEE studies.

Finally, the hardware tests and the software fault-injection methods also do not take into account the SET susceptibilities. However, unlike the logic simulation and emulation methods, they are not suitable for incorporating this capability due to their restricted access to the circuit nodes.

2.5 CLB Architectures

The commercial Field Programmable Gate Arrays emerged in the mid-'80s. The classic structure of an FPGA has four elements: the Configurable Logic Blocks, also referred to as Logic Cells (LCs), the configurable Input and Output Cell (IOC), the Routing Network, and the Configuration Memory, as illustrated in Fig. 2.9. The CLBs are used to implement the desired logic functions, the IOCs are used to implement the required interface characteristics, the Routing Network is used to interconnect the CLBs and IOCs, and the Configuration Memory holds the configured information.

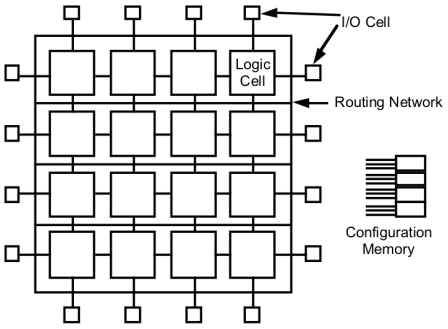

FIGURE 2.9 – Basic elements of an FPGA – Logic Cells, Input/Output Cells, Routing Network, and Configuration Memory. Source: reproduced from Goda *et al.* (2001)

A major classification of the FPGAs is related to the technology used in the Configuration Memory, for which there is three types: the antifuse, that is an One-Time Programmable (OTP), similar to the PROM (Programmable Read-Only Memory); the volatile re-programmable SRAM; and the non-volatile re-programmable flash memory.

Although there may be some differences in this nomenclature from one manufacturer to other, this basic elements are present in all FPGAs. However, due to the technology

advancements and the increasing interest for this devices, the manufactures started to add more resources, as memory banks, Phase-Locked Loops (PLLs), and dedicated circuits. Nevertheless, in this work, we focus only on the CLB.

Usually, the CLB has three parts: a memory element that implements the sequential functions, a Lookup Table (LUT) that implements the combinational functions, and some routing elements to interconnect them. Fig. 2.10 presents the CLB of the XC2000, the first FPGA family from Xilinx, that has this classical structure. Fig. 2.11 presents the Programmable Logic Block (PLB) of the iCE40 UltraPlus, an FPGA family from Lattice Semiconductor. It has eight LCs that are similar to the previous one.

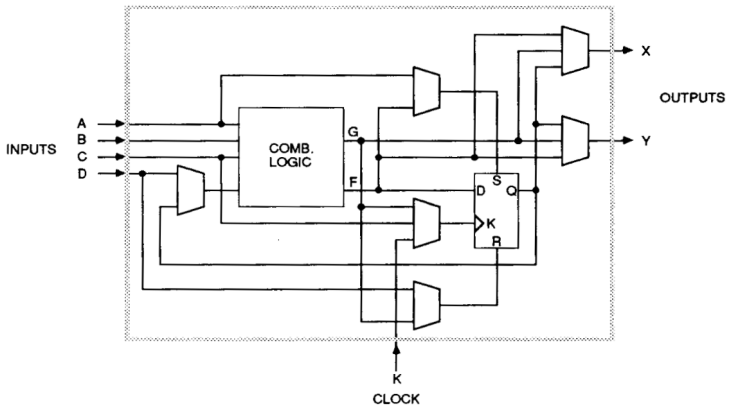

FIGURE 2.10 – Configurable Logic Block of the Xilinx XC2000 FPGA family. Source: reproduced from XILINX (1985).

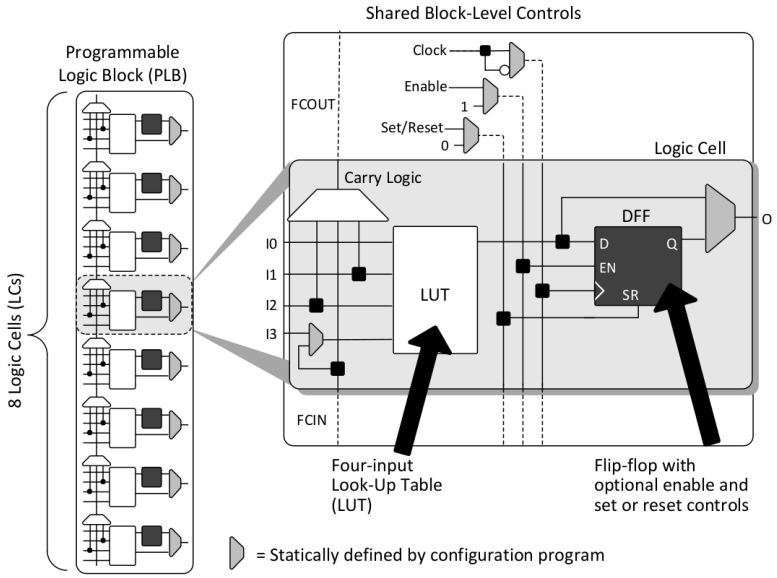

FIGURE 2.11 – Programmable Logic Block (PLB) of the Lattice Semiconductor iCE40 UltraPlus FPGA family. Source: reproduced from LATTICE SEMICONDUCTOR (2018).

There are much more complex CLBs. For instance, Fig. 2.12 presents a quarter of the CLB used in the Spartan-3 FPGA family, from Xilinx. However, it also has LUTs, memory and routing elements.

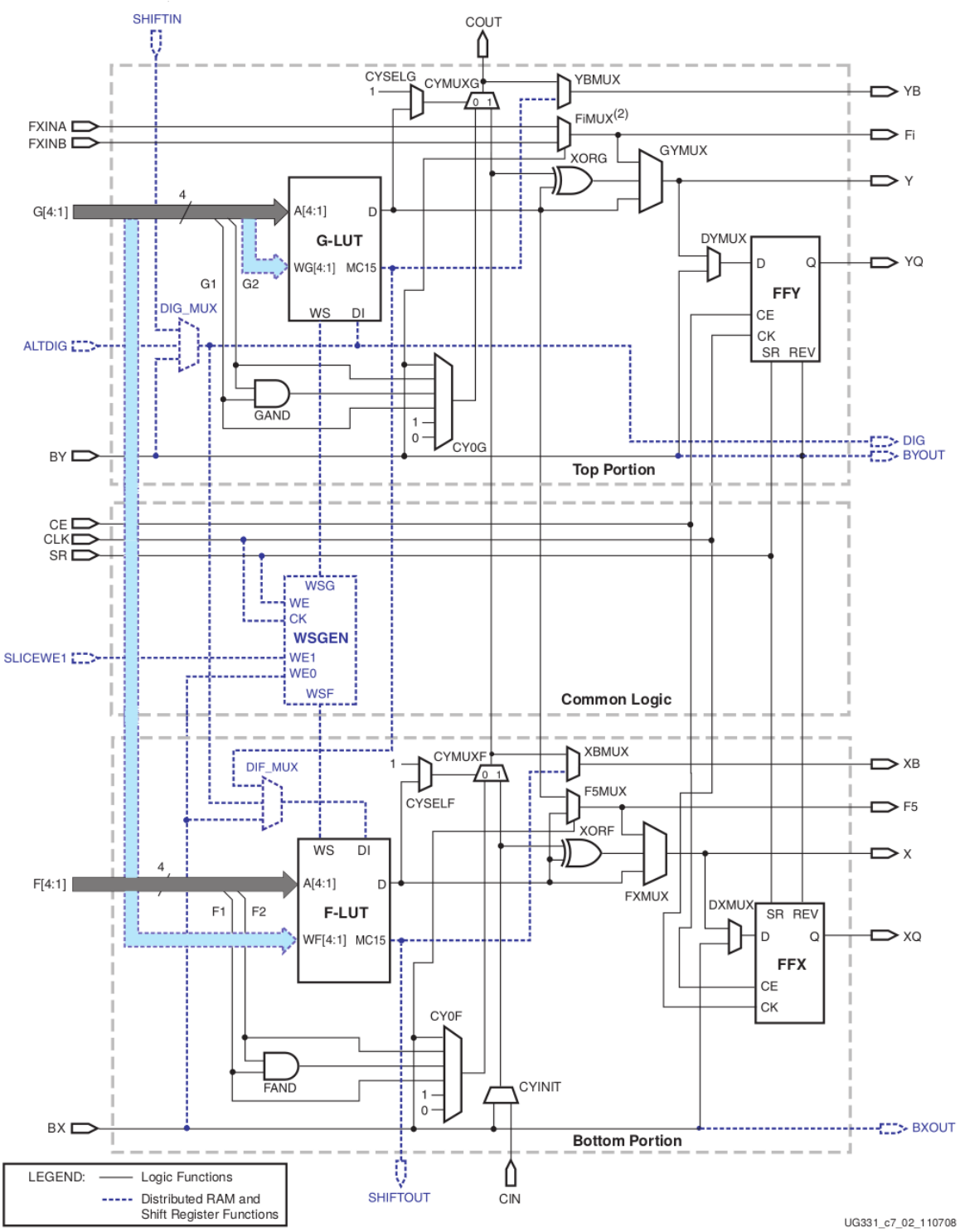

FIGURE 2.12 – Partial representation of the Configuration Logic Block of the Xilinx Spartan-3 FPGA family. The CLB of this family has four slices. This figure reproduces the structure of one of these slices, that Xilinx called SLICEM. Source: reproduced from XILINX (2011).

The advancements in the FPGA technology may result from small refinements to an existing CLB architecture. For example, Fig. 2.13 presents the Adaptive Logic Module (ALM) of the Stratix 10 FPGA family, while Fig. 2.14 presents the ALM of the Agilex FPGA family, both from Intel (former Altera).

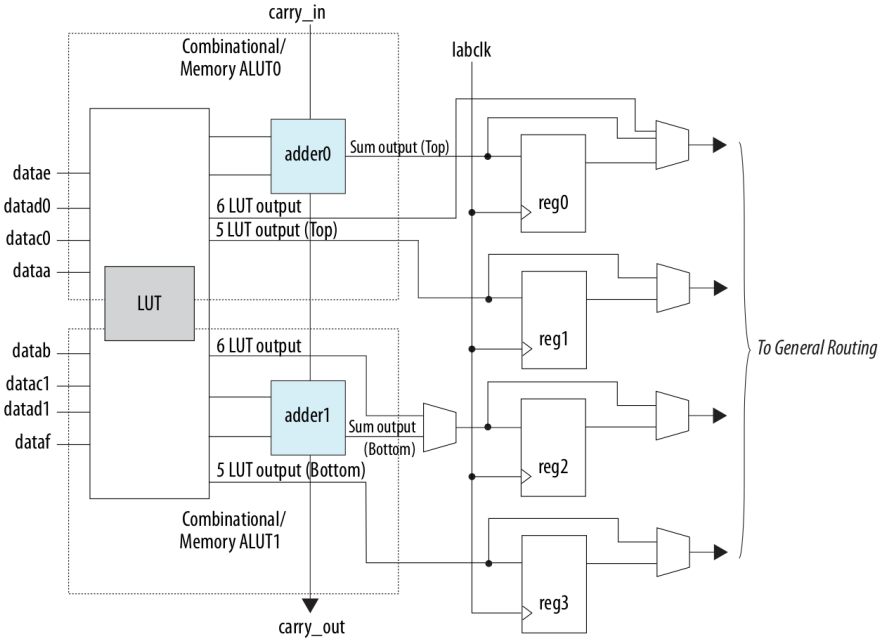

FIGURE 2.13 – Adaptive Logic Module (ALM) of the Logic Array Block (LAB) of the Intel Stratix 10 FPGA family. Source: reproduced from INTEL (2018).

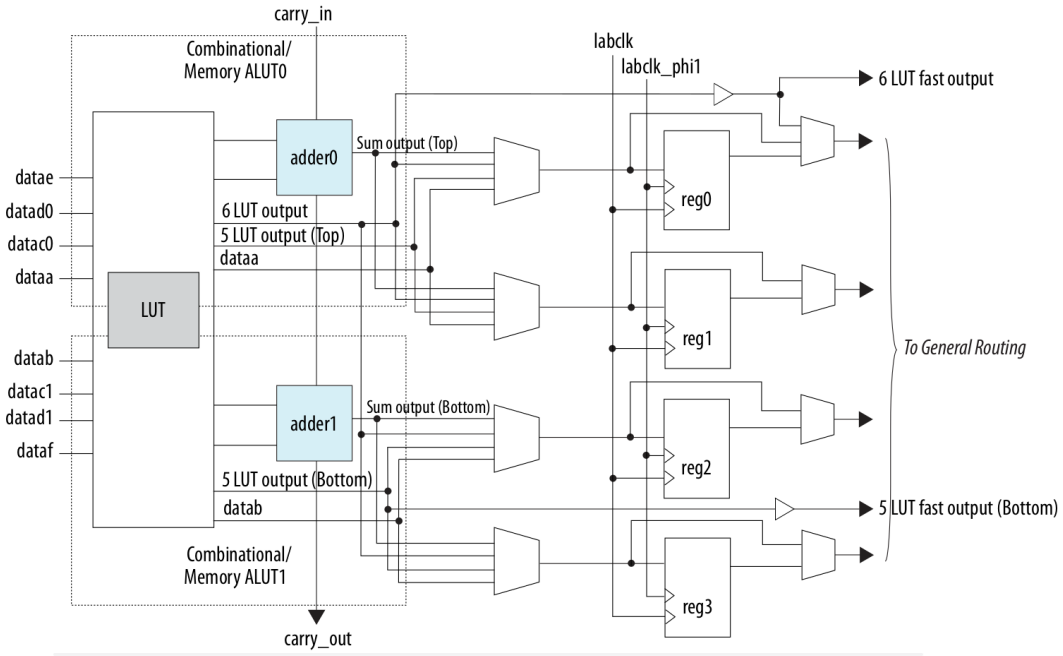

FIGURE 2.14 – Adaptive Logic Module (ALM) of the Logic Array Block (LAB) of the Intel Agilex FPGA family. Source: reproduced from INTEL (2019).

Finally, in Fig. 2.15, we present the CLB used by the IGLOO, ProASIC3 and Fusion FPGA families, and their variants, from Microsemi (former Actel, now a Microchip Company). This CLB has an unusual architecture, since it does not have a LUT and a dedicated memory element. Contrarily, it has a small set of logic gates and configuration switches that allows the implementation of combinational and sequential functions.

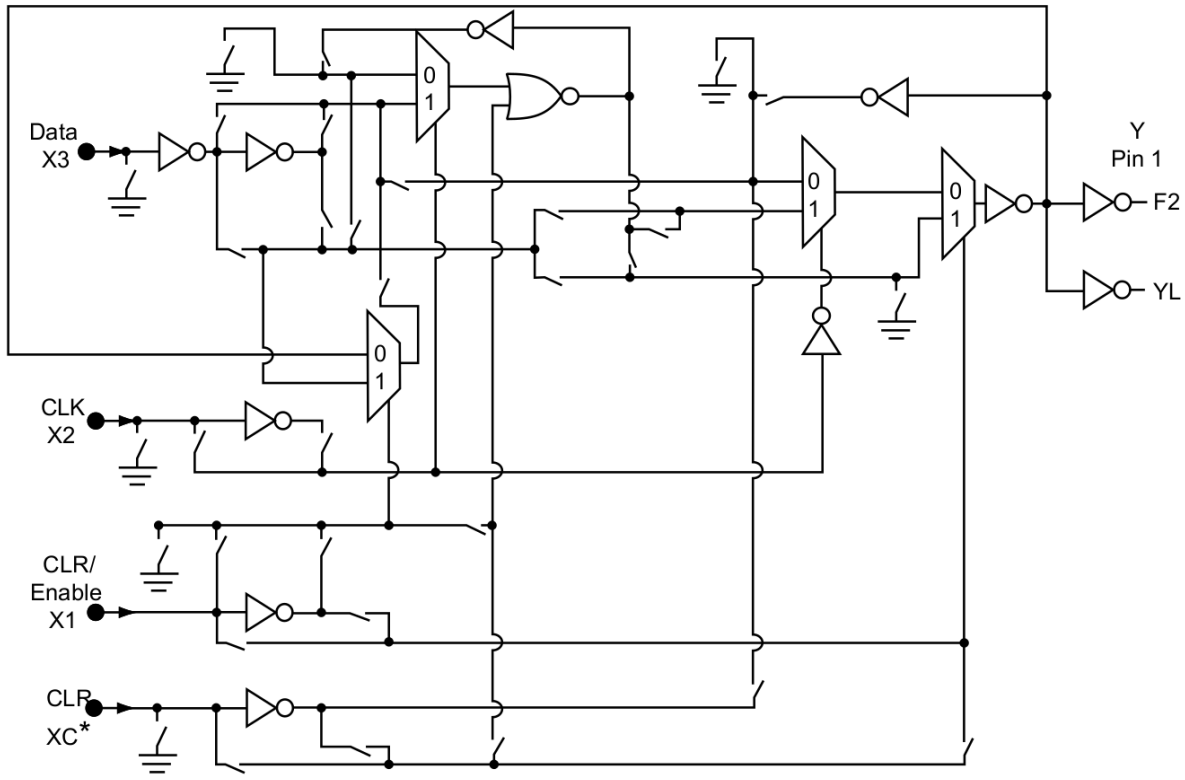

FIGURE 2.15 – Configuration Logic Block used in the Microsemi IGLOO, ProASIC3, and Fusion FPGA families and their variants. Source: reproduced from MICROSEMI (2012a).

This CLB is the chosen technology (detailed in Section 4.1) used for the evaluation of the proposed strategy, that is described in the next section. Since its architecture does not have a LUT, the effects of the logic-gate internal circuit are more evident and, thus, it is more suitable for the proposed analysis.

3 Proposed Strategy for SEV Estimation

As discussed in the previous chapter, the logic simulation and the emulation-based approaches for SEV estimation do not take the SET susceptibilities of the logic gates into consideration. This scenario motivated us to investigate a SEV estimation strategy that considers these susceptibilities and can be adopted in both logic simulation and emulation-based approaches. In this chapter, we present the proposed strategy for which we defined two versions: a simplified and a complete one.

The simplified version assumes that the SET susceptibility of a logic gate depends only on its internal circuitry. In other words, different types of logic gates (and different circuits for the same logic function) have different SET susceptibilities, e.g. an OR and an AND (or two AND with different circuits), but every instance of the same type of logic gate (with the same circuit) has the same SET susceptibility. Complementary, the complete version considers that logic gate operation also affects the SET susceptibility of the logic gate, in addition to the internal circuitry effect. In other words, each logic gate has a specific SET susceptibility, even though it is from the same type (with the same internal circuit) of another logic gate.

Both versions of the strategy rely on the same SEV estimation framework structure, illustrated in Fig. 3.1. This framework enables SET injection and SE detection comparing the results of two versions of the Device Under Test (DUT), while running the same operational scenario. In Fig. 3.1, the framework is represented as a testbench file, named

`dut_sev_tb.vhd`, intended to be used in logic simulations. However, a similar framework structure could be embedded for use in emulation-based approaches. Nonetheless, in the remainder of this chapter, the framework is considered as a testbench file, to exemplify the steps of the strategy, that also considers its application in a logic simulation environment. The components of the framework are described hereunder.

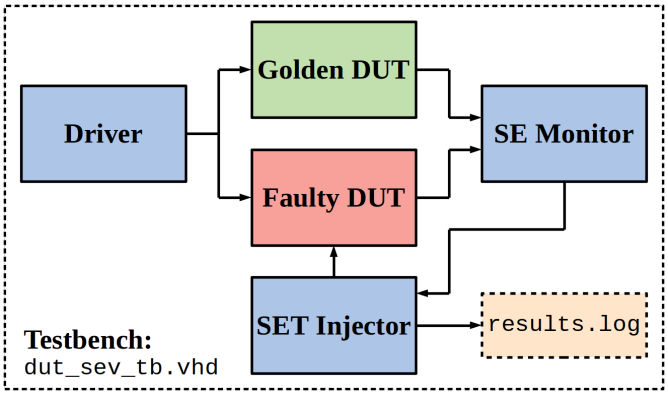

FIGURE 3.1 – Proposed framework for SEV estimation. This framework enables SET injection and SE detection comparing the results of two versions of the DUT, while running the same operational scenario. It is represented as a testbench for logic simulation environments, but could be embedded into emulation-based platforms.

Golden DUT is the version of the DUT that is not subjected to SET injection.

Faulty DUT is the version of the DUT in which the SETs are injected. The SET injection, at the output of the CLBs inside the Faulty DUT, is performed using a *saboteur* for each CLB. This element inverts the logic value of the CLB output while it is enabled and a SET is injected.

Driver is responsible for the operational scenario. It applies the same sequence of test vectors to the inputs of both Golden DUT and Faulty DUT.

SE Monitor compares the outputs of both DUTs, detecting the occurrence of a SE. When a SE occurs, the SE Monitor warns the SET Injector.

SET Injector injects the SETs into the Faulty DUT, through the saboteurs. It controls when, where and how many SETs must be injected. Additionally, it generates the report for the SEV estimation process, with the total number of injected SETs and observed SEs.

Section 3.1 and Section 3.2, respectively, present the simplified and complete versions of the proposed strategy, while in Sections 3.3 to 3.6, we detail some of the steps required by the strategy.

3.1 Simplified Version of the Strategy

The simplified version of the proposed SEV estimation strategy has five steps, as illustrated in Fig. 3.2, that are described in what follows. In this process, we adopted the VHDL language, but it could be used with another HDL, e.g., Verilog. This possibility is discussed in Chapter 7.

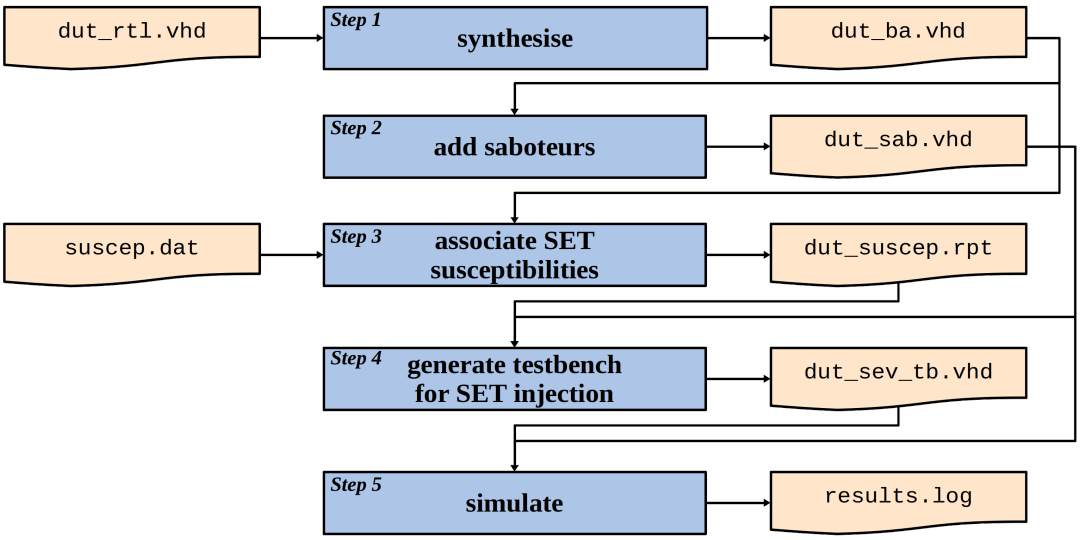

FIGURE 3.2 – Simplified proposed strategy for SEV estimation. The overall process has five steps that require two inputs to produce the SEV estimation and other four intermediate sub-products.

Step 1 is to synthesise the RTL description (`dut_rtl.vhd`) of the DUT to the target technology, generating the back-annotated post-synthesis netlist (`dut_ba.vhd`). It is essential to set the synthesis tool to do not insert I/O blocks so that the netlist will include only the CLBs. In this step, we used Synplify Pro L-2016.09M-2.

Step 2 is to insert the saboteurs into the DUT, generating the netlist for the Faulty DUT (`dut_sab.vhd`), as detailed in Section 3.3. The Golden DUT is the original post-synthesis netlist (`dut_ba.vhd`).

Step 3 is to associate to each CLB its respective SET susceptibility, based on the reference database (`suscep.dat`). The result is a report file (`dut_suscep.rpt`) with the SET susceptibilities of each CLB used in the DUT. The `suscep.dat` file is discussed in Chapter 4.

Step 4 is to generate the testbench used in the simulation framework (`dut_sev_tb.rpt`), incorporating the SET susceptibility of each CLB, as described in Section 3.6.

Step 5 is the actual simulation of the DUT injecting the weighted distribution of SETs, generating a report (`results.log`) with the estimated SEV for the circuit. In this step, we used ModelSim SE-64 10.6a.

As illustrated in Fig. 3.2, and described above, this simplified SEV estimation process requires two inputs:

1. a synthesisable description of the DUT; and
2. a reference database with the SET susceptibilities of each logic gate type.

In the next section we present the complete strategy, that has two additional steps and a modified one. After that, we deepen the discussion about the details of the steps.

3.2 Complete Version of the Strategy

The complete version of the proposed SEV estimation strategy has seven steps, as illustrated in Fig. 3.3, that are described below. Again, in this process, we adopted the VHDL language.

Step 1 is to synthesise the RTL description, as in the simplified version.

Step 2 is to insert the saboteurs into the DUT, as in the simplified version.

Step 3 is to simulate the DUT with the desired operational scenario to generate the Value Change Dump (VCD) file (`dut_tb.vcd`), from which the input patterns are

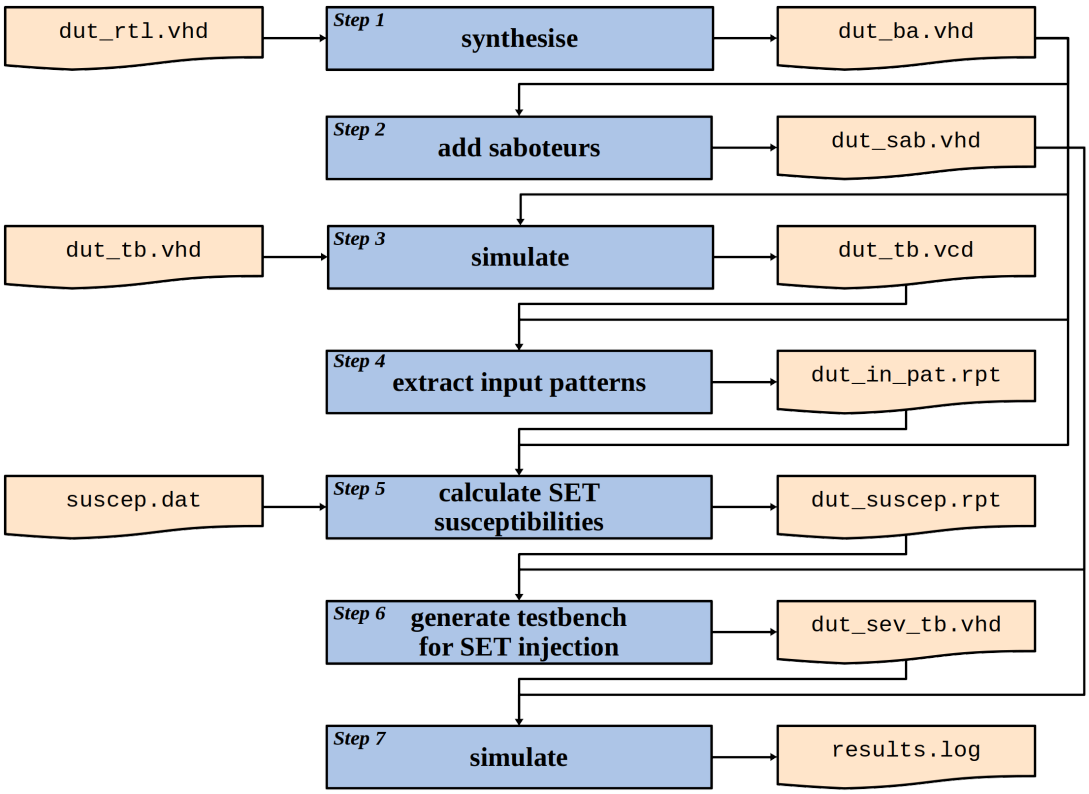

FIGURE 3.3 – Complete proposed strategy for SEV estimation. The overall process has seven steps that require three inputs to produce the SEV estimation and other six intermediate sub-products.

extracted. The testbench (`dut_tb.vhd`) for this simulation must use the same Driver used in the simulation framework, discussed at the beginning of this chapter, to assure the use of the same operational scenario. In this step, we used ModelSim SE-64 10.6a.

Step 4 is to extract the input patterns from the VCD file, generating a report (`dut_in_pat.rpt`) with the percentage of time that each CLB was submitted to each input pattern. The details of this process are presented in Section 3.4.

Step 5 is to calculate the specific SET susceptibilities of each CLB for the operational scenario, using the input patterns report (`dut_in_pat.rpt`) and the SET susceptibilities information (`suscep.dat`). The result is a report file (`dut_suscep.rpt`) with the SET susceptibilities of each CLB used in the DUT. The calculus done in this step is detailed in Section 3.5. This step is a modification of the Step 3 from the simplified version.

Step 6 is to generate the testbench used in the simulation framework, as in the Step 4 of the simplified version.

Step 7 is the actual simulation of the DUT injecting the weighted distribution of SETs, as in the Step 5 of the simplified version.

As illustrated in Fig. 3.3, and described above, the complete SEV estimation process requires three inputs:

1. a synthesisable description of the DUT;
2. a testbench that executes the intended operational scenario; and
3. a reference database with the SET susceptibilities of each CLB as a function of the input patterns.

The complete strategy requires two additional steps, to simulate the intended operational scenario and to extract the input patterns distribution, and a modification in the step that determine the specific SET susceptibility of each logic gate. While each susceptibility is directly get from a database in the simplified version, in the complete one each specific susceptibility is calculated using the database and the input patterns.

All seven steps can be automated by scripts. In fact, the validation of this approach, described in Chapter 5, used a set of scripts. The synthesis and simulations steps were automated using TCL (Tool Command Language) scripts. The steps for inserting saboteurs, extracting the input patterns, calculating the SET susceptibilities, and generating the testbench are based on file manipulation and basic mathematical operations. For them, we implemented scripts using Python 3.6, without any specific package.

In Section 3.3, we describe the process for adding saboteurs. Next, in Section 3.4, we detail the extraction of the input patterns, while in Section 3.5, we demonstrate how we calculate the SET susceptibilities based on the input patterns. Finally, in Section 3.6, we discuss how to generate the testbench used as SEV estimation framework.

3.3 Adding Saboteurs

As introduction in the proposed framework discussion, at the begining of this chapter, the saboteur is an element that inverts the logic value of an intercepted signal, when it is enabled, and a SET is injected, as represented in (3.1)

$$output_{signal} = input_{signal} \oplus (sab\_enable \cdot fault), \tag{3.1}$$

where $output_{signal}$ is the intercepted signal that receives the transient, $input_{signal}$ is the original signal, sab_enable is the enabling signal of the saboteur, and $fault$ is the signal to inject the SET.

In the selected technology, described in Section 4.1, there is a CLB macro configuration that implements the logic function of (3.1): AX1C.

The process of inserting saboteurs into the DUT, that is exactly the same in both versions of the strategy, consists of:

1. at the port of the entity, add the fault (SET) signal and the enable signals for each CLB;
2. for each CLB, create a new signal, that replaces the original output of the CLB and is used as the input of the saboteurs;
3. connects the output of the CLB to this new signal; and
4. add an instance of the saboteur, mapping its input to the new signal, its output to the original output signal of the CLB, its enable signal to the respective enable port of the entity, and the fault signal to the fault port of the entity.

Fig. 3.4 exemplifies this process, comparing the VHDL files before and after adding the saboteurs, for the *b1* benchmark, from Yang (1991). This figure highlights the four modifications described above.

```
 1    library ieee;                          library ieee;
 2    use ieee.std_logic_1164.all;           use ieee.std_logic_1164.all;
 3    use ieee.numeric_std.all;              use ieee.numeric_std.all;
 4    library proasic3e;                     library proasic3e;
 5    use proasic3e.components.all;          use proasic3e.components.all;
 6
 7    entity b1 is                           entity b1 is
 8    port(                                  port(
 9 *                                           Fault : in std_logic;
10 *                                           Enable : in std_logic_vector(2 downto 0);
11      a :  in std_logic;                     a :  in std_logic;
12      b :  in std_logic;                     b :  in std_logic;
13      c :  in std_logic;                     c :  in std_logic;
14      d :  out std_logic;                    d :  out std_logic;
15      e :  out std_logic;                    e :  out std_logic;
16      f :  out std_logic;                    f :  out std_logic;
17      g :  out std_logic);                   g :  out std_logic);
18    end b1;                                end b1;
19
20    architecture beh of b1 is              architecture beh of b1 is
21 *                                           signal g_sab : std_logic;
22 *                                           signal f_sab : std_logic;
23 *                                           signal E_2_sab : std_logic;
24      signal E_2 : std_logic ;               signal E_2 : std_logic ;
25      signal NN_1 : std_logic ;              signal NN_1 : std_logic ;
26      signal NN_2 : std_logic ;              signal NN_2 : std_logic ;
27    begin                                  begin
28 *                                         Saboteur_0: AX1C port map (
29 *                                             Y => g,
30 *                                             A => Enable(0),
31 *                                             B => Fault,
32 *                                             C => g_sab);
33 *                                         Saboteur_1: AX1C port map (
34 *                                             Y => f,
35 *                                             A => Enable(1),
36 *                                             B => Fault,
37 *                                             C => f_sab);
38 *                                         Saboteur_2: AX1C port map (
39 *                                             Y => E_2,
40 *                                             A => Enable(2),
41 *                                             B => Fault,
42 *                                             C => E_2_sab);
43    C_RNI33: INV port map (                C_RNI33: INV port map (
44 *      Y => g,                                Y => g_sab,
45        A => c);                               A => c);
46    UN4_P: XA1B port map (                 UN4_P: XA1B port map (
47 *      Y => f,                                Y => f_sab,
48        A => a,                                A => a,
49        B => c,                                B => c,
50        C => E_2);                             C => E_2);
51    B_RNI36: XOR2 port map (               B_RNI36: XOR2 port map (
52 *      Y => E_2,                              Y => E_2_sab,
53        A => a,                                A => a,
54        B => b);                               B => b);
55    VCC_I: VCC port map (                  VCC_I: VCC port map (
56        Y => NN_2);                            Y => NN_2);
57    GND_I: GND port map (                  GND_I: GND port map (
58        Y => NN_1);                            Y => NN_1);
59    d <= c;                                d <= c;
60    e <= E_2;                              e <= E_2;
61    end beh;                               end beh;
```

FIGURE 3.4 – Comparison of VHDL files before and after adding the saboteurs to the *b1* benchmark (YANG, 1991). On the left is the original content, with some additional blank lines to facilitate the comparison, and on the right is the resulting content. The asterisks on the left margin mark the lines in which some content was inserted or modified: (1) the Fault and Enable ports were added in lines 9–10; (2) the new signals were created in lines 21–23, with suffix _sab; (3) the output of the CLBs were connected to the new signals in lines 44, 47 and 52; and (4) the saboteurs were added in lines 28–42.

3.4 Extraction of the Input Patterns Distribution

The VCD file is a dump file generated by the logic simulation tools, normally used for waveform visualisation. It is an ASCII-based file that includes ascending time-ordered value changes of the signals (INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, 1996). Essentially, it has all the time marks in which there are value changes of at least one signal. Following a time mark, there is a list with all the signals that have a value change at that time, with their new values.

The information to be extracted from the VCD file is the percentage of time that each input pattern is applied to each CLB. The first step is to transform this textual representation into a table, with a list of time intervals and the values of each signal in the respective interval. Then, the time intervals in which each pattern is applied to the CLB are summed, generating the total time for each pattern. With the total time of the simulated scenario, the percentages can be calculated.

To illustrate how the input patterns extraction occurs, consider the example circuit presented in Fig. 3.5, the graphic representation of a VCD file illustrated in Fig. 3.6, and the respective VCD file, containing the analysed operational scenario, reproduced in the Listing 3.1.

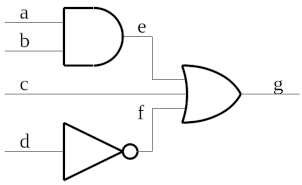

FIGURE 3.5 – Example circuit for input patterns extraction.

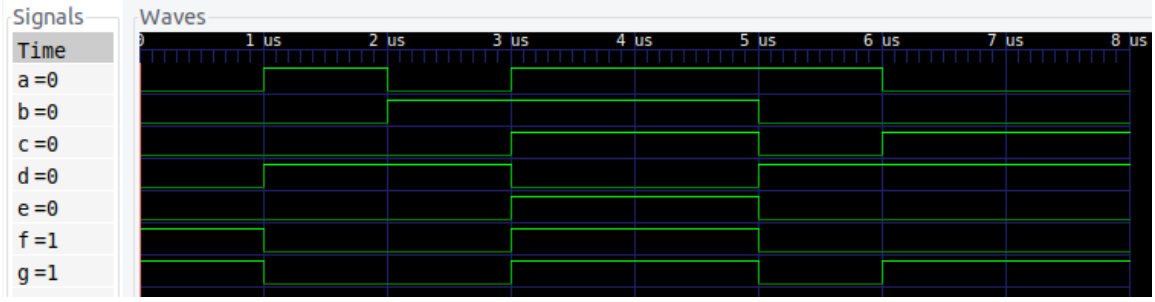

FIGURE 3.6 – Graphical visualisation of the example VCD file using the GtkWave tool.

```
1  $date [date] $end
2  $version [tool version] $end
3  $timescale 1ns $end
4  $scope module example $end
5  $scope module u_dut $end
6  $var wire 1 ! a $end
7  $var wire 1 " b $end
8  $var wire 1 # c $end
9  $var wire 1 $ d $end
10 $var wire 1 % e $end
11 $var wire 1 & f $end
12 $var wire 1 ’ g $end
13 $enddefinitions $end
14 #0
15 $dumpvars
16 0!
17 0"
18 0#
19 0$
20 0%
21 1&
22 1’
23 $end
24 #1000
25 1!
26 1$
27 0&
28 0’
29 #2000
30 0!
31 1"
32 #3000
33 1!
34 1#
35 0$
36 1%
37 1&
38 1’
39 #5000
40 0"
41 0#
42 1$
43 0%
44 0&
45 0’
46 #6000
47 0!
48 1#
49 1’
50 #8000
```

LISTING 3.1 – VCD file with the operational scenario used as example. The time marks starts with # (lines 14, 24, 29, 32, 39, 46, and 50), and are expressed in the timescale defined in line 3 (1 ns). Each signal is associated to a symbol to reduce the file size if long names are used (lines 6–12). The value change are represented by the new value followed by the signal symbol, e.g., in line 49, signal g changes to ‘1’.

For this VCD, the time marks are the instants 0, 1, 2, 3, 5, 6 and 8 μs (lines 14, 24, 29, 32, 39, 46, and 50 in the Listing 3.1). After the first time mark (0 μs), each signal receives its initial value (a = 0, b = 0, c = 0, d = 0, e = 0, f = 1, and g = 1). On the next time mark (1 μs), the signals with value changes receive their new values (a = 1, d = 1, f = 0, and g = 0). This process is repeated for the other time marks, except for the last one (8 μs), that is not followed by any value change assignment.

The intermediate table, with time intervals and signals values for this example, are illustrated in Table 3.1. To improve its readability, it also includes columns for the inputs of each logic gate: {a b} for the AND, {d} for the Inverter, and {e c f} for the OR gate.

TABLE 3.1 – Intermediate table of the input patterns extraction process. Tabular representation of the operational scenario defined in the VCD file reproduced in Listing 3.1, and illustrated in Fig. 3.6.

Interval [μs]	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**{a b}**	**{d}**	**{e c f}**
1	0	0	0	0	0	1	1	00	0	001
1	1	0	0	1	0	0	0	10	1	000
1	0	1	0	1	0	0	0	01	1	000
2	1	1	1	0	1	1	1	11	0	111
1	1	0	0	1	0	0	0	10	1	000
2	0	0	1	1	0	0	1	00	1	010

The AND gate has four input patterns 00, 01, 10, and 11. Assuming the time in which each pattern is applied is represented as $\{t_{00}, t_{01}, t_{10}, t_{11}\}$, processing the values presented in Table 3.1 results in {3, 1, 2, 2} μs. As the total time is 8 μs, the percentage of time in each input pattern $\{P_{00}, P_{01}, P_{10}, P_{11}\}$ is {37.5, 12.5, 25.0, 25.0}.

Similarly, the Inverter has two input patterns with time percentages of {37.5, 62.5}, and the OR gate has eight input patterns with the time percentages of {37.5, 12.5, 25.0, 0.0, 0.0, 0.0, 0.0, 25.0}.

3.5 Calculation of the SET Susceptibilities

The complete strategy requires the specific SET susceptibility of each CLB of the DUT, that is calculated as a weighted sum of the SET susceptibilities of the respective CLB configuration for each input pattern. The weights are the percentage of time that each input pattern is applied to the CLB. So, the SET susceptibility of a specific instance of a CLB configuration can be obtained with the equation below,

$$\chi_{CLB_n} = \sum_{pattern=0_b}^{(2^i-1)_b} \chi_{CLB_{pattern}} \cdot P_{CLB_{n,pattern}}, \tag{3.2}$$

where χ_{CLB_n} is the SET susceptibility of the specific instance n of the CLB configuration; i is the number of inputs of the CLB configuration; *pattern* is the possible input patterns for the CLB configuration; $\chi_{CLB_{pattern}}$ is the SET susceptibility of the CLB configuration for the *pattern*; and $P_{CLB_{n,pattern}}$ is the percentage of time that the instance n of the CLB configuration is submitted to *pattern*.

To exemplify this calculus, consider two instances of a 2-input AND gate, represented by the CLB configuration AND2 ($AND2_0$ and $AND2_1$). Additionally, consider the SET susceptibility of the four input patterns of the configuration AND2 to be {7%, 11%, 15%, 21%}. Finally, consider that the time distribution of the patterns are $P_{AND2_0} = \{10\%, 15\%, 25\%, 50\%\}$ and $P_{AND2_1} = \{40\%, 30\%, 15\%, 15\%\}$. The SET susceptibility of each instance can be calculated using (3.3), expanded in (3.4), resulting in $\chi_{AND2_0} = 16.6\%$ and $\chi_{AND2_1} = 11.5\%$ of the SET that produced in the transistors of the CLB.

$$\chi_{AND2_n} = \sum_{pattern=00_b}^{11_b} \chi_{AND2_{pattern}} \cdot P_{AND2_{n,pattern}} \tag{3.3}$$

$$\chi_{AND2_n} = \chi_{AND2_{00}} \cdot P_{AND2_{n,00}} + \chi_{AND2_{01}} \cdot P_{AND2_{n,01}} + \chi_{AND2_{10}} \cdot P_{AND2_{n,10}} + \chi_{AND2_{11}} \cdot P_{AND2_{n,11}} \tag{3.4}$$

3.6 Generating the Testbench for SEV Estimation

The testbench used as the SEV estimation framework is composed of five elements, as proposed at the beginning of this chapter, and has the same structure for both simplified and complete versions of the strategy. The generating process for this testbench consists on creating the specific SET Injector and interconnecting it with the other elements, as described in what follows.

The Driver defines the operational scenario and, for the complete version, must be the same used in the first simulation. For this version, it is an input for the overall SEV estimation process.

The Golden DUT and the Faulty DUT are sub-products of the overall process. The first one is obtained with the synthesis (Step 1), while the second is obtained adding the saboteurs (Step 2).

The SE Monitor is a parameterised comparator that can be used with any DUT, requiring only the adjustment of its input data width parameter to match the number of output signals from the DUT.

Regarding the SET Injector, it is specific for each operational scenario applied to the DUT. The SET distribution is performed using the Open Source VHDL Verification Methodology (OSVVM) (OSVVM.ORG, 2018). This methodology includes the RandomPkg package, that provides a set of functions to generate pseudo-random numbers, including a weighted distribution function for integer values (DistValInt). The SET Injector relies on this function to select the CLB in which the SET will be injected. As it is generated for any specific SET distribution, its generation process applies to both simplified and complete versions of the proposed strategy.

Considering the two instances of the CLB configuration AND2 used as an example in Section 3.5 ($AND2_0$ and $AND2_1$), the selection of in which instance a SET would be injected can be obtained with the VHDL attribution exemplified in Listing 3.2. The variable TargetCLB, used as an index to select the CLB in which the SET will be injected,

can receive 0 or 1, for $AND2_0$ and $AND2_1$, with higher probability of receiving 0: for every 281 SETs, 166 are targeted to $AND2_0$, and 155 to $AND2_1$.

```
TargetCLB := RV.DistValInt(( (0, 166), (1, 115) ));
```

LISTING 3.2 – DistValInt function of the RandomPkg package. Example of how to obtain a weighted distribution.

For the same example, the average susceptibility of the two instances of AND2 is 14.05%. Assuming that the SEV estimation process would inject 100000 SETs in the transistors of the CLB, on average, 14.05% of the 100000 SETs should be injected at the output of each CLB. With two instances, the total number of SETs that should be injected is 28100, that would be distributed as 16600 in $AND2_0$ and 11500 in $AND2_1$. The total number of SETs that needs to be injected, $N_{SET_{Total}}$, is given by (3.5),

$$N_{SET_{Total}} = N_{CLB} \cdot N_{Transistor_{CLB}} \cdot N_{SET_{Transistor}} \cdot \chi_{CLB_{average}} \tag{3.5}$$

where N_{CLB} is quantity of CLBs used in the DUT; $N_{Transistor_{CLB}}$ is the quantity of transistors per CLB; $N_{SET_{Transistor}}$ is the desired number of SETs at the transistor level; and $\chi_{CLB_{average}}$ is the average susceptibility of the CLBs used in the DUT.

In addition to where (in which CLB) and how many SETs are injected, the SET Injector also controls when they are injected. The SET injections must be distributed evenly through the clock cycles of the operational scenario.

4 Estimation of the SET Susceptibilities

The proposed strategy for SEV estimation requires the SET susceptibility of the elements that compose the circuit. However, unfortunately, this information is not usually available. For this reason, we estimated the SET susceptibility of the configurable functions of an FPGA technology family that we chose.

In Section 4.1, we describe the chosen technology while, in Section 4.2, we detail the logic simulation model of the respective CLB that we developed to estimate the SET susceptibilities.

This CLB can be configured to combinational and sequential functions with the appropriate configuration vector. Section 4.3 describes how we defined the configuration vector of each function. These functions were grouped as components in a VHDL library that can substitute the original library of the chosen technology, as detailed in Section 4.4.

The configured functions added to this library were submitted to the SET susceptibility estimation process described in Section 4.5, resulting in the susceptibilities presented in Section 4.6.

This estimation provides the average SET susceptibility of each logic gate required by the simplified version of the proposed strategy (see Section 3.1), and the parameterised SET susceptibilities required by the complete version (see Section 3.2).

4.1 Analysed Technology

This work focuses on the ProASIC3E FPGA family (MICROSEMI, 2015). It is a 130 nm flash-based CMOS device. As already discussed, unlike most other technologies, this CLB is not based on a sequential element and a LUT. Instead, it has a small set of logic gates that can be configured as combinational and sequential functions. The CLB of this family is called VersaTile (illustrated in Fig. 2.15).

The devices from this family have a dedicated Flash ROM memory that can be programmed when the device is configured. They also have dedicated dual-port SRAM blocks, each of them with its own FIFO controller.

Additionally, all devices have six Clock Conditioning Circuits (CCC), with integrated PLL, and has an Input/Output structure that allows its configuration from many options (1.5 V, 1.8 V, 2.5 V and 3.3 V; single-ended/differential).

The synthesis target device is an A3PE1500-PQ208. Table 4.1 presents the main features of this device.

TABLE 4.1 – ProASIC3E A3PE1500-PQ208 main features. Source: MICROSEMI (2012a).

Feature	**A3PE1500-PQ208**
System Gates	1,500,000
VersaTiles	38,400
SRAM [kbits]	270
SRAM Blocks [4,608 bits]	60
FlashROM [kbits]	1
CCC	6
PLL	2
Global Nets	18
I/O banks	8
Max. Single Ended I/O	147
Max. Differential I/O pairs	65

The VersaTile CLB has four inputs and one logic output, with two physical options: F2 for local nets; and YL for long connections. This CLB is composed of 16 CMOS logic

gates (eleven Inverters, four 2-Input Multiplexers, and one 2-Input NOR) interconnected through 32 flash-based switches. Its internal structure allows the configuration of any 3-input combinational function and a set of sequential functions, as latches and flip-flops, with enable and set/clear options (MICROSEMI, 2012a). The configurations available for this CLB are presented as macros in (MICROSEMI, 2010).

The design documents of this technology do not present the implementation details of the CLB at the transistor level – like all other researched commercial technologies. For this reason, we make some assumptions to make this study feasible. First, we assumed some CMOS circuits for the internal logic gates of the Versatile (described in Section 4.2). Second, we arbitrarily chose the values of the configuration vector for each function configured in a CLB (discussed in Section 4.3). Third, we assumed that all the transistors of the CLB have the same cross-section, without taking into account possible layout effects (described in Section 4.5). Finally, as the focus of SET susceptibility estimation is on the logical masking, the electrical and the timing characteristics were not modelled. Although these assumptions impact the quantitative values presented in this work, they do not undermine the general qualitative analyses.

4.2 CLB Simulation Model

The VersaTile CLB was modelled at the transistor level to reproduce its logical behaviour, with no concerns related to the electrical and timing characteristics. It was described in a structural VHDL as an interconnection of PMOS and NMOS transistors, implementing both logic gates and configuration switches.

For the logic gates, we assumed the circuits illustrated in Fig. 4.1, because they are compact and suitable for the modelling with logic switches, as described in this section. This assumption results in 50 transistors for the logic gates of the CLB, being 25 PMOS and 25 NMOS (22 for the eleven Inverters, 24 for the four 2-Input Multiplexers, and 4 for the 2-Input NOR). Additionally, each configuration switch was modelled as an NMOS transistor. The resulting CLB model has 82 transistors. Fig. 4.2 illustrates the CLB with

the adopted IDs, while Table 4.2 list the 82 transistors and their functions on the model. Lastly, the entity of the VersaTile model is presented in Listing 4.1.

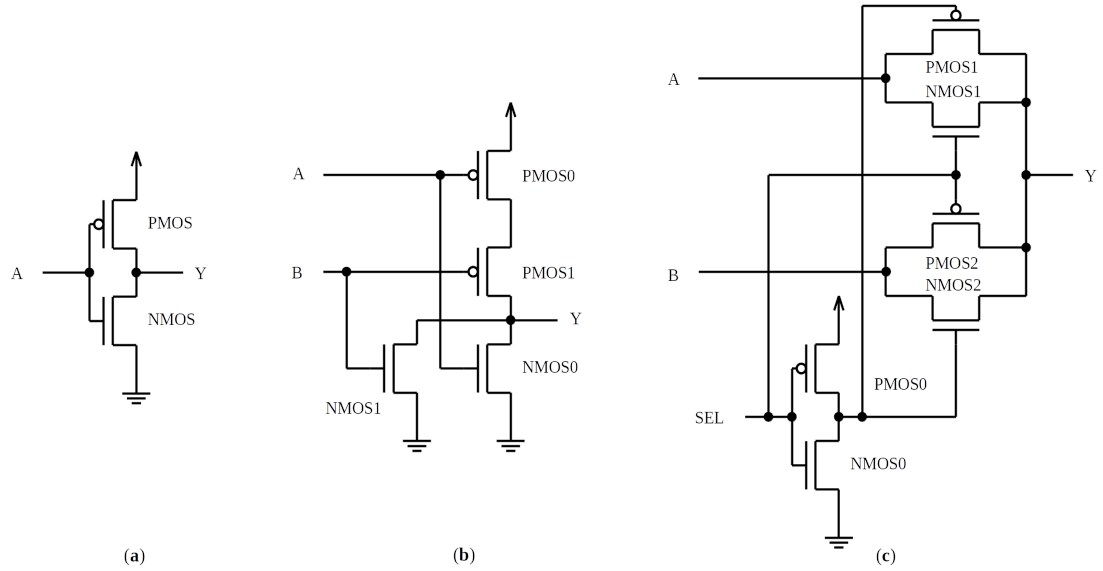

FIGURE 4.1 – Adopted circuits for the internal logic gates of the Versatile CLB – (**a**) Inverter; (**b**) 2-Input NOR; and (**c**) 2-Input Multiplexer.

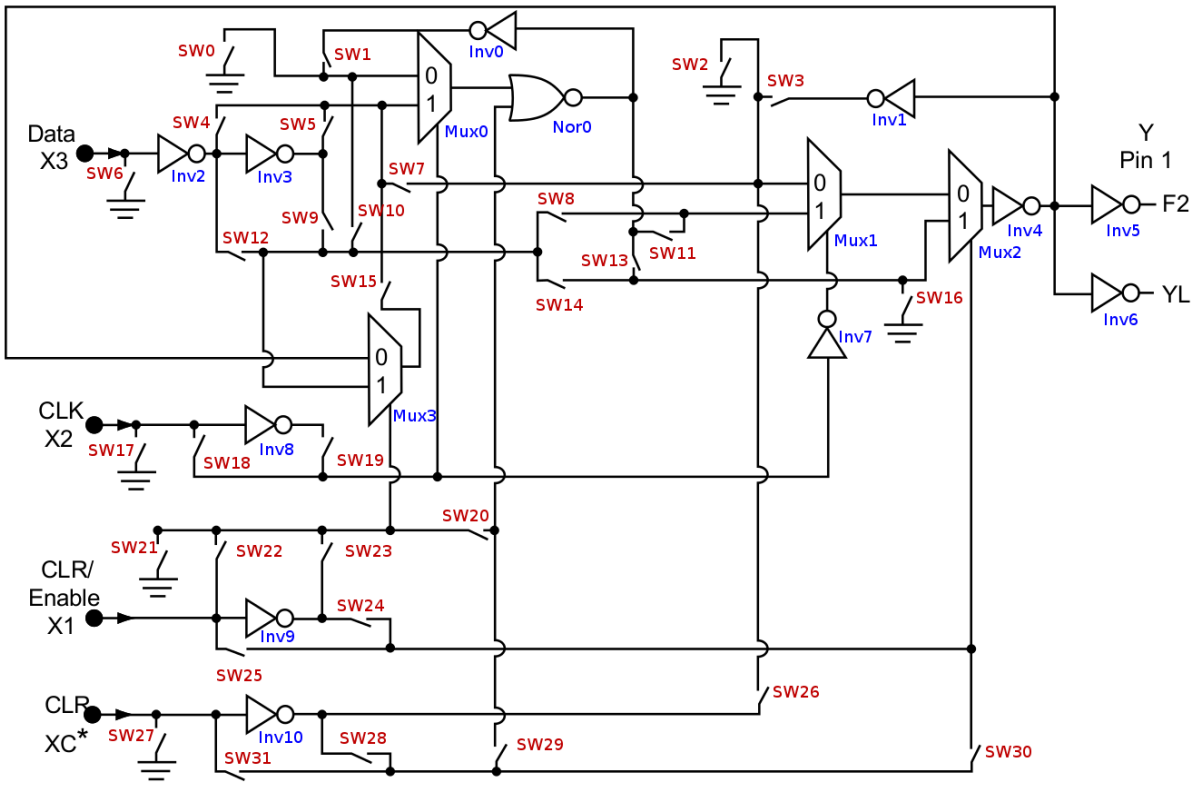

FIGURE 4.2 – VersaTile internal structure with the adopted IDs. This CLB is composed by 16 CMOS logic gates (INV10..INV0; MUX3..MUX0; NOR0), that can be interconnected through 32 switches (SW31..SW0). Source: adapted from (MICROSEMI, 2012a).

TABLE 4.2 – VersaTile transistors. For each transistor, it is presented its ID and function in the VersaTile model.

ID	**Function**	**ID**	**Function**	**ID**	**Function**
Q0	PMOS of Inv0	Q28	PMOS0 of Mux1	Q56	NMOS SW6
Q1	NMOS of Inv0	Q29	NMOS0 of Mux1	Q57	NMOS SW7
Q2	PMOS of Inv1	Q30	PMOS1 of Mux1	Q58	NMOS SW8
Q3	NMOS of Inv1	Q31	NMOS1 of Mux1	Q59	NMOS SW9
Q4	PMOS of Inv2	Q32	PMOS2 of Mux1	Q60	NMOS SW10
Q5	NMOS of Inv2	Q33	NMOS2 of Mux1	Q61	NMOS SW11
Q6	PMOS of Inv3	Q34	PMOS0 of Mux2	Q62	NMOS SW12
Q7	NMOS of Inv3	Q35	NMOS0 of Mux2	Q63	NMOS SW13
Q8	PMOS of Inv4	Q36	PMOS1 of Mux2	Q64	NMOS SW14
Q9	NMOS of Inv4	Q37	NMOS1 of Mux2	Q65	NMOS SW15
Q10	PMOS of Inv5	Q38	PMOS2 of Mux2	Q66	NMOS SW16
Q11	NMOS of Inv5	Q39	NMOS2 of Mux2	Q67	NMOS SW17
Q12	PMOS of Inv6	Q40	PMOS0 of Mux3	Q68	NMOS SW18
Q13	NMOS of Inv6	Q41	NMOS0 of Mux3	Q69	NMOS SW19
Q14	PMOS of Inv7	Q42	PMOS1 of Mux3	Q70	NMOS SW20
Q15	NMOS of Inv7	Q43	NMOS1 of Mux3	Q71	NMOS SW21
Q16	PMOS of Inv8	Q44	PMOS2 of Mux3	Q72	NMOS SW22
Q17	NMOS of Inv8	Q45	NMOS2 of Mux3	Q73	NMOS SW23
Q18	PMOS of Inv9	Q46	PMOS0 of Nor0	Q74	NMOS SW24
Q19	NMOS of Inv9	Q47	PMOS1 of Nor0	Q75	NMOS SW25
Q20	PMOS of Inv10	Q48	NMOS0 of Nor0	Q76	NMOS SW26
Q21	NMOS of Inv10	Q49	NMOS1 of Nor0	Q77	NMOS SW27
Q22	PMOS0 of Mux0	Q50	NMOS SW0	Q78	NMOS SW28
Q23	NMOS0 of Mux0	Q51	NMOS SW1	Q79	NMOS SW29
Q24	PMOS1 of Mux0	Q52	NMOS SW2	Q80	NMOS SW30
Q25	NMOS1 of Mux0	Q53	NMOS SW3	Q81	NMOS SW31
Q26	PMOS2 of Mux0	Q54	NMOS SW4		
Q27	NMOS2 of Mux0	Q55	NMOS SW5		

```
entity VersaTile_82 is
    generic(
        Config      : std_logic_vector( 31 downto 0 ) :=
                    "00000000000000000000000000000000"
        );
    port(
        X3Data      : in    std_logic;
        X2Clk       : in    std_logic;
        X1Enable    : in    std_logic;
        XcClr       : in    std_logic;
        F2          : out   std_logic;
        YL          : out   std_logic;
        Faults      : in    bit_vector( 81 downto 0 )
        );
end VersaTile_82;
```

LISTING 4.1 – VersaTile entity. The CLB function can be configured by the generic Config, and the SETs can be injected to each transistor through the port Faults. The other ports are the same illustrated in Fig. 4.2.

The MOS transistors were modelled as unidirectional logic switches: the output (Drain) receives the input (Source) value when the switch control (Gate) is activated (low logic level for PMOS, and high logic level for NMOS). Additionally, a fault model is incorporated to enable the injection of the SETs: while the Fault signal is '1', the drain is forced to '0', for the NMOS, or '1', for the PMOS; otherwise, the transistor operates with no fault. This fault modelling is based on the electrical modelling described by Buard and Anghel (2011), and discussed in Section 2.1.3 (see Fig. 2.3), in which a transient current source is connected between the body and the drain of the MOS transistor. It implies that the drain of a NMOS can be forced only to '0', while the drain of a PMOS can be forced only to '1'. The entity and architecture of the NMOS transistor model are presented in Listing 4.2.

The model uses the IEEE std_logic_1164 multi-value logic system that defines different strengths to represent the logic level of the signals. The nominal operation of the CLB model is based on the weak values ('H' and 'L'), while the fault effect uses strong forcing values ('1' and '0') to represent high- and low-level logic values in both cases, respectively. In this way, the fault effect can be imposed on any signal.

```
entity NMos is
    generic(
        OUTPUT_DELAY    : time := 0 ps
        );
    port(
        Gate            : in    std_logic;
        Source          : in    std_logic;
        Drain           : out   std_logic;
        Fault           : in    bit
        );
end entity NMos;

architecture Faulty_BH of NMos is
    signal  NoDelayOutput   : std_logic;
begin
    with Fault select NoDelayOutput <=
        '0'     when '1',
        'Z'     when '0';

    with Gate select NoDelayOutput <=
        Source  when '1',
        Source  when 'H',
        'Z'     when '0',
        'Z'     when 'L',
        'Z'     when 'Z',
        'U'     when 'U',
        'X'     when others;

    Drain <= NoDelayOutput after OUTPUT_DELAY;
end architecture Faulty_BH;
```

LISTING 4.2 – NMOS VHDL model. Both nominal and faulty behaviours were modelled with with-select constructions.

For example, for an Inverter, as illustrated in Fig. 4.1 (**a**), operating without fault (Fault = '0'), while the input value is 'L', the output value is 'H'. However, if a SET is injected in the NMOS transistor (Fault = '1'), the output will temporarily change to '0' during the desired pulse width the SET, resulting in the sequence {'H';'0';'H'}. However, if a similar SET is injected in the PMOS transistor, the output will temporarily change to '1', resulting in the sequence {'H';'1';'H'}. In the first case, there is a temporary inversion of the logic value. In the second one, there is no inversion, but it is possible to identify that a SET was injected in that node.

This modelling avoids that a signal receives unknown values, represented by 'X' or 'W' in the IEEE std_logic_1164 system, when a SET is injected on its node. However, there is a specific situation in which an unknown value may appear. When a SET injected in the Inverter of the 2-Input Multiplexer (Fig. 4.1 (**c**)), causing the momentarily logic inversion of this Inverter output, the input and the output of the Inverter will have the same logic value. In this case, the transmission gates of both inputs of the Multiplexer will transmit their values to the output. If these inputs have different logic values, the output will be unknown ('W'). To avoid that this unknown value propagates outside the VersaTile model, a process handles this situation. Whenever an unknown value ('X' or 'W') would be propagated to the output, the output receives the logic inversion of its immediately previous value. For example, if the VersaTile output value is 'H', and during a SET injection it would temporarily change to 'W' (sequence {'H';'W';'H'}), it will in fact change to 'L' (sequence {'H';'L';'H'}).

Finally, a small variation of the VersaTile model were created for the sequential functions with Preset inputs. In this variation, the configuration switch SW16 connects to the 'H' value, instead of the 'L', illustrated in Fig. 4.2.

4.3 Definition of the Configuration Vectors

As the design and manufacturing documents of this FPGA family do not provide the configuration vector data for each function, we defined possible configuration vectors as described hereunder.

For the combinational functions, a VHDL script analyses all 2^{32} possible values for the configuration vector, validating two aspects. First, the switches that configure loops shall be open (SW1, SW3 and SW15), since it characterises a possible sequential function. Second, every node has to have one, and only one, source. For example, considering the switches SW13, SW14 and SW16, if none of them is closed, the input 1 of the Mux2 will be left open. However, if more than one is closed, there will be a conflict between the values provided by the switches.

Then, the valid configurations are tested with all input patterns, and the results of the combinational function are logged in a csv (comma-separated values) file. There are many valid configuration vectors for each combinational function. For instance, the AND2 macro has 1815 possible configurations, while the XOR3 has 32.

The selection of which possible configurations should be used followed these arbitrary criteria:

1. have the least number of closed switches;
2. use these inputs, in order, X3Data, X2Clk, X1Enable, and XcClr; and
3. be the first in the list to meet the previous criteria.

Additionally, there are some combinational macro functions presented in (MICROSEMI, 2010) that has the same logic function. As an example, AND2 and NOR2B. The first one implements the function $Y = A \cdot B$, while the second implements $Y = \overline{\overline{A} + \overline{B}}$, that is the same. In these cases, we adopted the criteria presented above for the first macro (in alphabetical order). For the second one, we adopted the same criteria, except for the third item: be the *last* in the list to meet the previous criteria.

For the sequential functions, the 32-bit configuration vectors were obtained directly from the VersaTile circuit analysis. For example, the Mux1 is used for the Latch function (with SW3 closed). In conjunction with the Mux0 they are used for the Flip-Flop construction (with SW1 also closed). Moreover, the Mux2 adds the Clear/Preset functionality, while the Mux3 adds the Enable function (with SW15 closed). Lastly, the polarity of the input signals can be configured through the switches around the Inverter close to the respective input. The sequential macros that requires more than one CLB to implement its function were not modeled. These macros are the Latches/Flip-Flops with both Clear and Preset inputs (MICROSEMI, 2010).

To illustrate these configurations, Fig. 4.3 shows the resulting circuit for the combinational AND3 macro, while Fig. 4.4 shows the resulting circuit for the sequential DFN1C0 macro.

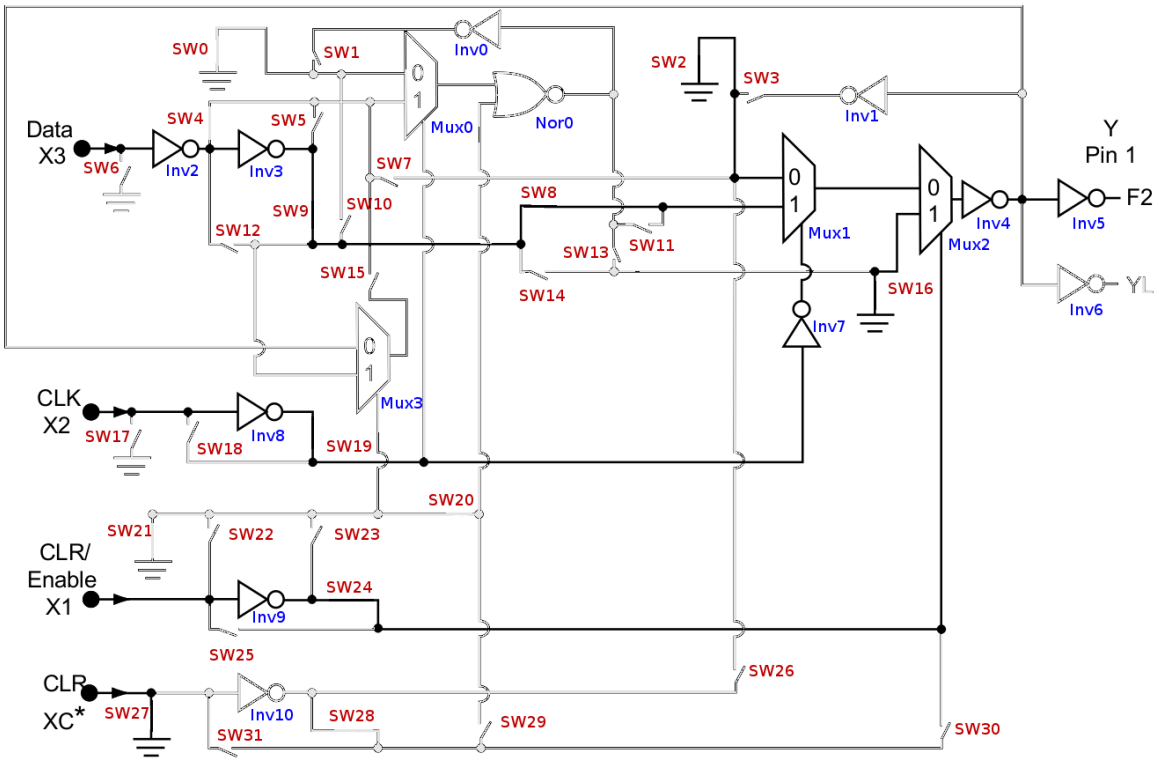

FIGURE 4.3 – VersaTile configured as an AND3 macro. Source: adapted from (MICROSEMI, 2012a).

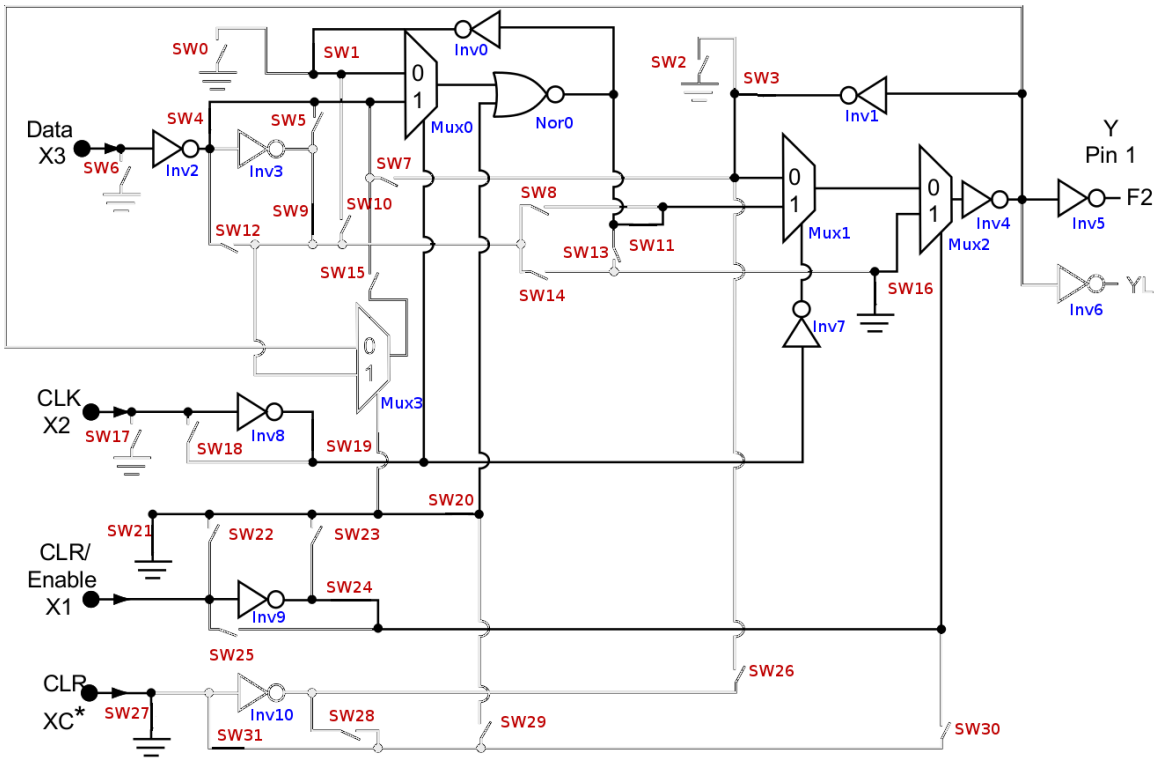

FIGURE 4.4 – VersaTile configured as a DFN1C0 macro, a D-flip-flop with non-inverting output, active high clock, and active low clear. Source: adapted from (MICROSEMI, 2012a).

The detailed configuration data of each VersaTile function is presented in Appendix A, divided in three tables: Table A.1 for the combinational functions; Table A.2 for the flip-flops; and Table A.3 for the latches. For each macro, it presents the configuration vector and the input mapping.

4.4 CLB Library

In order to simplify the substitution of the original ProASIC3E macros by our modelled CLB, we created a VHDL library of components with all modelled CLB functions. In fact, two versions of this library were created. In the first one (*proasic3e_nofault*), the macros have the same ports from the original library. They do not provide access to the transistor. Thus, with this library, it is not possible to inject the SETs. However, they were useful to verify the proper functionality of the modelled macros, by comparing the results with the original library. In the second one (*proasic3e_faulty*), each macro has an additional Faults port, that gives access to the 82 transistors of the CLB model. The output of all macros in the libraries was mapped to the F2 output, with the YL output left opened. To compare the structures, Listing 4.3 presents a version of the entity and architecture of the AND2 macro in the original library, without the VITAL modelling, while Listing 4.4 presents the entity and the architecture of the modelled AND2 macro, with the Faults port.

```
entity AND2 is
    port(
        A       : in    STD_ULOGIC;
        B       : in    STD_ULOGIC;
        Y       : out   STD_ULOGIC);
end AND2;

architecture VITAL_ACT of AND2 is
begin
        Y <= ( A  AND  B );
end VITAL_ACT;
```

LISTING 4.3 – Entity and architecture of the AND2 macro in the original library, without the VITAL timing modelling.

```
entity AND2 is
    port(
        A           : in    std_ulogic;
        B           : in    std_ulogic;
        Y           : out   std_ulogic;
        Faults      : in    bit_vector( 81 downto 0 )
    );
end AND2;

architecture VersaTile_ST of AND2 is
begin
    CLB: VersaTile_82
    generic map(
        Config      => "11001000001110000010001100010101"
        )
    port map(
        X3Data      => A,
        X2Clk       => B,
        X1Enable    => 'Z',
        XcClr       => 'Z',
        F2          => Y,
        YL          => open,
        Faults      => Faults
        );
end VersaTile_ST;
```

LISTING 4.4 – Entity and architecture of the AND2 macro using the VersaTile model at the transistor level. The Faults port enable the access to inject the SETs in the transistors. Other ports are the same from the original macro.

4.5 Estimation of the SET Susceptibilities

We estimated the SET susceptibility of every macro contained in our version of the *proasic3e* library. The estimation process is based on VHDL simulations with SET injections at the transistor level.

In this process, we injected the same number of SETs in each transistor, adopting a uniform distribution of SETs. In other words, we assume that all transistors have the same cross-section. It is a simplification, since, for example, F2 and YL signals requires different transistor characteristics. However, it is reasonable for this study, since it is consistently applied through all the work (for both CLB and circuit analysis).

Another simplification is that the transistors are treated separately as if they are part of a discrete circuit. As discussed in Buard and Anghel (2011), for the integrated circuits, the cross-section (sensitive zone) depends on the circuit layout. We did not consider the layout effect. However, for the same rationale presented above, it is reasonable for this study.

Regarding the SET pulse widths, there are some published radiation-test results for the selected technology. In Rezgui *et al.* (2007), the authors reported that most of the observed SET pulse widths are in the range from 1 ns to 4 ns. In this work, we performed all SET injections considering pulse widths of 1 ns.

The estimation of the SET susceptibility of every macro takes into account each input pattern independently. Except for the Gate/Clock signal, the input values are kept static during the SET injection. In this way, every combinational macro of n inputs is submitted to 2^n estimation processes, one for each input pattern. On the other hand, every sequential macro of n inputs is submitted to 2^{n-1} estimation process, since the Gate/Clock is not kept static.

The frequency of the Gate/Clock signal, for the estimation processes of every sequential macro, was 40 MHz, with a duty cycle of 50 %. The SET susceptibility of the sequential macros should be affected by the Gate/Clock frequency and the SET pulse width. However, since this study do not deal with this timing effect, all reference estimations adopts these values. We choose the frequency of 40 MHz because it is the value available on the ProASIC3E starter kit (MICROSEMI, 2012b), that was used in the emulation-based approach proposed in Armelin *et al.* (2018a), and discussed in Chapter 6.

4.6 Results

The estimated SET susceptibilities for the VersaTile functions are presented in Appendix B, divided in three tables: Table B.1, for the combinational functions; Table B.2, for the flip-flops; and Table B.4, for the latches. For each macro, they present the SET

susceptibilities for each input pattern. Additionally, they also present the average SET susceptibility for each macro, considering the average of the values for each input pattern.

The values of these tables are the reference database (`suscep.dat`) of SET susceptibilities required by the proposed strategy. The average values compose the reference database for the simplified version of the strategy, while the values parameterised by the input patterns compose the reference database for the complete one.

Taking into account the input patterns, the SET susceptibilities of the combinational functions are in the range of ∼5–30%. For the average values, that considers only the internal circuitry influence, they are in the range of ∼10–27%. For the flip-flops, considering the input patterns resulted in SET susceptibilities in the range of ∼5–21%, while the average values are in the range of ∼10–17%. For the latches, considering the input patterns resulted in SET susceptibilities in the range of ∼5–18%, while the average values are in the range of ∼9–15%.

Additionally, a percentage of the SEs observed in the flip-flops and the latches is composed by SEUs, not only SETs. For the flip-flops, considering the input patterns, the SEU percentage is in the range of ∼0–76%, while the average values are in the range of ∼33–68%. Finally, for the latches, considering the input patterns, the SEU percentage is in the range of ∼0–50%, while the average values are in the range of ∼21–49%. Table B.3 presents the percentage value of observed SEs that are SEUs, for the flip-flops, while Table B.5 presents the same information for the latches.

To illustrate these differences in SET susceptibilities, Fig. 4.5 presents the obtained values for the 2-input combinational macros, Fig. 4.6 presents the values for some 3-input flip-flop macros, and Fig. 4.7 presents the values for some 3-input latch macros. Finally, to illustrate these differences in the percentage of generated SEUs, Fig. 4.8 presents the obtained values for the same 3-input flip-flop macros, while Fig. 4.9 presents the values for the same 3-input latch macros.

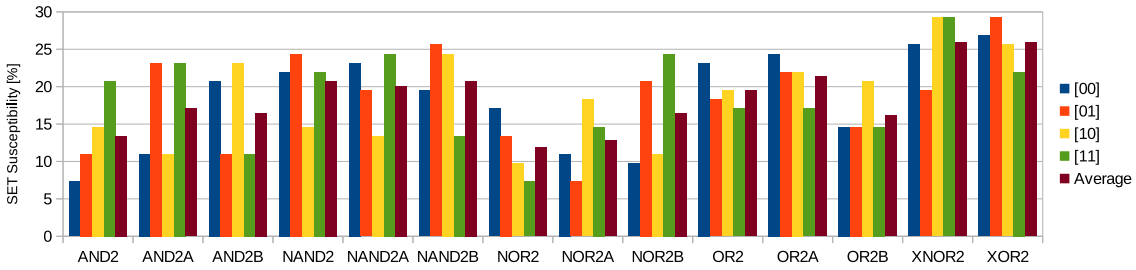

FIGURE 4.5 – SET susceptibilities of the 2-input combinational functions. For each macro, it presents the susceptibility for each input pattern and their average value. The input patterns refer to inputs A and B [A B].

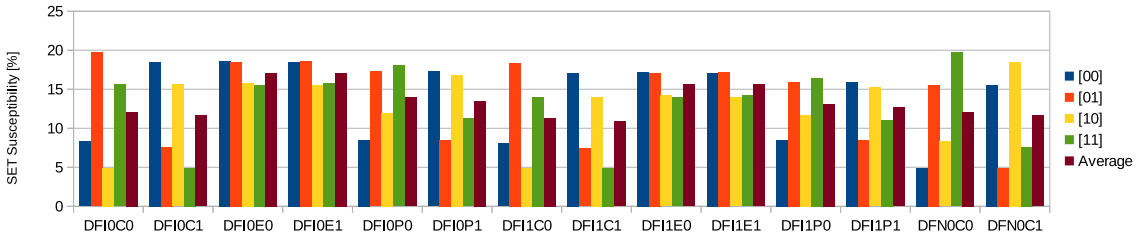

FIGURE 4.6 – SET susceptibilities of some 3-input flip-flop macros. For each macro, it presents the susceptibility for each input pattern and their average value. The input patterns refer to input Data and the Other input used [Data Other], that may be Clear (C), Preset (P) or Enable (E), identified in the penultimate character of the macro name.

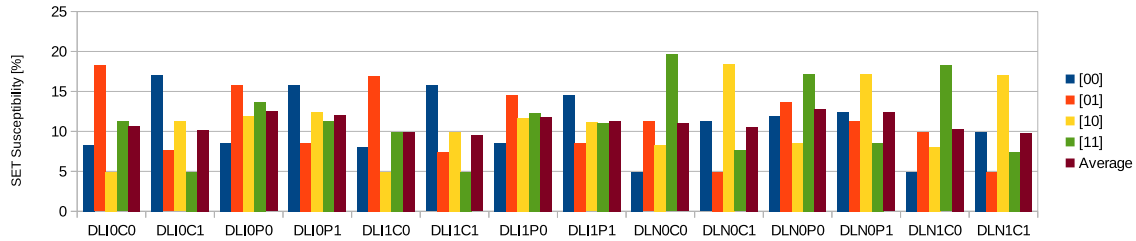

FIGURE 4.7 – SET susceptibilities of some 3-input latch macros. For each macro, it presents the susceptibility for each input pattern and their average value. The input patterns refer to input Data and the Other input used [Data Other], that may be Clear (C) or Preset (P), identified in the penultimate character of the macro name.

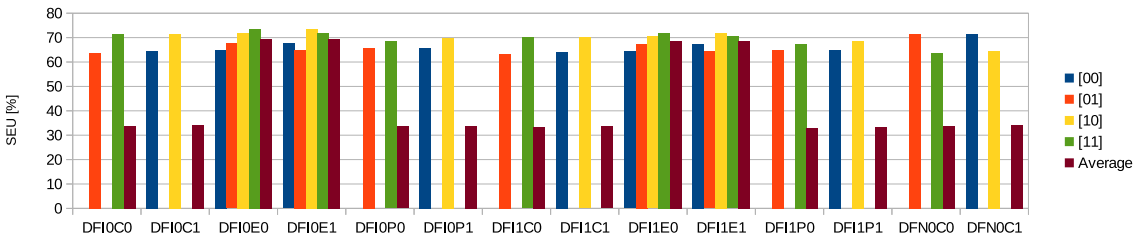

FIGURE 4.8 – Percentage of generated SEUs for some 3-input flip-flop macros. For each macro, it presents the percentage for each input pattern and the average value. The input patterns refer to input Data and the Other input used [Data Other], that may be Clear (C), Preset (P) or Enable (E), identified in the penultimate character of the macro name.

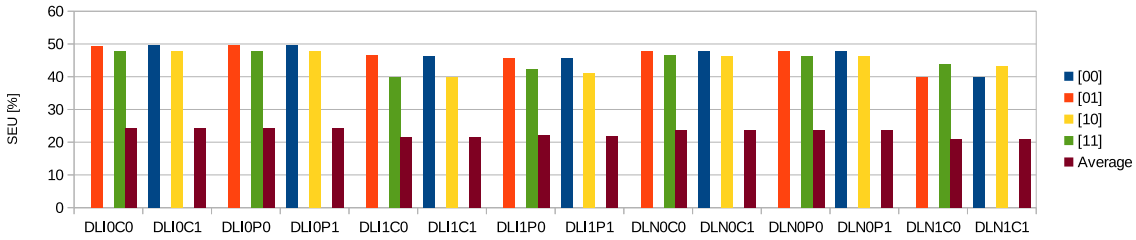

FIGURE 4.9 – Percentage of generated SEUs for some 3-input latch macros. For each macro, it presents the percentage for each input pattern and the average value. The input patterns refer to input Data and the Other input used [Data Other], that may be Clear (C) or Preset (P), identified in the penultimate character of the macro name.

The specific susceptibility, in any case, depends on the amount of transistors effectively used by the specific logic function and on the logical masking produced by the inputs. Thus, it is directly related to the complexity of the resulting circuit used to implement the logic function.

For example, to compare the quantity of used transistors, the AND3 macro illustrated in Fig. 4.3 has 26 transistors for the logic gates (7 Inverters, Inv2, Inv3, Inv4, Inv5, Inv7, Inv8 and Inv9, and 2 MUX, Mux1 and Mux2), and 12 transistors for the configuration switches. On the other hand, the DFN1C0 macro illustrated in Fig. 4.4 has 38 transistors for the logic gates (8 Inverters, Inv0, Inv1, Inv2, Inv4, Inv5, Inv7, Inv8 and Inv9, 1 NOR, and 3 MUX, Mux0, Mux1 and Mux2), and 12 transistors for the configuration switches. The average SET susceptibility of the AND3 is 9.909%, while of the DFN1C0 is 11.293%.

Regarding the logical masking, considering the AND3 macro illustrated in Fig. 4.3, when the input X1 has logic value ‘0’, all the SETs produced before Mux2 (Mux1, Inv2, Inv3, Inv7 and Inv8) are masked. Contrarily, when the input X2 has logic value ‘0’, only the SETs produced before Mux1 (Inv2 and Inv3) are masked. The SET susceptibility of AND3 for the input pattern [111] is 23.171%, while for the input pattern [110] (X1 = ‘0’, and the others ‘1’) is 10.976%.

5 Evaluation of the Proposed Strategy

In this chapter we present the evaluation of the proposed strategy using a set of benchmark circuits. To have a reference for comparison, we estimated the SEV at the transistor level, using the model described in Section 4.2. Then we performed three estimations at the CLB level:

1. ignoring the SET susceptibility of each CLB, as the logic-simulation and emulation-based approaches usually do;
2. using the simplified strategy, that considers the effects of the internal circuitry of the logic gates on the SET susceptibilities; and
3. using the complete strategy, that additionally considers the effects of the operational scenarios on the SET susceptibilities.

The analysed benchmark circuits are described in Section 5.1, while the evaluated operational scenarios are discussed in Section 5.2.

All SEV estimations used the structure of the simulation framework presented in Chapter 3, with some specificities for each case. For instance, for each DUT, the Driver and the SE Monitor were the same in the four categories of SEV estimations, while the other blocks have some adaptions.

The four categories of SEV estimation processes are detailed in Sections 5.3 to 5.6,

that present the specificities of each case and the resulting data from the SEV estimation of the analysed benchmarks.

Finally, in Section 5.7, we compare the estimated SEVs in terms of estimation error and estimation time.

5.1 Analysed Benchmarks

To evaluate the proposed strategy, we selected 38 combinational benchmark circuits from the LGSynth91 list (YANG, 1991) to be used as DUTs. We limited the analysis to the benchmarks that require less than 100 CLBs, due to the simulation time at the transistor level. The choice for the LGSynth91 list was arbitrary, but this selection is not important for this analysis since the proposed approach is applicable to any circuit. The selected benchmark circuits have a good range of complexity:

- the number of inputs varies from 3 to 47;
- the number of outputs varies from 1 to 36; and
- the number of CLBs, for the analysed technology, varies from 3 to 95.

The details for each of the selected benchmark circuits are presented in Table 5.1.

TABLE 5.1 – Combinational benchmark circuits. For each benchmark, this table presents the number of inputs, outputs, and required CLBs of the target technology.

Benchmark	**Inputs**	**Outputs**	**CLBs**
b1	3	4	3
cm82a	5	3	4
majority	5	1	5
tcon	17	16	8
cm152a	11	1	10
parity	16	1	10
cm42a	4	10	13
cm138a	6	8	13
t481	16	1	13
cmb	16	4	17
cm163a	16	5	19
9symml	9	1	21
cm85a	11	3	21
cm162a	14	5	21
cordic	23	2	22
decod	5	16	22
i1	25	16	22
z4ml	7	4	22
cu	14	11	24
pm1	16	13	24
pcle	19	9	27
cc	21	20	28
sct	19	15	28
x2	10	7	28
mux	21	1	29
f51m	8	8	32
frg1	28	3	35
lal	26	19	36
c8	28	18	44
unreg	36	16	48
pcler8	27	17	51
count	35	16	55
b9	41	21	59
comp	32	3	62
term1	34	10	68
cht	47	36	82
ttt2	24	21	85
my_adder	33	17	95

5.2 Operational Scenarios

The selected benchmark circuits do not have detailed documentation about their functionality. For this reason, in the Driver, we chose to use a pseudo-random sequence of test vectors to define the operational scenarios. Instead of using the RandomPkg from the OSVVM, we decided to use Linear-Feedback Shift Registers (LFSR) to generate the pseudo-random sequence. Contrarily to the RandomPkg functions, the LFSR structures are synthesisable, which is interesting for an emulation-based SEV estimation approach, as proposed in Armelin *et al.* (2018a), and discussed in Chapter 6.

We implemented an LFSR VHDL component with a configurable data width, n, ranging from 2 to 31 bits, according to the polynomials presented in MAXIM INTEGRATED (2010). In addition to the data width parameter, one can configure its initial value through the seed parameter.

For each of the analysed DUTs, the respective Driver is a set of these LFSRs, to fit the input data width of the DUT. The number of inputs of the DUT is divided by 31. The quotient gives the number of 31-bits LFSRs, while the remainder gives the width of the last LFSR. If the last LFSR would be one-bit wide, the previous one is reduced to 30 bits, and the last is incremented to 2 bits. Finally, if more than one LFSR is needed, each of them is set to a different seed.

All operational scenarios span for 1,000 cycles, i.e., they consist of a sequence of 1,000 pseudo-random values. If the required LFSR data width is 9-bit or less, the LFSR sequence is restarted and repeated until achieving the 1,000 cycles. In other cases, the sequence is truncated at 1,000 cycles.

5.3 SEV Estimation at the Transistor Level

The SEV estimation at the transistor level is the reference for comparison of the proposed strategy with the approach that ignores the SET susceptibilities. This approach is reasonable since the CLB model at the transistor level used for these SEV estimations

is the same adopted for estimating the SET susceptibilities.

In these estimations, both Golden DUT and Faulty DUT uses the *proasic3e_faulty* library, described in Section 4.4. While the Faults port of the CLBs of the Faulty DUT are controlled by the SET injector, the Faults port of the CLBs of the Golden DUT are fixed in '0', without SET injection.

The SET injector is the block that has more changes among the four SEV estimation categories, although, for consistency, in all cases, we adopted an equivalent to 1,000 injected SETs per transistor. To exemplify the particularities of the SET injector, we describe the test data for the *majority* benchmark. This benchmark requires 5 CLBs (2 NOR3C, 1 OR3, and 2 OR2), identified as $\{NOR3C_1, NOR3C_2, OR3_1, OR2_1, OR2_2\}$.

The quantity of SETs that needs to be injected ($N_{SET_{Total}}$) is obtained with (3.5), discussed in Section 3.6, and reproduced below.

$$N_{SET_{Total}} = N_{CLB} \cdot N_{Transistor_{CLB}} \cdot N_{SET_{Transistor}} \cdot \chi_{CLB_{average}} \tag{5.1}$$

The quantity of CLBs (N_{CLB}) is 5, the quantity of transistors per CLB ($N_{Transistor_{CLB}}$) is 82, and the quantity of SETs per transistor ($N_{SET_{Transistor}}$) is 1,000. At the transistor level, the average susceptibility of the CLBs ($\chi_{CLB_{average}}$) has no meaning, since all SET at this level need to be injected. The resulting $N_{SET_{Total}}$ is 410,000.

Table 5.2 presents the SEV estimation data at the transistor level. For each benchmark, this table presents the number of injected SETs, and observed SEs, the resulting SEV, that is the fraction of injected SETs that produces a SE, and the simulation time. The estimated SEV ranges from 2.23% (*comp* benchmark) to 24.33% (*parity* benchmark). There is no correlation between the estimated SEV and the complexity of the benchmark. As explained in Chapter 4, the susceptibilities depend on the number of transistors used by each CLB (depending on the used logic functions, a circuit with less CLBs may use more transistors than a circuit with more CLBs), and on the logical masking due to the operational scenario. However, the estimation time increases with the benchmark size.

TABLE 5.2 – SEV estimation at the transistor level. For each benchmark, this table presents the number of injected SETs, the observed SEs, the resulting SEV, and the simulation time.

Benchmark	**Injected SETs**	**Observed SEs**	**SEV [%]**	**Sim. Time [s]**
b1	246,000	56,274	22.88	16
cm82a	328,000	64,750	19.74	20
majority	410,000	29,154	7.11	32
tcon	656,000	102,504	15.63	46
cm152a	820,000	55,739	6.80	86
parity	820,000	199,540	24.33	141
cm42a	1,066,000	153,992	14.45	113
cm138a	1,066,000	121,281	11.38	106
t481	1,066,000	96,625	9.06	228
cmb	1,394,000	48,211	3.46	188
cm163a	1,558,000	113,452	7.28	242
9symml	1,722,000	85,420	4.96	406
cm85a	1,722,000	111,091	6.45	423
cm162a	1,722,000	136,472	7.93	338
cordic	1,804,000	95,024	5.27	545
decod	1,804,000	162,133	8.99	367
i1	1,804,000	155,015	8.59	336
z4ml	1,804,000	191,826	10.63	434
cu	1,968,000	175,629	8.92	387
pm1	1,968,000	258,680	13.14	492
pcle	2,214,000	242,821	10.97	677
cc	2,296,000	291,514	12.70	844
sct	2,296,000	243,121	10.59	690
x2	2,296,000	184,694	8.04	826
mux	2,378,000	58,684	2.47	845
f51m	2,624,000	319,996	12.19	1,455
frg1	2,870,000	147,293	5.13	1,640
lal	2,952,000	294,254	9.97	1,769
c8	3,608,000	330,306	9.15	2,839
unreg	3,936,000	358,056	9.10	4,425
pcler8	4,182,000	488,513	11.68	3,499
count	4,510,000	459,377	10.19	4,671
b9	4,838,000	510,557	10.55	6,484
comp	5,084,000	113,509	2.23	8,285
term1	5,576,000	260,615	4.67	10,036
cht	6,724,000	621,771	9.25	15,792
ttt2	6,970,000	601,227	8.63	16,979
my_adder	7,790,000	1,050,020	13.48	28,482

5.4 SEV Estimation adopting Uniform Distribution

The SEV estimation using the uniform distribution of SETs was performed to mimic the approach that ignores the SET susceptibilities.

In these estimations, both Golden DUT and Faulty DUT uses the original library of the ProASIC3E family. The Fault and Enable ports of the saboteurs added to the Faulty DUT are controlled by the SET injector, that distributes the SETs as described below.

Since there is no SET susceptibility in this case, we needed to adopt a value for the average susceptibility, $\chi_{CLB_{average}}$, for consistency in the comparison. We adopted the average value of all the combinational functions in our library. This average value, presented in Table B.1, is $\chi_{CLB_{average}} = 0.17182$.

Again, the quantity of SETs that needs to be injected ($N_{SET_{Total}}$) is obtained with (3.5). The quantity of CLBs is 5, the quantity of transistors per CLB is 82, the quantity of SETs per transistor is 1,000, and the average susceptibility of the CLBs is 0.17182. The resulting $N_{SET_{Total}}$ is 70,446.

$$N_{SET_{Total}} = 5 \cdot 82 \cdot 1,000 \cdot 0.17182 = 70,446 \tag{5.2}$$

Table 5.3 presents the resulting SEV estimation data using uniform distribution. For each benchmark, this table presents the equivalent number of injected SETs at the transistor level, the effective number of injected SETs at the gate level, the number of observed SEs, the resulting SEV, that is the fraction of injected SETs at the transistor level that produces a SE, and the simulation time. The estimated SEV ranges from 2.57% (*comp* benchmark) to 17.18% (*b1*, *cm82a*, *tcon*, and *parity* benchmarks). Again, there is no correlation between the estimated SEV and the complexity of the benchmark. However, as the *b1*, *cm82a*, *tcon*, and *parity* benchmarks are 'transparent', since they do not filter any injected SET, their estimated SEV is equivalent to the average SEV that was adopted in the uniform distribution. Finally, the estimation times are shorter than at the transistor level, and their increase with circuit size is less accentuated.

TABLE 5.3 – SEV estimation adopting uniform distribution. For each benchmark, this table presents the equivalent injected SETs at the transistor level, the effective injected SETs at the CLB level, the observed SEs, the resulting SEV, and the simulation time.

Benchmark	**Inj. Trans.**	**Inj. CLB**	**Obs. SEs**	**SEV [%]**	**Sim. Time [s]**
b1	246,000	42,267	42,267	17.18	4.3
cm82a	328,000	56,356	56,356	17.18	4.6
majority	410,000	70,446	27,743	6.77	4.6
tcon	656,000	112,713	112,713	17.18	5.0
cm152a	820,000	140,892	68,282	8.33	4.7
parity	820,000	140,892	140,892	17.18	5.1
cm42a	1,066,000	183,160	176,536	16.56	5.0
cm138a	1,066,000	183,160	133,799	12.55	5.0
t481	1,066,000	183,160	94,802	8.89	5.2
cmb	1,394,000	239,517	58,979	4.23	5.2
cm163a	1,558,000	267,695	128,205	8.23	5.5
9symml	1,722,000	295,874	94,073	5.46	5.9
cm85a	1,722,000	295,874	115,786	6.72	5.6
cm162a	1,722,000	295,874	150,604	8.75	5.9
cordic	1,804,000	309,963	119,779	6.64	5.9
decod	1,804,000	309,963	282,627	15.67	5.6
i1	1,804,000	309,963	196,462	10.89	6.0
z4ml	1,804,000	309,963	200,437	11.11	6.0
cu	1,968,000	338,141	240,885	12.24	6.1
pm1	1,968,000	338,141	267,277	13.58	6.0
pcle	2,214,000	380,409	274,071	12.38	6.2
cc	2,296,000	394,498	306,625	13.35	6.9
sct	2,296,000	394,498	275,822	12.01	6.7
x2	2,296,000	394,498	191,141	8.32	6.3
mux	2,378,000	408,587	64,726	2.72	6.6
f51m	2,624,000	450,855	282,514	10.77	7.3
frg1	2,870,000	493,123	169,121	5.89	7.4
lal	2,952,000	507,212	318,219	10.78	7.7
c8	3,608,000	619,926	379,865	10.53	9.3
unreg	3,936,000	676,283	393,843	10.01	9.8
pcler8	4,182,000	718,551	537,869	12.86	9.4
count	4,510,000	774,908	418,720	9.28	11.2
b9	4,838,000	831,265	525,984	10.87	13.3
comp	5,084,000	873,532	130,455	2.57	12.2
term1	5,576,000	958,068	314,802	5.65	13.5
cht	6,724,000	1,155,317	800,148	11.90	16.8
ttt2	6,970,000	1,197,585	683,075	9.80	19.5
my_adder	7,790,000	1,338,477	925,060	11.87	29.0

5.5 SEV Estimation adopting the Simplified Strategy

In this section, we present the SEV estimation with the simplified version of the proposed strategy. Again, both Golden DUT and Faulty DUT uses the original library of the ProASIC3E family. The Fault and Enable ports of the saboteurs added to the Faulty DUT are controlled by the SET injector, that distributes the SETs as described below.

The average SET susceptibilities of the 5 macros required by the *majority* benchmark ($\{NOR3C_1,\ NOR3C_2,\ OR3_1,\ OR2_1,\ OR2_2\}$) are {0.13567, 0.13567, 0.16768, 0.19512, 0.19512} (see Table B.1), that leads to an average SET susceptibility of the circuit of 0.16585.

The quantity of SETs that needs to be injected ($N_{SET_{Total}}$) is obtained, once again, with (3.5). The quantity of CLBs is 5, the quantity of transistors per CLB is 82, the quantity of SETs per transistor is 1,000, and the average susceptibility of the CLBs is 0.16585. The resulting $N_{SET_{Total}}$ is 67,998.

$$N_{SET_{Total}} = 5 \cdot 82 \cdot 1,000 \cdot 0.16585 = 67,998 \tag{5.3}$$

Table 5.4 presents the SEV estimation data for the simplified strategy. For each benchmark, this table presents the equivalent number of injected SETs at the transistor level, the effective number of injected SETs at the gate level, the number of observed SEs, the resulting SEV, and the simulation time. The estimated SEV ranges from 2.22% (*comp* benchmark) to 24.31% (*parity* benchmark). Once again, there is no correlation between the estimated SEV and complexity of the benchmark. Finally, the required estimation times are slightly greater than using uniform distribution.

TABLE 5.4 – SEV estimation adopting the simplified strategy. For each benchmark, this table presents the equivalent injected SETs at the transistor level, the effective injected SETs at the CLB level, the observed SEs, the resulting SEV, and the simulation time.

Benchmark	**Inj. Trans.**	**Inj. CLB**	**Obs. SEs**	**SEV [%]**	**Sim. Time [s]**
b1	246,000	56,501	56,501	22.97	4.7
cm82a	328,000	64,248	64,248	19.59	4.8
majority	410,000	67,998	25,442	6.21	5.1
tcon	656,000	102,998	102,998	15.70	5.2
cm152a	820,000	109,740	50,780	6.19	5.0
parity	820,000	199,374	199,374	24.31	5.4
cm42a	1,066,000	167,244	162,654	15.26	6.0
cm138a	1,066,000	189,246	133,016	12.48	5.9
t481	1,066,000	184,620	93,403	8.76	5.6
cmb	1,394,000	195,369	60,189	4.32	5.8
cm163a	1,558,000	213,243	110,202	7.07	6.0
9symml	1,722,000	261,744	83,462	4.85	6.7
cm85a	1,722,000	301,504	115,181	6.69	7.5
cm162a	1,722,000	244,747	131,212	7.62	6.6
cordic	1,804,000	285,879	93,648	5.19	6.8
decod	1,804,000	220,250	198,810	11.02	5.8
i1	1,804,000	244,874	152,493	8.45	6.7
z4ml	1,804,000	266,504	180,538	10.01	6.8
cu	1,968,000	270,481	189,681	9.64	6.6
pm1	1,968,000	336,862	273,931	13.92	7.5
pcle	2,214,000	349,258	257,181	11.62	7.6
cc	2,296,000	389,378	299,260	13.03	9.0
sct	2,296,000	374,248	258,943	11.28	8.0
x2	2,296,000	351,127	183,796	8.01	7.4
mux	2,378,000	306,119	53,283	2.24	7.0
f51m	2,624,000	473,369	322,889	12.31	10.3
frg1	2,870,000	442,123	151,430	5.28	9.0
lal	2,952,000	524,629	310,205	10.51	9.3
c8	3,608,000	544,338	339,958	9.42	10.1
unreg	3,936,000	643,496	366,789	9.32	11.5
pcler8	4,182,000	654,984	494,574	11.83	11.2
count	4,510,000	824,608	465,387	10.32	13.7
b9	4,838,000	777,127	511,516	10.57	15.3
comp	5,084,000	798,493	113,111	2.22	14.1
term1	5,576,000	877,495	263,336	4.72	16.1
cht	6,724,000	867,396	594,967	8.85	17.0
ttt2	6,970,000	1,074,216	586,786	8.42	21.4
my_adder	7,790,000	1,394,721	1,011,741	12.99	38.4

5.6 SEV Estimation adopting the Complete Strategy

In this section, we present the SEV estimation with the complete version of the proposed strategy. As in all cases at the gate level, both Golden DUT and Faulty DUT uses the original library of the ProASIC3E family. The Fault and Enable ports of the saboteurs added to the Faulty DUT are controlled by the SET injector, that distributes the SETs as described below.

The specific SET susceptibilities of the 5 macros required by *majority* benchmark ($\{NOR3C_1$, $NOR3C_2$, $OR3_1$, $OR2_1$, $OR2_2\}$), for the adopted operational scenario, are {0.16206, 0.16203, 0.19674, 0.19391, 0.19395}, that leads to an average value of 0.18174.

The quantity of SETs that needs to be injected ($N_{SET_{Total}}$) is obtained, as in the other cases, with (3.5). The quantity of CLBs is 5, the quantity of transistors per CLB is 82, the quantity of SETs per transistor is 1,000, and the average susceptibility of the CLBs is 0.18174. The resulting $N_{SET_{Total}}$ is 74,513.

$$N_{SET_{Total}} = 5 \cdot 82 \cdot 1,000 \cdot 0.18174 = 74,513 \tag{5.4}$$

Table 5.5 presents the SEV estimation data for the complete strategy. For each benchmark, this table presents the equivalent number of injected SETs at the transistor level, the effective number of injected SETs at the gate level, the number of observed SEs, the resulting SEV, and the simulation time. The estimated SEV ranges from 2.23% (*comp* benchmark) to 24.33% (*parity* benchmark). As in the previous cases, there is no correlation between the estimated SEV and complexity of the benchmark. Finally, the required estimation times are similar to those obtained with the simplified strategy.

TABLE 5.5 – SEV estimation adopting the complete strategy. For each benchmark, this table presents the equivalent injected SETs at the transistor level, the effective injected SETs at the CLB level, the observed SEs, the resulting SEV, and the simulation time.

Benchmark	**Inj. Trans.**	**Inj. CLB**	**Obs. SEs**	**SEV [%]**	**Sim. Time [s]**
b1	246,000	56,274	56,274	22.88	4.6
cm82a	328,000	64,760	64,760	19.74	4.9
majority	410,000	74,513	29,484	7.19	5.3
tcon	656,000	102,532	102,532	15.63	5.0
cm152a	820,000	111,897	53,417	6.51	5.0
parity	820,000	199,538	199,538	24.33	5.8
cm42a	1,066,000	158,258	153,819	14.43	5.7
cm138a	1,066,000	177,254	121,104	11.36	5.3
t481	1,066,000	186,965	95,785	8.99	5.5
cmb	1,394,000	175,476	48,067	3.45	5.1
cm163a	1,558,000	207,915	110,635	7.10	7.2
9symml	1,722,000	261,434	85,332	4.96	5.9
cm85a	1,722,000	289,950	111,612	6.48	6.4
cm162a	1,722,000	247,141	135,535	7.87	6.2
cordic	1,804,000	280,918	95,421	5.29	6.2
decod	1,804,000	183,737	162,248	8.99	5.6
i1	1,804,000	243,233	156,465	8.67	6.3
z4ml	1,804,000	276,012	192,457	10.67	6.4
cu	1,968,000	251,943	175,590	8.92	6.5
pm1	1,968,000	316,769	256,468	13.03	6.6
pcle	2,214,000	311,332	240,347	10.86	6.7
cc	2,296,000	375,556	290,970	12.67	7.8
sct	2,296,000	352,803	243,129	10.59	7.7
x2	2,296,000	347,155	182,951	7.97	7.1
mux	2,378,000	325,643	55,647	2.34	7.1
f51m	2,624,000	469,827	319,432	12.17	9.0
frg1	2,870,000	431,963	148,369	5.17	8.4
lal	2,952,000	503,817	294,657	9.98	9.2
c8	3,608,000	523,665	331,218	9.18	10.2
unreg	3,936,000	635,152	358,096	9.10	11.3
pcler8	4,182,000	601,078	485,876	11.62	10.5
count	4,510,000	805,035	460,524	10.21	13.5
b9	4,838,000	779,595	513,199	10.61	16.4
comp	5,084,000	788,630	113,175	2.23	16.0
term1	5,576,000	850,395	260,610	4.67	16.9
cht	6,724,000	893,485	619,778	9.22	19.8
ttt2	6,970,000	1,062,785	593,077	8.51	23.3
my_adder	7,790,000	1,455,249	1,058,428	13.59	35.5

5.7 Comparative Analysis

In this section, we present a comparative analisys of the resulting SEV estimations at the gate level, taking the SEV estimation at the transistor level as our reference. Fig. 5.1 presents the absolute value of the errors in the SEV estimation process. For the analysed benchmarks, in the chosen operational scenarios, the average error of the SEV estimation obtained ignoring the SET susceptibilities was 15.27%. The adoption of the simplified version of the strategy leads to an average error of 4.70%. Lastly, the adoption of the complete version leads to an average error of 0.68%.

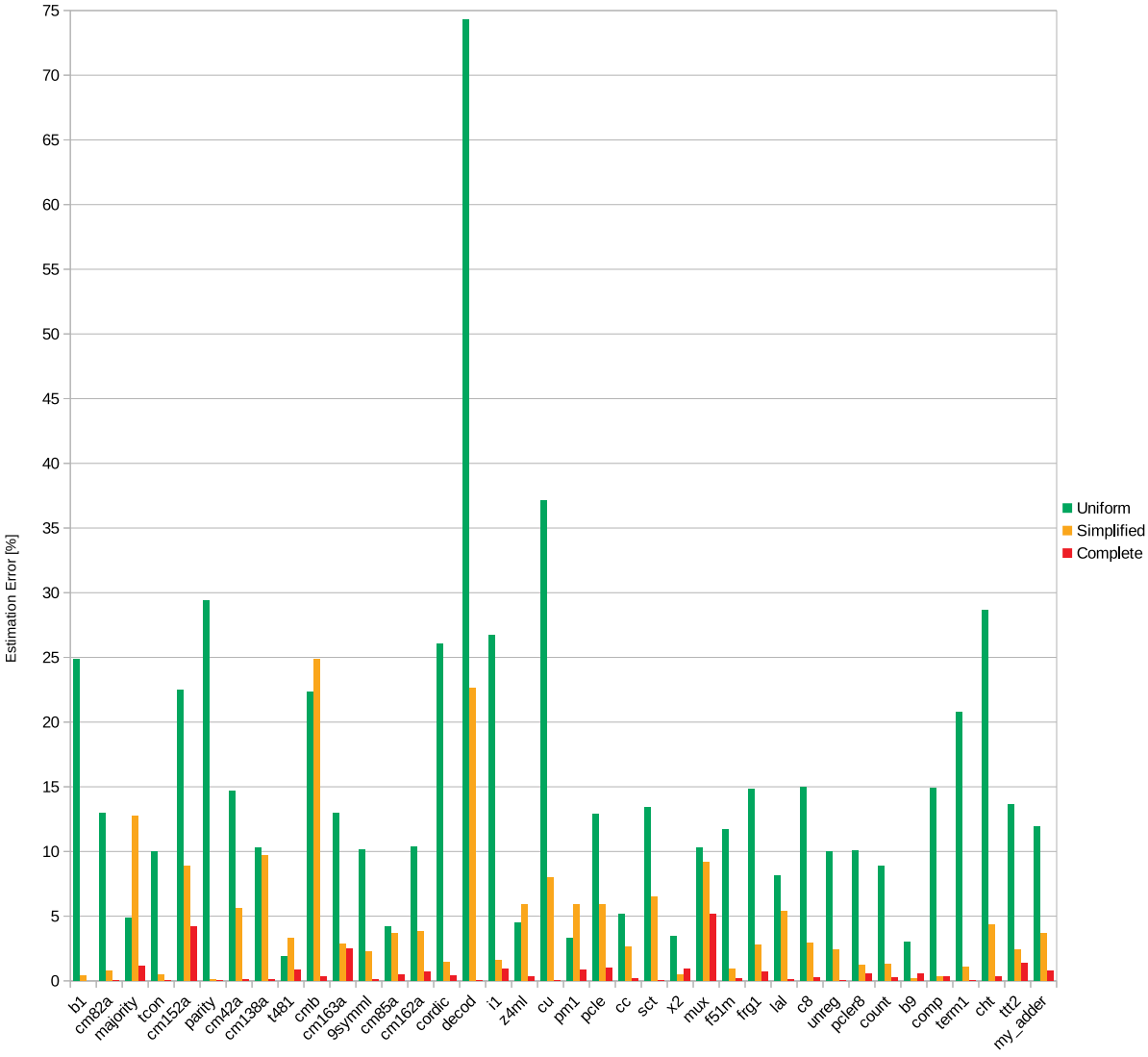

FIGURE 5.1 – Errors of the SEV estimation processes. For each benchmark circuit, it presents the absolute value of the error of the three SEV estimations at the CLB level in comparison to the reference estimation at the transistor level.

Complementary, ignoring the SET susceptibilities lead to an error of up to 74.32% (*decod* benchmark), while the best agreement was −1.89% of error (*t481* benchmark). Considering the simplified strategy had a maximum error of 24.85% (*cmb* benchmark) and best agreement of −0.08% (*parity* benchmark). Finally, considering the complete strategy leads to a maximum error of −5.17% (*mux* benchmark), and a minimum error of 0.00% (*b1*, *parity*, *sct*, and *term1* benchmarks).

The estimation errors are directly related to the average susceptibility used to determine the amount of SETs to be injected. The uniform distribution has higher estimation errors because, in this case, the average susceptibility considers all combinational functions of the library. For instance, for the *decod* benchmark, the average susceptibility of the used macros is 12.209%, while the average susceptibility of the library is 17.182%.

Fig. 5.2 compares the resulting simulation times. At the transistor level, there is an exponential growth of the time required to estimate the SEV as a function of the circuit size. At the CLB level, there is also a growth in the simulation time, as a function of the circuit size, but it is less accentuated. For the largest circuit analysed (*my_adder* benchmark), the simulation time for the estimation at the transistor level is three orders of magnitude higher than those at the CLB level. For the estimation methods at the CLB level, the simulation times are similar, but the required time for the methods with weighted distributions are higher than those with uniform distribution.

This simulation time increase is mainly due to the growth in signal-attribution operations, resulting from the inclusion of new entities. It is more accentuated at the transistor level because, for each CLB, it has additional 82 entities for the transistors.

Regarding the comparison between weighted and uniform distributions at the CLB level, the results show a small penalty in simulation time for using the weighted function.

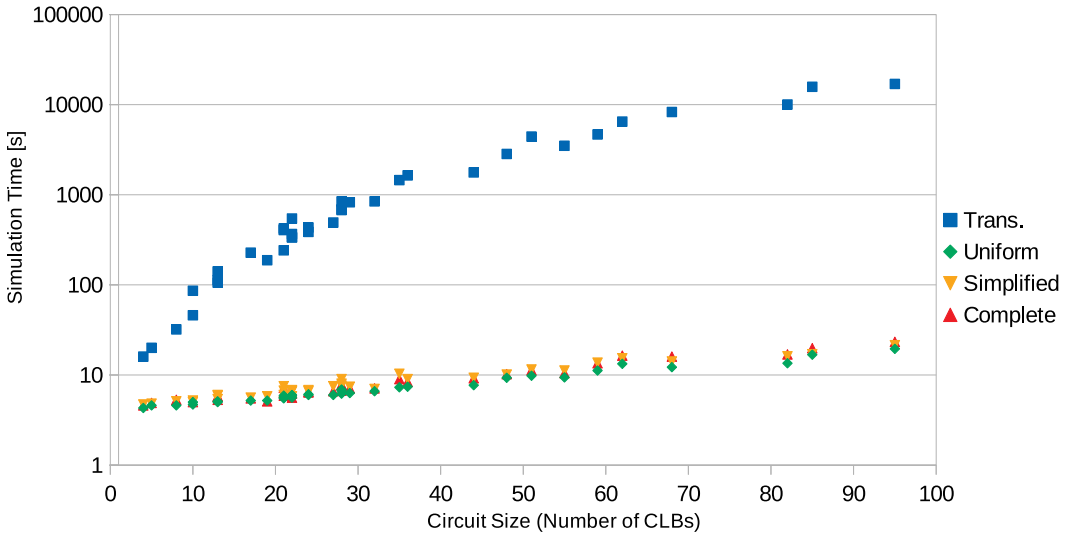

FIGURE 5.2 – Simulation time of the SEV estimation processes. It presents the simulation time for the four SEV estimations as a functions of the circuit size.

6 Using the Proposed Strategy with Emulation

The adoption of the proposed strategy with a logic simulation approach was demonstrated in the previous chapter, along with the evaluation of the simplified and complete versions of the strategy. Although the strategy is well suited in this simulation-based SEV estimation approach, it would be interesting to benefit from the advantage of rapidly estimation proportioned of the emulation. For this reason, in this chapter, we discuss the adoption of this strategy with some emulation approaches.

Both simplified and complete versions apply for emulation. For the simplified version (see Section 3.1), Step 4 changes from generating testbench for SET injection to generate the embedded emulation framework for SET injection, and Step 5 changes from simulating to executing the emulation framework. Similarly, for the complete version (see Section 3.2), the same changes apply for Step 6 and Step 7, respectively.

Section 6.1 brings a discussion related to this adoption with host-based emulation platforms, while Section 6.2 presents an analysis of the use of this strategy with autonomous platforms. Then, in Section 6.3, we propose an autonomous emulation platform, focusing on the weighted distribution of SETs.

6.1 Use with Host-Based Emulation Approach

As described in Section 2.3.4, the host-based emulation approaches are those in which an external host computer controls the fault-injection process. In this scenario, the weighted distribution that represents the distinct SET susceptibilities needs to be generated and controlled by the host. It is expected to be easier than generating and controlling the weighted distribution inside the emulation platform.

In this way, the works presented by Civera *et al.* (2001), Antoni *et al.* (2002), and Ebrahimi *et al.* (2014) could be adapted to use the proposed strategy impacting only the host software that controls the fault-injection process.

6.2 Use with Autonomous Emulation Approach

In opposition to the host-based approaches, in the autonomous emulation approaches, the control of the fault-injection process is inside the platform. In this scenario, although the weighted distribution can be configured at a previous step, it needs to be generated and controlled by the platform. It requires more resources, but the estimation can still be faster than using the host-based approach.

Thus, with this approach, the works presented by Lopez-Ongil *et al.* (2007), Naviner *et al.* (2011), and Entrena *et al.* (2012) would require significant changes in the emulation platform to be adapted to use the proposed strategy.

On the other hand, the emulation approach described by May and Stechele (2012) and May and Stechele (2013) already adopts a probability-aware fault-injection. They use a Pseudo-Random Number Generator (PRNG) and a comparator, for each circuit node, to decide when to inject a fault on the respective node. It grants concomitant and independent fault-injection, suitable for their intended application: stochastic computing and voltage over-scaling.

However, their approach is not suitable for SEE studies, in which either the analysed

SE occurs in a single node or the multiple effects are not independent. In this context, in the next section, we propose a different mechanism for the SET injection following a weighted distribution.

6.3 Proposed Emulation-Based Framework

This section describes the proposed autonomous, golden-circuit emulation platform with direct-access saboteurs. Fig. 6.1 illustrates the proposed platform, implemented in the target device A3PE1500-PQ208, using the ProASIC3/E Starter Kit (MICROSEMI, 2012b).

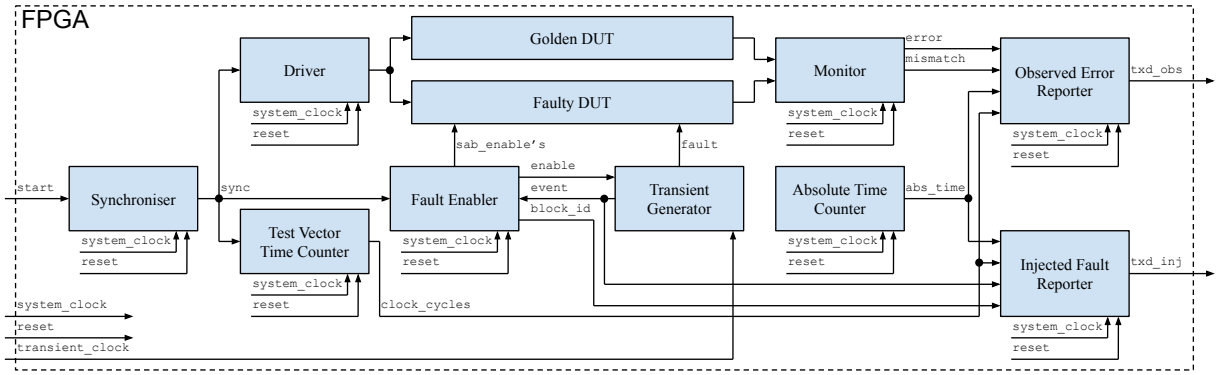

FIGURE 6.1 – Proposed emulation-based platform. It is based on the proposed framework (see Fig. 3.1), with the SET Injector decomposed in many sub-blocks.

The proposed emulation platform is based on the general structure proposed by the SEV estimation framework, in Chapter 3 (see Fig. 3.1). In this platform, the SET Injector is decomposed in other sub-blocks required for the proper operation of the platform. Each block is described below:

Golden DUT is the fault-free version of the DUT. In this case, the Golden DUT is the same circuit used as Faulty DUT, with the saboteurs. However, each saboteur gets permanently disabled and without receiving SETs. This approach leads both versions of the DUT to have the most possible similar timing characteristics.

Faulty DUT is the instrumented version of the DUT, with saboteurs, obtained with the

same process described in Section 3.3. However, in this case the Macro AX1 is used, instead of AX1C, due to the analysis presented in Appendix C.

Driver generates the test vector sequence of the operational scenario. For the complete version of the strategy, needs to apply the same scenario used to estimate the input patterns distribution.

Monitor compares the outputs from the Golden DUT and the Faulty DUT. In case of outputs mismatch, the Monitor triggers the Observed Error Reporter with the signal `mismatch`, passing the result of the comparison, `error`.

Synchroniser synchronise the Driver, the Fault Enabler and the Test Vector Time Counter, to assure that only one fault is injected for each execution of the operational scenario.

Fault Enabler controls the Transient Generator and the saboteurs of the Faulty DUT. For the first, it determines when the SET is injected, which can be any clock cycle of the operational scenario. For the second, it determines the target CLB. In both cases, the values are pseudo-random. The distribution of the place can be defined as uniform or weighted (see Section 6.4), while the distribution of the time is always uniform.

Transient Generator generates the transient pulse to be injected in the Faulty DUT, when enabled by the Fault Enabler. It also generates a flag indicating that a transient was issued. This block is described in details in Appendix C, that evaluates the use of such transients.

Test Vector Time Counter counts the clock cycles of the execution of the operational scenario. Its value is used to generate a *test vector timestamp* for the Reporters (signal `clock_cycles`).

Absolute Time Counter counts the clock cycles of the entire fault injection run. Its value is used to generate a *absolute timestamp* for the Reporters (signal `abs_time`), enabling the correlation between injected SETs and observed SEs.

Injected Fault Reporter sends a serial message containing the place where and the time when the SET was injected. The place is reported with the CLB ID (signal `block_id`), and the time, with the *test vector timestamp* (signal `clock_cycles`). The *absolute timestamp* is also reported (signal `abs_time`).

Observed Error Reporter sends a serial message containing the observed mismatch between outputs of the Golden DUT and the Faulty DUT, and the relative and absolute time when it was observed (*test vector timestamp* and *absolute timestamp*).

Besides this platform, implemented in the FPGA, embedded in a development kit, the framework requires a external clock source, for the `transient_clock`, two converters from LVTTL (Low-Voltage Transistor Transistor Logic) to USB (Universal Serial Bus), for the serial interfaces (signals `txd_inj` and `txd_obs`), and a computer to receive the serial communications.

The `system_clock`, `reset`, and `start` signals are driven by the development kit. The `system_clock` receives a clock signal of 40 MHz, while `reset` and `start` are connected to dedicate buttons.

The external clock source for `transient_clock` is set to 39.9 MHz, that is the closest frequency to `system_clock` that can be adjusted in the used clock source. The rationale for this choice is discussed in Appendix C.

Once the SEV estimation is initiated by the `start` signal, the Golden DUT and Faulty DUT starts executing the operational scenario continuously. For each operational scenario execution, one SET may be injected at any saboteur of the Faulty DUT, at any cycle of the operational scenario. Each injected SET generates a serial message with the content illustrated in Fig. 6.2. If a SE is detected, a serial message with the content illustrated in Fig. 6.3 is generated. This process continues until the platform is restarted of powered off. In the next section, the fault distribution scheme is described.

```
I2A@034BT00EA9FC532
```

FIGURE 6.2 – Illustration of the serial message reporting a SET injection. The first character, 'I', denotes it is a SET injection report, while the next two characters indicate, in hexadecimal, the CLB ID, in this case, 0x2A represents the CLB 42. The forth character, '@', is a delimiter, while the next four characters reports, in hexadecimal, the *test vector timestamp*, in this case, 0x034B represents the cycle 843. Finally, 'T' is another delimiter, while the last ten characters reports, in hexadecimal, the *absolute timestamp*, in this case, 0x00EA9FC532 represents the cycle 3936339250 since the start of the estimation process.

```
O10@012CT00EAA23417
```

FIGURE 6.3 – Illustration of the serial message reporting an observed SE. The message structure is similar to the message for injected SETs, with the first character, 'O', denoting it is an observed SE report, while the next two characters indicate, in hexadecimal, the mismatch value from the selected outputs of the circuit, in this case, 0x10. The forth character, '@', is a delimiter, while the next four characters reports, in hexadecimal, the *test vector timestamp* in which the SE was observed, in this case, 0x012C represents the cycle 300. Finally, 'T' is another delimiter, while the last ten characters reports, in hexadecimal, the *absolute timestamp* in which the SE was observed, in this case, 0x00EAA23417 represents the cycle 3936498711 since the start of the estimation process.

6.4 SET Distribution Scheme

The emulation platform was implemented in three versions, with the same SET distributions adopted with the simulation framework, at the gate level:

1. ignoring the SET susceptibility of each CLB, as the logic-simulation and emulation-based approaches usually do;
2. using the simplified strategy, that considers the effects of the internal circuitry of the logic gates on the SET susceptibilities; and
3. using the complete strategy, that additionally considers the effects of the operational scenarios on the SET susceptibilities.

The fault-distribution is implemented using a PRNG, to pseudo-randomly distribute the SETs, and a decoder to correlate the pseudo-random numbers with the associated saboteur. For a uniform distribution, each value generated by the PRNG is used to enable one saboteur, in a one-to-one decoding process, while for a weighted distribution, a range of values generated by the PRNG is used to enable one saboteur, in a range decoding process. Different weights are defined with different ranges.

The ranges are proportional to the SET susceptibility of each logic gate. Considering the *majority* benchmark, already used as an example in Chapter 5, this circuit requires five CLBs of the adopted technology: $\{NOR3C_1, NOR3C_2, OR3_1, OR2_1, OR2_2\}$. For the uniform distribution, a 3-bit PRNG is enough to generate a pseudo-random sequence of this five IDs.

However, considering the simplified strategy, the SET susceptibility of each of these macros is {0.13567, 0.13567, 0.16768, 0.19512, 0.19512}. With a 17-bit PRNG, it would be possible to define the following ranges for each macro {[1..13567]; [13568..27134]; [27135..43902]; [43903..63414]; [63415..82926]}, with the other values not leading to any SET injection. This ranges can be proportionally expanded or reduced, impacting the PRNG data width, the decoder complexity and the SET distribution accuracy. For the complete strategy, the same scheme is used.

Finally, a second PRNG determines the clock cycle of the operational scenario in which the SET will be injected, to guarantee a uniform SET distribution over time.

6.5 Evaluation of the Proposed Emulation Platform

In this section, we present and discuss the obtained results from the implementation of the proposed emulation platform for the SEV estimation of some benchmark circuits. First, we present the analysed circuits, then we report the resources consumption, followed by the analysis of the proposed scheme for the SET distribution, and the discussion of the estimated SEV.

6.5.1 Analysed Circuits

Differently from the simulation framework, the steps for generating the emulation platform are not fully automated by scripts. For this reason, we chose four benchmarks from those used for the evaluation of the strategy, and manually coded their emulation platforms, considering the uniform distribution and the two versions of the strategy. Two of then were selected from the ten smaller and two from the ten bigger benchmarks used previously. Table 6.1 presents the selected benchmarks, with the details reproduced from Table 5.1. These benchmarks cover a good range of complexity from the list used with the simulation framework: the number of inputs ranges from 11 to 32, the number of outputs from 1 to 18, and the required amount of CLBs from 10 to 62.

TABLE 6.1 – Benchmark circuits used in emulation. For each benchmark, this table presents the number of inputs, outputs, and required CLBs of the target technology.

Benchmark	**Inputs**	**Outputs**	**CLBs**
cm152a	11	1	10
cmb	16	4	17
c8	28	18	44
comp	32	3	62

6.5.2 Resources Consumption

In this section, we present the resources consumption of the emulation platform, for the four benchmarks, considering the three distinct SET distribution (uniform, and simplified and complete strategies). Tables 6.2, 6.3, 6.4, and 6.5 reports the resources for the benchmarks *cm152a*, *cmb*, *c8*, and *comp*, respectively. Each table brings the information for the three analysed SET distributions.

For all benchmarks, the unique functional block that has significant and consistent difference in the required resources is the Fault Enabler, that is the block that implements the SET distribution. The uniform distribution has the lowest consumption, while the complete strategy leads to the highest resources consumption.

TABLE 6.2 – Resources consumption of the emulation platform with the *cm152a* benchmark. For each functional block of the platform, it presents the required amount of CLBs, for the uniform distribution and the weighted distribution with the simplified and complete versions of the proposer strategy.

Block	**Uniform**	**Simplified**	**Complete**
Golden DUT	20	20	20
Faulty DUT	20	20	20
Driver	15	15	15
Monitor	5	5	5
Synchroniser	57	57	57
Fault Enabler	246	283	312
Transient Generator	3	3	3
Test Vector Time Counter	39	39	39
Absolute Time Counter	103	103	103
Injected Fault Reporter	454	459	462
Observed Error Reporter	863	884	859
Total	1825	1888	1895

TABLE 6.3 – Resources consumption of the emulation platform with the *cmb* benchmark. For each functional block of the platform, it presents the required amount of CLBs, for the uniform distribution and the weighted distribution with the simplified and complete versions of the proposer strategy.

Block	**Uniform**	**Simplified**	**Complete**
Golden DUT	34	34	34
Faulty DUT	34	34	34
Driver	21	21	21
Monitor	17	17	17
Synchroniser	57	57	57
Fault Enabler	257	323	369
Transient Generator	3	3	3
Test Vector Time Counter	39	39	39
Absolute Time Counter	103	103	103
Injected Fault Reporter	501	502	504
Observed Error Reporter	833	852	862
Total	1899	1985	2043

TABLE 6.4 – Resources consumption of the emulation platform with the *c8* benchmark. For each functional block of the platform, it presents the required amount of CLBs, for the uniform distribution and the weighted distribution with the simplified and complete versions of the proposer strategy.

Block	**Uniform**	**Simplified**	**Complete**
Golden DUT	88	88	88
Faulty DUT	88	88	88
Driver	39	39	39
Monitor	61	61	61
Synchroniser	57	56	56
Fault Enabler	296	431	549
Transient Generator	4	4	4
Test Vector Time Counter	39	39	39
Absolute Time Counter	103	103	103
Injected Fault Reporter	910	900	877
Observed Error Reporter	856	868	873
Total	2540	2677	2777

TABLE 6.5 – Resources consumption of the emulation platform with the *comp* benchmark. For each functional block of the platform, it presents the required amount of CLBs, for the uniform distribution and the weighted distribution with the simplified and complete versions of the proposer strategy.

Block	**Uniform**	**Simplified**	**Complete**
Golden DUT	124	124	124
Faulty DUT	124	124	124
Driver	44	44	44
Monitor	12	12	12
Synchroniser	57	57	57
Fault Enabler	325	483	660
Transient Generator	5	5	5
Test Vector Time Counter	39	39	39
Absolute Time Counter	103	103	103
Injected Fault Reporter	460	463	464
Observed Error Reporter	877	859	863
Total	2170	2313	2495

The Reporters have some fluctuations on the consumption, due the placing and routing tool, since the input parameters for these block, for each benchmark, are the same.

The Test Vector Time Counter and the Absolute Time Counter have the same size in all cases, independently of the benchmark and the type of distribution. Similarly, the Synchroniser has almost the same size in all cases, with a small reduction for two distributions with the *c8* benchmark. The Transient Generator presents a increasing size with the increase of the DUT size.

The size of the Driver, the Monitor, and, obviously, of two versions of the DUT, depends on the analysed DUT. However, for each benchmark, the resource consumption is the same, independent of the adopted SET distribution.

6.5.3 SET Distribution Analysis

The focus of the proposed platform is on its ability to generate and control an weighted distribution of SETs. To evaluate its ability, we verified two aspects, considering the uniform distribution and the weighted distribution for the complete strategy.

First, we check if there is any correlation between the injection time (Clock Cycle of the *test vector timestamp*) and the injection place (Logic Block ID). For this purpose, we plotted the reported injected SETs in a place and time graph, for each benchmark, for uniform and weighted distribution. We verified that the injected SETs are randomly distributed in the place and time graph over time.

To illustrate it, Fig. 6.4 and Fig. 6.5 bring a plot of the SET injection place and time, after 100, 1,000, and 10,000 injected SETs, for the uniform and weighted distribution on the *c8* benchmark, respectively. Similarly, Fig. 6.6 and Fig. 6.7 bring the same information for the *comp* benchmark.

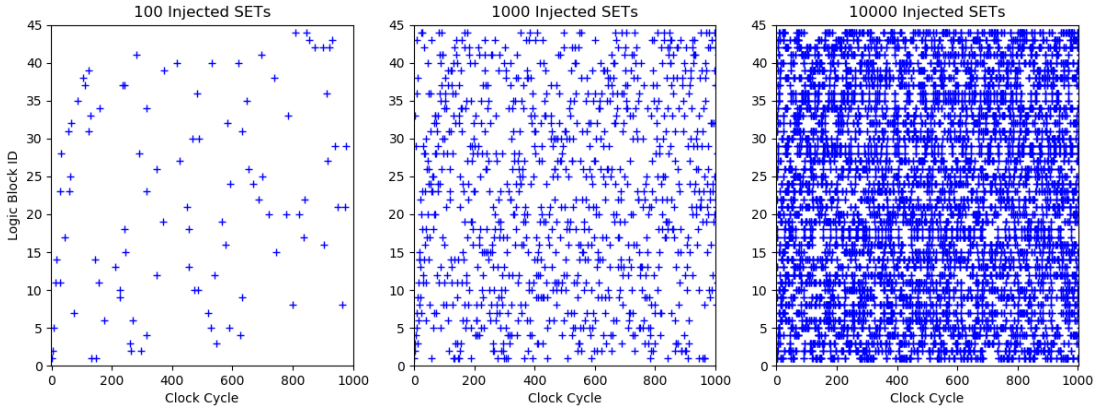

FIGURE 6.4 – Time (Clock cycle) vs. place (Logic Block ID) uniform SET distribution for the *c8* benchmark. The left panel presents the distribution after 100 injected SETs, while the center panel presents the distribution after 1,000, and the right panel after 10,000.

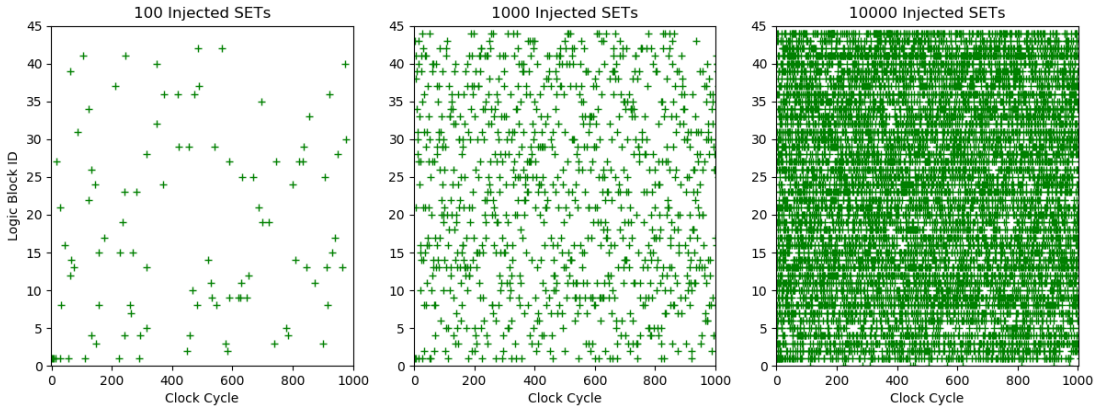

FIGURE 6.5 – Time (Clock cycle) vs. place (Logic Block ID) weighted SET distribution for the *c8* benchmark. The left panel presents the distribution after 100 injected SETs, while the centre panel presents the distribution after 1,000, and the right panel after 10,000.

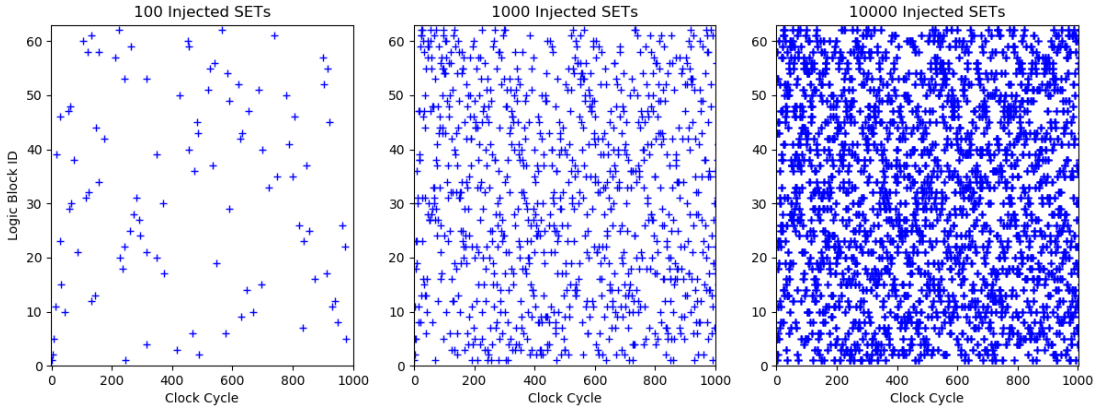

FIGURE 6.6 – Time (Clock cycle) vs. place (Logic Block ID) uniform SET distribution for the *comp* benchmark. The left panel presents the distribution after 100 injected SETs, while the centre panel presents the distribution after 1,000, and the right panel after 10,000.

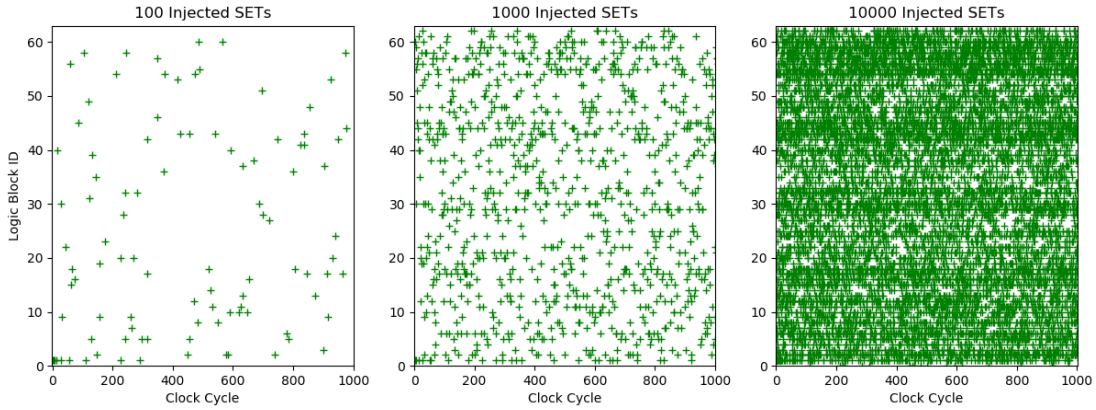

FIGURE 6.7 – Time (Clock cycle) vs. place (Logic Block ID) weighted SET distribution for the *comp* benchmark. The left panel presents the distribution after 100 injected SETs, while the centre panel presents the distribution after 1,000, and the right panel after 10,000.

The visual analysis of these plots indicates that there is no clear correlation in the weighted distribution. However, the uniform distribution on the *comp* benchmark seems to repeat time and places, leaving some gaps.

The second verification is about the number of SETs distributed for each Logic Block ID and Clock Cycle. For each benchmark, the expected average value of SETs per clock cycle is the total amount of injected SETs divided by the 1,000 clock cycles of the operational scenario. The critical parameter, in this case, is the standard deviation of the distribution.

Concerning the CLBs, for the uniform distribution, the expected average value of SETs per CLB is the total amount of injected SETs divide by the number of CLBs of each benchmark. Again, the critical parameter is the standard deviation of this distribution.

However, for the weighted distribution, each CLB has a specific expected value of SETs per CLB, that depends on its SET susceptibility. In this case, the essential parameters are the average error between the expected and injected SETs of each CLB and the standard deviation of these errors.

For the *cm152a* benchmark, with the uniform distribution, 498,078 SETs were injected. The average value of SETs per CLB was $\approx$ 49,808 (10 CLBs), with a standard deviation of 0.42, and SETs per clock cycle is $\approx$ 498, with a standard deviation of 17.59. With the

weighted distribution, 498,645 SETs were injected. The average value of SETs per clock cycle was ≈ 499, with a standard deviation of 18.26, while the average error between the expected an injected SETs per CLB was 54.91, with a standard deviation of 33.17.

For the *cmb* benchmark, with the uniform distribution, 496,189 SETs were injected. The uniform distribution resulted in ≈ 29,188 SETs per CLB (17 CLBs), with a standard deviation of 0.51, and ≈ 496 SETs per clock cycle, with a standard deviation of 17.75. With the weighted distribution, 498,645 SETs were injected, resulting in ≈ 499 SETs per clock cycle, with a standard deviation of 18.18, while the average error between the expected an injected SETs per CLB was 30.78, with a standard deviation of 20.06.

For the *c8* benchmark, with the uniform distribution, 496,000 SETs were injected. The uniform distribution resulted in ≈ 11,273 SETs per CLB (44 CLBs), with a standard deviation of 0.45, and ≈ 496 SETs per clock cycle, with a standard deviation of 18.27. With the weighted distribution, 499,401 SETs were injected, resulting in ≈ 499 SETs per clock cycle, with a standard deviation of 18.33, while the average error between the expected an injected SETs per CLB was 15.62, with a standard deviation of 12.63.

For the *comp* benchmark, with the uniform distribution, 497,889 SETs were injected. The uniform distribution resulted in ≈ 8,030 SETs per CLB (62 CLBs), with a standard deviation of 0.50, and ≈ 498 SETs per clock cycle, with a standard deviation of 18.26. With the weighted distribution, 499,401 SETs were injected, resulting in ≈ 499 SETs per clock cycle, with a standard deviation of 17.82, while the average error between the expected an injected SETs per CLB was 14.95, with a standard deviation of 11.41.

Fig. 6.8 and Fig. 6.9 illustrate the uniform and weighted SET distribution for the *c8* benchmark, respectively. Similarly, Fig. 6.10 and Fig. 6.11 illustrate the uniform and weighted SET distribution for the *comp* benchmark, respectively.

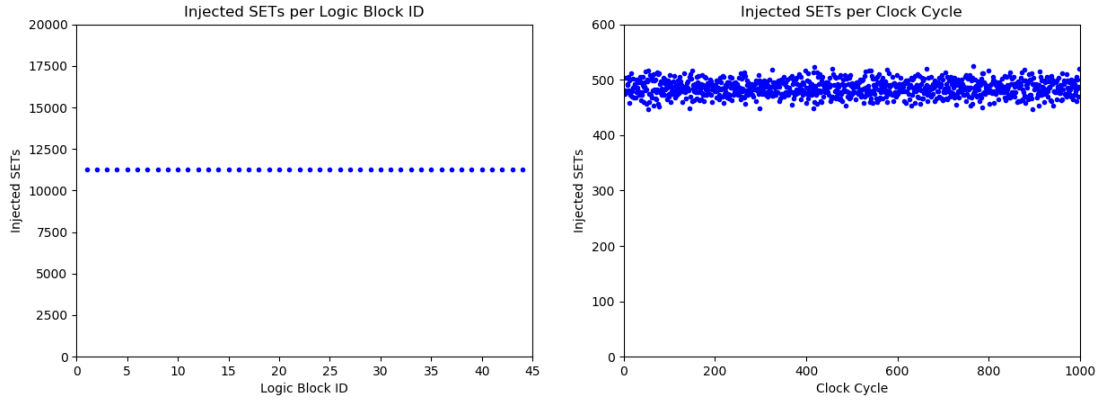

FIGURE 6.8 – Uniform SET distribution for the *c8* benchmark. The left panel presents the SET distribution per CLBs, while the right panel presents the SET distribution per clock cycle.

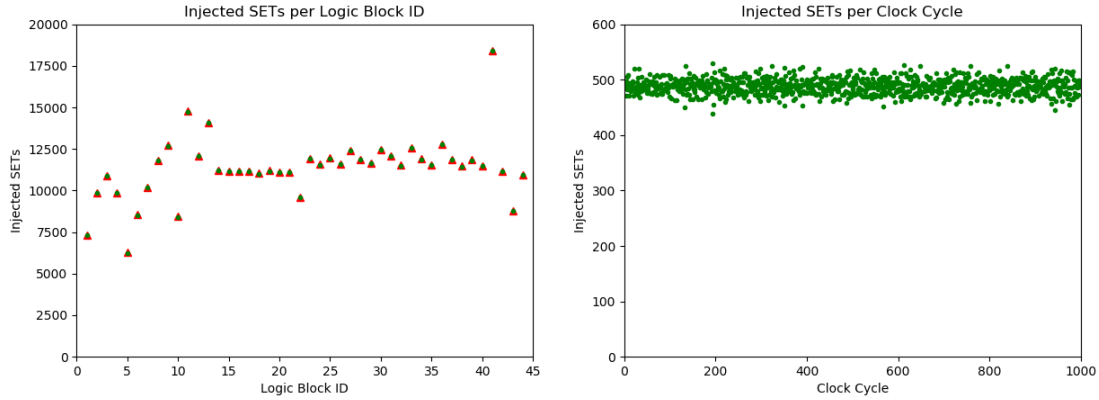

FIGURE 6.9 – Weighted SET distribution for the *c8* benchmark. The left panel presents the SET distribution per CLBs, while the right panel presents the SET distribution per clock cycle. The red triangles, added to the left panel, indicate the expected distribution for each CLB.

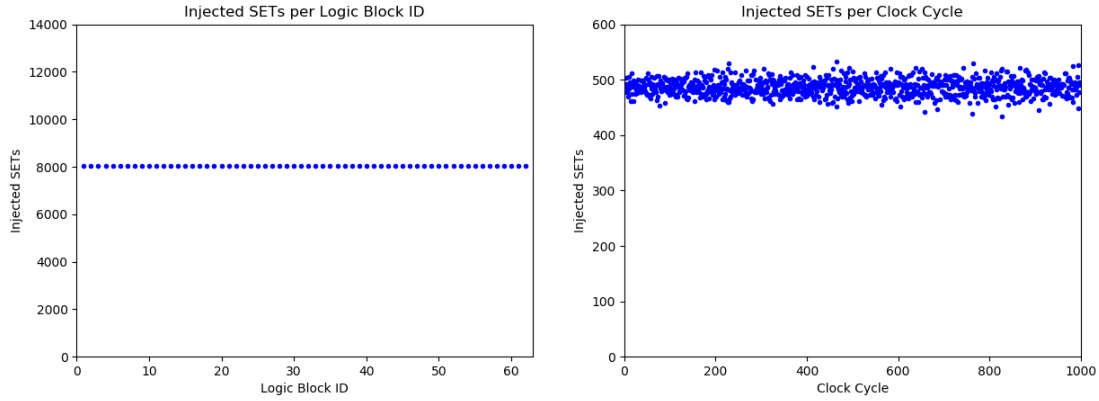

FIGURE 6.10 – Uniform SET distribution for the *comp* benchmark. The left panel presents the SET distribution per CLBs, while the right panel presents the SET distribution per clock cycle.

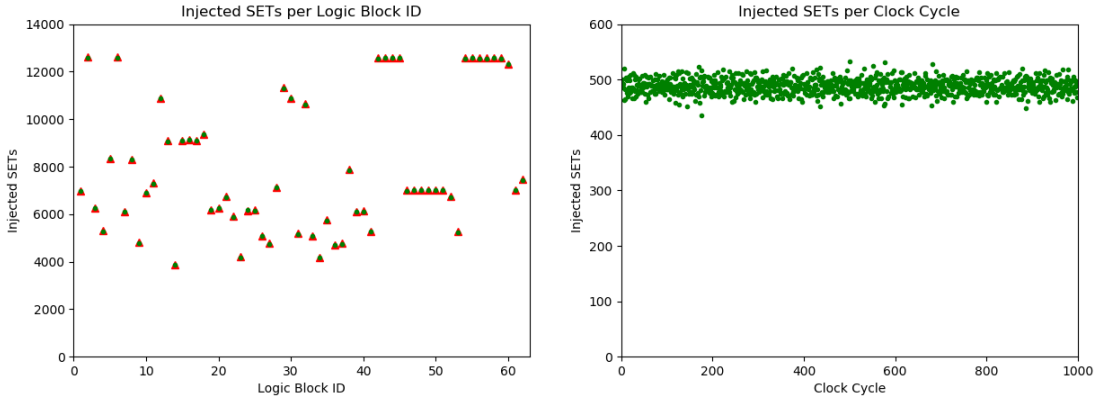

FIGURE 6.11 – Weighted SET distribution for the *comp* benchmark. The left panel presents the SET distribution per CLBs, while the right panel presents the SET distribution per clock cycle. The red triangles, added to the left panel, indicate the expected distribution for each CLB.

Regarding the CLBs, the reported SET distribution is very close to the reference values use to define then. However, concerning the clock cycles, the reported SET distribution presents a higher dispersion when compared to the dispersion observed with the CLBs.

6.5.4 Estimated SEV

This section brings the estimated SEV for each benchmark, obtained using the complete version of the strategy. These SEV estimations are compared to the estimations obtained with the simulation framework (Chapter 5).

Table 6.6 presents the data related to the SEV estimation of the benchmarks. With the average SET susceptibility of the circuit and the quantity of SETs injected at the gate level, it is possible to obtain the equivalent number of SETs injected at the transistor level.

The SEV would be the fraction of this equivalent injected SETs that produces SEs. However, the Monitor is a synchronous circuit that cannot observe all produced SEs, since the analysed benchmarks are combinational circuits propagating SETs. For this reason, this fraction is considered a partial SEV.

TABLE 6.6 – Estimated SEV using the emulation platform with the complete version of the proposed strategy. For each benchmark, it presents the average SET susceptibility of the circuit (Suscep.), the number of injected SETs (Inj. SETs), the equivalent injected SETs at the transistor level (Eq. SETs), the number of observed SEs (Obs. SEs), the partial SEV (Part. SEV), and the equivalent SEV (Eq. SEV).

Circuit	**Suscep.**	**Inj. SETs**	**Eq. SETs**	**Obs. SEs**	**Part. SEV**	**Eq. SEV**
cm152a	0.13646	498,645	3,654,148	16,219	0.4439%	5.04%
cmb	0.12588	498,645	3,961,273	13,394	0.3381%	3.84%
c8	0.14500	499,401	3,444,145	74,870	2.1738%	24.70%
comp	0.15383	499,401	3,246,447	10,430	0.3213%	3.65%

To estimate the equivalent SEV, we need the period of the system clock (25 ns) and width of the SET. The details of the transient generation are described in Appendix C. However, in all cases described in this chapter, the delay element used to produce the transient pulse, CLKDLY, is configured to '10000', that produces transients with pulse widths of $\approx$ 2.2 ns, when used with the same saboteur used in the platform. Since the pulse width is $\approx$ 8.8% of the system clock period, we assume the partial SEV is 8.8% of the equivalent SEV.

Fig. 6.12 brings a comparison between the estimated SEV using the logic simulation approach, described in the previous chapter, and using the emulation approach described in this chapter. The SEV estimations are relatively close, except for the *c8* benchmark, for which the estimation with the emulation platform is much bigger than obtained with the simulation approach.

The cause of this discrepancy is not clear. It can be related to the implementation of the emulation platform, that was manually coded. However, the analysis of the implementation did not disclose any possible cause so far, and it requires further investigation.

On the other hand, the results for the *cm152a*, *cmb*, and *comp* benchmarks suggest that it is a promising emulation platform, that deserves further researching efforts.

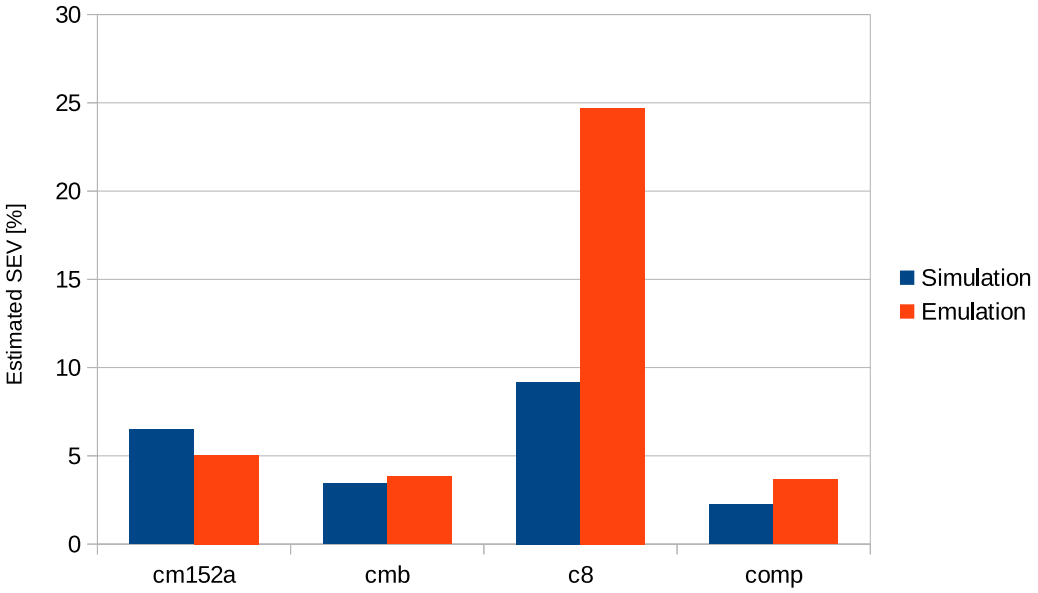

FIGURE 6.12 – Comparison between SEV estimation using logic simulation and emulation.

7 Conclusions and Perspectives

This work presents a strategy to incorporate the SET susceptibilities of the logic gates into the SEV estimation of digital circuits. This strategy is proposed in two versions, a simplified and a complete one. The first has the advantages of requiring less information and data processing, but delivers a limited gain in the estimation accuracy, while the second one, even though demanding more information and data processing, provides a more accurate SEV estimation.

As the strategy requires the SET susceptibilities of the logic gates, and this information is not available for the selected technology, this work also provides an estimation of the SET susceptibility of each CLB configuration of the chosen FPGA family.

The proposed strategy was evaluated with a logic-simulation approach applied to a set of benchmark circuits. The results show an incremental gain in the SEV estimation accuracy from an approach that ignores the SET susceptibilities to the simplified version of the strategy and, then, to the complete version of the strategy.

The work also includes a discussion about the adoption of the proposed strategy with emulation approaches. A prototype of an emulation platform is used to demonstrate the applicability of a proposed scheme for a probability-aware distribution of SETs.

In the remainder of this chapter, we discuss the main contributions of this work and perspectives for future works. Finally, we list the publications produced during the development of this work.

7.1 Main Contributions

The main contribution of this work is the proposed strategy, that leads to a reduction of the SEV estimation error, by incorporating the specific SET susceptibility of each logic gate into the estimation process.

The development of the proposed strategy for SEV estimation aimed four features, introduced in the objectives of this work (Section 1.2), and evaluated bellow.

Flexible in terms of HDL and tools: The implementation of the strategy used the VHDL language, the Modelsim simulator, the Synplify Pro synthesiser, and the Designer for placing and routing. However, the same strategy can be adapted to use other languages and tools, since it does not rely on specific features of the selected language and tools.

Regarding the adoption of another HDL, e.g., Verilog, it is essential to note that the first Step is the synthesis, after that we only use the back-annotated post-synthesis netlist of the DUT. In this way, the main concern about the language would be the testbench file. A significant issue would be the implementation of the weighted distribution of pseudo-random numbers, for which in VHDL we used a third-party package, OSVVM, as described in Section 3.6.

Concerning the synthesiser, it only needs to work with the selected language and technology. Additionally, it should be able to do not include the input and output blocks, so the synthesised circuit has only CLBs. Otherwise, the I/O blocks need to be posteriorly removed.

The logic simulator needs to work with the selected HDL and the VCD file format, or similar. The VCD format is an industry standard and is usually supported by the simulation tools. However, if it is replaced by another format, the critical issue would be able to extract the input patterns distribution from the ‘waveforms’ file, and this information must be in the data.

The place and route tool is required for the emulation platform, but once again, it

only needs to support the selected language and technology.

Finally, for the other steps, the developed scripts would need to be adapted for the new language, file format and tools.

General use, applicable to any circuit architecture: The proposed strategy does not rely on any characteristic of the analysed circuit. It can be used with any circuit architecture. The implementation of the strategy and its evaluation focused on combinational circuits. However, as shown by the estimation of SET susceptibilities of sequential elements, their susceptibilities also depend on the logic gate internal circuitry and the input patterns distribution. The inclusion of sequential elements on the analysed circuit requires that the fault-injection framework also can inject SEUs, instead of only SETs.

Additionally, regarding the generality, the proposed strategy works at the gate level, but it could be extended to work at the RTL or system level, with the ability to inject MBUs, MCUs, and multiple SETs, instead of only SETs and SEUs.

Can be incorporated to verification flow: The proposed strategy can be easily incorporated into a dynamic verification flow, in which the logic simulations are used. The first simulation, required in the complete version, would already be part of the verification flow since it is natural to verify the proper nominal behaviour of the design before verifying its behaviour in the presence of faults. However, the proposed strategy would not be suitable for another verification paradigm, e.g., formal verification.

Can be automated by scripts: Scripts can automate all the steps of the proposed strategy. As described in Section 3.2, the steps for synthesis and simulation were automated by TCL scripts, following the documentation provided by the tool vendors, while for the other steps we developed python scripts for:

- Adding saboteurs to the DUT netlist;
- Extracting the input patterns from the VCD file;

- Calculate the specific SET susceptibility of each logic gate; and
- Generate the testbench for SET injection.

For the analysed benchmarks, we also developed a script for generating their Drivers, responsible for the operational scenario. However, it is possible in this scenario, in which we use generic benchmarks that have no detailed documentation about their functionality. For the application of the proposed strategy, the Drivers must guarantee the desired operational scenario.

Finally, the generation of the emulation platform was not automated, but it can be automated.

Regarding the estimated SET susceptibilities, they show the impact of the internal circuitry and the input patterns on the probability of internally generated SEE propagate to the output of the logic gate, for both combinational and sequential elements. They also contribute to the comparative analysis that supports the evaluation of the proposed strategy. Lastly, they can be used by other studies, as a reference SET susceptibilities, especially in comparison with the proposed strategy.

Finally, concerning the emulation platform, the proposed strategy can be easily adapted to host-based approaches, but can also be incorporated into the autonomous approaches. The proposed scheme for SET distribution seems suitable for this purpose since it does not present a prohibitive consumption of resources. However, the overall emulation platform demands more research and development. The use of self-produced transients for SET emulation and the dispersion in the SET distribution over time requires further investigation.

7.2 Future Work

In the next topics, we present some perspectives for future research derived from the development of this work.

Inclusion of SEUs: The simulation framework used with the proposed strategy currently injects only SETs to the outputs of the CLBs. It will be extended to include the injection of SEUs. However, as the sequential elements can present both SETs and SEUs at their outputs, this inclusion needs to take into account the specific probability of each sequential element to produce a SET or a SEU. It is also applicable to the emulation platform.

Electrical and latching-window masking: The estimation of the SET susceptibilities of the logic gates took into account only the logical masking effect, as it is the most relevant for the analysis of the internal circuitry and input patterns effect. This SET susceptibilities can be refined with the inclusion of the electrical and latching-window masking effects. It would also lead to additional parameterisation regarding pulse widths and clock frequency. However, this modelling requires detailed documentation of the selected technology.

Other FPGAs and ASIC analysis: The presented work focused on a specific FPGA technology. The adoption of the proposed strategy for the SEV estimation of circuits implemented in other FPGAs technologies and ASICs can be explored.

SoC analysis: As discussed about the generality of this strategy, it could be extended to work at the RTL or system level, for the analysis of SoCs. It would require the ability to inject MBUs, MCUs, and multiple SETs, instead of only SETs and SEUs.

Emulation Platform: The proposed emulation platform is a prototype that demonstrates the feasibility of the SET distribution scheme. It requires more research and development efforts to become a robust platform. An alternative to the SET distribution would be the adoption of scan-based saboteurs, and the SET emulation could be replaced by the model adopted in Entrena *et al.* (2012). Lastly, the Reporters can be redesigned, as they consume too many resources.

Radiation Tests: Finally, we intend to compare the SEV estimation obtained with our proposed approach with the results from radiation tests. It requires access to the

details of the technology of the device submitted to the radiation test, for the proper estimation of the SET susceptibilities of their logic gates.

7.3 Publications

This section list the publications produced during the development of this work, divided into conference proceedings and journal article.

7.3.1 Conference Proceedings

1. ARMELIN, F. B.; NAVINER, L. A. B.; D'AMORE, R.; AZEVEDO, I. A. Impact evaluation of logic blocks configuration on FPGA's soft error rate estimation. In: **IEEE International Conference on Electronics, Circuits and Systems (ICECS). Proceedings...** Monaco: IEEE, 2016. p. 277–280.

2. ARMELIN, F. B.; NAVINER, L. A. B.; D'AMORE, R. Probability aware fault-injection approach for SER estimation. In: **19th IEEE Latin-American Test Symposium (LATS). Proceedings...** Sao Paulo: IEEE, 2018. p. 1–3.

3. ARMELIN, F. B.; NAVINER, L. A. B.; D'AMORE, R. Using FPGA self-produced transients to emulate SETs for SER estimation. In: **19th IEEE Latin-American Test Symposium (LATS). Proceedings...** Sao Paulo: IEEE, 2018. p. 1–3.

7.3.2 Journal Article

1. ARMELIN, F. B.; NAVINER, L. A. B.; D'AMORE, R. Soft-Error Vulnerability Estimation Approach Based on the SET Susceptibility of Each Gate. **Electronics**, v. 8(7), n. 749, p. 1–18, jul 2019.

Bibliography

AMENDOLA, A.; BENSO, A.; CORNO, F.; IMPAGLIAZZO, L.; MARMO, P.; PRINETTO, P.; REBAUDENGO, M.; SONZA REORDA, M. Fault behavior observation of a microprocessor system through a VHDL simulation-based fault injection experiment. In: **European Design Automation Conference with EURO-VHDL '96 and Exhibition. Proceedings...** Geneva: IEEE Comput. Soc. Press, 1996. p. 536–541.

ANGHEL, L.; REBAUDENGO, M.; SONZA REORDA, M.; VIOLANTE, M. Multi-level fault effects evaluation. In: VELAZCO, R.; FOUOLLAT, P.; REIS, R. (Ed.). **Radiation Effects on Embedded Systems**. Dordrecht: Springer, 2007. cap. 4, p. 69–88.

ANTONI, L.; LEVEUGLE, R.; FEHER, B. Using run-time reconfiguration for fault injection in hardware prototypes. In: **17th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2002. DFT 2002. Proceedings...** Vancouver: IEEE Comput. Soc, 2002. p. 245–253.

ARLAT, J.; AGUERA, M.; AMAT, L.; CROUZET, Y.; FABRE, J.-C.; LAPRIE, J.-C.; MARTINS, E.; POWELL, D. Fault injection for dependability validation: a methodology and some applications. **IEEE Transactions on Software Engineering**, v. 16, n. 2, p. 166–182, 1990.

ARMANI, J. M.; LERAY, J. L.; GAILLARD, R.; ILUTA, V. TID Response of Various Field Programmable Gate Arrays and Memory Devices. In: **15th European Conference on Radiation and Its Effects on Components and Systems (RADECS). Proceedings...** Moscow: IEEE, 2015. p. 1–4.

ARMELIN, F. B.; NAVINER, L. A. B.; D'AMORE, R. Probability aware fault-injection approach for SER estimation. In: **19th IEEE Latin-American Test Symposium (LATS). Proceedings...** Sao Paulo: IEEE, 2018. p. 1–3.

ARMELIN, F. B.; NAVINER, L. A. B.; D'AMORE, R. Using FPGA self-produced transients to emulate SETs for SER estimation. In: **19th IEEE Latin-American Test Symposium (LATS). Proceedings...** Sao Paulo: IEEE, 2018. p. 1–3.

ARMELIN, F. B.; NAVINER, L. A. B.; D'AMORE, R. Soft-Error Vulnerability Estimation Approach Based on the SET Susceptibility of Each Gate. **Electronics**, v. 8(7), n. 749, p. 1–18, jul 2019.

ARMELIN, F. B.; NAVINER, L. A. B.; D'AMORE, R.; AZEVEDO, I. A. Impact evaluation of logic blocks configuration on FPGA's soft error rate estimation. In: **IEEE International Conference on Electronics, Circuits and Systems (ICECS). Proceedings...** Monaco: IEEE, 2016. p. 277–280.

ASADI, G.; TAHOORI, M. An Accurate SER Estimation Method Based on Propagation Probability. In: **Design, Automation and Test in Europe. Proceedings...** Munich: IEEE, 2005. p. 306–307.

BARAZA, J.; GRACIA, J.; GIL, D.; GIL, P. A prototype of a VHDL-based fault injection tool: description and application. **Journal of Systems Architecture**, v. 47, n. 10, p. 847–867, apr 2002.

BARAZA, J.; GRACIA, J.; GIL, D.; GIL, P. Improvement of fault injection techniques based on VHDL code modification. In: **10th IEEE International High-Level Design Validation and Test Workshop. Proceedings...** Napa Valley: IEEE, 2005. p. 19–26.

BARNABY, H. J. Total-Ionizing-Dose Effects in Modern CMOS Technologies. **IEEE Transactions on Nuclear Science**, v. 53, n. 6, p. 3103–3121, dec 2006.

BERROJO, L.; GONZALEZ, I.; CORNO, F.; SONZA REORDA, M.; SQUILLERO, G.; ENTRENA, L.; LOPEZ, C. New techniques for speeding-up fault-injection campaigns. In: **Design, Automation and Test in Europe. Proceedings...** Paris: IEEE Comput. Soc, 2002. p. 847–852.

BINDER, D.; SMITH, E. C.; HOLMAN, A. B. Satellite Anomalies from Galactic Cosmic Rays. **IEEE Transactions on Nuclear Science**, v. 22, n. 6, p. 2675–2680, 1975.

BOTTONI, C.; COEFFIC, B.; DAVEAU, J.-M.; NAVINER, L.; ROCHE, P. Partial triplication of a SPARC-V8 microprocessor using fault injection. In: **6th IEEE Latin American Symposium on Circuits & Systems (LASCAS). Proceedings...** Montevideo: IEEE, 2015. p. 1–4.

BOTTONI, C.; GLORIEUX, M.; DAVEAU, J.; GASIOT, G.; ABOUZEID, F.; CLERC, S.; NAVINER, L.; ROCHE, P. Heavy ions test result on a 65nm Sparc-V8 radiation-hard microprocessor. In: **IEEE International Reliability Physics Symposium. Proceedings...** Waikoloa: IEEE, 2014. p. 1–6.

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

BOUE, J.; PETILLON, P.; CROUZET, Y. MEFISTO-L: a VHDL-based fault injection tool for the experimental assessment of fault tolerance. In: **28th Annual International Symposium on Fault-Tolerant Computing. Proceedings...** Munich: IEEE Comput. Soc, 1998. p. 168–173.

BUARD, N.; ANGHEL, L. Gate Level Modeling and Simulation. In: NICOLAIDIS, M. (Ed.). **Soft Errors in Modern Electronic Systems**. New York: Springer, 2011. cap. 4, p. 77–102.

CAI, C.; FAN, X.; LIU, J.; LI, D.; LIU, T.; KE, L.; ZHAO, P.; HE, Z. Heavy-Ion Induced Single Event Upsets in Advanced 65 nm Radiation Hardened FPGAs. **Electronics**, v. 8(3), n. 323, p. 1–13, mar 2019.

CARREIRA, J.; MADEIRA, H.; SILVA, J. Xception: a technique for the experimental evaluation of dependability in modern computers. **IEEE Transactions on Software Engineering**, v. 24, n. 2, p. 125–136, feb 1998.

CIVERA, P.; MACCHIARULO, L.; REBAUDENGO, M.; SONZA REORDA, M.; VIOLANTE, M. Exploiting circuit emulation for fast hardness evaluation. **IEEE Transactions on Nuclear Science**, v. 48, n. 6, p. 2210–2216, 2001.

CLAEYS, C.; SIMOEN, E. **Radiation Effects in Advanced Semiconductor Materials and Devices**. Berlin: Springer, 2002. (Springer Series in Materials Science, v. 57).

DABIRI, F.; NAHAPETIAN, A.; MASSEY, T.; POTKONJAK, M.; SARRAFZADEH, M. General methodology for soft-error-aware power optimization using gate sizing. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, IEEE, v. 27, n. 10, p. 1788–1797, sep 2008.

DIEHL, S. E.; VINSON, J. E.; SHAFER, B. D.; MNICH, T. M. Considerations for Single Event Immune VLSI Logic. **IEEE Transactions on Nuclear Science**, v. 30, n. 6, p. 4501–4507, dec 1983. ISSN 0018-9499.

DING, L.; GERARDIN, S.; BAGATIN, M.; BISELLO, D.; MATTIAZZO, S.; PACCAGNELLA, A. Investigation of total ionizing dose effect and displacement damage in 65nm CMOS transistors exposed to 3MeV protons. **Nuclear Instruments and Methods in Physics Research Section A**, v. 796, p. 104–107, oct 2015.

DODD, P. Device simulation of charge collection and single-event upset. **IEEE Transactions on Nuclear Science**, v. 43, n. 2, p. 561–575, apr 1996.

DODD, P. E.; MASSENGILL, L. W. Basic mechanisms and modeling of single-event upset in digital microelectronics. **IEEE Transactions on Nuclear Science**, v. 50, n. 3, p. 583–602, jun 2003.

DODD, P. E.; SHANEYFELT, M. R.; FELIX, J. A.; SCHWANK, J. R. Production and propagation of single-event transients in high-speed digital logic ICs. **IEEE Transactions on Nuclear Science**, v. 51, n. 6, p. 3278–3284, dec 2004.

EBRAHIMI, M.; MOHAMMADI, A.; EJLALI, A.; MIREMADI, S. G. A fast, flexible, and easy-to-develop FPGA-based fault injection technique. **Microelectronics Reliability**, Elsevier, v. 54, n. 5, p. 1000–1008, may 2014.

ENTRENA, L.; GARCIA-VALDERAS, M.; FERNANDEZ-CARDENAL, R.; PORTELA-GARCIA, M.; LOPEZ-ONGIL, C. SET Emulation Considering Electrical Masking Effects. **IEEE Transactions on Nuclear Science**, v. 56, n. 4, p. 2021–2025, aug 2009.

ENTRENA, L.; GARCIA-VALDERAS, M.; FERNANDEZ-CARDENAL, R.; LINDOSO, A.; PORTELA-GARCIA, M.; LOPEZ-ONGIL, C. Soft Error Sensitivity

Evaluation of Microprocessors by Multilevel Emulation-Based Fault Injection. **IEEE Transactions on Computers**, v. 61, n. 3, p. 313–322, mar 2012. ISSN 0018-9340.

EVANS, A.; ALEXANDRESCU, D.; FERLET-CAVROIS, V.; NICOLAIDIS, M. New Techniques for SET Sensitivity and Propagation Measurement in Flash-Based FPGAs. **IEEE Transactions on Nuclear Science**, v. 61, n. 6, p. 3171–3177, dec 2014.

FACCIO, F.; CERVELLI, G. Radiation-induced edge effects in deep submicron CMOS transistors. **IEEE Transactions on Nuclear Science**, v. 52, n. 6, p. 2413–2420, dec 2005.

FAZELI, M.; MIREMADI, S. G.; ASADI, H.; TAHOORI, M. B. A Fast Analytical Approach to Multi-cycle Soft Error Rate Estimation of Sequential Circuits. In: **13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools. Proceedings...** Lille: IEEE, 2010. p. 797–800.

FEDERICO, C.; GONÇALEZ, O.; FONSECA, E.; MARTIN, I.; CALDAS, L. Neutron spectra measurements in the south Atlantic anomaly region. **Radiation Measurements**, v. 45, n. 10, p. 1526–1528, dec 2010.

FERLET-CAVROIS, V.; MASSENGILL, L. W.; GOUKER, P. Single Event Transients in Digital CMOS - A Review. **IEEE Transactions on Nuclear Science**, v. 60, n. 3, p. 1767–1790, jun 2013.

FERLET-CAVROIS, V.; POUGET, V.; MCMORROW, D.; SCHWANK, J.; FEL, N.; ESSELY, F.; FLORES, R.; PAILLET, P.; GAILLARDIN, M.; KOBAYASHI, D.; MELINGER, J.; DUHAMEL, O.; DODD, P.; SHANEYFELT, M. Investigation of the Propagation Induced Pulse Broadening (PIPB) Effect on Single Event Transients in SOI and Bulk Inverter Chains. **IEEE Transactions on Nuclear Science**, v. 55, n. 6, p. 2842–2853, dec 2008.

FOLKESSON, P.; SVENSSON, S.; KARLSSON, J. A comparison of simulation based and scan chain implemented fault injection. In: **28th Annual International Symposium on Fault-Tolerant Computing. Proceedings...** Munich: IEEE Comput. Soc, 1998. p. 284–293.

FRANCO, D. T.; VASCONCELOS, M. C.; NAVINER, L.; NAVINER, J.-F. Reliability analysis of logic circuits based on signal probability. In: **15th IEEE International Conference on Electronics, Circuits and Systems**. St. Julien's: IEEE, 2008. p. 670–673.

GAILLARD, R. Single Event Effects: Mechanisms and Classification. In: NICOLAIDIS, M. (Ed.). **Soft Errors in Modern Electronic Systems**. New York: Springer, 2011. cap. 2, p. 27–54.

GARCÍA-VALDERAS, M.; ENTRENA, L.; FERNÁNDEZ-CARDENAL, R.; LÓPEZ-ONGIL, C.; PORTELA-GARCÍA, M. SET Emulation Under a Quantized Delay Model. **Journal of Electronic Testing**, v. 25, n. 1, p. 107–116, feb 2009.

GHOSH, S.; CHAKRABORTY, T. J. On behavior fault modeling for digital designs. **Journal of Electronic Testing**, v. 2, n. 2, p. 135–151, jun 1991.

GODA, B. S.; KRAFT, R. P.; CARLOUGH, S. R.; Krawczyk Jr., T. W.; MCDONALD, J. F. Gigahertz Reconfigurable Computing Using SiGe HBT BiCMOS FPGAs. In: BREBNER, G.; WOODS, R. (Ed.). **Field-Programmable Logic and Applications**. Berlin: Springer-Verlag, 2001. p. 59–69.

HAN, J.; CHEN, H.; BOYKIN, E.; FORTES, J. Reliability evaluation of logic circuits using probabilistic gate models. **Microelectronics Reliability**, Elsevier Ltd, v. 51, n. 2, p. 468–476, feb 2011.

HAN, J.; CHEN, H.; LIANG, J.; ZHU, P.; YANG, Z.; LOMBARDI, F. A Stochastic Computational Approach for Accurate and Efficient Reliability Evaluation. **IEEE Transactions on Computers**, IEEE, v. 63, n. 6, p. 1336–1350, jun 2014.

HINRICHSEN, P.; HOUDAYER, A.; BARRY, A.; VINCENT, J. Proton induced damage in SiC light emitting diodes. **IEEE Transactions on Nuclear Science**, v. 45, n. 6, p. 2808–2812, dec 1998.

HOLMES-SIEDLE, A.; ADAMS, L. **Handbook of radiation effects**. 2nd.. ed. New York: Oxford University Press, 2002. 642 p.

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. **1149.1-1990 - IEEE Standard Test Access Port and Boundary-Scan Architecture**. New York, 1993. 139 p.

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. **1364-1995 - IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language**. New York, 1996. 676 p.

INTEL. **Stratix 10 Logic Array Blocks and Adaptive Logic Modules User Guide**. Santa Clara, 2018. 18 p.

INTEL. **Agilex Logic Array Blocks and Adaptive Logic Modules User Guide**. Santa Clara, 2019. 20 p.

JENN, E.; ARLAT, J.; RIMEN, M.; OHLSSON, J.; KARLSSON, J. Fault injection into VHDL models: the MEFISTO tool. In: **24th IEEE International Symposium on Fault- Tolerant Computing. Proceedings...** Austin: IEEE Comput. Soc. Press, 1994. p. 66–75.

JEONG, Y. S.; LEE, S. M.; LEE, S. E. A Survey of Fault-Injection Methodologies for Soft Error Rate Modeling in Systems-on-Chips. **Bulletin of Electrical Engineering and Informatics**, v. 5, n. 2, p. 169–177, jun 2016.

KANAWATI, G.; KANAWATI, N.; ABRAHAM, J. FERRARI: a tool for the validation of system dependability properties. In: **22nd International Symposium on Fault-Tolerant Computing. Proceedings...** Boston: IEEE, 1992. p. 336–344.

KARLSSON, J.; LIDEN, P.; DAHLGREN, P.; JOHANSSON, R.; GUNNEFLO, U. Using heavy-ion radiation to validate fault-handling mechanisms. **IEEE Micro**, v. 14, n. 1, p. 8–23, feb 1994.

KOGA, R.; PINKERTON, S.; MOSS, S.; MAYER, D.; LALUMONDIERE, S.; HANSEL, S.; CRAWFORD, K.; CRAIN, W. Observation of single event upsets in analog microcircuits. **IEEE Transactions on Nuclear Science**, v. 40, n. 6, p. 1838–1844, dec 1993.

KOOLI, M.; DI-NATALE, G. A survey on simulation-based fault injection tools for complex systems. In: **9th IEEE International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS). Proceedings...** Santorini: IEEE, 2014. p. 1–6.

LATTICE SEMICONDUCTOR. **iCE40 UltraPlus Family**. Portland, 2018. 51 p.

LEVEUGLE, R.; HADJIAT, K. Optimized generation of VHDL mutants for injection of transition errors. In: **13th Symposium on Integrated Circuits and Systems Design. Proceedings...** Manaus: IEEE Comput. Soc, 2000. p. 243–248.

LOPES FILHO, A.; D'AMORE, R. Analysis of the error susceptibility of a field programmable gate array-based image compressor through random event injection simulation. **IET Computers & Digital Techniques**, v. 6, n. 3, p. 160–165, may 2012.

LOPEZ-ONGIL, C.; GARCIA-VALDERAS, M.; PORTELA-GARCIA, M.; ENTRENA, L. Autonomous Fault Emulation: A New FPGA-Based Acceleration System for Hardness Evaluation. **IEEE Transactions on Nuclear Science**, v. 54, n. 1, p. 252–261, feb 2007.

MADEIRA, H.; RELA, M.; MOREIRA, F.; SILVA, J. G. RIFLE: A general purpose pin-level fault injector. In: ECHTLE, K.; HAMMER, D.; POWELL, D. (Ed.). **Dependable Computing – EDCC-1**. Berlin: Springer Berlin Heidelberg, 1994. cap. 12, p. 197–216.

MAXIM INTEGRATED. **Application Note 4400 - Pseudo Random Number Generation Using Linear Feedback Shift Registers**. San Jose, 2010. 7 p.

MAY, D.; STECHELE, W. An FPGA-based probability-aware fault simulator. In: **International Conference on Embedded Computer Systems (SAMOS). Proceedings...** Samos: IEEE, 2012. p. 302–309.

MAY, D.; STECHELE, W. A resource-efficient probabilistic fault simulator. In: **23rd International Conference on Field programmable Logic and Applications. Proceedings...** Porto: IEEE, 2013. p. 1–4.

MICROSEMI. **IGLOO, ProASIC3, SmartFusion and Fusion Macro Library Guide for Software v10.1**. Aliso Viejo, 2010. 180 p.

MICROSEMI. **ProASIC3E FPGA Fabric User's Guide**. Rev. 4. Aliso Viejo, 2012. 348 p.

MICROSEMI. **ProASIC3/E Starter Kit User's Guide**. Aliso Viejo, 2012. 54 p.

MICROSEMI. **ProASIC3E Flash Family FPGAs with Optional Soft ARM Support**. Rev. 15. Aliso Viejo, 2015. 160 p.

MOHAMMADI, A.; EBRAHIMI, M.; EJLALI, A.; MIREMADI, S. G. SCFIT: A FPGA-based fault injection technique for SEU fault model. In: **Design, Automation & Test in Europe Conference & Exhibition (DATE). Proceedings...** Dresden: IEEE, 2012. p. 586–589.

NATELLA, R.; COTRONEO, D.; MADEIRA, H. S. Assessing Dependability with Software Fault Injection. **ACM Computing Surveys**, v. 48, n. 3, p. 1–55, feb 2016.

NAVINER, L. A. B.; NAVINER, J. F.; SANTOS JR., G. G. dos; MARQUES, E. C.; PAIVA JR., N. M. FIFA: A fault-injectionâfault-analysis-based tool for reliability assessment at RTL level. **Microelectronics Reliability**, Elsevier, v. 51, n. 9-11, p. 1459–1463, sep 2011.

NICHITIU, F.; DRUMMOND, J. R.; ZOU, J.; DESCHAMBAULT, R. Solar particle events seen by the MOPITT instrument. **Journal of Atmospheric and Solar-Terrestrial Physics**, v. 66, n. 18, p. 1797–1803, dec 2004.

OKUNO, E.; YOSHIMURA, E. M. **Física das Radiações**. Sao Paulo: Oficina de Textos, 2010. 296 p.

OSVVM.ORG. **Open Source VHDL Verification Methodology**. 2018. Available at: <http://osvvm.org/>.

PARROTTA, B.; REBAUDENGO, M.; SONZA REORDA, M.; VIOLANTE, M. New techniques for accelerating fault injection in VHDL descriptions. In: **6th IEEE International On-Line Testing Workshop. Proceedings...** Palma de Mallorca: IEEE Comput. Soc, 2000. p. 61–66.

PEREIRA JUNIOR, E. C. F. **Desenvolvimento de uma plataforma para ensaios de efeitos da radiação ionizante em memórias**. 126 p. Thesis (Master) — Instituto Tecnológico de Aeronáutica, 2015.

PICKEL, J. C.; BLANDFORD, J. T. Cosmic Ray Induced in MOS Memory Cells. **IEEE Transactions on Nuclear Science**, v. 25, n. 6, p. 1166–1171, dec 1978.

PORTELA-GARCIA, M.; LOPEZ-ONGIL, C.; GARCIA-VALDERAS, M.; ENTRENA, L. A Rapid Fault Injection Approach for Measuring SEU Sensitivity in Complex Processors. In: **13th IEEE International On-Line Testing Symposium (IOLTS). Proceedings...** Crete: IEEE, 2007. p. 101–106.

RAJI, M.; PEDRAM, H.; GHAVAMI, B. A practical metric for soft error vulnerability analysis of combinational circuits. **Microelectronics Reliability**, Elsevier, v. 55, n. 2, p. 448–460, feb 2015.

REITZ, G. Characteristic of the radiation field in low earth orbit and in deep space. **Zeitschrift für Medizinische Physik**, v. 18, n. 4, p. 233–243, dec 2008.

REZAEI, S.; MIREMADI, S. G.; ASADI, H.; FAZELI, M. Soft error estimation and mitigation of digital circuits by characterizing input patterns of logic gates. **Microelectronics Reliability**, Elsevier, v. 54, n. 6, p. 1412–1420, jun 2014.

REZGUI, S.; WANG, J. J.; TUNG, E. C.; CRONQUIST, B.; MCCOLLUM, J. New Methodologies for SET Characterization and Mitigation in Flash-Based FPGAs. **IEEE Transactions on Nuclear Science**, v. 54, n. 6, p. 2512–2524, dec 2007.

SAJJADE, F. M.; GOYAL, N. K.; VARAPRASAD, B.; MOOGINA, R. Radiation Hardened by Design Latches – A Review and SEU Fault Simulations. **Microelectronics Reliability**, Elsevier, v. 83, n. July 2017, p. 127–135, apr 2018.

SIEH, V.; TSCHACHE, O.; BALBACH, F. VERIFY: evaluation of reliability using VHDL-models with embedded fault descriptions. In: **27th IEEE International Symposium on Fault Tolerant Computing. Proceedings...** Seattle: IEEE Comput. Soc, 1997. p. 32–36.

SLAYMAN, C. JEDEC Standards on Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray Induced Soft Errors. In: NICOLAIDIS, M. (Ed.). **Soft Errors in Modern Electronic Systems**. New York: Springer, 2011. cap. 3, p. 55–76.

SNOW, E.; GROVE, A.; FITZGERALD, D. Effects of ionizing radiation on oxidized silicon surfaces and planar devices. **Proceedings of the IEEE**, v. 55, n. 7, p. 1168–1185, jul 1967.

SOOTKANEUNG, W.; SALUJA, K. K. Gate input reconfiguration for combating soft errors in combinational circuits. In: **International Conference on Dependable Systems and Networks Workshops (DSN-W). Proceedings...** Chicago: IEEE, 2010. p. 107–112.

SRINIVASAN, V.; STERNBERG, A.; DUNCAN, A.; ROBINSON, W.; BHUVA, B.; MASSENGILL, L. Single-event mitigation in combinational logic using targeted data path hardening. **IEEE Transactions on Nuclear Science**, v. 52, n. 6, p. 2516–2523, dec 2005.

SROUR, J.; MARSHALL, C.; MARSHALL, P. Review of displacement damage effects in silicon devices. **IEEE Transactions on Nuclear Science**, v. 50, n. 3, p. 653–670, jun 2003.

STASSINOPOULOS, E.; RAYMOND, J. The space radiation environment for electronics. **Proceedings of the IEEE**, v. 76, n. 11, p. 1423–1442, nov 1988.

STERPONE, L.; BATTEZZATI, N.; FERLET-CAVROIS, V. Analysis of SET Propagation in Flash-Based FPGAs by Means of Electrical Pulse Injection. **IEEE Transactions on Nuclear Science**, v. 57, n. 4, p. 1820–1826, aug 2010.

TAMBARA, L. A.; AKHMETOV, A.; BOBROVSKY, D. V.; KASTENSMIDT, F. L. On the Characterization of Embedded Memories of Zynq-7000 All Programmable SoC under Single Event Upsets Induced by Heavy Ions and Protons. In: **15th European Conference on Radiation and Its Effects on Components and Systems (RADECS). Proceedings...** Moscow: IEEE, 2015. p. 1–4.

TAMBARA, L. A.; KASTENSMIDT, F. L.; MEDINA, N. H.; ADDED, N.; AGUIAR, V. A. P.; AGUIRRE, F.; MACCHIONE, E. L. A.; SILVEIRA, M. A. G. Heavy Ions Induced Single Event Upsets Testing of the 28 nm Xilinx Zynq-7000 All Programmable SoC. In: **IEEE Radiation Effects Data Workshop (REDW). Proceedings...** Boston: IEEE, 2015. p. 1–6.

TURFLINGER, T. Single-event effects in analog and mixed-signal integrated circuits. **IEEE Transactions on Nuclear Science**, v. 43, n. 2, p. 594–602, apr 1996.

VELAZCO, R.; FOUILLAT, P.; REIS, R. **Radiation Effects on Embedded Systems**. Dordrecht: Springer Netherlands, 2007. 269 p.

VIOLANTE, M. Accurate single-event-transient analysis via zero-delay logic simulation. **IEEE Transactions on Nuclear Science**, v. 50, n. 6, p. 2113–2118, dec 2003.

WARD, P. C.; ARMSTRONG, J. R. Behavioral fault simulation in VHDL. In: **27th ACM/IEEE design automation conference – DAC'90. Proceedings...** New York: ACM Press, 1990. p. 587–593.

WIRTH, G.; KASTENSMIDT, F. L.; RIBEIRO, I. Single Event Transients in Logic Circuits – Load and Propagation Induced Pulse Broadening. **IEEE Transactions on Nuclear Science**, v. 55, n. 6, p. 2928–2935, dec 2008.

XILINX. **XC2064-XC2018 Logic Cell Array**. San Jose, 1985. 41 p.

XILINX. **Spartan-3 Generation FPGA User Guide**. San Jose, 2011. 512 p.

YANG, S. **Logic Synthesis and Optimization Benchmarks User Guide Version 3.0**. North Carolina, 1991. 45 p.

ZIADE, H.; AYOUBI, R.; VELAZCO, R. A Survey on Fault Injection Techniques. **The International Arab Journal of Information Technology**, v. 1, n. 2, p. 171–186, jul 2004.

Appendix A - Configurations of the VersaTile Functions

This Appendix presents the configuration vector and the associated input mapping for each function implemented with the VersaTile model. Table A.1 presents the data for the combinational macros, while Table A.2 presents the data for the flip-flops, and Table A.3 presents the data for the latches.

TABLE A.1 – Configuration vectors for the combinational macros. For each Macro ID, it presents the adopted configuration vector, [SW31..SW0], and the corresponding input mapping, correlating the macro inputs (A, B, and C/S) with the Versatile inputs (X3, X2, X1, and XC). The table has three parts, for 1-, 2-, and 3-input functions.

Macro ID	**A**	**B**	**C/S**	**Configuration Vector**
BUFD	X3	–	–	[1110.1100.0010.0111.0001.1100.0010.0000]
BUFF	X3	–	–	[0101.1000.0011.0110.0011.0101.0001.0100]
INV	X3	–	–	[0101.1000.0011.0110.0010.0111.0001.0100]
INVD	X3	–	–	[1110.1100.0010.0111.0001.0101.0010.0000]
AND2	X3	X2	–	[1100.1000.0011.1000.0010.0011.0001.0101]
AND2A	X3	X2	–	[1100.1000.0011.1000.0010.1110.0001.0100]
AND2B	X3	X2	–	[1100.1000.0011.0100.0010.1110.0001.0100]
NAND2	X3	X2	–	[0101.1000.0011.0100.0010.0011.0010.0101]
NAND2A	X3	X2	–	[0101.1000.0011.0100.0010.0011.0001.0101]
NAND2B	X3	X2	–	[0101.1000.0011.1000.0010.0011.0001.0101]
NOR2	X3	X2	–	[1110.1000.0010.0101.0001.0101.0010.0100]
NOR2A	X3	X2	–	[1100.1000.0011.0100.0010.0011.0001.0101]
NOR2B	X3	X2	–	[1110.1000.0010.1001.0001.1100.0010.0100]
OR2	X3	X2	–	[1110.1100.0010.0101.0001.1100.0010.0000]

Continue on next page

TABLE A.1 – continued from previous page

Macro ID	A	B	C/S	Configuration Vector
OR2A	X3	X2	–	[0101.1000.0011.1000.0010.0011.0010.0101]
OR2B	X3	X2	–	[1110.1100.0010.1001.0001.0101.0010.0000]
XNOR2	X3	X2	–	[0101.1000.0011.0100.0010.0111.0001.0100]
XOR2	X3	X2	–	[0101.1000.0011.0100.0011.0101.0010.0100]
AND3	X3	X2	X1	[0001.1001.0011.1001.0000.0011.0001.0101]
AND3A	X3	X2	X1	[0001.1001.0011.1001.0000.1110.0001.0100]
AND3B	X3	X2	X1	[0001.1001.0011.0101.0000.1110.0001.0100]
AND3C	X3	X2	X1	[0001.1010.0011.0101.0000.1110.0001.0100]
AO1	X2	X1	XC	[0101.0000.1001.1000.0011.0001.1101.0001]
AO12	X3	X2	X1	[0001.1001.0011.1000.0011.0001.1010.0001]
AO13	X3	X2	X1	[0001.1001.0011.1000.0010.0011.0001.0101]
AO14	X3	X2	X1	[0001.1001.0011.1000.0010.0011.1001.0001]
AO15	X3	X2	X1	[0001.1010.0011.0100.0011.0101.0010.0100]
AO16	X2	X1	XC	[0011.0101.0010.0100.0010.0011.0101.0001]
AO17	X3	X2	X1	[0001.1010.0011.1000.0011.0101.0010.0100]
AO18	X3	X2	X1	[0001.1001.0011.1000.0011.0001.0010.0101]
AO1A	X2	X1	XC	[0101.0000.1001.0100.0011.0001.1101.0001]
AO1B	X2	X1	XC	[0101.0000.1001.1000.0101.1000.0101.0101]
AO1C	X2	X1	XC	[0101.0000.1001.0100.0101.1000.0101.0101]
AO1D	X2	X1	XC	[0101.0000.0101.0100.0011.0001.1101.0001]
AO1E	X2	X1	XC	[0101.0000.0101.0100.0101.1000.0101.0101]
AOI1	X3	X2	X1	[0001.1001.0101.0100.0010.1010.0010.0101]
AOI1A	X3	X2	X1	[0001.1001.0101.0100.0010.1010.0001.0101]
AOI1B	X3	X2	X1	[0001.1001.0011.0101.0000.1010.1001.0001]
AOI1C	X3	X2	X1	[0001.1001.0101.1000.0010.1010.0001.0101]
AOI1D	X3	X2	X1	[0001.1001.0011.1001.0000.1010.1010.0001]
AOI5	X3	X2	X1	[0001.1101.0011.1000.0010.0011.0010.0001]
AX1	X3	X2	X1	[0001.1010.1001.0100.0010.1010.1001.0001]
AX1A	X3	X2	X1	[0001.1001.0101.0100.0010.1010.1001.0001]
AX1B	X3	X2	X1	[0001.1010.1001.1000.0010.1010.1001.0001]
AX1C	X3	X2	X1	[0001.1010.1001.0100.0010.1010.1010.0001]
AX1D	X3	X2	X1	[0001.1001.0101.1000.0010.1010.1001.0001]
AX1E	X3	X2	X1	[0001.1001.0101.0100.0010.1010.1010.0001]
AXO1	X3	X2	X1	[0001.1110.1001.0100.0010.1010.0001.0001]
AXO2	X3	X2	X1	[0001.1110.1001.0100.0010.1010.0010.0001]

Continue on next page

TABLE A.1 – continued from previous page

Macro ID	A	B	C/S	Configuration Vector
AXO3	X3	X2	X1	[0001.1101.0101.1000.0010.1010.0001.0001]
AXO5	X3	X2	X1	[0001.1101.0101.0100.0010.1010.0010.0001]
AXO6	X3	X2	X1	[0001.1110.1001.1000.0010.1010.0001.0001]
AXO7	X3	X2	X1	[0001.1101.0101.1000.0010.1010.0010.0001]
AXOI1	X3	X1	XC	[0011.0110.0010.1010.0010.0011.0010.0001]
AXOI2	X3	X1	XC	[0011.0110.0010.1010.0010.0011.0001.0001]
AXOI3	X2	X1	XC	[0011.0110.0010.1000.0010.0011.0110.0001]
AXOI4	X2	X1	XC	[0011.0101.0010.1000.0010.0011.0110.0001]
AXOI5	X2	X1	XC	[0011.0101.0010.0100.0010.0011.0110.0001]
AXOI7	X2	X1	XC	[0011.0110.0010.0100.0010.0011.0110.0001]
MAJ3	X3	X2	X1	[0001.1010.0011.1000.0010.0011.0001.0101]
MAJ3X	X3	X2	X1	[0001.1010.0011.1000.0010.0111.0001.0100]
MAJ3XI	X3	X2	X1	[0001.1001.0011.0100.0011.0001.1010.0001]
MIN3	X3	X2	X1	[0001.1001.0011.0100.0011.0001.0010.0101]
MIN3X	X3	X2	X1	[0001.1001.0011.0100.0011.0101.0010.0100]
MIN3XI	X3	X2	X1	[0001.1010.0011.1000.0010.0011.1001.0001]
MX2	X3	X2	X1	[0001.1001.0011.1000.0100.1010.0001.0101]
MX2A	X3	X2	X1	[0001.1001.0011.1000.0101.1000.0001.0101]
MX2B	X3	X2	X1	[0001.1001.0011.0100.0100.1010.0001.0101]
MX2C	X3	X2	X1	[0001.1001.0011.0100.0101.1000.0001.0101]
NAND3	X3	X2	X1	[0001.1110.0011.0100.0010.1010.0010.0001]
NAND3A	X3	X2	X1	[0001.1110.0011.0100.0010.1010.0001.0001]
NAND3B	X3	X2	X1	[0001.1110.0011.1000.0010.1010.0001.0001]
NAND3C	X3	X2	X1	[0001.1101.0011.1000.0010.1010.0001.0001]
NOR3	X3	X2	X1	[1100.1000.0101.0101.0000.1110.0010.0100]
NOR3A	X3	X2	X1	[0001.1010.0011.0101.0000.0011.0001.0101]
NOR3B	X3	X2	X1	[0001.1010.0011.1001.0000.0011.0001.0101]
NOR3C	X3	X2	X1	[1100.1000.1001.1001.0001.1100.0010.0100]
OA1	X3	X2	X1	[1010.1101.1000.0101.0001.1100.0010.0000]
OA1A	X3	X2	X1	[0001.1001.0011.1001.0000.1010.1001.0001]
OA1B	X3	X2	X1	[1010.1110.1000.0101.0001.1100.0010.0000]
OA1C	X3	X2	X1	[0001.1001.0101.1000.0010.1010.0010.0101]
OAI1	X3	X1	XC	[1100.0100.0101.1010.0011.1100.0010.0000]
OR3	X3	X2	X1	[1010.1101.1000.1000.0011.1000.0001.0001]
OR3A	X3	X2	X1	[0001.1101.0011.1000.0010.1010.0010.0001]

Continue on next page

TABLE A.1 – continued from previous page

Macro ID	**A**	**B**	**C/S**	**Configuration Vector**
OR3B	X3	X2	X1	[0001.1101.0011.0100.0010.1010.0010.0001]
OR3C	X3	X2	X1	[1010.1110.1000.0100.0011.1000.0010.0001]
XA1	X3	X2	X1	[0001.1001.0011.0101.0000.0011.1001.0001]
XA1A	X3	X2	X1	[0001.1001.0011.0101.0000.1110.1010.0000]
XA1B	X3	X2	X1	[0001.1001.0101.0100.0011.1100.0010.0100]
XA1C	X3	X2	X1	[0001.1001.0101.0100.0010.1110.0001.0100]
XAI1	X1	X2	X3	[0001.1101.0011.0100.0011.0001.0010.0001]
XAI1A	X1	X2	X3	[0001.1101.0011.1000.0011.0001.0010.0001]
XNOR3	X3	X2	X1	[0001.1001.0011.0100.0010.0111.1001.0000]
XO1	X1	X2	X3	[0001.1101.0011.1000.0010.0011.0001.0001]
XO1A	X1	X2	X3	[0001.1101.0011.0100.0010.0011.0001.0001]
XOR3	X3	X2	X1	[0001.1001.0011.0100.0011.0101.1010.0000]
ZOR3	X2	X1	XC	[0011.0110.0010.1000.0010.0011.0101.0001]
ZOR3I	X3	X2	X1	[0001.1101.0011.1000.0011.0001.0001.0001]

TABLE A.2 – Configuration vectors for the flip-flops. For each Macro ID, it presents the adopted configuration vector, [SW31..SW0], and the corresponding input mapping, correlating the macro inputs (D – Data, CLK – Clock, E – Enable, and C/P – Clear or Preset) with the Versatile inputs (X3, X2, X1, and XC). The table has three parts, for 2-, 3-, and 4-input functions.

Macro ID	**D**	**CLK**	**E**	**C/P**	**Configuration Vector**
DFI0	X3	X2	–	–	[1110.1000.0010.0101.0000.1010.0010.1010]
DFI1	X3	X2	–	–	[1110.1000.0010.1001.0000.1010.0010.1010]
DFN0	X3	X2	–	–	[1110.1000.0010.0101.0000.1010.0001.1010]
DFN1	X3	X2	–	–	[1110.1000.0010.1001.0000.1010.0001.1010]
DFI0C0	X3	X2	–	X1	[1000.1001.0011.0101.0000.1010.0010.1010]
DFI0C1	X3	X2	–	X1	[1000.1010.0011.0101.0000.1010.0010.1010]
DFI0E0	X3	X2	X1	–	[1110.1000.1000.0101.1000.1010.0000.1010]
DFI0E1	X3	X2	X1	–	[1110.1000.0100.0101.1000.1010.0000.1010]
DFI0P0	X3	X2	–	X1	[1000.1001.0011.0101.0000.1010.0010.1010]
DFI0P1	X3	X2	–	X1	[1000.1010.0011.0101.0000.1010.0010.1010]
DFI1C0	X3	X2	–	X1	[1000.1001.0011.1001.0000.1010.0010.1010]
DFI1C1	X3	X2	–	X1	[1000.1010.0011.1001.0000.1010.0010.1010]
DFI1E0	X3	X2	X1	–	[1110.1000.1000.1001.1000.1010.0000.1010]
DFI1E1	X3	X2	X1	–	[1110.1000.0100.1001.1000.1010.0000.1010]
DFI1P0	X3	X2	–	X1	[1000.1001.0011.1001.0000.1010.0010.1010]
DFI1P1	X3	X2	–	X1	[1000.1010.0011.1001.0000.1010.0010.1010]
DFN0C0	X3	X2	–	X1	[1000.1001.0011.0101.0000.1010.0001.1010]
DFN0C1	X3	X2	–	X1	[1000.1010.0011.0101.0000.1010.0001.1010]
DFN0E0	X3	X2	X1	–	[1110.1000.1000.0101.1001.1000.0000.1010]
DFN0E1	X3	X2	X1	–	[1110.1000.0100.0101.1001.1000.0000.1010]
DFN0P0	X3	X2	–	X1	[1000.1001.0011.0101.0000.1010.0001.1010]
DFN0P1	X3	X2	–	X1	[1000.1010.0011.0101.0000.1010.0001.1010]
DFN1C0	X3	X2	–	X1	[1000.1001.0011.1001.0000.1010.0001.1010]
DFN1C1	X3	X2	–	X1	[1000.1010.0011.1001.0000.1010.0001.1010]
DFN1E0	X3	X2	X1	–	[1110.1000.1000.1001.1001.1000.0000.1010]
DFN1E1	X3	X2	X1	–	[1110.1000.0100.1001.1001.1000.0000.1010]
DFN1P0	X3	X2	–	X1	[1000.1001.0011.1001.0000.1010.0001.1010]
DFN1P1	X3	X2	–	X1	[1000.1010.0011.1001.0000.1010.0001.1010]
DFI0E0C0	X3	X2	X1	XC	[0111.0000.1000.0101.1000.1010.0000.1010]
DFI0E0C1	X3	X2	X1	XC	[1110.0000.1000.0101.1000.1010.0000.1010]
DFI0E0P0	X3	X2	X1	XC	[0111.0000.1000.0101.1000.1010.0000.1010]

Continue on next page

TABLE A.2 – continued from previous page

Macro ID	D	CLK	E	C/P	Configuration Vector
DFI0E0P1	X3	X2	X1	XC	[1110.0000.1000.0101.1000.1010.0000.1010]
DFI0E1C0	X3	X2	X1	XC	[0111.0000.0100.0101.1000.1010.0000.1010]
DFI0E1C1	X3	X2	X1	XC	[1110.0000.0100.0101.1000.1010.0000.1010]
DFI0E1P0	X3	X2	X1	XC	[0111.0000.0100.0101.1000.1010.0000.1010]
DFI0E1P1	X3	X2	X1	XC	[1110.0000.0100.0101.1000.1010.0000.1010]
DFI1E0C0	X3	X2	X1	XC	[0111.0000.1000.1001.1000.1010.0000.1010]
DFI1E0C1	X3	X2	X1	XC	[1110.0000.1000.1001.1000.1010.0000.1010]
DFI1E0P0	X3	X2	X1	XC	[0111.0000.1000.1001.1000.1010.0000.1010]
DFI1E0P1	X3	X2	X1	XC	[1110.0000.1000.1001.1000.1010.0000.1010]
DFI1E1C0	X3	X2	X1	XC	[0111.0000.0100.1001.1000.1010.0000.1010]
DFI1E1C1	X3	X2	X1	XC	[1110.0000.0100.1001.1000.1010.0000.1010]
DFI1E1P0	X3	X2	X1	XC	[0111.0000.0100.1001.1000.1010.0000.1010]
DFI1E1P1	X3	X2	X1	XC	[1110.0000.0100.1001.1000.1010.0000.1010]
DFN0E0C0	X3	X2	X1	XC	[0111.0000.1000.0101.1001.1000.0000.1010]
DFN0E0C1	X3	X2	X1	XC	[1110.0000.1000.0101.1001.1000.0000.1010]
DFN0E0P0	X3	X2	X1	XC	[0111.0000.1000.0101.1001.1000.0000.1010]
DFN0E0P1	X3	X2	X1	XC	[1110.0000.1000.0101.1001.1000.0000.1010]
DFN0E1C0	X3	X2	X1	XC	[0111.0000.0100.0101.1001.1000.0000.1010]
DFN0E1C1	X3	X2	X1	XC	[1110.0000.0100.0101.1001.1000.0000.1010]
DFN0E1P0	X3	X2	X1	XC	[0111.0000.0100.0101.1001.1000.0000.1010]
DFN0E1P1	X3	X2	X1	XC	[1110.0000.0100.0101.1001.1000.0000.1010]
DFN1E0C0	X3	X2	X1	XC	[0111.0000.1000.1001.1001.1000.0000.1010]
DFN1E0C1	X3	X2	X1	XC	[1110.0000.1000.1001.1001.1000.0000.1010]
DFN1E0P0	X3	X2	X1	XC	[0111.0000.1000.1001.1001.1000.0000.1010]
DFN1E0P1	X3	X2	X1	XC	[1110.0000.1000.1001.1001.1000.0000.1010]
DFN1E1C0	X3	X2	X1	XC	[0111.0000.0100.1001.1001.1000.0000.1010]
DFN1E1C1	X3	X2	X1	XC	[1110.0000.0100.1001.1001.1000.0000.1010]
DFN1E1P0	X3	X2	X1	XC	[0111.0000.0100.1001.1001.1000.0000.1010]
DFN1E1P1	X3	X2	X1	XC	[1110.0000.0100.1001.1001.1000.0000.1010]

TABLE A.3 – Configuration vectors for the latches. For each Macro ID, it presents the adopted configuration vector, [SW31..SW0], and the corresponding input mapping, correlating the macro inputs (D – Data, G – Gate, and C/P – Clear or Preset) with the Versatile inputs (X3, X2, X1, and XC). The table has two parts, for 2- and 3-input functions.

Macro ID	**D**	**G**	**C/P**	**Configuration Vector**
DLI0	X3	X2	–	[1110.1000.0010.0101.0001.0001.0001.1001]
DLI1	X3	X2	–	[1110.1000.0010.1001.0001.0001.0001.1001]
DLN0	X3	X2	–	[1110.1000.0010.0101.0000.0011.0010.1001]
DLN1	X3	X2	–	[1110.1000.0010.1001.0000.0011.0010.1001]
DLI0C0	X3	X2	X1	[1000.1001.0011.0101.0001.0001.0001.1001]
DLI0C1	X3	X2	X1	[1000.1010.0011.0101.0001.0001.0001.1001]
DLI0P0	X3	X2	X1	[1000.1001.0011.0101.0001.0001.0001.1001]
DLI0P1	X3	X2	X1	[1000.1010.0011.0101.0001.0001.0001.1001]
DLI1C0	X3	X2	X1	[1000.1001.0011.1001.0001.0001.0001.1001]
DLI1C1	X3	X2	X1	[1000.1010.0011.1001.0001.0001.0001.1001]
DLI1P0	X3	X2	X1	[1000.1001.0011.1001.0001.0001.0001.1001]
DLI1P1	X3	X2	X1	[1000.1010.0011.1001.0001.0001.0001.1001]
DLN0C0	X3	X2	X1	[1000.1001.0011.0101.0000.0011.0010.1001]
DLN0C1	X3	X2	X1	[1000.1010.0011.0101.0000.0011.0010.1001]
DLN0P0	X3	X2	X1	[1000.1001.0011.0101.0000.0011.0010.1001]
DLN0P1	X3	X2	X1	[1000.1010.0011.0101.0000.0011.0010.1001]
DLN1C0	X3	X2	X1	[1000.1001.0011.1001.0000.0011.0010.1001]
DLN1C1	X3	X2	X1	[1000.1010.0011.1001.0000.0011.0010.1001]
DLN1P0	X3	X2	X1	[1000.1001.0011.1001.0000.0011.0010.1001]
DLN1P1	X3	X2	X1	[1000.1010.0011.1001.0000.0011.0010.1001]

Appendix B - SET Susceptibilities of the VersaTile Functions

This Appendix presents the parameterised and the average SET susceptibilities for each function implemented with the VersaTile model and, for the sequential functions, the percentage of observed SEs that are SEUs. Table B.1 presents the SET susceptibilities for the combinational macros. Table B.2 and Table B.3 present the SET susceptibilities and the SEU percentage for the flip-flops, respectively. Finally, Table B.4 and Table B.5 present the SET susceptibilities and the SEU percentage for the latches, respectively.

TABLE B.1 – SET susceptibilities for the combinational macros. For each Macro, it presents the percentage SET susceptibilities for each input pattern and their average value. The table has three parts, for 1-, 2-, and 3-input functions, with [A], [A B], or [A B C/S] input patterns.

Macro ID	**Susceptibility for each Input Pattern**				**Average**
	[0]	**[1]**			
BUFD	23.171	19.512			21.341
BUFF	25.610	24.390			25.000
INV	25.610	29.268			27.439
INVD	14.634	12.195			13.415
	[00]	**[01]**	**[10]**	**[11]**	
AND2	7.317	10.976	14.634	20.732	13.415
AND2A	10.976	23.171	10.976	23.171	17.073
AND2B	20.732	10.976	23.171	10.976	16.463
NAND2	21.951	24.390	14.634	21.951	20.732
NAND2A	23.171	19.512	13.415	24.390	20.122

Continue on next page

TABLE B.1 – continued from previous page

Macro ID	**Susceptibility for each Input Pattern**								**Average**
NAND2B	19.512	25.610	24.390	13.415					20.732
NOR2	17.073	13.415	9.756	7.317					11.890
NOR2A	10.976	7.317	18.293	14.634					12.805
NOR2B	9.756	20.732	10.976	24.390					16.463
OR2	23.171	18.293	19.512	17.073					19.512
OR2A	24.390	21.951	21.951	17.073					21.341
OR2B	14.634	14.634	20.732	14.634					16.159
XNOR2	25.610	19.512	29.268	29.268					25.915
XOR2	26.829	29.268	25.610	21.951					25.915
	[000]	**[001]**	**[010]**	**[011]**	**[100]**	**[101]**	**[110]**	**[111]**	
AND3	4.878	7.317	4.878	9.756	4.878	13.415	10.976	23.171	9.909
AND3A	4.878	10.976	10.976	25.610	4.878	9.756	4.878	23.171	11.890
AND3B	10.976	23.171	4.878	10.976	4.878	23.171	4.878	9.756	11.585
AND3C	21.951	9.756	10.976	4.878	23.171	4.878	9.756	4.878	11.280
AO1	15.854	20.732	28.049	20.732	23.171	17.073	18.293	13.415	19.665
AO12	21.951	18.293	13.415	18.293	28.049	28.049	23.171	17.073	21.037
AO13	19.512	7.317	23.171	12.195	21.951	15.854	13.415	20.732	16.768
AO14	25.610	25.610	23.171	17.073	21.951	18.293	13.415	20.732	20.732
AO15	17.073	23.171	14.634	25.610	10.976	21.951	7.317	21.951	17.835
AO16	13.415	23.171	12.195	12.195	13.415	19.512	21.951	9.756	15.701
AO17	14.634	25.610	19.512	25.610	7.317	21.951	10.976	24.390	18.750
AO18	21.951	15.854	13.415	18.293	21.951	7.317	23.171	12.195	16.768
AO1A	23.171	17.073	15.854	13.415	15.854	20.732	28.049	20.732	19.360
AO1B	20.732	10.976	20.732	14.634	20.732	20.732	12.195	21.951	17.835
AO1C	20.732	20.732	12.195	19.512	20.732	10.976	20.732	14.634	17.530
AO1D	14.634	13.415	21.951	17.073	28.049	20.732	15.854	20.732	19.055
AO1E	12.195	18.293	20.732	19.512	20.732	14.634	20.732	10.976	17.226
AOI1	13.415	14.634	21.951	7.317	14.634	15.854	21.951	7.317	14.634
AOI1A	14.634	15.854	19.512	7.317	13.415	14.634	21.951	7.317	14.329
AOI1B	10.976	18.293	10.976	21.951	10.976	19.512	4.878	15.854	14.177
AOI1C	19.512	7.317	17.073	15.854	21.951	7.317	13.415	14.634	14.634
AOI1D	4.878	15.854	10.976	21.951	10.976	24.390	10.976	18.293	14.787
AOI5	15.854	19.512	19.512	17.073	28.049	15.854	17.073	15.854	18.598
AX1	19.512	15.854	24.390	24.390	15.854	14.634	9.756	21.951	18.293
AX1A	14.634	18.293	25.610	25.610	13.415	14.634	21.951	9.756	17.988

Continue on next page

TABLE B.1 – continued from previous page

Macro ID	**Susceptibility for each Input Pattern**								**Average**
AX1B	24.390	24.390	21.951	18.293	9.756	21.951	15.854	14.634	18.902
AX1C	15.854	14.634	9.756	21.951	19.512	15.854	26.829	26.829	18.902
AX1D	25.610	25.610	17.073	20.732	21.951	9.756	13.415	14.634	18.598
AX1E	13.415	14.634	21.951	9.756	14.634	18.293	28.049	28.049	18.598
AXO1	19.512	15.854	20.732	24.390	18.293	14.634	20.732	17.073	18.902
AXO2	18.293	14.634	20.732	17.073	19.512	15.854	20.732	26.829	19.207
AXO3	25.610	21.951	17.073	20.732	15.854	21.951	13.415	19.512	19.512
AXO5	13.415	17.073	15.854	21.951	14.634	18.293	28.049	21.951	18.902
AXO6	20.732	24.390	21.951	18.293	20.732	17.073	20.732	14.634	19.817
AXO7	15.854	21.951	13.415	19.512	28.049	21.951	17.073	20.732	19.817
AXOI1	20.732	10.976	17.073	23.171	14.634	15.854	12.195	21.951	17.073
AXOI2	20.732	9.756	12.195	19.512	14.634	17.073	17.073	23.171	16.768
AXOI3	20.732	10.976	17.073	23.171	14.634	10.976	12.195	20.732	16.311
AXOI4	18.293	24.390	21.951	12.195	12.195	21.951	14.634	12.195	17.226
AXOI5	12.195	21.951	12.195	12.195	18.293	24.390	21.951	12.195	16.921
AXOI7	12.195	10.976	12.195	20.732	20.732	10.976	17.073	23.171	16.006
MAJ3	7.317	19.512	10.976	21.951	14.634	20.732	20.732	13.415	16.159
MAJ3X	7.317	19.512	10.976	24.390	14.634	25.610	21.951	28.049	19.055
MAJ3XI	13.415	15.854	21.951	18.293	20.732	14.634	28.049	28.049	20.122
MIN3	13.415	15.854	21.951	15.854	20.732	12.195	21.951	7.317	16.159
MIN3X	24.390	18.293	26.829	15.854	23.171	12.195	21.951	7.317	18.750
MIN3XI	24.390	24.390	15.854	21.951	17.073	20.732	20.732	13.415	19.817
MX2	7.317	10.976	13.415	23.171	20.732	15.854	14.634	19.512	15.701
MX2A	18.293	13.415	12.195	20.732	7.317	13.415	13.415	21.951	15.091
MX2B	13.415	20.732	7.317	10.976	14.634	17.073	20.732	15.854	15.091
MX2C	12.195	18.293	18.293	13.415	13.415	19.512	7.317	13.415	14.482
NAND3	14.634	13.415	13.415	15.854	15.854	14.634	17.073	26.829	16.463
NAND3A	15.854	14.634	17.073	24.390	14.634	13.415	13.415	15.854	16.159
NAND3B	17.073	24.390	15.854	17.073	13.415	15.854	14.634	13.415	16.463
NAND3C	25.610	18.293	17.073	15.854	15.854	13.415	13.415	14.634	16.768
NOR3	20.732	14.634	13.415	7.317	21.951	8.537	7.317	7.317	12.652
NOR3A	9.756	4.878	7.317	4.878	19.512	9.756	13.415	4.878	9.299
NOR3B	7.317	4.878	9.756	4.878	13.415	4.878	21.951	9.756	9.604
NOR3C	7.317	9.756	9.756	20.732	7.317	10.976	17.073	25.610	13.567
OA1	4.878	23.171	10.976	19.512	10.976	20.732	10.976	18.293	14.939

Continue on next page

TABLE B.1 – continued from previous page

Macro ID	**Susceptibility for each Input Pattern**								**Average**
OA1A	10.976	21.951	10.976	18.293	4.878	15.854	10.976	21.951	14.482
OA1B	23.171	4.878	18.293	9.756	19.512	9.756	17.073	9.756	14.024
OA1C	21.951	7.317	13.415	14.634	21.951	7.317	17.073	15.854	14.939
OAI1	17.073	28.049	20.732	14.634	17.073	23.171	20.732	8.537	18.750
OR3	25.610	18.293	17.073	15.854	15.854	13.415	13.415	14.634	16.768
OR3A	15.854	13.415	13.415	14.634	28.049	18.293	17.073	15.854	17.073
OR3B	13.415	14.634	15.854	13.415	14.634	15.854	28.049	18.293	16.768
OR3C	14.634	13.415	13.415	15.854	15.854	14.634	17.073	26.829	16.463
XA1	4.878	12.195	10.976	25.610	10.976	20.732	4.878	15.854	13.262
XA1A	10.976	21.951	4.878	15.854	4.878	24.390	10.976	28.049	15.244
XA1B	18.293	9.756	26.829	7.317	17.073	15.854	21.951	7.317	15.549
XA1C	17.073	15.854	19.512	7.317	20.732	9.756	26.829	7.317	15.549
XAI1	13.415	20.732	15.854	28.049	13.415	14.634	13.415	21.951	17.683
XAI1A	15.854	28.049	13.415	23.171	13.415	21.951	13.415	17.073	18.293
XNOR3	23.171	14.634	25.610	25.610	26.829	20.732	26.829	18.293	22.713
XO1	25.610	15.854	23.171	13.415	21.951	13.415	17.073	15.854	18.293
XO1A	20.732	13.415	25.610	15.854	14.634	15.854	21.951	13.415	17.683
XOR3	24.390	18.293	26.829	18.293	23.171	14.634	28.049	28.049	22.713
ZOR3	20.732	9.756	12.195	19.512	14.634	10.976	13.415	24.390	15.701
ZOR3I	25.610	15.854	17.073	13.415	15.854	19.512	19.512	17.073	17.988
	Average susceptibility of all combinational macros in the library								17.182

TABLE B.2 – SET susceptibilities for the flip-flop. For each Macro, it presents the percentage SET susceptibilities for each input pattern and their average value. The table has three parts, for 2-, 3-, and 4-input functions, with [D], [D C/E/P], or [D E C/P] input patterns.

Macro ID	**Susceptibility for each Input Pattern**				**Average**
	[0]	**[1]**			
DFI0	18.488	15.611			17.049
DFI1	17.122	14.000			15.561
DFN0	15.513	18.487			17.000
DFN1	13.902	17.122			15.512
	[00]	**[01]**	**[10]**	**[11]**	
DFI0C0	8.293	19.707	4.878	15.610	12.122
DFI0C1	18.488	7.610	15.611	4.878	11.647
DFI0E0	18.585	18.488	15.806	15.561	17.110
DFI0E1	18.488	18.585	15.562	15.805	17.110
DFI0P0	8.537	17.268	11.951	18.049	13.951
DFI0P1	17.268	8.537	16.830	11.268	13.476
DFI1C0	8.049	18.341	4.878	14.000	11.317
DFI1C1	17.122	7.415	14.000	4.878	10.854
DFI1E0	17.220	17.122	14.195	13.951	15.622
DFI1E1	17.122	17.220	13.951	14.195	15.622
DFI1P0	8.537	15.902	11.707	16.439	13.146
DFI1P1	15.902	8.537	15.220	11.073	12.683
DFN0C0	4.878	15.512	8.293	19.707	12.098
DFN0C1	15.513	4.878	18.487	7.610	11.622
DFN0E0	15.709	15.561	18.584	18.488	17.085
DFN0E1	15.562	15.707	18.487	18.585	17.085
DFN0P0	11.951	17.951	8.537	17.268	13.927
DFN0P1	16.733	11.268	17.267	8.537	13.451
DFN1C0	4.878	13.902	8.049	18.341	11.293
DFN1C1	13.902	4.878	17.122	7.415	10.829
DFN1E0	14.098	13.951	17.220	17.122	15.598
DFN1E1	13.951	14.098	17.122	17.220	15.598
DFN1P0	11.707	16.341	8.537	15.902	13.122
DFN1P1	15.122	11.073	15.902	8.537	12.659

Continue on next page

TABLE B.2 – continued from previous page

Macro ID	**Susceptibility for each Input Pattern**								**Average**
	[000]	**[001]**	**[010]**	**[011]**	**[100]**	**[101]**	**[110]**	**[111]**	
DFI0E0C0	5.024	21.023	4.878	20.926	4.878	15.805	4.878	15.561	11.622
DFI0E0C1	18.585	5.024	18.487	4.878	15.805	4.878	15.561	4.878	11.012
DFI0E0P0	13.171	17.365	13.171	17.267	13.317	19.463	13.171	19.220	15.768
DFI0E0P1	17.366	13.171	17.267	13.171	17.024	13.317	16.780	13.171	15.158
DFI0E1C0	4.878	20.926	5.024	21.023	4.878	15.561	4.878	15.805	11.622
DFI0E1C1	18.488	4.878	18.584	5.024	15.561	4.878	15.805	4.878	11.012
DFI0E1P0	13.171	17.267	13.171	17.365	13.171	19.220	13.317	19.463	15.768
DFI0E1P1	17.268	13.171	17.365	13.171	16.780	13.171	17.024	13.317	15.158
DFI1E0C0	5.024	19.659	4.878	19.561	4.878	14.195	4.878	13.951	10.878
DFI1E0C1	17.220	5.024	17.122	4.878	14.195	4.878	13.951	4.878	10.268
DFI1E0P0	12.829	16.000	12.829	15.902	12.976	17.854	12.829	17.610	14.854
DFI1E0P1	16.000	12.829	15.902	12.829	15.415	12.976	15.171	12.829	14.244
DFI1E1C0	4.878	19.561	5.024	19.659	4.878	13.951	4.878	14.195	10.878
DFI1E1C1	17.122	4.878	17.220	5.024	13.951	4.878	14.195	4.878	10.268
DFI1E1P0	12.829	15.902	12.829	16.000	12.829	17.610	12.976	17.854	14.854
DFI1E1P1	15.902	12.829	16.000	12.829	15.171	12.829	15.415	12.976	14.244
DFN0E0C0	4.878	15.707	4.878	15.561	5.024	21.023	4.878	20.926	11.609
DFN0E0C1	15.709	4.878	15.561	4.878	18.584	5.024	18.487	4.878	11.000
DFN0E0P0	13.317	19.366	13.171	19.220	13.171	17.365	13.171	17.267	15.756
DFN0E0P1	16.928	13.317	16.780	13.171	17.365	13.171	17.267	13.171	15.146
DFN0E1C0	4.878	15.561	4.878	15.707	4.878	20.926	5.024	21.023	11.609
DFN0E1C1	15.562	4.878	15.707	4.878	18.487	4.878	18.584	5.024	11.000
DFN0E1P0	13.171	19.220	13.317	19.366	13.171	17.267	13.171	17.365	15.756
DFN0E1P1	16.782	13.171	16.927	13.317	17.267	13.171	17.365	13.171	15.146
DFN1E0C0	4.878	14.098	4.878	13.951	5.024	19.659	4.878	19.561	10.866
DFN1E0C1	14.098	4.878	13.951	4.878	17.220	5.024	17.122	4.878	10.256
DFN1E0P0	12.976	17.756	12.829	17.610	12.829	16.000	12.829	15.902	14.841
DFN1E0P1	15.317	12.976	15.171	12.829	16.000	12.829	15.902	12.829	14.232
DFN1E1C0	4.878	13.951	4.878	14.098	4.878	19.561	5.024	19.659	10.866
DFN1E1C1	13.951	4.878	14.098	4.878	17.122	4.878	17.220	5.024	10.256
DFN1E1P0	12.829	17.610	12.976	17.756	12.829	15.902	12.829	16.000	14.841
DFN1E1P1	15.171	12.829	15.317	12.976	15.902	12.829	16.000	12.829	14.232

TABLE B.3 – SEU susceptibilities for the flip-flop. For each Macro, it presents the percentage of the SET susceptibilities that produce a SEU for each input pattern and their average value. The table has three parts, for 2-, 3-, and 4-input functions, with [D], [D C/E/P], or [D E C/P] input patterns.

Macro ID	**Susceptibility for each Input Pattern**				**Average**
	[0]	**[1]**			
DFI0	64.6	71.6			68.1
DFI1	64.1	70.0			67.1
DFN0	71.4	64.6			68.0
DFN1	69.8	64.1			67.0
	[00]	**[01]**	**[10]**	**[11]**	
DFI0C0	0.0	63.6	0.0	71.6	33.8
DFI0C1	64.6	0.0	71.6	0.0	34.1
DFI0E0	64.8	67.5	71.9	73.4	69.4
DFI0E1	67.5	64.8	73.4	71.9	69.4
DFI0P0	0.0	65.8	0.0	68.4	33.5
DFI0P1	65.8	0.0	69.9	0.0	33.9
DFI1C0	0.0	63.3	0.0	70.0	33.3
DFI1C1	64.1	0.0	70.0	0.0	33.5
DFI1E0	64.3	67.2	70.4	72.0	68.5
DFI1E1	67.2	64.3	72.0	70.4	68.5
DFI1P0	0.0	65.0	0.0	67.4	33.1
DFI1P1	65.0	0.0	68.6	0.0	33.4
DFN0C0	0.0	71.4	0.0	63.6	33.7
DFN0C1	71.4	0.0	64.6	0.0	34.0
DFN0E0	71.7	73.4	64.8	67.5	69.4
DFN0E1	73.4	71.7	67.5	64.8	69.4
DFN0P0	0.0	68.2	0.0	65.8	33.5
DFN0P1	69.7	0.0	65.8	0.0	33.9
DFN1C0	0.0	69.8	0.0	63.3	33.3
DFN1C1	69.8	0.0	64.1	0.0	33.5
DFN1E0	70.2	72.0	64.3	67.2	68.5
DFN1E1	72.0	70.2	67.2	64.3	68.5
DFN1P0	0.0	67.2	0.0	65.0	33.0
DFN1P1	68.4	0.0	65.0	0.0	33.4

Continue on next page

TABLE B.3 – continued from previous page

Macro ID	**Susceptibility for each Input Pattern**								**Average**
	[000]	**[001]**	**[010]**	**[011]**	**[100]**	**[101]**	**[110]**	**[111]**	
DFI0E0C0	0.0	68.9	0.0	71.3	0.0	71.9	0.0	73.4	35.7
DFI0E0C1	64.8	0.0	67.5	0.0	71.9	0.0	73.4	0.0	34.7
DFI0E0P0	0.0	73.3	0.0	76.0	0.0	67.4	0.0	68.8	35.7
DFI0E0P1	66.0	0.0	68.6	0.0	70.2	0.0	71.8	0.0	34.6
DFI0E1C0	0.0	71.3	0.0	68.9	0.0	73.4	0.0	71.9	35.7
DFI0E1C1	67.5	0.0	64.8	0.0	73.4	0.0	71.9	0.0	34.7
DFI0E1P0	0.0	76.0	0.0	73.3	0.0	68.8	0.0	67.4	35.7
DFI0E1P1	68.6	0.0	66.0	0.0	71.8	0.0	70.2	0.0	34.6
DFI1E0C0	0.0	68.7	0.0	71.3	0.0	70.4	0.0	72.0	35.3
DFI1E0C1	64.3	0.0	67.2	0.0	70.4	0.0	72.0	0.0	34.3
DFI1E0P0	0.0	72.6	0.0	75.5	0.0	66.7	0.0	68.1	35.4
DFI1E0P1	65.2	0.0	68.1	0.0	69.0	0.0	70.7	0.0	34.1
DFI1E1C0	0.0	71.3	0.0	68.7	0.0	72.0	0.0	70.4	35.3
DFI1E1C1	67.2	0.0	64.3	0.0	72.0	0.0	70.4	0.0	34.3
DFI1E1P0	0.0	75.5	0.0	72.6	0.0	68.1	0.0	66.7	35.4
DFI1E1P1	68.1	0.0	65.2	0.0	70.7	0.0	69.0	0.0	34.1
DFN0E0C0	0.0	71.7	0.0	73.4	0.0	68.9	0.0	71.3	35.7
DFN0E0C1	71.7	0.0	73.4	0.0	64.8	0.0	67.5	0.0	34.7
DFN0E0P0	0.0	67.3	0.0	68.8	0.0	73.3	0.0	76.0	35.7
DFN0E0P1	70.0	0.0	71.8	0.0	66.0	0.0	68.6	0.0	34.6
DFN0E1C0	0.0	73.4	0.0	71.7	0.0	71.3	0.0	68.9	35.7
DFN0E1C1	73.4	0.0	71.7	0.0	67.5	0.0	64.8	0.0	34.7
DFN0E1P0	0.0	68.8	0.0	67.3	0.0	76.0	0.0	73.3	35.7
DFN0E1P1	71.8	0.0	70.0	0.0	68.6	0.0	66.0	0.0	34.6
DFN1E0C0	0.0	70.2	0.0	72.0	0.0	68.7	0.0	71.3	35.3
DFN1E0C1	70.2	0.0	72.0	0.0	64.3	0.0	67.2	0.0	34.2
DFN1E0P0	0.0	66.5	0.0	68.1	0.0	72.6	0.0	75.5	35.3
DFN1E0P1	68.8	0.0	70.7	0.0	65.2	0.0	68.1	0.0	34.1
DFN1E1C0	0.0	72.0	0.0	70.2	0.0	71.3	0.0	68.7	35.3
DFN1E1C1	72.0	0.0	70.2	0.0	67.2	0.0	64.3	0.0	34.2
DFN1E1P0	0.0	68.1	0.0	66.5	0.0	75.5	0.0	72.6	35.3
DFN1E1P1	70.7	0.0	68.8	0.0	68.1	0.0	65.2	0.0	34.1

TABLE B.4 – SET susceptibilities for the latches. For each Macro, it presents the percentage SET susceptibilities for each input pattern and their average value. The table has two parts, for 2-, and 3-input functions, with [D] or [D C/P] input patterns.

Macro ID	**Susceptibility**				**Average**
	[0]	**[1]**			
DLI0	17.024	11.220			14.122
DLI1	15.756	9.902			12.829
DLN0	11.220	18.390			14.805
DLN1	9.902	17.024			13.463
	[00]	**[01]**	**[10]**	**[11]**	
DLI0C0	8.293	18.244	4.878	11.220	10.659
DLI0C1	17.024	7.610	11.220	4.878	10.183
DLI0P0	8.537	15.805	11.951	13.659	12.488
DLI0P1	15.805	8.537	12.439	11.268	12.012
DLI1C0	8.049	16.976	4.878	9.902	9.951
DLI1C1	15.756	7.415	9.902	4.878	9.488
DLI1P0	8.537	14.537	11.707	12.341	11.780
DLI1P1	14.537	8.537	11.122	11.073	11.317
DLN0C0	4.878	11.220	8.293	19.610	11.000
DLN0C1	11.220	4.878	18.390	7.610	10.524
DLN0P0	11.951	13.659	8.537	17.171	12.829
DLN0P1	12.439	11.268	17.171	8.537	12.354
DLN1C0	4.878	9.902	8.049	18.244	10.268
DLN1C1	9.902	4.878	17.024	7.415	9.805
DLN1P0	11.707	12.341	8.537	15.805	12.098
DLN1P1	11.122	11.073	15.805	8.537	11.634

TABLE B.5 – SEU susceptibilities for the latches. For each Macro, it presents the percentage of the SET susceptibilities that produce a SEU for each input pattern and their average value. The table has two parts, for 2-, and 3-input functions, with [D] or [D C/P] input patterns.

Macro ID	**Susceptibility**				**Average**
	[0]	**[1]**			
DLI0	49.6	47.8			48.7
DLI1	46.1	39.9			43.0
DLN0	47.8	46.4			47.1
DLN1	39.9	43.3			41.6
	[00]	**[01]**	**[10]**	**[11]**	
DLI0C0	0.0	49.5	0.0	47.8	24.3
DLI0C1	49.6	0.0	47.8	0.0	24.3
DLI0P0	0.0	49.7	0.0	47.9	24.4
DLI0P1	49.7	0.0	47.8	0.0	24.4
DLI1C0	0.0	46.6	0.0	39.9	21.6
DLI1C1	46.1	0.0	39.9	0.0	21.5
DLI1P0	0.0	45.6	0.0	42.3	22.0
DLI1P1	45.6	0.0	41.2	0.0	21.7
DLN0C0	0.0	47.8	0.0	46.5	23.6
DLN0C1	47.8	0.0	46.4	0.0	23.6
DLN0P0	0.0	47.9	0.0	46.3	23.5
DLN0P1	47.8	0.0	46.3	0.0	23.5
DLN1C0	0.0	39.9	0.0	43.9	20.9
DLN1C1	39.9	0.0	43.3	0.0	20.8
DLN1P0	0.0	42.3	0.0	42.6	21.2
DLN1P1	41.2	0.0	42.6	0.0	21.0

Appendix C - Self-Produced Transients for SET Emulation

The autonomous emulation approach described in Section 6.3 makes use of self-produced transient pulses for SET emulation. In this appendix we describe the analysis that supports this use, that were proposed in Armelin *et al.* (2018b). First, in the Section C.1, we present the context in which this analysis is inserted, briefly describing the related works. In the Section C.2, we describe how we evaluated the use of these self-produced transient pulses. Finally, in the Section C.3, we present the results of this evaluation.

C.1 Context

The SET propagation studies dedicated to analyse the its broadening and quenching effects rely on the injection of pulses in some combinational circuits. These pulses can be simulated, as in Wirth *et al.* (2008), induced by laser radiation, as in Ferlet-Cavrois *et al.* (2008), induced by particle radiation, as in Rezgui *et al.* (2007), Ferlet-Cavrois *et al.* (2008), Evans *et al.* (2014), or generated in the device itself, as in Sterpone *et al.* (2010).

On the other hand, for SER and SEV analyses, the SET is also usually simulated or induced by laser or particle irradiation, but it is not emulated as a pulse generated inside the device. The main reasons are the difficulty in control when the SET is injected inside the clock cycle of a sequential circuit, and the difference of the electrical interaction of

the pulse in the emulated circuit when compared to the real one.

To overcome these problems, some techniques were developed to control when the SET occurs along a clock cycle (GARCÍA-VALDERAS *et al.*, 2009), and to consider the electrical interaction with the real device (ENTRENA *et al.*, 2009). These techniques are based on shift-registers, to emulate subcycles of the clock cycle (see Fig. C.1), and non-linear counters, to emulate the electrical interaction (see Fig. C.2). Both cases provide good control of the emulated SET. However, the use of these techniques for SEV estimation slows down the emulation process.

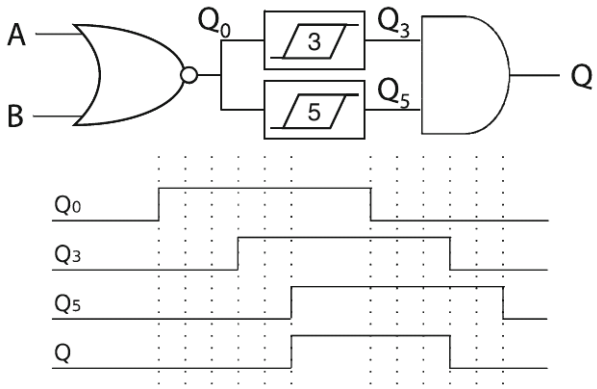

FIGURE C.1 – SET quantised delay model. In this case, two shift-registers, of three and five stages, are used to delay the propagation of a signal in three and five quantised delays (period of the clock used to generate the sub-cycles). Source: reproduced from García-Valderas *et al.* (2009).

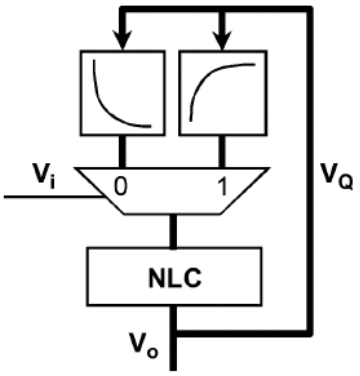

FIGURE C.2 – SET emulation considering electrical masking effects. Depending on the value of the the input Vi, the non-linear counter (NLC) receives its next value from a different table (emulating rising or falling interaction). Counting stops when upper or lower limits are reached, however, Vo receives the value of the most significant bit of NLC at all times. Source: adapted from Entrena *et al.* (2009).

In this context, this research topic investigates the use of an FPGA to self-produce transients to emulate SETs for SER estimation. The goal is to evaluate if the produced transients have the characteristics needed for SEV estimation: uncorrelated with the system clock and distributed with low distortion. The results indicate that is possible to adopt this approach when the target device is the same FPGA. The analysed Saboteurs, which are needed to inject the SET into the circuit nodes, have low interference over the pulse widths. In this analysis, we used the same device described in Section 4.1 (A3PE1500-PQ208, from the ProASIC3E family), and the same development kit describe in Section 6.3 (ProASIC3/E Starter Kit).

C.2 Evaluation Process

The process for evaluating the SET emulation is based on applying self-produced transient pulses to some Saboteur Candidates and verifying how they are broadened or quenched by these Saboteurs.

Unfortunately, it is not possible to directly measure the pulses widths. For this reason, an indirect technique is applied to measure them. A D-flip-flop is used as a pulse Monitor, with pulses being applied to its D input, and a Counter registers how many pulses were captured by this Monitor. The ratio between captured and applied pulses is assumed to be approximately equivalent to the ratio between the pulse width and the period of the clock applied to the Monitor, and the results show this assumption is reasonable. For example, if 10% of the applied pulses are captured, and the clock is 40 MHz, the pulse width is approximately 2.5 ns.

C.2.1 Evaluation Environment

The environment for the evaluation of the proposed SET emulation approach is composed by a Pulse Generator, the Saboteurs Candidates, with their respective SET Monitors (D-flip-flops) and SET Counters, and a Reporter.

The Pulse Generator produces transient electrical pulses with the width controlled by a delay element. These pulses may occur at any moment of a System Clock cycle, and they are applied to the Saboteurs Candidates. Each SET Monitor observes the output of a Saboteur Candidate and registers the occurrence of a SET. This occurrence increments the respective SET Counter. At the end of an evaluation test cycle, the Reporter sends the SET Counters values through a serial interface.

One SET Monitor, and respective SET Counter, is connected directly to the transient pulses source to provide a reference to characterise the pulse without the interference of any Saboteur.

The transient electrical pulse is generated inside the FPGA using a D-flip-flop (macro DFN1E1C1) and a Clock Delay block (macro CLKDLY), as illustrated in Fig. C.3. If the flip-flop is enabled, the pulse starts with the rising edge of the transient clock and ends with the rising edge of the delayed transient clock.

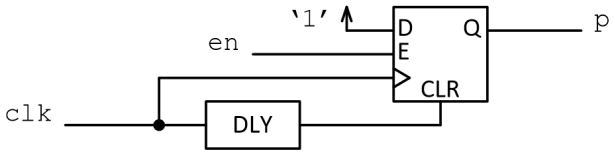

FIGURE C.3 – Electrical pulse generation circuit. When the flip-flop is enabled, the pulse starts at the rising edge of the clock and ends at the rising edge of the delayed clock.

The Clock Delay block statically controls the transient electrical pulse width. The CLKDLY macro has a 5-bit input for static configuration of the delay value. In this analysis, all the 32 possible delay values are used. This delay block may be substituted by a PLL block, enabling dynamic control of the pulse width.

To guarantee that the pulse may occur at any moment in the system clock cycle (used by the SET Monitors, Counters, and Reporter), a distinct clock source is used to generate the pulse, with a slightly different frequency. We adopted the closest frequency that our external source could provide: The kit has a 40 MHz source and the external source were adjusted to 39.9 MHz (approximately 0.25% of difference).

C.2.2 Analysed Characteristics

Two possible sources of pulse distortion were analysed: the saboteurs and the pulse distribution. To evaluate the distortion caused by the Saboteur, six distinct Saboteur Candidates were selected, as illustrated in Fig. C.4. All candidates have a similar behavior: they invert the logic value from `in` to `out`, when enabled by `en` and a pulse `p` occurs. The differences reside in the `en` and `p` active logic values.

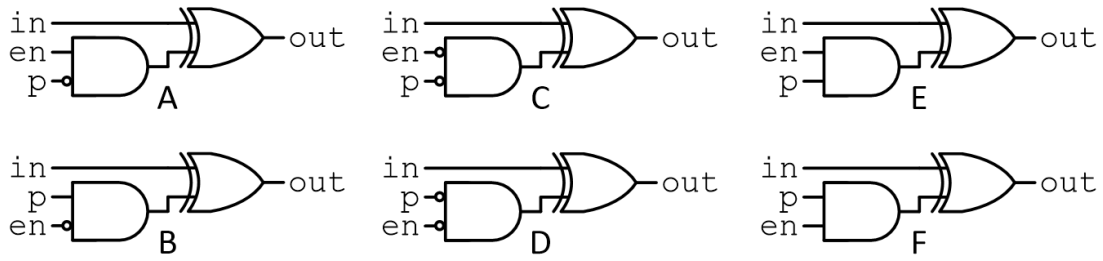

FIGURE C.4 – Saboteur Candidates. The candidates A and B are instances of the macro AX1, C and D are instances of the macro AX1B, while E and F are instances of the macro AX1C. Although C and D have the same logic function, we tested it with the signals `p` and `en` assigned to inverted inputs, to check if there is some electrical difference. The same occurs for E and F.

Additionally, each Saboteur Candidate was evaluated with its input `in` set to '0' and to '1', to verify the broadening and quenching effects. The Saboteurs configurations are labeled with the combination of the letter presented in Fig. C.4 and the logic value set at their input, as A0, A1, B0, and so on.

Some Saboteurs Candidates requires a low-level pulse, instead of a high-level pulse. For these cases, the D-flip-flop used to generate the pulse (Fig. C.3) has an inverted output: DFI1E1C1 replaces the macro DFN1E1C1.

Another possible source of distortion is how the pulse is distributed inside the device. To evaluate it, the same Saboteur Candidate (A0 for the low-level pulse, B0 for high) is used in different locations, with their respective SET Monitors and Counters. Additionally, the pulse was distributed using a local and a global net, for comparison.

C.2.3 SET Emulations

For the evaluation of the Saboteur Candidates, two different setups were used: one with the high-level pulse and the Saboteurs B0, B1, E0, E1, F0, and F1; and another one with the low-level pulse and the Saboteurs A0, A1, C0, C1, D0, and D1. In both cases, the pulse was distributed via the global net. Each of these setups was configured with the all the 32 possible delay values.

For the Pulse Distribution evaluation, it was also used two different setups for the pulse level, with six instances of the same Saboteur Candidate (A0 for the setup with low-level pulse, and B0 for the setup with high-level pulse), and only one delay value (10000_b). Each of these two setups was replicated to use local and global nets to distribute the pulse.

For all evaluation test cycles, the system clock was 40 MHz (from the evaluation kit), and the external clock for generating the transient pulse was 39.9 MHz. Finally, each configuration was submitted to 10 evaluation test cycles, each of them with 100,000,000 generated pulses. The results are presented in the next section.

C.3 Evaluation Test Cycles

These results are organised in (i) external acquisitions of the generated pulse, to verify that their widths vary with the static control; (ii) verification of the dissociation of the generated pulses with respect to the system clock; (iii) generated pulse widths and the distortions caused by the saboteurs candidates; and (iv) the pulse distortions caused by the pulse distribution inside the device.

The generated transient pulse was connected to an output of the device for external observation. This external pulse is not equivalent to the internal one, due to the output driver distortion and the impedance mismatch of the probe, but it gives an idea of the transient curve and how it changes with the delay variation. The Fig. C.5 presents five external pulse acquisitions, for the delay set to 01010_b, 01110_b, 10100_b, 11000_b and 11110_b.

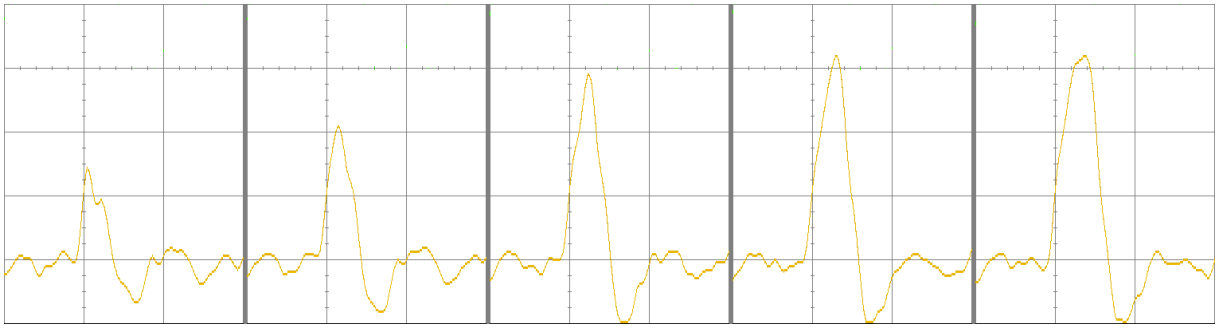

FIGURE C.5 – Transient electrical pulses observed at an output of the FPGA. From left to right, acquisitions for the delay set to 01010_b, 01110_b, 10100_b, 11000_b and 11110_b. The amplitude axis scale is 1 V per division, and the time axis scale is 10 ns per division.

The dissociation between the generated pulses and the system clock was verified by observing the external pulse and the clock. Fig. C.6 shows the relation between the clock and the pulse after 1, 5, 10 and 20 accumulated captures.

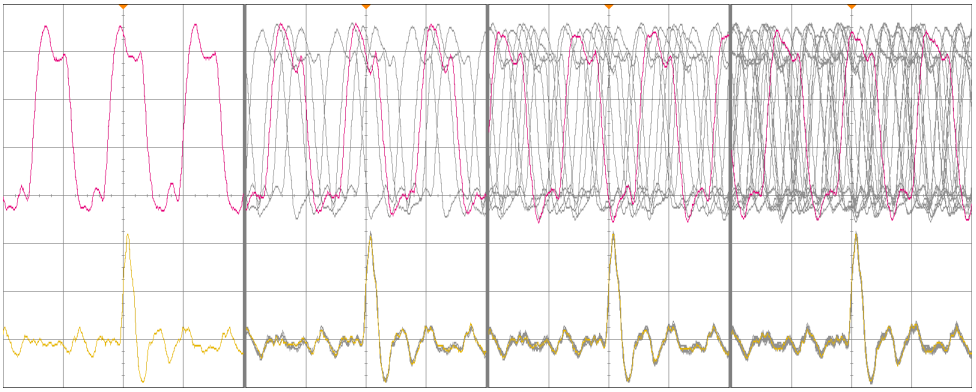

FIGURE C.6 – Dissociation between pulses and system clock. From left to right, acquired images after 1, 5, 10 and 20 accumulated captures of the system clock (top) and generated pulse (bottom). The amplitude axis scale is 1 V per division, and the time axis scale is 20 ns per division.

The resulting pulse widths for the Saboteur Candidates are reported in Table C.1. The column Delay presents the binary value configured in the clock delay block. Its value starts from 00111_b because no pulse was generated with smaller values. The column Ref0 presents the width of the generated low-level pulse, monitored without the saboteur interference, while Ref1 presents the width for the high-level pulse. As explained in Section C.2.3, Ref0 is associated with the Saboteur Candidates A, C and D, while the Ref1 is associated with B, E, and F.

TABLE C.1 – Resulting pulse widths for SET emulation. For each configured Delay (the binary value used to configure CLKDLY), it presents the estimated reference pulse widths (Ref0 for low-level and Ref1 for high-level pulse) and the pulse widths estimated after each of the twelve saboteurs configurations. All width values are in ns.

Delay	**Ref0**	**Ref1**	**A0**	**A1**	**B0**	**B1**	**C0**	**C1**	**D0**	**D1**	**E0**	**E1**	**F0**	**F1**
00111	–	0.57	–	–	0.47	0.68	–	–	–	–	0.50	0.66	0.49	0.62
01000	0.39	1.18	0.47	0.59	1.15	1.33	0.43	0.62	0.38	0.63	1.15	1.32	1.15	1.27
01001	0.59	1.37	0.69	0.80	1.34	1.51	0.65	0.83	0.63	0.85	1.34	1.51	1.34	1.46
01010	0.78	1.55	0.88	0.99	1.52	1.69	0.84	1.02	0.83	1.04	1.52	1.68	1.52	1.64
01011	0.99	1.76	1.10	1.21	1.72	1.90	1.06	1.24	1.05	1.25	1.72	1.89	1.72	1.84
01100	1.20	1.98	1.31	1.42	1.94	2.11	1.27	1.45	1.26	1.47	1.94	2.10	1.94	2.06
01101	1.40	2.18	1.50	1.61	2.13	2.30	1.46	1.64	1.45	1.66	2.13	2.29	2.12	2.24
01110	1.61	2.37	1.70	1.81	2.31	2.48	1.67	1.84	1.65	1.86	2.31	2.47	2.31	2.43
01111	1.82	2.58	1.90	2.00	2.51	2.69	1.87	2.03	1.86	2.05	2.51	2.68	2.51	2.64
10000	2.10	2.83	2.16	2.25	2.78	2.96	2.13	2.28	2.12	2.29	2.78	2.95	2.78	2.90
10001	2.23	2.98	2.31	2.42	2.95	3.12	2.27	2.44	2.26	2.46	2.95	3.11	2.95	3.07
10010	2.36	3.13	2.43	2.54	3.10	3.27	2.39	2.57	2.38	2.58	3.11	3.24	3.10	3.21
10011	2.57	3.33	2.63	2.73	3.28	3.44	2.61	2.76	2.59	2.77	3.28	3.44	3.28	3.39
10100	2.76	3.52	2.84	2.93	3.45	3.61	2.80	2.96	2.79	2.97	3.46	3.60	3.46	3.56
10101	2.94	3.72	2.99	3.09	3.63	3.78	2.95	3.12	2.94	3.14	3.64	3.77	3.63	3.73
10110	3.12	3.91	3.17	3.27	3.80	3.96	3.13	3.30	3.12	3.32	3.81	3.95	3.81	3.91
10111	3.31	3.97	3.36	3.47	3.89	4.05	3.32	3.50	3.31	3.52	3.89	4.02	3.89	4.00
11000	3.59	4.37	3.67	3.77	4.26	4.42	3.63	3.80	3.62	3.82	4.27	4.41	4.27	4.37
11001	3.75	4.52	3.84	3.94	4.42	4.59	3.80	3.07	3.79	3.98	4.43	4.58	4.42	4.54
11010	3.91	4.67	3.99	4.10	4.58	4.76	3.96	4.13	3.95	4.14	4.58	4.76	4.58	4.71
11011	4.11	4.86	4.19	4.30	4.79	4.97	4.15	4.33	4.14	4.35	4.79	4.97	4.79	4.92
11100	4.33	5.05	4.41	4.52	5.02	5.21	4.37	4.56	4.36	4.58	5.02	5.20	5.02	5.15
11101	4.52	5.24	4.61	4.72	5.22	5.41	4.56	4.76	4.55	4.77	5.23	5.41	5.22	5.36
11110	4.71	5.45	4.81	4.92	5.44	5.63	4.76	4.96	4.75	4.97	5.44	5.63	5.43	5.58
11111	4.91	5.67	5.01	5.13	5.67	5.87	4.97	5.17	4.96	5.19	5.67	5.86	5.66	5.81

The proposed approach for SET emulation covers a good range of pulse widths, from approximately 500 ps to more than 5 ns, with small pulse distortions. This range is in agreement with that presented in Rezgui *et al.* (2007), which were characterised by radiation tests, for the same device technology.

In general, the pulse distortions caused by the Saboteurs are small. The important parameter is the unbalanced distortion regarding the original values to be inverted by the

Saboteur. The Saboteur Candidate that presented the smaller distortion was the A, with the mean value for the absolute difference between A1 and A0 of 0.08 ns. On the other hand, the worst case was the Saboteur D, with the mean value of 0.16 ns.

Fig. C.7 presents the results for the pulse distribution analysis. The labels 'a' to 'f' represent the six instances of the same Saboteur. The use of the global net led to a smaller dispersion in the observed pulses than using a local net. Regardless, in both cases, the mean values observed in the six instances of the Saboteur are very similar.

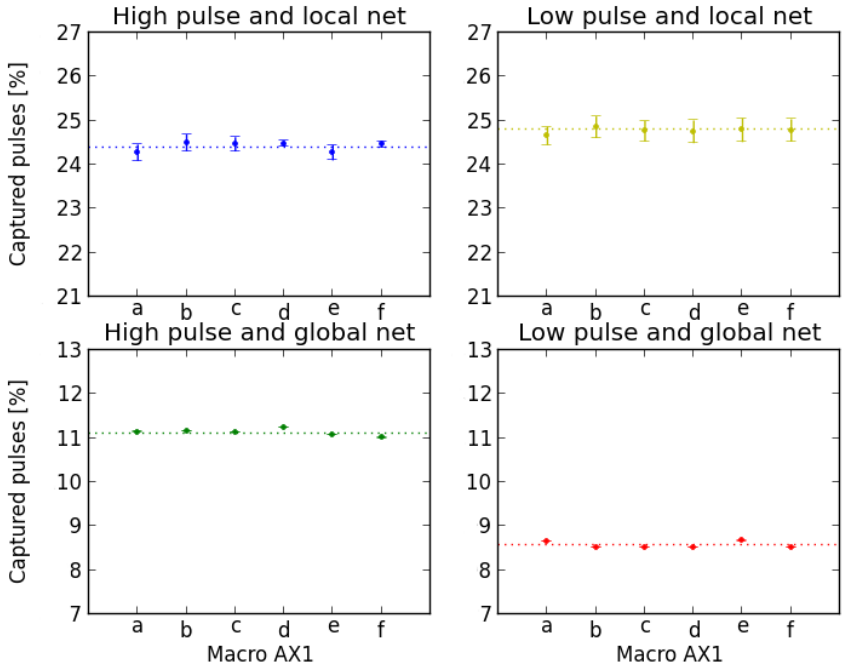

FIGURE C.7 – Comparison between local and global net for pulse distribution. The y-axis presents the percentage of pulses that were captured, and the x-axis presents the six instances ('a' to 'f') of the macro AX1. The top row presents the results for using an ordinary local net to distribute the pulse, while the bottom row presents the results for using a global net. The left column is for the high-level pulses and the right column for the low-level.

In summary, this analysis showed that seems feasible to use an FPGA self-produced transient for SET emulation. The analysed Saboteurs have small pulse distortion, and the pulse can be disassociated from the system clock. Nonetheless, this approach is restricted to the cases in which the device used for the SEV estimation is from the same type of the device targeted for the real application.

Appendix D - Résumé Français

D.1 Introduction

Dans les systèmes numériques, une erreur 'soft' (SE – Soft Error) est une valeur erronée dans un registre de signal ou de données qui n'est pas permanente, c'est-à-dire que le système peut restaurer son état opérationnel après le traitement de la SE. Au contraire, une erreur 'hard' est un effet permanent, généralement causé par des dommages, mais peut également être une erreur de conception, par exemple.

Les causes de SE sont nombreuses, notamment la réduction de tension, le vieillissement, la diaphonie, les interférences électromagnétiques, les effets de *hazard* et les effets de rayonnements. L'occurrence d'un SE dépend généralement de certaines caractéristiques du dispositif et de certains facteurs externes, tels que l'environnement et le fonctionnement du dispositif.

Par exemple, les SE produits en raison de la réduction de la tension dépendent de la température de fonctionnement du dispositif. Le vieillissement entraîne un changement permanent dans les caractéristiques du dispositif, ce qui peut entraîner des problèmes de synchronisation pour certains scénarios opérationnels et certaines conditions environnementales, générant des SE. Si l'erreur due aux effets de vieillissement se produit indépendamment de l'environnement et du fonctionnement, elle devient une erreur 'hard'.

De même, la diaphonie dépend de la susceptibilité du circuit et des caractéristiques du signal qui induit le SE entre les lignes du circuit. Les interférences électromagnétiques dépendent des caractéristiques du signal électromagnétique et de la susceptibilité

du circuit.

Les effets de *hazard* sont une source de préoccupation importante pour les circuits asynchrones, dans lesquels ils peuvent se produire du fait de la mise en œuvre du circuit et de la séquence des modifications du signal, comme dans les cas précédents.

Les effets de rayonnement peuvent induire des SE en raison de l'ionisation du dispositif électronique. Ainsi, ils dépendent également de la susceptibilité du circuit et des caractéristiques de l'environnement de rayonnement.

Bien que la définition adoptée de SE couvre toutes les causes, dans de nombreux travaux, le terme Soft Error est exclusivement associé aux effets des rayonnements. Ce travail se concentre également sur les SE causées par les radiations. Plus précisément, les SE directement produits par les effets à événement unique (SEE – Single Event Effect), c'est-à-dire la classe d'effet des rayonnements ionisants générés par une seule particule d'ionisation. De manière différente, la dose totale ionisante (TID – Total Ionising Dose) est une autre classe d'effet ionisant qui peut ressembler à l'effet de vieillissement et peut indirectement causer des SE, mais qui sort du cadre de ce travail.

La première SEE d'intérêt est le transitoire d'événement unique (SET – Single-Event Transient), c'est-à-dire un changement temporaire de la valeur logique d'un nœud, provoqué par une excursion sur sa tension. Cette excursion de tension résulte de l'impulsion de courant transitoire produite par la recombinaison des ions induits par la particule d'ionisation.

Le SET est généré dans la zone sensible des transistors qui composent les éléments logiques et peut les affecter de différentes manières. Pour les éléments combinatoires (portes logiques), le SET peut se propager à la sortie ou peut être filtré en interne. Pour les éléments séquentiels (registres – flip-flop et latch), le SET peut également se propager vers la sortie ou être filtré, mais il ne s'agit que d'effets mineurs. Pour eux, l'effet principal est le SEU (Single-Event Upset), c'est-à-dire le changement de la valeur stockée, qui est valable jusqu'au prochain cycle d'écriture. De même, pour les cellules de mémoire, le SEU est également la principale préoccupation.

L'évaluation de la fiabilité des systèmes numériques adopte le taux d'erreur 'soft' (SER – Soft-Error Rate) comme métrique associée aux SE. Ce paramètre est intrinsèquement lié au taux d'impulsions électriques transitoires R générées dans les transistors du circuit. Cependant, le SER du circuit n'est pas la somme, ou la combinaison, de la valeur R de chaque transistor. Les effets de masquage logique, électrique et temporel filtrent de nombreuses impulsions transitoires. Il est difficile d'obtenir une formule mathématique pour le SER, basée sur les caractéristiques de rayonnement et de circuit, incluant également ces effets de masquage. Cette difficulté a conduit au développement de nombreuses méthodes d'estimation de SER.

Les méthodes d'estimation du SER peuvent, entre autres, adopter des approches d'analyse, de simulation, d'émulation et de test de rayonnement. Cependant, à l'exception des tests de rayonnement, ils ne traitent pas directement avec le SER, car il nécessite le flux de particules ionisantes. Au lieu de cela, ils traitent de la probabilité d'apparition d'une SE, en adoptant des termes comme sensibilité, susceptibilité ou vulnérabilité aux SE. Dans ce travail, nous adoptons deux termes. Pour la probabilité d'apparition de SE à la sortie des composants de niveau de porte dus aux SET générés dans leurs transistors internes, nous utilisons le terme susceptibilité SET. Pour la probabilité d'apparition de SE à la sortie du circuit, nous adoptons le terme vulnérabilité aux erreurs 'soft' (SEV – Soft-Error Vulnerability).

Nous différencions ces deux probabilités parce que nous nous concentrons sur l'influence des susceptibilités SET lors de l'évaluation du SEV au niveau de la porte. Cette influence est prise en compte par certaines classes de méthodes d'estimation (méthode analytique et simulation électrique) mais ignorée par d'autres (simulation logique et émulation).

Dans ce contexte, nous proposons une stratégie d'estimation SEV qui prend en compte les susceptibilités SET des éléments logiques du circuit. Cette approche est destinée à être utilisée avec une approche de simulation logique, mais peut être adaptée pour être utilisée avec des plateformes d'émulation. Le processus d'injection SET suit une distribution pondérée, pour laquelle les poids sont les susceptibilités SET des éléments logiques.

Cette approche prenant en compte les probabilités nécessitant l'accès à la sortie des éléments logiques, la simulation nécessite l'utilisation du fichier annoté post-synthèse, c'est-à-dire une netlist contenant les éléments logiques de la technologie cible. Cette approche peut être appliquée à la fois aux circuits intégrés d'application spécifique (ASIC – Application Specific Integrated Circuit) et aux matrices de portes programmables par l'utilisateur (FPGA – Field Programmable Gate Array). Cependant, dans ce travail, nous n'avons ciblé qu'une famille de FPGA spécifique. Pour cette raison, ci-après, nous appelons généralement les éléments logiques directement les CLB (Configuration Logic Block) d'un FPGA.

L'adoption de la méthode d'estimation SEV proposée nécessite les susceptibilités SET pour la technologie cible. Comme ces informations ne sont pas disponibles, nous avons estimé les susceptibilités SET des CLB de la technologie sélectionnée.

Pour évaluer l'approche proposée, nous avons estimé la SEV pour un ensemble de circuits de référence. Enfin, nous explorons l'adoption de la stratégie proposée avec une plateforme d'émulation.

D.2 Concepts et revue de littérature

La stratégie proposée pour l'estimation SEV des circuits numériques en tenant compte des susceptibilités SET de leurs portes constitue une contribution incrémentielle aux efforts de la communauté scientifique pour analyser et prévoir correctement le comportement des dispositifs électroniques dans un environnement irradiant.

Dans cette section, nous présentons les concepts et la revue de la littérature concernant les principaux sujets d'intérêt de ce travail.

Effets des radiations sur les dispositifs électroniques

Les effets du rayonnement sur les dispositifs électroniques constituent un domaine de connaissances complexe comprenant les sources du rayonnement, leur interaction avec les

semi-conducteurs et les effets de cette interaction sur le comportement de ces dispositifs.

L'environnement de rayonnement est un mélange hétérogène de nombreux types de rayonnements, avec une large gamme d'énergie et de flux, provenant de différentes sources. Pour cette raison, chaque application envisagée, par exemple l'espace, l'aéronautique, les opérations au sol et les installations radioactives, possède un environnement de rayonnement associé spécifique.

Le rayonnement peut être divisé en cinq types : électromagnétique (photons – rayons X et rayons gamma) ; particules chargées légers (électrons et positrons) ; particules chargées lourdes (protons, particules alpha, fragments de fission et autres ions) ; particules neutres (neutrons) ; et des particules élémentaires (résultant de l'interaction du rayonnement cosmique avec l'atmosphère).

L'interaction du rayonnement avec la matière peut être divisée en deux classes, en fonction des caractéristiques du rayonnement : les particules chargées et les particules non chargées. Les particules chargées, sont subdivisées en particules lourdes et légères. Les particules non chargées, sont subdivisées en électromagnétiques (photons - considérés comme des particules d'énergie et de quantité de mouvement) et en neutrons.

Il existe trois classes d'effets de rayonnement sur les dispositifs électroniques : les dommages de déplacement (DD – Displacement Damage), la dose totale ionisante et l'effet à événement unique. Les DD et TID sont des effets cumulatifs, ce qui signifie que les effets ne seront perceptibles qu'après l'interaction de nombreuses particules de rayonnement avec la matière, tandis que le SEE est un effet singulier, provoqué par l'interaction d'une seule particule de rayonnement.

Le SEE est la classe des effets de rayonnement d'intérêt dans ce travail. Le SEE principal est le transitoire à événement unique, c'est-à-dire une excursion de tension dans un nœud de circuit à l'intérieur du dispositif. Lorsque le rayonnement ionisant traverse une jonction polarisée en inverse, le champ électrique induit un courant transitoire avec les charges libres produites. Un autre SEE intéressant pour ce travail est le SEU déjà mentionné, qui est un changement de la valeur logique stockée d'un élément de mémoire

(cellule de mémoire, flip-flop ou latch).

Les SEE sont divisés en erreurs 'soft' et 'hard'. Les SE sont les erreurs récupérables sur les signaux ou les registres de données, tandis que les 'hard' erreur sont un changement irréversible du fonctionnement du dispositif.

Le SER est un paramètre essentiel pour évaluer correctement la fiabilité des systèmes électroniques fonctionnant sous les effets des particules de rayonnement. Ce paramètre est intrinsèquement lié au taux d'impulsions électriques transitoires, R, générées dans les éléments internes sensibles du dispositif,

$$R = \int_0^\infty \phi(E) \cdot \sigma(E) \cdot dE, \tag{D.1}$$

où ϕ est le flux de particules, σ est la section transversale de l'élément et E est l'énergie des particules.

Dans les systèmes numériques, ces éléments sensibles sont les transistors de commutation, pour lesquels la section transversale est directement liée à leurs zones de drain.

Cependant, le SER du dispositif n'est pas la somme, ou la combinaison, de la valeur R de chaque transistor. Les effets de masquage logiques, temporels et électriques, décrits dans la section suivante, éliminent de nombreuses impulsions transitoires.

Facteurs de masquage d'erreur

Les facteurs de masquage d'erreur éliminent certains des SE générés à l'intérieur d'un dispositif électronique. Trois types de facteurs de masquage d'erreur sont pertinents pour les circuits numériques : le masquage logique, le masquage électrique et le masquage temporel. Chacune d'entre elles dispose de mécanismes différents par lesquels le SE peut être filtré.

Un SE peut être masqué logiquement dans des circuits combinatoires lorsque l'une des entrées a une valeur qui impose la valeur de sortie, quelle que soit la valeur des autres

entrées.

L'effet de masquage électrique est l'atténuation successive de l'impulsion transitoire jusqu'à ce qu'elle soit complètement éteinte. Elle est causée par l'interaction électrique de l'impulsion transitoire avec les composants électroniques. Son apparition dépend des caractéristiques du dispositif, telles que ses capacités parasites, et du transitoire, de son amplitude et de sa largeur.

L'effet de masquage temporel, ou effet de masquage de la fenêtre de verrouillage, est la "perte" de l'impulsion transitoire due à son apparition hors de la fenêtre de verrouillage des flip-flops et des latches. La probabilité de masquage dépend de la fréquence de fonctionnement de l'élément de verrouillage et de la largeur de l'impulsion transitoire.

Méthodes d'analyse d'erreur logicielle

Les méthodes d'analyse SE sont basées sur de nombreuses techniques différentes, des modèles mathématiques aux tests physiques. Dans cette section, nous décrivons les techniques les plus courantes liées à l'estimation SEV, réparties en sept catégories : analytique, simulation électrique, simulation logique, émulation, injection de fautes de logiciel, tests de matériel et tests de rayonnement.

Les méthodes analytiques utilisent un modèle mathématique pour prédire l'apparition de SE. Ces modèles diffèrent de la modélisation à semi-conducteurs utilisée dans les simulations électriques. Ils traitent généralement les probabilités associées, telles que probabilité de génération d'impulsions, probabilité de propagation d'erreur, probabilité de verrouillage, fenêtre de vulnérabilité probabiliste, matrices de transfert, modèles de porte probabilistes, et probabilité de signal. Ces méthodes fonctionnent au niveau du système, du RTL (Register Transfer Level) et, dans la plupart des cas, au niveau de la porte, avec la représentation du dispositif dans le domaine structurel.

Les méthodes de simulation électrique sont divisées en deux parties : celles dédiées à l'analyse de l'interaction de la particule de rayonnement avec le dispositif électronique, basées sur des simulations tridimensionnelles (3D), et celles dédiées à l'analyse du

comportement électrique du dispositif électronique, basées sur des simulations de circuits similaires à SPICE. Ces méthodes fonctionnent aux niveaux des transistors et des portes, dans le domaine physique.

Les méthodes de simulation logique utilisent des modèles HDL (Hardware Description Language) et un simulateur logique. Elles peuvent être divisées en deux groupes principaux : celles basées sur les commandes du simulateur et celles basées sur la modification du code. En outre, ils incluent un groupe générique mineur pour d'autres techniques. Les deux groupes peuvent travailler sur des domaines comportementaux et structurels, au niveau système, RTL ou au niveau de la porte.

Les méthodes d'émulation pour l'analyse SE sont celles dans lesquelles une plateforme matérielle, généralement un FPGA, est utilisée pour implémenter la fonctionnalité de circuit souhaitée, en ajoutant des ressources pour injecter des erreurs et pour détecter les erreurs propagées. Ils travaillent dans le domaine physique, au niveau du système, de la RTL ou de la porte.

Les méthodes d'injection de défaut de logiciel sont celles implémentées par le logiciel qui s'exécute dans le microprocesseur évalué ou fait partie du système évalué. Ainsi, il peut être divisé en méthodes dédiées à l'analyse des effets des défauts matériels et à celles dédiées aux bogues logiciels. Dans les deux cas, ils travaillent dans le domaine physique, au niveau RTL ou au niveau du système.

Les tests matériels d'analyse d'erreur requièrent que le matériel dispose de ressources spécifiques pour l'injection d'erreur dans la conception finale. Comme dans les méthodes d'injection logicielle de défaut, cette exigence limite le nombre de nœuds de circuit dans lesquels un SE peut être injecté. Ces méthodes fonctionnent également dans le domaine physique, au niveau RTL ou système.

Les méthodes de test de rayonnement nécessitent le dispositif réel et une source de rayonnement. Puisqu'ils s'appuient sur l'interaction du rayonnement avec le dispositif électronique, ils constituent la méthode la plus proche de l'interaction réelle dans l'application réelle. Ces tests peuvent être utilisés pour analyser les trois classes d'effets de rayonnement

(TID, DD et SEE), nécessitant différentes sources de rayonnement. Pour les études SEE, ils utilisent généralement des ions lourds et des protons, mais également des sources de photons. Ces méthodes fonctionnent au niveau des transistors, dans le domaine physique.

SET Susceptibilité sur l'estimation SEV

Parmi les méthodes discutées pour l'analyse SE, les tests de rayonnement sont les seuls qui prennent intrinsèquement en compte les susceptibilités SET des portes du circuit, car lors de ces tests, la particule de rayonnement interagit avec les transistors de circuit pendant que le circuit exécute le scénario de fonctionnement souhaité. Ainsi, tous les effets de masquage d'erreur sont considérés intrinsèquement par cette méthode.

Cependant, certaines méthodes de simulation analytique et électrique prennent également en compte les susceptibilités SET. Dans Buard and Anghel (2011), les auteurs présentent une méthode de modélisation de porte et de simulation électrique au niveau des transistors utilisant SPICE. Leur méthode inclut intrinsèquement le masquage logique interne, avec l'influence de la topologie de la porte et des valeurs d'entrée. Il comprend également le masquage électrique, via la modélisation électrique des transistors et des SET. Leurs résultats montrent que les effets de masquage logique et électrique affectent la susceptibilité SET de la porte NAND2 analysée, ce qui a entraîné une augmentation d'environ 63% du taux de défaillance le plus faible au plus élevé en fonction de la variation de la distribution des valeurs d'entrée. De même, la méthode analytique proposée dans Rezaei *et al.* (2014) inclut également les effets de masquage interne des portes. Une des préoccupations majeures des auteurs est de savoir comment la distribution des valeurs d'entrée influencent la génération SET à la sortie des portes. Leurs résultats montrent également que les susceptibilités SET sont affectées par ces facteurs et, en outre, leur influence sur le SEV du circuit. Par exemple, ils montrent une augmentation d'environ 900 % de la vulnérabilité la plus faible à la plus élevée d'une porte NAND2 en fonction des valeurs d'entrée.

D'autre part, les méthodes basées sur la simulation logique et sur l'émulation ne pren-

nent pas en compte les susceptibilités SET des portes du circuit. En effet, après toutes les recherches documentaires effectuées, la seule méthode que nous avons trouvée qui adopte une distribution de erreur probabiliste, qui ressemblerait à celle prise en compte des susceptibilités SET, est l'approche d'émulation décrite dans May and Stechele (2012) et May and Stechele (2013). Cependant, leurs travaux portent sur l'informatique stochastique et la réduction de tension, pour lesquels il est essentiel d'analyser les effets de défauts multiples indépendants et concomitants, dans un scénario différent de celui des études SEE.

Enfin, les tests du matériel et les méthodes d'injection de logiciel ne prennent pas non plus en compte les susceptibilités SET. Cependant, contrairement aux méthodes de simulation et d'émulation logiques, elles ne conviennent pas pour incorporer cette fonctionnalité en raison de leur accès restreint aux nœuds du circuit.

D.3 Stratégie proposée pour l'estimation SEV

Comme indiqué dans la section précédente, la simulation logique et les approches basées sur l'émulation pour l'estimation SEV ne prennent pas en compte les susceptibilités SET des portes. Ce scénario nous a motivés à étudier une stratégie d'estimation SEV tenant compte de ces susceptibilités et pouvant être adoptée à la fois par la simulation logique et par les approches basées sur l'émulation. Dans cette section, nous présentons la stratégie proposée pour laquelle nous avons défini deux versions : une simplifiée et une complète.

La version simplifiée suppose que la susceptibilité au SET d'une porte dépend uniquement de sa topologie. En d'autres termes, différents types de porte ont différentes susceptibilités SET, par ex. un OR et un AND, mais chaque instance du même type de porte a la même susceptibilité au SET. La version complète considère que, à l'instar de la topologie, le fonctionnement de la porte a également un effet sur sa susceptibilité au SET. En d'autres termes, chaque porte à une susceptibilité SET spécifique, même si elle provient du même type que l'autre porte.

Les deux versions de la stratégie reposent sur la même structure de cadre d'estimation SEV, illustrée à la Fig. D.1. Cette infrastructure permet l'injection SET et la détection SE en comparant les résultats de deux versions du dispositif à tester (DUT – Device Under Test), tout en exécutant le même scénario opérationnel. Dans la Fig. D.1, la structure est représentée par un fichier de testbench, nommé `dut_sev_tb.vhd`, destiné à être utilisé dans des simulations logiques. Cependant, une structure de cadre similaire pourrait être intégrée pour être utilisée dans les approches basées sur l'émulation. Néanmoins, dans la suite de cette section, la structure est considérée comme un fichier de testbench, illustrant les étapes de la stratégie, qui considère également son application dans un environnement de simulation logique. Les composants du cadre sont décrits ci-dessous.

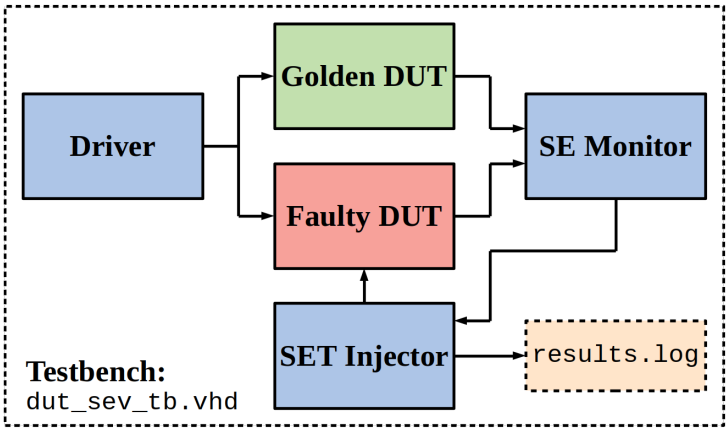

FIGURE D.1 – Cadre proposé pour l'estimation SEV.

Golden DUT est la version du DUT qui n'est pas soumise à l'injection SET.

Faulty DUT est la version du DUT dans laquelle les SET sont injectés. L'injection SET, à la sortie des CLB à l'intérieur du Faulty DUT, est effectuée à l'aide d'un saboteur pour chaque CLB. Cet élément inverse la valeur logique de la sortie CLB lorsqu'elle est activée et qu'un SET est injecté.

Driver est responsable du scénario opérationnel. Il applique la même séquence de vecteurs de test aux entrées de Golden DUT et de Faulty DUT.

SE Monitor compare les sorties des deux DUT, en détectant l'occurrence d'un SE. Lorsqu'un SE se produit, le SE Monitor avertit le SET Injector.

SET Injector injecte les SET dans le Faulty DUT, via les saboteurs. Il contrôle quand, où et combien de SET doivent être injectés. En outre, il génère le rapport pour le processus d'estimation SEV, avec le nombre total de SET injectés et de SE observés.

Version simplifiée de la stratégie proposée

La version simplifiée de la stratégie d'estimation SEV proposée comporte cinq étapes, décrites dans ce qui suit.

Étape 1 consiste à synthétiser la description RTL du dispositif testé sous la technologie cible, générant ainsi la netlist post-synthèse annotée.

Étape 2 consiste à insérer les saboteurs dans le DUT, en générant la netlist du Faulty DUT. Le Golden DUT est la netlist d'origine post-synthèse.

Étape 3 consiste à associer à chaque CLB sa susceptibilité SET respective, basée sur la base de données de référence. Le résultat est un fichier de rapport avec les susceptibilités SET de chaque CLB utilisé dans le DUT.

Étape 4 consiste à générer le testbench utilisé dans le cadre de simulation, en incorporant la susceptibilité SET de chaque CLB.

Étape 5 correspond à la simulation réelle du DUT injectant la distribution pondérée de SET, et génère un rapport avec l'estimation SEV du circuit.

Version complète de la stratégie proposée

La version complète de la stratégie d'estimation SEV proposée comporte sept étapes décrites ci-dessous.

Étape 1 consiste à synthétiser la description RTL, comme dans la version simplifiée.

Étape 2 consiste à insérer les saboteurs dans le DUT, comme dans la version simplifiée.

Étape 3 consiste à simuler le DUT avec le scénario opérationnel souhaité pour générer le fichier de modification de la valeur (VCD – Value Change Dump), à partir duquel la distribution des valeurs d'entrée sont extraits. Le testbench de cette simulation doit utiliser le même Driver utilisé dans le cadre de la simulation, abordé au début de ce section, pour garantir l'utilisation du même scénario opérationnel.

Étape 4 consiste à extraire la distribution des valeurs d'entrée du fichier VCD, en générant un rapport avec le pourcentage de temps pendant lequel chaque CLB a été soumis à chaque valeurs d'entrée.

Étape 5 consiste à calculer les susceptibilités SET spécifiques de chaque CLB pour le scénario opérationnel, à l'aide du rapport sur la distribution des valeurs d'entrée et des informations sur les susceptibilités SET. Le résultat est un fichier de rapport avec les susceptibilités SET de chaque CLB utilisé dans le DUT. Cette étape est une modification de l'étape 3 de la version simplifiée.

Étape 6 consiste à générer le testbench utilisé dans le cadre de la simulation, comme dans l'étape 4 de la version simplifiée.

Étape 7 correspond à la simulation réelle du DUT injectant la distribution pondérée des SET, comme à l'étape 5 de la version simplifiée.

Ajout des saboteurs

Le processus d'insertion de saboteurs dans le DUT, qui est exactement identique dans les deux versions de la stratégie, se traduit par :

1. au port de l'entité, ajouter le signal d'erreur (SET) et les signaux d'activation pour chaque CLB ;
2. pour chaque CLB, créer un nouveau signal, qui remplace la sortie d'origine du CLB et est utilisé comme entrée des saboteurs ;
3. connecter la sortie du CLB à ce nouveau signal ; et

4. ajouter une instance du saboteur, mappant son entrée sur le nouveau signal, sa sortie sur le signal de sortie d'origine du CLB, son signal d'activation sur le port d'activation correspondant de l'entité et le signal d'erreur sur le port d'erreur de l'entité.

Extraction de la distribution des valeurs d'entrée

Le fichier VCD est généré par les outils de simulation logique, normalement utilisés pour la visualisation de forme d'onde. C'est un fichier basé sur ASCII qui inclut les changements de valeur par ordre croissant dans le temps des signaux. Il contient essentiellement toutes les marques de temps dans lesquelles il y a des changements de valeur d'au moins un signal. Pour chaque repère temporel, il existe une liste avec tous les signaux dont la valeur change à ce moment-là, avec leurs nouvelles valeurs.

Les informations à extraire du fichier VCD correspondent au pourcentage de temps pendant lequel chaque valeur d'entrée est appliquée à chaque CLB. La première étape consiste à transformer cette représentation textuelle en un tableau, avec une liste d'intervalles de temps et les valeurs de chaque signal dans l'intervalle respectif. Ensuite, les intervalles de temps dans lesquels chaque valeur est appliquée au CLB sont additionnés, générant le temps total pour chaque valeur. Avec le temps total du scénario simulé, les pourcentages peuvent être calculés.

Calcul des susceptibilités SET

La stratégie complète nécessite la susceptibilité SET spécifique de chaque CLB du dispositif sous test, qui est calculée comme une somme pondérée des susceptibilités SET de la configuration de CLB respective pour chaque valeur d'entrée. Les poids représentent le pourcentage de temps pendant lequel chaque valeur d'entrée est appliquée au CLB.

Générer le testbench pour l'estimation SEV

Le testbench utilisé comme cadre d'estimation SEV est composé de cinq éléments, comme proposé au début de cette section, et présente la même structure pour les versions simplifiée et complète de la stratégie. Le processus de génération de ce testbench consiste à créer le SET Injector spécifique et à l'interconnecter avec les autres éléments, comme décrit dans la suite.

Driver définit le scénario opérationnel et, pour la version complète, doit être identique à celui utilisé dans la première simulation.

Le Golden DUT et le Faulty DUT sont des sous-produits du processus global. Le premier est obtenu avec la synthèse (étape 1), tandis que le second est obtenu en ajoutant les saboteurs (étape 2).

Le SE Monitor est un comparateur paramétrable qui peut être utilisé avec n'importe quel DUT. Il suffit pour cela de régler le paramètre de largeur des données d'entrée en fonction du nombre de signaux de sortie du DUT.

En ce qui concerne SET Injector, il est spécifique à chaque scénario opérationnel appliqué au DUT. La distribution SET est effectuée à l'aide de la méthodologie de vérification Open Source VHDL (OSVVM – Open Source VHDL Verification Methodology). Cette méthodologie inclut le package RandomPkg, qui fournit un ensemble de fonctions permettant de générer des nombres pseudo-aléatoires, y compris une fonction de distribution pondérée pour les valeurs entières (DistValInt). Le SET Injector s'appuie sur cette fonction pour sélectionner le CLB dans lequel le SET sera injecté. Comme il est généré pour toute distribution SET spécifique, son processus de génération s'applique à la fois aux versions simplifiées et complètes de la stratégie proposée.

D.4 Estimation des susceptibilités SET

La stratégie proposée pour l'estimation SEV nécessite la susceptibilité SET des éléments qui composent le circuit. Malheureusement, ces informations ne sont généralement

pas disponibles. Pour cette raison, nous avons estimé la susceptibilité SET des fonctions configurables d'une famille de technologies FPGA que nous avons choisie.

Bien que la méthodologie adoptée s'applique à n'importe quelle technologie FPGA, ces travaux portent sur la famille des FPGA ProASIC3E. Il s'agit d'un dispositif CMOS à mémoire flash de 130 nm. Contrairement à la plupart des autres technologies, ce CLB n'est pas basé sur un élément séquentiel et une table de conversion. Au lieu de cela, il a un petit ensemble de portes logiques qui peuvent être configurées en tant que fonctions combinatoires et séquentielles. Le CLB de cette famille s'appelle VersaTile.

Le VersaTile CLB a été modélisé au niveau du transistor pour reproduire son comportement logique, sans aucune préoccupation liée aux caractéristiques électriques et temporelles. Il a été décrit dans un VHDL structurel comme une interconnexion de transistors PMOS et NMOS, mettant en œuvre à la fois des portes logiques et des commutateurs de configuration.

Comme les documents de conception et de fabrication de cette famille FPGA ne fournissent pas les données de vecteur de configuration pour chaque fonction, nous avons défini les vecteurs de configuration possibles.

Afin de simplifier la substitution des macros ProASIC3E d'origine par notre CLB modélisé, nous avons créé une bibliothèque de composants VHDL avec toutes les fonctions CLB modélisées.

Nous avons estimé la susceptibilité SET de chaque macro contenue dans notre version de la bibliothèque *proasic3e*. Le processus d'estimation est basé sur des simulations VHDL avec des injections SET au niveau du transistor.

Les susceptibilités au SET estimées pour les fonctions VersaTile sont présentées dans l'Appendice B, divisé en trois tableaux : Table B.1, pour les fonctions combinatoires ; Table B.2, pour les flip-flops ; et Table B.4, pour les latches. Pour chaque macro, ils présentent les susceptibilités SET pour chaque valeur d'entrée. De plus, ils présentent également la susceptibilité au SET moyenne pour chaque macro, en considérant la moyenne des valeurs pour chaque valeur d'entrée.

En tenant compte des valeurs d'entrée, les susceptibilités au SET des fonctions combinatoires sont comprises entre $\sim$ 5–30 %. Pour les valeurs moyennes, qui prennent en compte que l'influence de la topologie, elles se situent dans $\sim$ 10–27 %. Pour les flip-flops, la prise en compte des valeurs d'entrée a eu pour résultat des susceptibilités SET comprises entre $\sim$ 5–21 %, tandis que les valeurs moyennes se situaient dans $\sim$ 10–17 %. Pour les latches, le fait de prendre en compte les valeurs d'entrée a eu pour résultat des susceptibilités SET comprises entre $\sim$ 5–18 %, tandis que les valeurs moyennes se situaient dans $\sim$ 9–15 %.

De plus, un pourcentage des SE observées dans les flip-flop et les latches est composé par des SEU, pas seulement par des SET. Pour les flip-flop, compte tenu des valeurs d'entrée, le pourcentage du SEUs se situe dans $\sim$ 0–76 %, tandis que les valeurs moyennes se situent dans $\sim$ 33–68 %. Enfin, pour les latches, en considérant les valeurs d'entrée, le pourcentage du SEUs se situe dans $\sim$ 0–50 %, tandis que les valeurs moyennes se situent dans $\sim$ 21–49 %. La Table B.3 présente la valeur en pourcentage des SE observés qui sont des SEU pour les flip-flops, tandis que la Table B.5 présente les mêmes informations pour les latches.

D.5 Évaluation de la stratégie proposée

Pour évaluer l'impact de la prise en compte de la susceptibilité SET de chaque CLB sur la vulnérabilité d'erreur 'soft' des circuits, nous avons sélectionné 38 circuits combinatoires de référence de la liste LGSynth91 pour utiliser en tant que DUT. Nous avons limité l'analyse aux circuits de référence nécessitant moins de 100 CLB, en raison du temps de simulation au niveau du transistor. Enfin, le choix de la liste LGSynth91 était arbitraire, mais cette sélection n'est pas importante pour la présente analyse car l'approche proposée est applicable à n'importe quel circuit.

Les circuits sélectionnés ont un large éventail de complexité : le nombre d'entrées varie de 3 à 47 ; le nombre de sorties varie de 1 à 36 ; et le nombre de CLB, pour la technologie analysée, varie de 3 à 95. Les détails de chacun des circuits de référence sélectionnés sont

présentés dans le Table D.1.

Les circuits de référence sélectionnés ne disposent pas de documentation détaillée sur leurs fonctionnalités. Pour cette raison, dans le Driver, nous avons choisi d'utiliser une séquence pseudo-aléatoire de vecteurs de test pour définir les scénarios opérationnels.

Nous avons effectué quatre estimations SEV distinctes, une au niveau du transistor, qui fournit le SEV de référence, et trois au niveau du CLB, afin d'évaluer l'effet de la prise en compte de la susceptibilité au SET des CLB.

Toutes les estimations SEV ont utilisé la même structure du cadre de simulation, avec certaines spécificités pour chaque catégorie. Pour chaque DUT, Driver et le SE Monitor étaient les mêmes dans les quatre catégories.

Pour le Golden DUT et le Faulty DUT, nous avons deux situations. L'estimation SEV au niveau du transistor utilise la bibliothèque présentée dans la section D.4 pour modéliser le DUT. Le port de défauts du Golden DUT est défini sur 0 (i.e. sans injection de SET), alors que le port de défauts du Faulty DUT est utilisé pour injecter les SET. Dans ce cas, la netlist avec les saboteurs n'est pas utilisée. Par ailleurs, les trois estimations SEV au niveau des CLB utilisent une variante de la bibliothèque de technologies originale sans le modèle de VITAL. Le Golden DUT est la netlist post-synthèse et le Faulty DUT est la netlist avec les saboteurs ajoutés.

Le SET Injector est l'élément qui nécessite davantage d'ajustements d'une catégorie à l'autre, notamment en ce qui concerne le nombre de SET injectés et la distribution de SET.

Au niveau des transistors, tous les SET sont distribués uniformément sur chaque transistor, conformément à l'estimation de susceptibilité au SET (Section D.4). Au niveau des CLB, en ignorant les susceptibilités au SET, les SET ont également été uniformément distribués à chaque CLB. D'autre part, les estimations SEV considérant les susceptibilités au SET utilisent la distribution pondérée décrite dans la section D.3. Le processus d'estimation SEV qui prend en compte uniquement l'effet de topologie utilise la susceptibilité au SET moyenne de chaque CLB (stratégie simplifiée). Enfin, le processus

d'estimation SEV prenant en compte à la fois la topologie et la distribution des valeurs d'entrée utilise les susceptibilités SET obtenues comme décrit dans la section D.3 (stratégie complète).

Pour toutes les estimations SEV, nous avons injecté l'équivalent de 1000 SET par transistor, ce qui donne 82000 SET à l'intérieur de chaque CLB, car le modèle CLB comporte 82 transistors. Au niveau des transistors, le nombre total de SET injectés est de 82000 $\cdot N_{CLB}$. Au niveau des CLB, nous avons adopté le calcul présenté dans la section D.3, mais avec des susceptibilités moyennes différentes. Pour le processus à distribution uniforme, nous avons adopté la susceptibilité moyenne de toutes les configurations combinatoires de la bibliothèque. Pour le processus avec distribution pondérée, nous avons calculé la susceptibilité moyenne des CLB utilisés dans le DUT.

Les estimations SEV obtenues pour chaque circuit de référence analysé sont présentées dans le Table D.1. Pour chaque référence, Table D.1 inclut la quantité de SE observées dans chaque catégorie d'estimation SEV et la SEV estimée respective, en pourcentage. Pour améliorer la lisibilité, Table D.1 n'inclut pas le nombre total de SET injectés.

Pour les circuits analysés, dans les scénarios opérationnels choisis, l'erreur moyenne de l'estimation SEV obtenue en ignorant les susceptibilités SET était de 15,27 %. En considérant uniquement l'effet topologique (stratégie simplifiée), l'erreur moyenne est de 4,70 %. Enfin, la prise en compte de la topologie et de l'effet de distribution des valeurs d'entrée (stratégie complète) entraîne une erreur moyenne de 0,68 %. Le calcul de ces erreurs moyennes prend en compte le module des erreurs présenté dans Table D.1.

TABLE D.1 – Circuits combinatoire de référence – Caractéristiques et résultats de l'estimation SEV. Pour chaque référence, le tableau présente (i) le nombre d'entrées, de sorties et de CLB requis pour la technologie sélectionnée ; et (ii) le nombre de SE observées et le SEV estimé. Colonnes Trans. fait référence aux résultats au niveau du transistor; Unif., à distribution uniforme, en ignorant les susceptibilités SET; Simp., pour la stratégie simplifiée; et Comp., pour la stratégie complète.

Benchmark				**Observed Soft Errors**				**Estimated Vulnerability [%]**			
Name	**in**	**out**	**CLB**	**Trans.**	**Unif.**	**Simp.**	**Comp.**	**Trans.**	**Unif.**	**Simp.**	**Comp.**
b1	3	4	3	56274	42267	56501	56274	22,88	17,18	22,97	22,88
cm82a	5	3	4	64750	56356	64248	64760	19,74	17,18	19,59	19,74
majority	5	1	5	29154	27743	25442	29484	7,11	6,77	6,21	7,19
tcon	17	16	8	102504	112713	102998	102532	15,63	17,18	15,70	15,63
cm152a	11	1	10	55739	68282	50780	53417	6,80	8,33	6,19	6,51
parity	16	1	10	199540	140892	199374	199538	24,33	17,18	24,31	24,33
cm42a	4	10	13	153992	176536	162654	153819	14,45	16,56	15,26	14,43
cm138a	6	8	13	121281	133799	133016	121104	11,38	12,55	12,48	11,36
t481	16	1	13	96625	94802	93403	95785	9,06	8,89	8,76	8,99
cmb	16	4	17	48211	58979	60189	48067	3,46	4,23	4,32	3,45
cm163a	16	5	19	113452	128205	110202	110635	7,28	8,23	7,07	7,10
9symml	9	1	21	85420	94073	83462	85332	4,96	5,46	4,85	4,96
cm85a	11	3	21	111091	115786	115181	111612	6,45	6,72	6,69	6,48
cm162a	14	5	21	136472	150604	131212	135535	7,93	8,75	7,62	7,87
cordic	23	2	22	95024	119779	93648	95421	5,27	6,64	5,19	5,29
decod	5	16	22	162133	282627	198810	162248	8,99	15,67	11,02	8,99
i1	25	16	22	155015	196462	152493	156465	8,59	10,89	8,45	8,67
z4ml	7	4	22	191826	200437	180538	192457	10,63	11,11	10,01	10,67
cu	14	11	24	175629	240885	189681	175590	8,92	12,24	9,64	8,92
pm1	16	13	24	258680	267277	273931	256468	13,14	13,58	13,92	13,03
pcle	19	9	27	242821	274071	257181	240347	10,97	12,38	11,62	10,86
cc	21	20	28	291514	306625	299260	290970	12,70	13,35	13,03	12,67
sct	19	15	28	243121	275822	258943	243129	10,59	12,01	11,28	10,59
x2	10	7	28	184694	191141	183796	182951	8,04	8,32	8,01	7,97
mux	21	1	29	58684	64726	53283	55647	2,47	2,72	2,24	2,34
f51m	8	8	32	319996	282514	322889	319432	12,19	10,77	12,31	12,17
frg1	28	3	35	147293	169121	151430	148369	5,13	5,89	5,28	5,17
lal	26	19	36	294254	318219	310205	294657	9,97	10,78	10,51	9,98
c8	28	18	44	330306	379865	339958	331218	9,15	10,53	9,42	9,18
unreg	36	16	48	358056	393843	366789	358096	9,10	10,01	9,32	9,10
pcler8	27	17	51	488513	537869	494574	485876	11,68	12,86	11,83	11,62
count	35	16	55	459377	418720	465387	460524	10,19	9,28	10,32	10,21
b9	41	21	59	510557	525984	511516	513199	10,55	10,87	10,57	10,61
comp	32	3	62	113509	130455	113111	113175	2,23	2,57	2,22	2,23
term1	34	10	68	260615	314802	263336	260610	4,67	5,65	4,72	4,67
cht	47	36	82	621771	800148	594967	619778	9,25	11,90	8,85	9,22
ttt2	24	21	85	601227	683075	586786	593077	8,63	9,80	8,42	8,51
my_adder	33	17	95	1050020	925060	1011741	1058428	13,48	11,87	12,99	13,59

D.6 Utilisation de la stratégie proposée avec émulation

L'adoption de la stratégie proposée avec une approche de simulation logique a été démontrée dans la section précédent, de même que l'évaluation des versions simplifiée et complète de la stratégie. Bien que la stratégie soit bien adaptée à cette approche d'estimation SEV basée sur la simulation, il serait intéressant de tirer parti de l'avantage d'une estimation rapide avec l'émulation. Pour cette raison, dans cette section, nous discutons de l'adoption de cette stratégie avec certaines approches d'émulation.

Les versions simplifiée et complète s'appliquent à l'émulation. Pour la version simplifiée, l'étape 4 passe de la génération du testbench pour l'injection SET à la structure d'émulation intégrée pour l'injection SET, et l'étape 5 de la simulation à l'exécution du cadre d'émulation. De même, pour la version complète, les mêmes modifications s'appliquent aux étapes 6 et 7, respectivement.

Les approches d'émulation basées sur l'hôte sont celles dans lesquelles un ordinateur hôte externe contrôle le processus d'injection de fautes. Dans ce scénario, la distribution pondérée qui représente les différentes susceptibilités au SET doit être générée et contrôlée par l'hôte. Cela devrait être plus facile que de générer et de contrôler la distribution pondérée à l'intérieur de la plateforme d'émulation.

Contrairement aux approches basées sur l'hôte, dans les approches à émulation autonome, le contrôle du processus d'injection de fautes est à l'intérieur de la plateforme. Dans ce scénario, bien que la distribution pondérée puisse être configurée à une étape précédente, elle doit être générée et contrôlée par la plateforme. Cela nécessite plus de ressources, mais l'estimation peut encore être plus rapide que d'utiliser l'approche basée sur l'hôte.

Nous avons proposé un prototype de plate-forme émulée pour l'estimation SEV, capable d'injecter une distribution pondérée de SET, conformément à la stratégie proposée. Les résultats obtenus jusqu'à présent indiquent une bonne capacité de distribution des

SET, mais des efforts de recherche et de développement supplémentaires restent nécessaires.

D.7 Conclusions

Ce travail présente une stratégie pour incorporer les susceptibilités au SET des portes dans l'estimation SEV des circuits numériques. Cette stratégie est proposée en deux versions, une simplifiée et une complète. La première présente l'avantage d'exiger moins d'informations et de traitement de données, mais offre un gain limité en précision d'estimation, tandis que la seconde, même si elle requiert plus d'informations et de traitement de données, fournit une estimation SEV plus précise.

Comme la stratégie s'appuie sur les susceptibilités au SET des porte et que cette information n'est pas disponible pour la technologie sélectionnée, ce travail fournit également une estimation de la susceptibilité au SET de chaque configuration de CLB de la famille de FPGA choisie.

La stratégie proposée a été évaluée avec une approche de simulation logique appliquée à un ensemble de circuits de référence. Les résultats montrent l'accroissement de la précision de l'estimation SEV en commençant par une approche qui ignore les susceptibilités SET, puis avec la version simplifiée de la stratégie, et enfin grâce à la version complète de cette stratégie.

Le travail inclut également une discussion sur la mise en œuvre de la stratégie proposée avec des approches d'émulation. Un prototype de plateforme d'émulation a été utilisé pour démontrer l'applicabilité de la solution proposée dans le cas d'une distribution probabiliste de SET.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Stratégie d'estimation de la vulnérabilité aux erreurs 'soft' basée sur la susceptibilité aux événements transitoires de chaque porte logique

Mots clés : Taux d'erreur 'soft', Vulnérabilité aux erreurs 'soft', Injection de défauts, Transitoires d'événements uniques, FPGA

Résumé : La vulnérabilité aux erreurs 'soft' (SEV – Soft-Error Vulnerability) est un paramètre estimé qui, associé aux caractéristiques de l'environnement de radiation, est utilisé pour obtenir le taux d'erreur 'soft' (SER – Soft-Error Rate), c'est utilisé pour prédire le comportement des systèmes numériques en cet environnement. Actuellement, la méthode la plus précise pour l'estimation de SER est le test de radiation, car elle présente l'interaction réelle du rayonnement avec le dispositif électronique. Toutefois, il est coûteux et nécessite le dispositif réel, qui devient disponible tard dans le cycle de conception. Ces restrictions ont motivé le développement d'autres méthodes d'estimation des SER et SEV, notamment des méthodes analytiques, simulations électriques et logiques, ainsi que des approches basées sur l'émulation. Ces techniques incorporent généralement des effets de masquage logiques, électriques et temporels. Néanmoins, la plupart d'entre eux ne prennent pas en compte un facteur intrinsèque au test de radiation: la probabilité que le rayonnement ionisant produise une erreur 'soft' à la sortie des portes logiques du circuit, dénommé susceptibilités aux événements transitoires (SET). Dans ce contexte, nous proposons une stratégie d'estimation SEV basée sur ces susceptibilités SET, adaptée aux cadres basés sur la simulation et l'émulation. Dans une version simplifiée de cette stratégie, les susceptibilités SET prennent en compte les effets de la topologie des portes logiques, tandis que dans une version complète, ces susceptibilités prennent en compte la topologie et le fonctionnement du circuit. La stratégie proposée a été évaluée avec une approche basée sur la simulation, estimant la SEV de 38 circuits de référence. L'erreur d'estimation moyenne a été réduite de 15,27%, en ignorant les susceptibilités, à 4,70% pour la version simplifiée et à 0,68% pour la version complète. Enfin, nous montrons la faisabilité de l'adoption de la stratégie proposée dans un approche basé sur l'émulation.

Title : A strategy for soft-error vulnerability estimation using the single event transient susceptibilities of each gate

Keywords : Soft-Error Rate, Soft-Error Vulnerability, Fault-Injection, Single-Event Transient, FPGA

Abstract : The Soft-Error Vulnerability (SEV) is an estimated parameter that, in conjunction with the characteristics of the radiation environment, is used to obtain the Soft-Error Rate (SER), that is a metric used to predict how digital systems will behave in this environment. Currently, the most reliable method for SER estimation is the radiation test, since it has the actual interaction of the radiation with the electronic device. However, this test is expensive and requires the real device, that becomes available late on the design cycle. These restrictions motivated the development of other SER and SEV estimation methods, including analytical, electrical and logic simulations, and emulation approaches. These techniques usually incorporate the logical, electrical and latching-window masking effects into the estimation process. Nevertheless, most of them do not take into account a factor that is intrinsic to the radiation test: the probability of the radiation particle of producing a Soft-Error (SE) at the output of the logic gates of the circuit, referred to as Single-Event Transient (SET) susceptibility. In this context, we propose a strategy for SEV estimation based on these SET susceptibilities, suitable for simulation- and emulation-based frameworks. In a simplified version of this strategy, the SET susceptibilities take into account only the effects of the logic gate internal circuitry, while in a complete version, these susceptibilities consider both the internal circuitry and the operation of the circuit, that affects its input pattern distribution. The proposed strategy was evaluated with a simulation-based framework, estimating the SEV of 38 benchmark circuits. The average estimation error was reduced from 15.27%, ignoring the susceptibilities, to 4.70%, for the simplified version, and to 0.68%, for the complete version. Finally, we discuss the feasibility of adopting the proposed strategy with an emulation-based framework.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France