

A Framework for Estimating Vulnerability of Digital Circuits against Single Event Multiple Transients

Vivek Bansal

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Electrical and Computer Engineering) at

Concordia University

Montréal, Québec, Canada

August 2022

© Vivek Bansal, 2022

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Vivek Bansal**

Entitled: **A Framework for Estimating Vulnerability of Digital Circuits
against Single Event Multiple Transients**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Sébastien Le Beux

_____ External Examiner
Dr. Mohsen Ghafouri

_____ Internal Examiner
Dr. Sébastien Le Beux

_____ Supervisor
Dr. Otmane Ait Mohamed

Approved by

Yousef R. Shayan, Chair
Department of Electrical and Computer Engineering

_____ 2022

Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

A Framework for Estimating Vulnerability of Digital Circuits against Single Event Multiple Transients

Vivek Bansal

Recent deep-submicron-technology-based integrated circuits (ICs) are substantially more susceptible to transient faults. Thus, the soft errors that occurred due to transient faults are more important than they have ever been. As a result, it is critical to identify any functional inconsistencies and component failures as early in the design process as possible in order to avoid potentially threatening events. Early stage analysis of the integrated circuits enables designers to build fault tolerant systems by applying some fault-mitigation techniques. Existing methodologies for investigating single event transients (SETs) or single event multiple transients (SEMTs) faults often lack the ability to perform system analysis due to the state-explosion (such as simulation-based testing) or due to the ignoring of post-layout information (such as netlist-based analysis).

In this thesis work, we develop a methodology to model and analyze the effect of transient faults at higher-level of abstraction. In the proposed methodology, we include the effect of SEMT on the integrated circuit (IC) device by considering its post-layout information. It also includes checking SEMT propagation throughout the circuit and the effect of all three masking factors (e.g. logical masking, electrical masking, and latching-window effect). The methodology also provides insights into design vulnerabilities and critical areas of the IC layout. It is further extended to estimate the vulnerabilities of registers in openMSP430 core while running a specific application.

Acknowledgments

I would like to acknowledge and give my warmest thanks to my supervisor Dr. Otmane Ait Mohamed to become a member of Hardware Verification Group (HVG) at Concordia University. His guidance and advice always carried me throughout all stages of completing my master's thesis. Working under him gave me invaluable knowledge and experience.

Also, I would like to thank my family and friends who always guided and motivated me throughout this journey. Their unconditional support made this all possible in more ways than one.

Finally, I would like to thank God, for letting me pass through all the difficulties. I have your blessing day by day. You are the one who let me finish my degree. I will keep trusting you with my future.

Contents

List of Figures	ix
List of Tables	xi
List of Acronyms	xiii
1 Introduction	1
1.1 Thesis Objective	3
1.2 Thesis Contribution	4
1.3 Thesis Organization	7
2 Background Information	9
2.1 History of Radiation Effects	9
2.2 Single Event Effects	10
2.3 Types of Single Event Effects	11
2.3.1 Single Event Upset	12
2.3.2 Single Event Transient	13
2.3.3 Single Event Multiple Transient	15
3 Literature Review	16
3.1 Netlist Based Analysis	17
3.2 Layout Based Analysis	17

4	Article I: META: A Layout Based Tool to Estimate the Vulnerability of Digital Circuits to Multiple Event Transient	19
4.1	Introduction	20
4.2	Related Art	21
4.3	META - Multiple Event Transient Analysis Tool	22
4.3.1	Pre Characterization	24
4.3.2	Synthesis and Place & Route	25
4.3.3	Transient Fault Pattern Identification	26
4.3.4	Fault Propagation Analyzer	26
4.4	Test And Results	28
4.4.1	Performance Analysis	29
4.4.2	SEMT Fault Error Sites	30
4.5	Conclusion	31
5	Article II: Layout-based Vulnerability Analysis of LEON3 Processor to Single Event Multiple Transients using Satisfiability Modulo Theories	32
5.1	Introduction	33
5.2	Related Works	34
5.3	Proposed Method	35
5.3.1	LEON3 Architecture, Synthesis, and Place & Route	35
5.3.2	Single Event Multiple Transient (SEMT) pattern identification	38
5.3.3	SMT Modelling of Logic Gates	39
5.3.4	SMT Modelling of LEON3 Processor's Sub-Modules	41
5.3.5	SEMT Propagation analysis using SMT Solvers	44
5.4	Results	46
5.5	Conclusions	48

6	Article III: Application-Specific Formal Approach to Estimate Soft Error Vulnerability of Register File using Satisfiability Modulo Theories	49
6.1	Introduction	50
6.2	Architecture	51
6.3	Preliminaries	52
6.4	Proposed Methodology	54
6.4.1	Satisfiability Modulo Theory (SMT) Analysis	56
6.4.2	Register File Vulnerability	57
6.5	Analytical Experiment	58
6.6	Conclusion	61
7	Conclusion and Future Work	62
7.1	Conclusion	62
7.2	Future Work	63
	Appendix A User Manual: META	64
A.1	Introduction	65
A.2	Running META tool using Graphical User Interface (GUI):	66
A.2.1	Creation of directory	66
A.2.2	Launch META tool	66
A.2.3	Load the Design	67
A.2.4	Select SEMT Pattern Parameters	69
A.2.5	Fault Analysis Evaluation Type	70
A.2.6	Testbench Generation	71
A.2.7	Configuration File	72
A.2.8	Run SEMT Analysis	73
A.2.9	Output - SEMT Fault Pattern	73

List of Figures

Figure 1.1	An illustration of an electron-hole pair generation.	2
Figure 1.2	Overview of proposed framework.	5
Figure 2.1	Energy Particle Strike on Transistor and resulting transient change in current.	11
Figure 2.2	Categorization of Single Event Effects.	12
Figure 2.3	Single Event Upset in D Flip-flop.	13
Figure 2.4	Logical Masking in NAND gate.	14
Figure 2.5	Example for Electrical Masking	14
Figure 2.6	Latching Window Masking Example	15
Figure 2.7	Single Event Multiple Transient Effect	15
Figure 4.1	META - Proposed Tool	23
Figure 4.2	Comparison of simulation time with proposed methodology while calculating SER	29
Figure 4.3	SEMT Faulty Error Sites	30
Figure 5.1	The proposed methodology for SEMT modelling and propagation analysis.	36
Figure 5.2	LEON3 processor's hierarchy view seen after synthesis.	37
Figure 5.3	Extraction of SEMT Patterns.	39
Figure 5.4	An example of 2-Input NAND gate in the proposed SMT modelling.	39
Figure 5.5	Fault re-converging example in 2-input NAND gate.	40

Figure 5.6	Example of a fault re-converging scenarios.	41
Figure 5.7	A pipeline stage of IU seen in Synopsys DC tool after synthesis. . .	43
Figure 5.8	Constructed graph of the pipeline stage seen in Fig. 5.7	44
Figure 5.9	Fatal SEMT Patterns marked on the LEON3 core Layout.	47
Figure 6.1	MSP430 Architecture	52
Figure 6.2	A typical Sample Program with its Block-dependency Graph (BDG). .	53
Figure 6.3	Register Data Flow Graph.	54
Figure 6.4	Basic Steps of Proposed Methodology.	55
Figure 6.5	Design Hierarchy after synthesis.	56
Figure 6.6	Glimpse of Fibonacci Series Program	59
Figure 6.7	Register Vulnerabilities.	60
Figure A.1	META Tool: Welcome Page	67
Figure A.2	META Tool: Mode Selection	68
Figure A.3	META Tool: Design layout Files Selection	69
Figure A.4	META Tool: SEMT Pattern Size Selection	70
Figure A.5	META Tool: Fault Evaluation Type Selection	71
Figure A.6	META Tool – Verilog Test-bench	72
Figure A.7	META Tool - Configuration File	72
Figure A.8	META Tool - Start Analysis	73
Figure A.9	META Tool Output - SEMT Pattern Error Sites	74
Figure A.10	META Tool: Thank You Note	75

List of Tables

Table 4.1	Characteristics of ISCAS’85 benchmark circuits synthesized using TSMC 65nm and SER	28
Table 5.1	Characteristics of sub-modules in the LEON3 core	37
Table 5.2	Comparision of average propagation probabilities due to SET and various SEMTs.	46
Table 5.3	CPU Time elapsed to build and analyze the SMT model.	48
Table 6.1	Experimental Results	61

List of Acronyms

SEMT	Single Event Multiple Transient
SET	Single Event Transient
IC	Integrated Circuit
SEU	Single Event Upset
CMOS	Complementary Metal-Oxide Semiconductor
SEL	Single Event Latch-up
SEE	Single Event Effects
SER	Soft Error Rate
EDA	Electronic Design Automation
META	Multiple Event Transient Analysis
SMT	Satisfiability Modulo Theory
RTL	Register Transfer Level
DRAM	Dynamic Random Access Memory
SEB	Single Event Induced Burnout
SRAM	Static Random Access Memory
GP	General Purpose
FP	Fault Pattern
SDF	Standard Delay Format
CPU	Central Processing Unit
RAM	Random Access Memory

ASIC	Application Specific Integrated Circuit
TMR	Triple Modular Redundancy
RAM	Random Access Memory
HDL	Hardware Description Language
GPL	General Public License
IP	Intellectual Property
IU	Integer Unit
PPO	Pseudo Primary Output
PPI	Pseudo Primary Input
COI	Cone of Influence
ECC	Error Correction Code
REC	Recovery Exiting Copy
PC	Program Counter
BDG	Block-based Dependency Graph
RDFG	Register Data Flow Graph Primary Input
RVF	Register Vulnerability Factor
SP	Stack Pointer
DEF	Data Exchange Format
TDF	Timing Delay Format
GUI	Graphical User Interface
SP	Stack Pointer

Chapter 1

Introduction

Due to rapid growth in the semiconductor industry over the last several decades, computing systems have become important and necessary in human society. The computing system's application area varies from daily user devices such as mobile phones, and laptops to higher-end mainframes like aerospace, and military devices. The fundamental reason behind the growth in a computing system is an increasingly aggressive reduction in the semiconductor device dimensions. Moore's law states that the number of transistors per integrated circuit chip doubles every two years, it was first described by Gordon E. Moore and later proven by the semiconductor industry [1]. The continuous advancement in the field of semiconductor devices leads to increased reliability concerns. One of the major concerns is transient faults. As the size of semiconductor devices decreases, resulting in nanoscale devices, integrated circuits become more prone to soft errors induced by transient faults. When a high-energy particle such as alpha or cosmic ray particles hits the device, it generates a large number of electron-hole pairs, which can be absorbed by the source and drain regions of the transistor on the semiconductor device (illustrated by Figure 1.1) and causes voltage transients at the associated nodes. Also, it may change the logical value of the corresponding combinational circuit components or change the state of flip-flops and latches for sequential circuit components [2] [3].

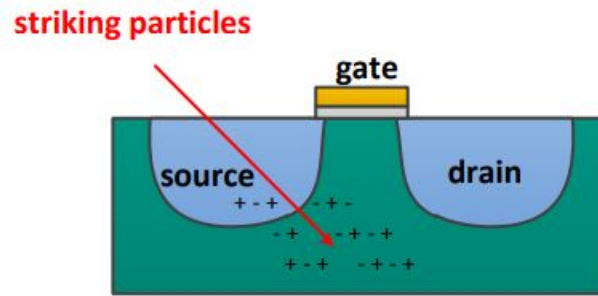


Figure 1.1: An illustration of an electron-hole pair generation.

A transient fault in a combinational logic circuit can be due to a single perturbation event on an integrated circuit (IC), which is called a single-event transient (SET) [4]. If the SET is captured or latched by a memory storage element at the output, it is called a single event upset (SEU). In Complementary Metal-Oxide Semiconductor (CMOS) Bulk technologies, the charge generated by a particle strike may also activate a parasitic thyristor structure and create a short, direct current path between power and ground. This is known as a Single Event Latch-up (SEL). There is significantly less probability that one transient fault effect only one component in the logic circuit due to the nanoscale of semiconductor devices. Therefore, fault can affect more than one component, known as single event multiple transients (SEMT). In the microelectronics domain, these phenomena are known as Single Event Effects (SEEs).

The errors in the circuit caused by radiation particle strikes are considered "soft errors" because they don't change the physical layout of the circuit itself. Circuit vulnerability against transient faults is represented in terms of Soft Error Rate (SER) and its calculation is crucial for the integrated circuit chip design process. This information can help the designer to apply various mitigation techniques to develop fault-resistant integrated circuit chips.

Transient-fault effects generally result in circuit failures, unless they are masked before reaching the primary outputs of the circuit. Masking effects, in fact, can eliminate SETs

or SEMTs before they cause circuit failure. There are three types of masking factors [5] mentioned below:

- (1) **Logical Masking:** When a transient fault induced is masked due to the combinational logic of the component, it is called logical masking.
- (2) **Electrical Masking:** When a transient fault width is attenuated due to electrical properties of gates throughout its propagation, it is called electrical masking
- (3) **Latching window masking:** A latching window masking occurs when a transient fault reaches the input of the flip-flops or memory cells but does not meet its timing requirement such as set-up and hold time.

1.1 Thesis Objective

Various techniques have been published to analyze transient faults at radiation level testing (i.e. after device manufacturing) or circuit level testing (simulation techniques). The main objective of this thesis is to build an automated framework to analyze transients fault at an early stage of development and to reduce the possibility of circuit failure at a later stage. Specifically, this thesis aims to answer the following questions:

Q1: *How to analyze circuit design against single event transients (SETs) and single event multiple transients (SEMTs) ?*

Q2: *How to efficiently utilize formal verification methods to build model and analyze the propagation of transient faults at high-level of abstraction ?*

Q3: *How to measure the vulnerability of circuit designs at a high level without compromising accuracy ?*

Q4: *How to utilize formal verification methods to help design engineer to build fault-tolerant circuit design ?*

Q5: *Is it possible to enhance design scalability while maintaining its accuracy ?*

1.2 Thesis Contribution

In order to accomplish the above-mentioned objectives, a fully automated electronic design automation (EDA) tool has been developed to calculate an accurate early estimation of the design's sensitivity to transient faults like single event transients (SET) and single event multiple transients (SEMT). Post-fabrication layout information is utilized while analyzing the transient faults. It will assist us in identifying and isolating the most critical component of the layout of the design as well as determining the most effective methods for the design engineer to mitigate the faults. This is achieved by investigating the possible faulty sites based on fault propagation throughout the circuit design. The developed tool can additionally explore the design's vulnerability by providing the soft error rate of the circuit design. The analysis is performed through Satisfiability Modulo Theory (SMT), which provides an automatic investigation of how fault is propagating to the final output of the design. This work includes constructing an SMT library of the logic gates. To the best of our knowledge, no other existing tools provide the aforementioned capabilities. Furthermore, such capabilities will help us to assist the designers with a much more exhaustive review of the system and its vulnerabilities before the system is manufactured.

The overview of the proposed methodology is presented in Fig.1.2. It is composed of three major phases: **Synthesis:** Synopsys design compiler is utilized to get gate level netlist of the design as per specified technology node in library files. **Place & Route:** Placement and timing information is extracted using Cadence Encounter Tool. **META (Multiple Event Transient Analysis) tool:** Information extracted in synthesis and place & route are passed to META tool to perform fault propagation analysis using SMT solver. It includes various steps like building an SMT model, fault pattern extraction, and so on. Detailed information is provided in chapter 4.

The key contributions from this research are summarized as follows:

Article I: *META: A Layout Based Tool to Estimate the Vulnerability of Digital Circuits*

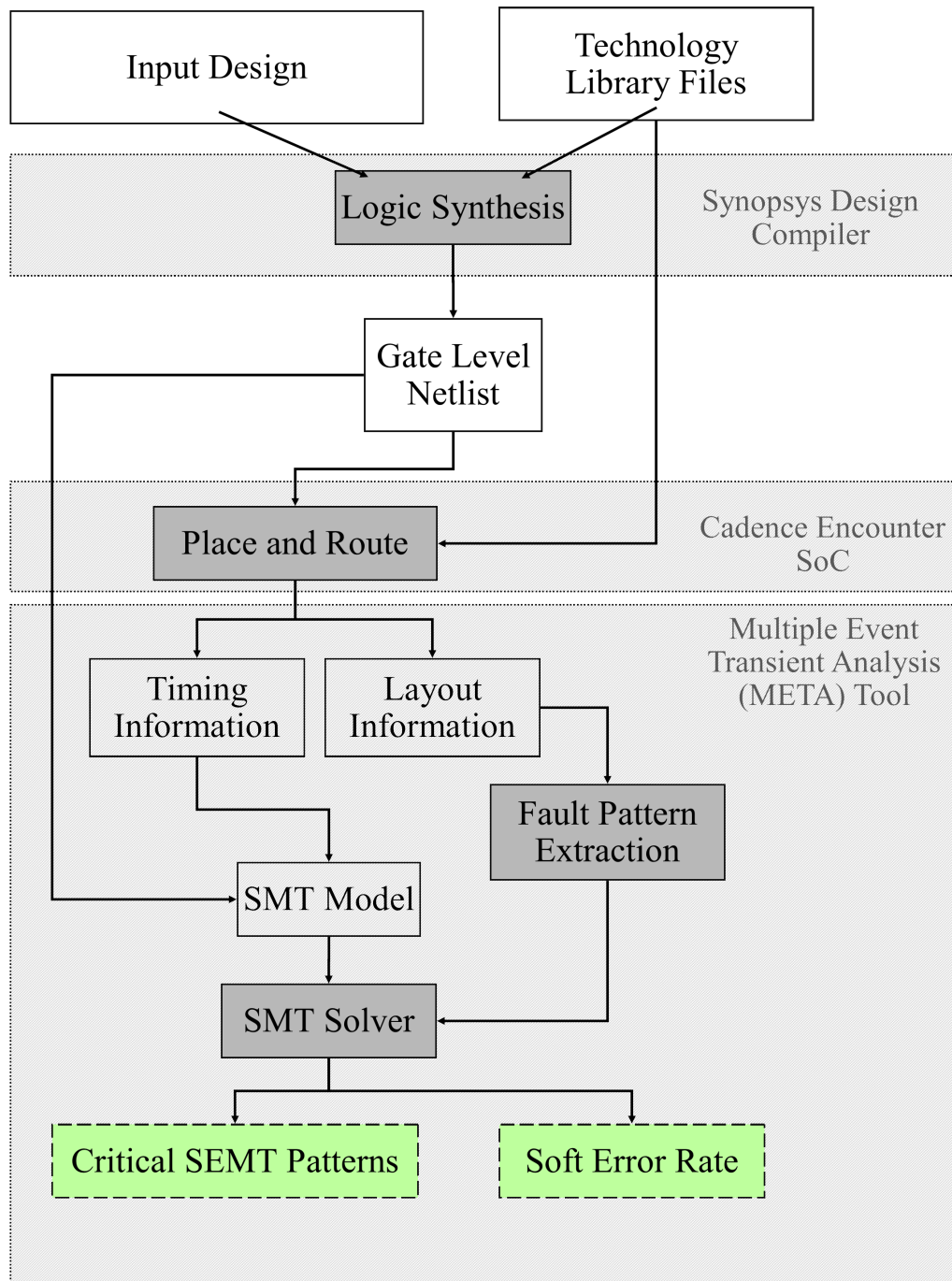


Figure 1.2: Overview of proposed framework.

to Multiple Event Transient

This published paper was the result of my effort to build the fully automated multiple-event analysis (META) tool. The idea was to apply a formal technique instead of simulation to check the vulnerability of the circuits against single event multiple transients. As this work is in continuation of previous work provided by co-author Sowmith Nethula, so I am responsible for the automation and standardization of this tool to work for any TSMC technology libraries. Also, SMT model for more gates is added to the SMT gate library of the tool. In this paper, I am solely responsible for all the experiments based on the TSMC 65nm technology library and interpretation of the results. As the first author, this work is published at the 20th IEEE International NEWCAS Conference.

This tool now includes a new feature that creates a design test bench automatically. Later on, this test bench along with delay information was utilized to perform simulation. This feature helps us to validate our tool by comparing the experiment findings from the META tool to simulation techniques. **Current version of this tool (META2.0) is fully supported by all TSMC standard cell libraries.**

Article II: *Layout-based Vulnerability Analysis of LEON3 Processor to Single Event Multiple Transients using Satisfiability Modulo Theories*

In this paper, I have extended a case study of LEON3 processor checking the vulnerability of various parts of the processor core. Processor core was synthesized at 65nm technology node and fault propagation analysis is performed using built META tool as mentioned in article I. To analyze the processor core, the tool was extended to add a new feature for removing feedback paths and topologically sorting. The co-author (Sowmith Nethula) offered help to utilize his work and extend it for standardization. This work was conceived and subsequently published at the 23rd International Symposium on Quality Electronic Design (ISQED) conference. The first author (Sowmith Nethula) is responsible for the characterization of the LEON3 architecture and for writing approximately 50

percent of the text. I am responsible for modeling and analysis, running experiments, and writing the rest of the text.

Article III: *Application-Specific Formal Approach to Estimate Soft Error Vulnerability of Register File using Satisfiability Modulo Theories*

This paper presents an application-specific approach to analyze the processor core against single event multiple transient faults using SMT theory. We calculate the vulnerability of registers in the processor while running a specific application on the core. This work utilizes the META tool to obtain the vulnerability factor of the register file module in the RTL design of the core. Later on, it was utilized to calculate register file vulnerability and program reliability. I am responsible to come up with the idea, and all the modeling, and experimental results.

1.3 Thesis Organization

The remainder of this thesis is organized as follows: Chapter 2 provide some preliminary information on the subject. We start by explaining the history of radiation effects and then followed by explaining single event effects. Chapter 3 discusses some of the related works in the field of analyzing transient faults and explains the motivation of this thesis. Chapter 4 presents the article *META: A Layout Based Tool to Estimate the Vulnerability of Digital Circuits to Multiple Event Transient*. In this publication, we present an automated in-house built electronic design automation tool for analyzing transient faults. Chapter 5 presents the article *Layout-based Vulnerability Analysis of LEON3 Processor to Single Event Multiple Transients using Satisfiability Modulo Theories*. In this work, we calculate fault propagation probabilities of various sub-modules of the LEON3 processor core, and also we demonstrate the most critical zones on the processor's layout against transients faults. Chapter 6 presents the article *Application-Specific Formal Approach to Estimate Soft Error Vulnerability of Register File using Satisfiability Modulo Theories*. In this work,

we extract the vulnerability of the register file of the core while running a specific program. We present the experimental result for three benchmark circuits: QuickSort, Matrix Multiplication, and Fibonacci Series. Chapter 7 provides general discussion about the present work which has been provided in chapters 4 through 6. It also summarizes the thesis and provides some future directions for future work.

Chapter 2

Background Information

The main objective of this chapter is to introduce the basic concept of radiation-induced effects in semiconductor devices. In this chapter, we explain the history of radiation effects and different types of single-event effects.

2.1 History of Radiation Effects

In 1975, the first study was reported by D.Binder [6] to discuss anomalies in communication satellite operation. The interaction of galactic cosmic rays with digital circuits was discussed. Using a scanning electron microscope, sensitive transistors which changed the state of the JK flip flop, are located. Later on, the effect of cosmic rays on memories was explored in 1979 by James F. Ziegler at IBM [7] in which the effect of package material on dynamic random access memory (DRAM) that emits alpha particles was studied. It was quickly followed by similar studies to discuss the radiation effects. In 1996, a researcher from IBM published an article [8] to present experimental results from 1979 to 1995. The purpose of these experiments is to study radiation-induced soft fails in electronics. A significant contribution was made to understand the mechanism by which a charged particle disrupts a circuit.

In 1995, aircraft manufacturers issued some guidelines to consider the effect of single event upsets (SEUs) due to radiation particles. International Electrotechnical Commission (IEC) also published technical specifications (TS 62396) in 2006 to accommodate atmospheric radiation effects via single event effects within all avionics electronic equipment. Thus, the study of the single event effects for circuits used in space, avionics, and military systems is essential to ensure a successful mission.

Several types of effects can be detected when a set of radiation particles interact within electronic devices through an exchange of energy. The influence of radiation effects on electronic equipment might produce a malfunction in the circuit's functionality. To apply strategies and mitigation methods to tolerate these faults and errors, their effects must be fully investigated. Given the high rate and smaller size of advanced technologies, the advanced system is projected to be more sensitive to radiation. As a result, having more information to assist the design engineer in using robust mitigation methods is important and necessary.

2.2 Single Event Effects

As mentioned in chapter 1, single event effects (SEEs) are caused by an energetic particle entering into a device's sensitive nodes, ionizing the material, and forming a track of electron-hole pairs. When an energetic particle hits a semiconductor, it loses energy due to interaction with the semiconductor lattice nuclei. The energy generated in the semiconductor makes electrons jump from the valence band to the conduction band based on its particle energy band and electron-hole (e-h) pairs are generated. This excess additional charge deposit in the sensitive node of the circuit causes unwanted changes in the voltage or current of the device.

Figure 2.1 illustrates the basic mechanism when a high-energy particle strike happens. As per the study by Baumann [4], a cylindrical trajectory will be formed after the particle

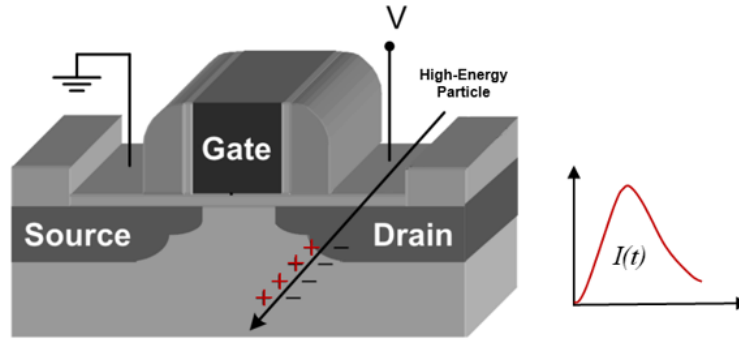


Figure 2.1: Energy Particle Strike on Transistor and resulting transient change in current.

strike whose radius is less than one micron. This cylindrical trajectory is extremely dense with electrons and holes and a high electric field will be created. Due to the electric field, electrons will be diffused in N-region and the hole will be absorbed in the P-substrate of the transistor. From the perspective of the circuit, a current pulse is generated in the corresponding node. Furthermore, voltage perturbation can be detected, which may cause this node's logic state to change and cause a soft error.

2.3 Types of Single Event Effects

Single event effects (SEEs) can be categorized into two categories (illustrated in Fig. 2.2: soft errors or hard errors. Soft errors are those errors that interrupt the device's regular operation without any physical damage to the device. These errors can be corrected by applying external input like reconfiguration or resetting the device. Soft error SEEs include single event upset (SEU), single event transient (SET), single event multiple transient (SEMT) and single event latchup (SEL). Hard errors, sometimes known as persistent errors, are another type of SEEs that cause physical damage to the system that cannot be repaired. When a hard error occurs, the device's function is permanently interrupted because the internal structure is compromised. We need to change the device in case of hard errors.

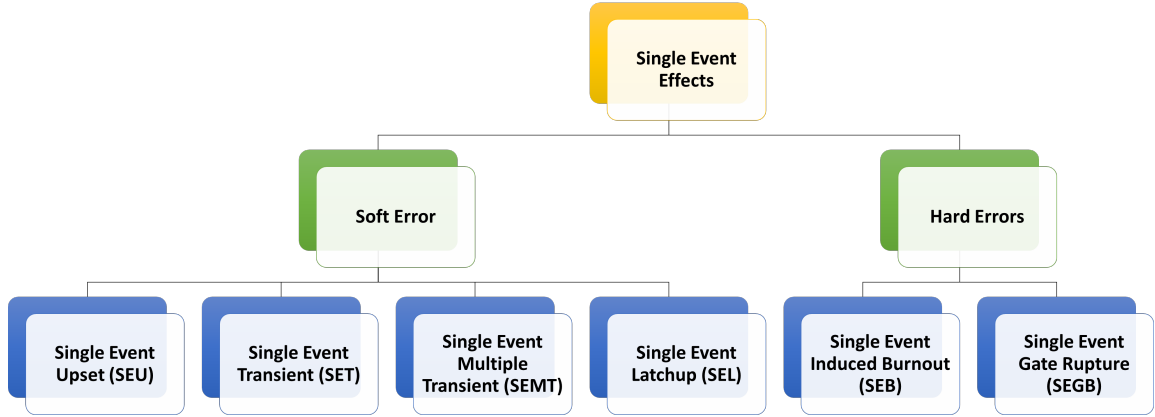


Figure 2.2: Categorization of Single Event Effects.

The hard error SEE category includes single event induced burnout (SEB) and single event gate rupture (SEGR). In the study of SEEs, both hard and soft errors are significant. But this thesis work is mainly focused on the study of soft errors: SET and SEMT.

2.3.1 Single Event Upset

Digital logic circuits can be classified into two types: sequential circuits or combinational circuits. Sequential circuits have memory elements such as flip-flops or latches to store data. SEEs act differently in these two kinds of circuits. There are no memory-like elements in combinational logic circuits. The collected charge will induce a transient change in current and voltage in the nodes, which we designate in the circuits as a single event transient (SET), which will be discussed in more detail in the section.

If an energetic particle strikes a sensitive node in the circuit and it changes the state of the memory element, it is called a single event upset (SEU) as illustrated in Fig. 2.3. First SEUs were observed from packaging material of dynamic random access memory (DRAM) as mentioned in section 2.1

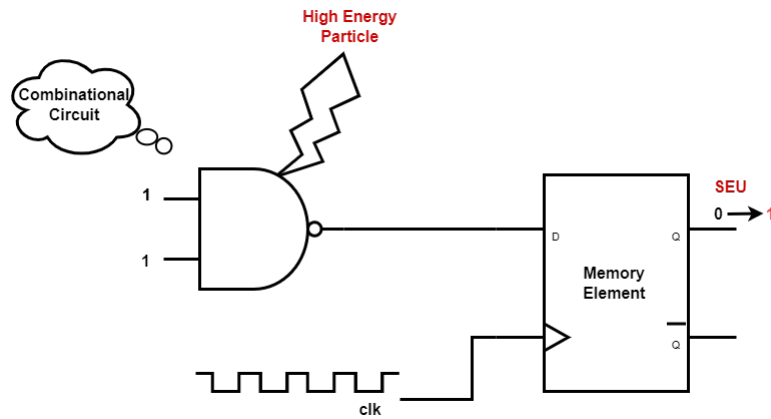


Figure 2.3: Single Event Upset in D Flip-flop.

2.3.2 Single Event Transient

Single event transient occurs when a radiation particle generates an erroneous signal pulse on non-memory elements such as a combinational circuit. Voltage transient generated by this phenomenon can propagate throughout the circuit and latched by sequential elements at the output, depending of the width of the transient pulse and fault masking mechanism. A fault generated by a transient does not always propagate to the output due to one of the following masking phenomena, including logical masking, electrical masking, and temporal masking.

Logical masking As briefly explained in the introduction section, logical masking occurs when a particle strikes a portion of combinational logic and reaches to an input of the logic gate whose output is determined by other controlling input values. An example is shown in figure 2.4 with the help of a NAND gate. If transient SET reaches an input of the NAND gate and another input of the gate has a controlling value (e.g. 0 for NAND logic gate), then, SET will be completely masked and output will not be affected by SET.

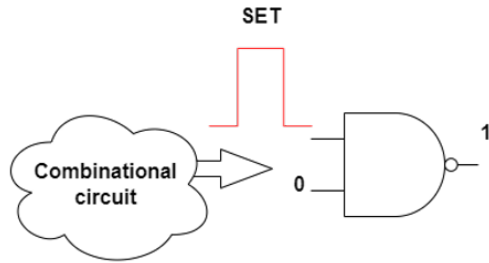


Figure 2.4: Logical Masking in NAND gate.

Electrical masking As the logic circuit in CMOS has limited bandwidth, thus transient pulse with bandwidth higher than the maximum cutoff frequency will be attenuated. It is possible that at some point in the circuit, the transient pulse will not affect the output of the corresponding gate. It is described with the help of a combinational circuit in Fig. 2.5.

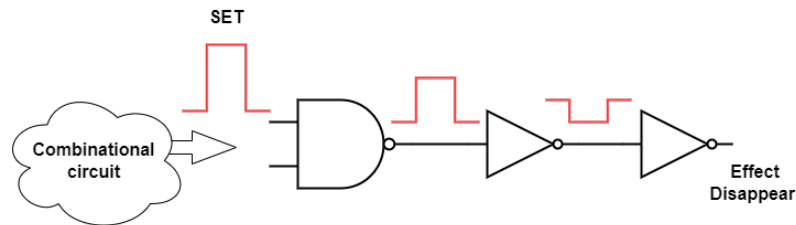


Figure 2.5: Example for Electrical Masking

Latching Window masking As the transient pulse propagates throughout the circuit and reaches a sequential element (e.g. flip-flop) at the output, it can reach outside of the clocking window. In this situation, the flip flop may fail to latch the transient pulse, resulting in no error. This is known as latching window masking. Figure 2.6 shows how SET reaches the flip-flop but not captured due to clock transition is not available at that time.

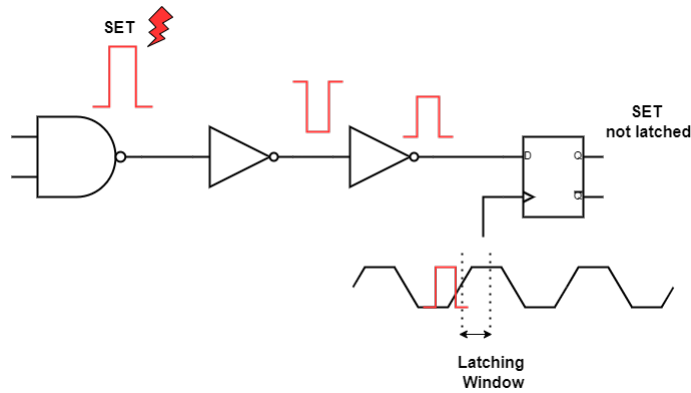


Figure 2.6: Latching Window Masking Example

2.3.3 Single Event Multiple Transient

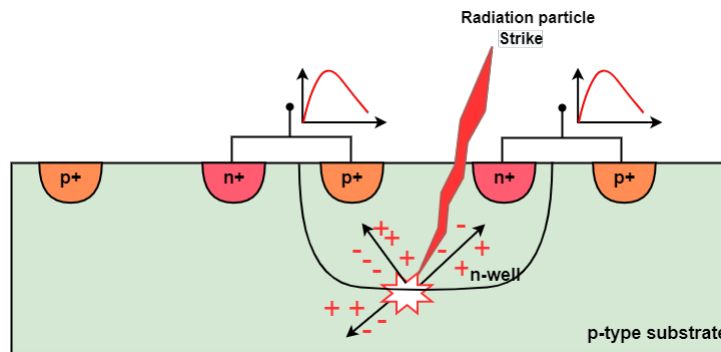


Figure 2.7: Single Event Multiple Transient Effect

When a single high-energy particle affects two or more adjacent nodes, a single event multiple transients (SEMT) occurs. As illustrated in Fig. 2.7, a single radiation particle generates two transient pulses because two adjacent transistors will collect the charge produced by the particle. Therefore, transient fault will be generated in multiple adjacents placed logical gates. This problem becomes more critical as the geometry is aggressively scaled down in the deep sub-micrometer era. In sequential circuits, it will lead to multiple bit upset (MBU) problem.

Chapter 3

Literature Review

Study of transient faults and their potential mitigation are being conducted with the goal of ensuring the reliability of electronic circuits. The first approach to evaluate circuit vulnerability against transient faults is through dynamic surface radiation experiments [9] [10] [11] [12] [13]. In radiation-based experiments, a prototype of the device is fabricated at the required technology node, then the device is exposed to different kinds of radiation tests to analyze its behavior. In [10] [11]-[13], test chips were fabricated at a 65-nm technology library and heavy ions experiments were conducted to analyze the effect of single event transients (SETs) and single event multiple event transients (SEMTs). In [9], heavy-ion experiments at two different technology nodes (e.g. 130-nm and 90-nm) were conducted to obtain the distribution of SET width. Although radiation-based testing provides the most accurate results as the real radiation environment is being applied to the testing device, facilities where radiation experiments are performed, are expensive. Also, we need a manufactured device prototype to perform those experiments.

Another method to analyze SETs or SEMTs, is circuit level fault propagation in which faults are injected into the circuit at logical or electrical levels. The fault propagation behaviour is analyzed using emulation [14] [15] or simulation-based techniques [5] [16] [17] [18] [19]. The benefit of this method is that we don't need a prototype of the device and no

hardware equipment or testing facilities are required to perform radiation testing. It reduces the cost of testing. These techniques are classified into two major categories as follows:

3.1 Netlist Based Analysis

In [5] [17]-[19], authors present different approaches to analyze multiple transient faults in logic (combinational and sequential) circuits. The most probable locations of multiple transients are extracted based on an output of neighboring gates. Gates and their fanins or fanouts neighbors are possible sites for transient fault generations. Inputs stimulus for the logic circuit are generated through simulation of the netlist to check the propagation of introduced transient faults. Error probabilities or soft error rates (SERs) for fault propagation are estimated as the reliability of the circuit. The major drawback of this technique to estimate SEU due to transient faults is related to the fact it is based on the logical level netlist to identify the possible error locations and neglects layout-based adjacency of error locations. As we will explain later in this thesis work, such assumptions can drastically underestimate the circuit SER. The distribution of affected error sites and the number of affected gates are entirely dependent on layout-level characteristics and cannot be derived from the logic-level netlist.

3.2 Layout Based Analysis

In the references [20] [21] [22] [23] [24] [25], authors discuss techniques to evaluate the soft error rate of digital circuits against multiple event transients by considering layout information. The possible error sites or single event multiple transient (SEMT) patterns are extracted based on the circuit layout. Transient faults are injected at high abstraction levels and analyzed using the circuit-level simulation method. In [21] [22], SEMT patterns

are extracted based on patterns observed on SRAM-based memory arrays, but circuit structure for SRAM memory arrays and combinational or sequential circuits are different and it would give us inaccurate results while calculating SER. [23] [24] [25] discuss layout-based fault injection approach to calculate SER. SET pulses are injected into corresponding hitting nodes of the circuit netlist and Monte Carlo simulations are conducted to check the fault propagation throughout the circuit. In the simulation, a limited number of different random primary input vectors are applied. For each input vector, injected transient pulse is analyzed and finally failure rate or SER is calculated.

However, the simulation-based technique has major drawbacks. This technique is not exhaustive, because testing is limited to very few input vectors which reduce its overall accuracy. Thus, this approach is restricted in terms of accuracy, in addition to the significant time necessary to model or check a scenario of injected faults. Few studies [26] [27] [28] have been published on formal approaches to investigate soft errors at the logical and higher abstraction levels. These studies provide more insights into the vulnerability of digital circuits against transient faults. That's mostly due to the fact that they are exhaustive and are not restricted by the number of test vectors, as simulation-based techniques. The authors in references [26]-[28] discuss the analysis of single event transient propagation at the register transfer logic level. In these works, a fault model was not developed by considering the layout information of the circuit, and also single event multiple transient (SEMT) faults were not considered.

In order to increase the accuracy and to include layout-adjacency information in fault analysis, we developed a framework to estimate the vulnerability of the design at the gate level using a formal technique. SEMT fault patterns are extracted based on the layout information and analyzed using Satisfiability Modulo Theory.

Chapter 4

Article I: META: A Layout Based Tool to Estimate the Vulnerability of Digital Circuits to Multiple Event Transient

Authors: Vivek Bansal, Otmane Ait Mohamed, Sowmith Nethula

Conference Name: 20th IEEE International NEWCAS Conference (NEWCAS'22)

Abstract: In this paper, an Electronic Design Automation tool to estimate the vulnerability of digital circuits (META) against transient faults is presented. META analyses both single event transient (SET) and single event multiple transient (SEMT) faults considering the layout of the circuit and the fabrication technology. Unlike simulation based tool, META uses satisfiability modulo theory to calculate soft error rate of the analyzed circuits. The complete analysis flow is automated and our results outperform those obtained by simulation by a average factor of four.

Index Terms - EDA, transient fault, masking, soft error, SEMT

4.1 Introduction

As CMOS integrated circuit (IC) technology progress by reduction in feature size, providing more integration density and higher operating frequency while maintaining less power consumption and low cost per transistor, it introduces more challenges for design engineer to check the reliability of integrated circuits. Due to reduction in transistor size as technology progress, the single event induced soft error becomes a major threat for reliability of the circuits.

When a radiation particle strikes a digital circuit, it can generate a single event transient (SET) if affecting a combinational standard cell and a single event upset (SEU) if affecting a memory element. The higher is the operation frequency of a circuit, the higher is the probability of propagation of the SET. In such cases, SETs become critical and should be considered in the digital design process. In addition, it is probable that single transient pulse affect more than one logical nodes in the circuit that results in single event multiple transient (SEMT) faults due to charge sharing mechanism between adjacent nodes [29]. The Number of gates affected by single transient fault depends on the energy of the particle and the physical characteristics of the circuit [30] [31].

These transient faults can be masked by three masking factors: logical, electrical and latching window masking.

1) **Logical masking:** when transient fault reaches to non-sensitive path of the circuit, then it will be logically masked.

2) **Electrical masking:** when transient fault width is not enough to propagate to final output of the circuit, then it will be attenuated or electrical masked.

3) **Latching window masking:** when fault fails to satisfy the setup time and hold time constraints of the flip-flop to capture fault at output, it will not affect the circuit.

In this work, we present an automated single event multiple transient fault analysis tool for digital circuit called META. Unlike simulation based tools, META is based on

satisfiability modulo theory to calculate the soft error rate of combinational circuit and taking into account the three masking factors. To evaluate the effectiveness of the proposed tool, we performed several experiments on ISCAS'85 benchmark circuits using TSMC 65nm General purpose (GP) Standard Cell Library files. Soft error rate of each design is calculated by META tool.

The reminder of this paper is organized as follows. Section II discusses the background of SEMT propagation in CMOS circuits. In section III, the analysis flow of the META tool is presented. The results of the analysis are discussed in Section IV, and finally, section V concludes the paper.

4.2 Related Art

There are several studies to estimate the soft error rate, it can be classified into two categories: netlist based analysis and layout based analysis. The papers [5] [18] provide a strategy for estimating the SER of a circuit in the presence of SEMTs. The analysis in this paper takes a netlist-based adjacency into account when extracting the SEMT faulty sites, ignoring layout information such as cell position, orientation, and distances between cells, which are significant in deciding the possible combination of nodes affected by transient pulse leading to a SEMT. The major disadvantage of netlist based techniques in estimating SER due to transient faults is that it utilizes logic-level netlist to identify the possible fault error sites, it neglects layout level adjacency of error sites. When particle hits the circuit design, it disrupts the logic state of one or more than one adjacent nodes based on its layout.

The references [21] [22] [23] discuss different methods for SEMT vulnerability analysis. Papers [21] [22] propose a layout-based SER estimation technique by taking account into single event and multiple event transient faults. The SEMT patterns in these papers are characterized based on the patterns observed on an SRAM device subjected to radiation

testing. However, the circuit structure of an SRAM and combinational circuits are different and would result in inaccuracies during the estimation of SER. References [23] [24] [25] also discuss SER estimation method based on layout and run the analysis based on simulation techniques. SEMT fault error sites are extracted based radiation experiment at particular technology node. The disadvantage of simulation-based technique is that it requires a large amount of time to check for all possible input possible combinations. Generally, it is tested against a limited number of input combination, which reduces the accuracy of the method. Thus, we present more effective and efficient layout based electronic design automation tool to estimate vulnerability of digital circuits using Satisfiability Modulo Theory (SMT).

4.3 META - Multiple Event Transient Analysis Tool

Multiple Event Transient Analysis (META) is an automated EDA layout based tool to estimate the vulnerability of the circuits against single event transient (SET) and single event multiple transient (SEMT) faults. META identifies multiple transient fault patterns based on design layout and checks its propagation throughout the circuit. META is based on TCL scripting written in Python programming language, which facilitates the modularity of the tool.

The tool is composed of four main parts as shown in Fig.4.1: pre-characterization to set up library files, synthesis and place route, the SEMT fault pattern identification and fault propagation analyzer which automatically builds SMT model for synthesized circuit design and calculate its soft error rate by taking account into both single event transient (SET) and single event multiple transient (SEMT) faults. Also, it will generate a graphical layout representation of the circuit to show critical patterns on its layout. In tool, we parse register transfer level (RTL) design description written in VHDL or Verilog language. In configuration set-up, we can set type of SEMT fault pattern discussed in section 4.3.3.

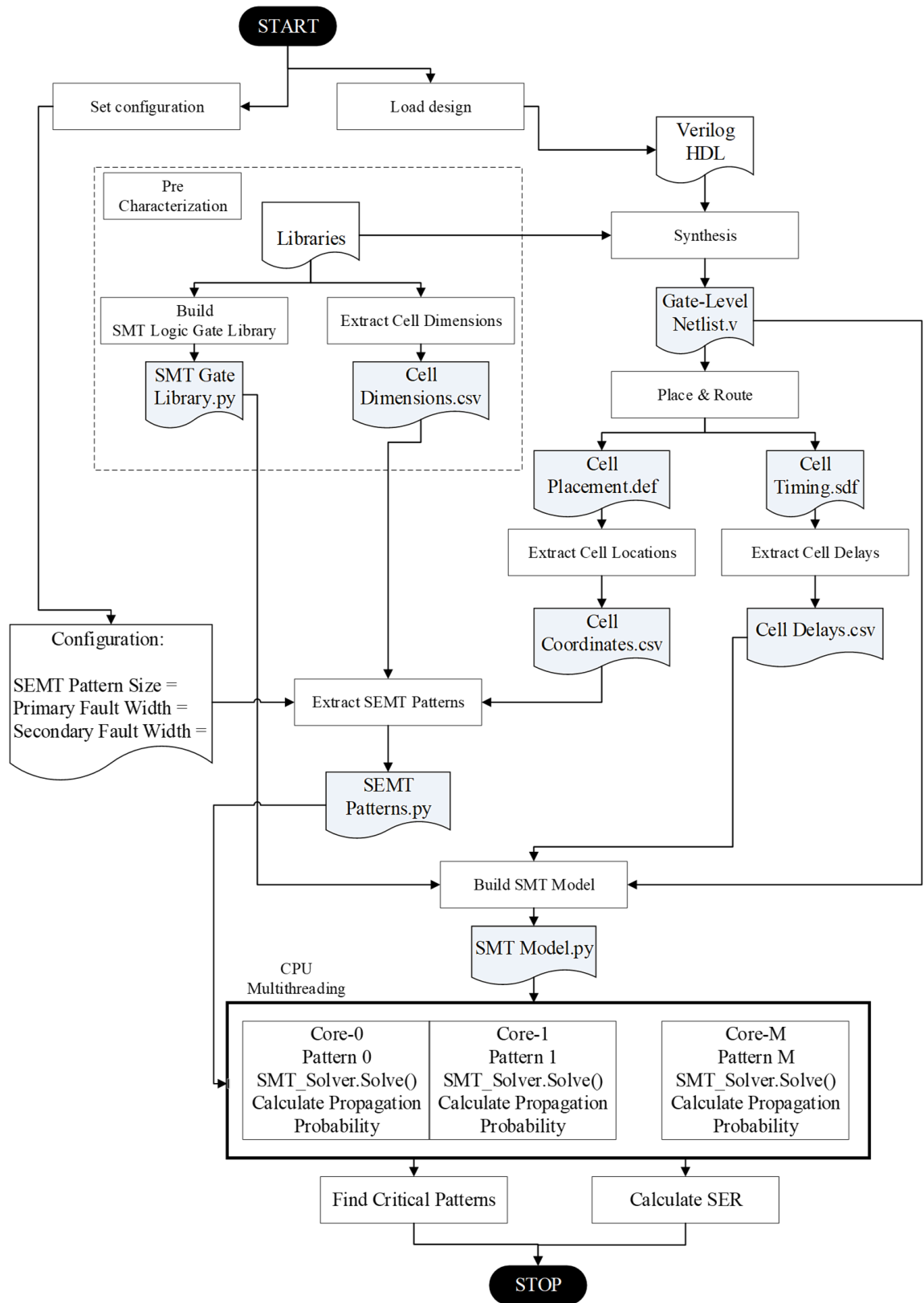


Figure 4.1: META - Proposed Tool

Also, we can inject transient fault widths as per radiation experiments [11] [12].

4.3.1 Pre Characterization

Prior to start of the transient fault propagation analysis in META tool, we characterized the library files provided in TSMC General purpose (GP) Standard Cell Libraries and identified all the standard cells. For each of the identified standard cell from libraries, we build a Satisfiability Modulo Theory (SMT) formula based on logic gate's behavior and transient fault propagation analysis. To build SMT model, we used python's PySMT API.

When transient fault propagates through a logic gate, the effects of the fault at the output of the gate can be masked by having a controlling value at another input of the gate (e.g. 1 for a OR gate). This phenomenon is called logical masking and such logical masking conditions for all the combinations of fault and fault free values at the gate input are modeled as a linear arithmetic SMT formula. The transient observed at a gate during a particle strike can possibly be attenuated and may completely disappear while propagating through a gate due to its electrical property. This property is called electrical masking. To model the electrical masking effects, the gate timing details are extracted and a linear arithmetic SMT formula is constructed that calculates the transient width at the output by considering the transient width at the input and the propagation delay of the gate through which the transient is propagating. An advantage of the SMT formulation for calculating the transient width at the output that it also includes the effect of transient pulse width broadening phenomenon besides the pulse attenuation or masking effects. If any of the injected SEMTs fails to reach the input of a state element within its latching window time, then the fault is temporally masked. This property is known as latching window masking. For a transient to be not temporally masked, it should propagate through the circuit and arrive within the latching window time of the state element or register which is the set-up time before the active edge of the clock. All these parameters e.g. logic value, propagation

delay, set-up & hold time are taken into account while building SMT model for each gate.

SMT formula for identified each gate will be saved in gate library (*SMT_gate_library.py*) of the tool. Also, the dimension for each cell is extracted from the LEF (Library Exchange Format) file available in the TSMC libraries and are saved in a comma-separated values file (*Cell_Dimension.csv*).

4.3.2 Synthesis and Place & Route

To perform the transient fault analysis, a register-transfer level (RTL) structural description of the design written in VHDL or verilog language is loaded into the META tool. The tool automatically invokes Synopsys Design Compiler to perform gate-level synthesis of the circuit design. TSMC 65 nm General Purpose (GP) Standard Cell Libraries are utilized to perform synthesis at 65-nanometer technology. After the logic synthesis of the design, gate-level netlist (*gate_level_netlist.v*) and a delay constraint file is extracted.

To extract layout information, Cadence Encounter RTL to GDSII is invoked to build layout of the synthesized circuit design by utilizing gate level netlist and delay constraint file. All standard cells in the design are placed on the layout and design is routed to extract the RC parameters for the timing analysis. Based on the layout and placement of cells, details related to the physical placement of cells (*Cell_Coordinates.csv*) and the gates exact timing (*Cell_Delays.csv*) are extracted. These details are required to better identify and model the masking effects. The knowledge of the physical adjacency of the gates helps us accurately extract SEMT sites, while knowing the timing details helps us model the electrical and temporal masking effects while propagating the transient faults throughout the circuit.

4.3.3 Transient Fault Pattern Identification

When a particle strikes at circuit node, it changes the logical behavior. To analyze the impact of transient fault at logical level, details of observed behavior from radiation experiment are utilized presented in references [11] [12]. In these references, radiation experiments are performed on different chain on inverters and XOR gates at 65nm technology node. It was observed that maximum five cells in horizontal direction and two cells in vertical direction are effected by a single transient fault pattern. This is due to presence of well between every vertical layer in the layout and exceptionally rare chance for charge sharing across the horizontal direction.

In this step, META will identify all possible SEMT fault patterns in the design. Based on the design layout, dimension of the cells and configurations (*SEMT fault size and fault width*), it will inject the fault at different nodes in the design and extract the cells effected by each fault separately. Each of these possible combinations is itemized with a number and a list for all design fault patterns (*SEMT_pattern.py*) will be saved internally. It will be utilized when fault propagation is analyzed for each injected fault.

4.3.4 Fault Propagation Analyzer

Fault Propagation

Based on the SMT gate library, the tool generates an SMT model (*SMT_Model.py*) for the synthesized circuit design by utilizing timing information extracted in place & route 4.3.2. To analyze the fault propagation to final output of the design, result in soft error, it assertions includes information about each SEMT pattern and its effected gates are injected. After that, it automatically invokes SMT solver to give us a model for which SEMT fault will be propagated to final output. If solver provides a satisfiable assignment in the model, it means fault is propagated to the circuit output for a particular SEMT fault pattern and if

not, then it is possible that fault has been masked due to any of masking factors (logical, electrical or latching window).

SER Calculation

In this step, tool will check all satisfying assignments for each SEMT fault pattern (Fp). The probability that an injected SEMT fault pattern (Fp) propagates to the output is calculated as shown in the equation (1).

$$Pp(Fp) = \frac{Number_Satisfying[Fp]}{N} \quad (1)$$

Here, the *Number_Satisfying* is the number of satisfiable assignments allowing the SEMT pattern Fp to appear at the output and *N* is the total number of possible input vectors.

The Soft Error Rate (SER) is calculated as the sum of propagation probabilities for all the fault injection scenarios in the design, as shown in the equation (2). For *n* number of fault injection scenarios (Fp1, Fp2, ..., Fpn), the SER is

$$SER = \sum_{i=1}^{i=n} Pp(Fp_i) \quad (2)$$

Graphical Plotter

Based on identified SEMT fault patterns those propagated the injected transient fault to final output, the tool will generate a design layout plot using cell location and cell dimension information. It shows vulnerable regions over the IC layout which is more sensitive to transient faults. Example is provided in figure 4.3.

4.4 Test And Results

To validate the scalability of proposed tool, we have performed several analytical experiments on different ISCAS'85 benchmark circuits written in verilog language. TSMC 65 nm General purpose (GP) Standard Cell Libraries are utilized in these experiments.

Table 4.1 reports the estimated soft error rates for all benchmark circuits. Benchmark circuit name is mentioned in first column. Second column tell us number of logic gates in synthesized circuit design. Third and fourth columns represent number of inputs and outputs in the circuit design. In last column, soft error rates are listed based on fault transient patterns.

Table 4.1: Characteristics of ISCAS'85 benchmark circuits synthesized using TSMC 65nm and SER

Benchmark	Gates	Inputs	Outputs	SER
C17	7	5	2	0.903
C432	180	36	7	0.651
C499	324	41	32	0.522
C880	512	60	26	0.417
C1355	654	41	32	0.598
C1908	689	33	25	0.621
C2670	1269	233	140	0.457
C3540	1436	50	22	0.354
C5315	2307	178	123	0.674
C6288	3579	32	32	0.794
C7552	2553	207	108	0.391

Each SEMT propagation analysis is unique, so we use multithreading to estimate the propagation probability for numerous patterns in parallel. All the experiments were conducted on a high-performance compute server powered by Intel XEON processor cluster with 160 CPU cores, each clocked at 2.83 GHz and a system RAM memory of 1 TB.

4.4.1 Performance Analysis

To validate our proposed layout based SMT approach, we investigated the performance in terms of computation time. We compare the time to estimate vulnerability using proposed tool with the traditional fault injection simulation method. In simulation method, SEMT faults are injected into the faulty version of the circuit and checked for propagation at the error flag for randomly selected input vectors. In the simulation model, fault is injected in verilog netlist based on extracted SEMT patterns. Standard Delay format (SDF) file is also included in the simulation in order to accommodate effect of electrical masking. Mentor Graphics QuestaSIM tool is used to perform simulation.

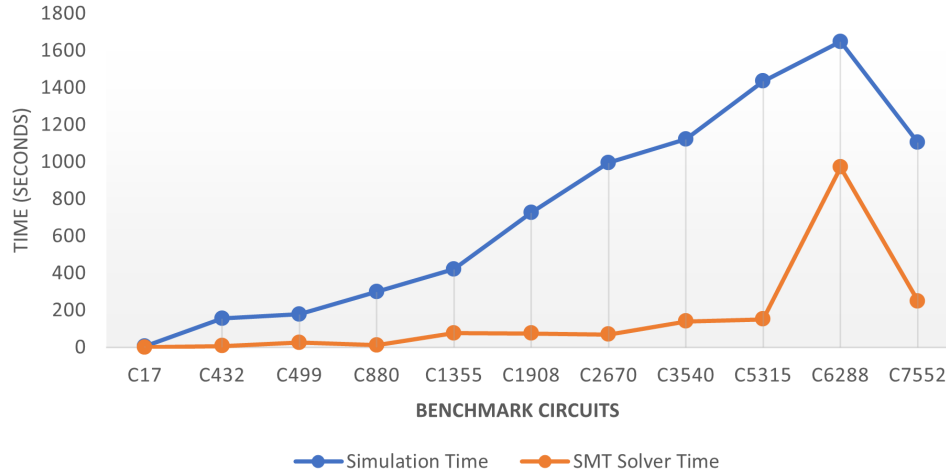


Figure 4.2: Comparison of simulation time with proposed methodology while calculating SER

Fig. 4.2 shows time comparison between time taken by proposed tool and simulation time. As the data showed in the figure, simulation time using the traditional fault injection method increases significantly, especially for complex larger circuits. On the other hand, time in our proposed tool increases slowly with circuit size. The proposed methodology is more effective for circuit design having larger and complex circuits. Due to implementation of multithreading technology, computation time can be improved more by utilizing more

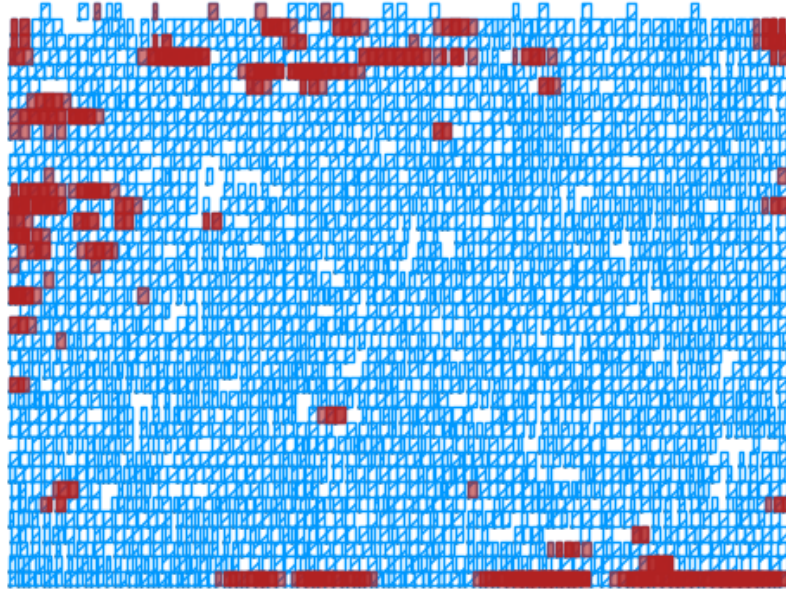


Figure 4.3: SEMT Faulty Error Sites

computer resources like increasing number of CPU and RAM memory. It is reasonable to expect that the proposed approach will become significant in the vulnerability estimation of combinational circuits as computer performance continues to improve and parallel computing continues to grow in popularity.

4.4.2 SEMT Fault Error Sites

Based on the vulnerability estimation for each identified SEMT patterns, we showed the IC layout to demonstrate most vulnerable regions on the design layout. Fig. 4.3 presented the design layout of benchmark circuit C5315. Darker region corresponds to most vulnerable area and less dark region represents less vulnerable area against SEMT faults. This type of analysis can only be accomplished using formal methods, and it's extremely difficult to do so using simulation-based fault injection approaches because the design must be exhaustively simulated. In the process of radiation hardening, this faulty error site information can help the design engineer to locate the SEMT fault sensitive regions. It will help to build and place transient filters to minimize the circuit failure in case of transient faults.

4.5 Conclusion

A layout based multiple event transient fault analysis tool, which allows to estimate the vulnerability of the circuit design was presented. Main feature of this modular tool has allowed us to easily run the vulnerability analysis check for any circuit design written in VHDL, verilog or system verilog language. In experimental work, we analyzed circuit designs from small program (c17) to larger program (c7552). These results allow us to identify the layout area which is most susceptible to propagate SETs or SEMTs. It helps designer engineer to concentrate on the most critical design parts and use it for radiation hardening to reduce the effect of radiation environment.

Furthermore, the proposed methodology can be applied to other technology nodes (i.e. 45nm or 15nm) as well. Also, it can be utilized to analyze and extract vulnerability of sequential benchmark circuits. The resulting findings will be reported in our future research works.

Chapter 5

Article II: Layout-based Vulnerability Analysis of LEON3 Processor to Single Event Multiple Transients using Satisfiability Modulo Theories

Authors: Sowmith Nethula, Vivek Bansal, Ghaith Bany Hamad, Otmane Ait Mohamed

Conference Name: 23rd International Symposium on Quality Electronic Design (ISQED'22)

Abstracts: With the reduction in transistor size, the radiation-induced soft error phenomenon in electronic circuits has become a significant reliability concern. In newer technologies, a high energy particle strike can affect multiple cells resulting in Single Event Multiple Transients (SEMTs). For an event affecting n cells horizontally, the SEMT pattern will be aliased as SEMT $_n$. In this paper, we present a framework to estimate the SEMT vulnerability of the LEON3 processor by using Satisfiability Modulo Theories. This framework considers layout-based cell adjacency as error sites and includes the Electrical,

Logical masking factors while estimating the vulnerability. The results indicate an overall average SEMT fault propagation probabilities of 30% and 54% due to the effects of SEMT3 and SEMT4, respectively. The total time to analyze the fault effects following our methodology is approximate 8.5 hours for the LEON3 ASIC Core on a server with 160 x Intel(R) Xeon(R) 10-core E7-8870 CPU running at 2.40GHz and with 1 TB of Main Memory.

5.1 Introduction

When a high energy particle strikes a transistor, it alters the logic value of the corresponding gate, resulting in a single event transient (SET). With the reduction in transistor size due to rapid scaling technology, it is possible that a single particle hit can cause transients to occur in two or more physically neighboring nodes, resulting in a Single Event Multiple Transient (SEMT). Several recent research has raised concerns about SEMT's influence on the system reliability [11] [13]. One reason for this concern is the fact that the set of gates involved in a SEMT can belong to different logic paths, making the implementation of error mitigation techniques, e.g., error-correcting codes (ECC) and triple modular redundancy (TMR) extremely difficult [32]. Therefore, an accurate method to analyze the vulnerability of embedded circuits due to SEMTs is crucial.

Since processors have a critical role in the development of embedded systems, the LEON3 processor is of great interest for embedded designers. LEON3 provides an accessible, affordable and non-proprietary CPU architecture capable of meeting the requirements for efficiency, software compliance and low cost. In our previous work, we model the SET fault propagation for combinational circuits using an SMT problem. We extend the modelling of SET propagation to SEMT propagation by including the layout details and also identifying the most critical sites on the LEON3's layout.

Our results show an estimated overall average SEMT fault propagation probabilities of 30% and 54% due to the effects of SEMT3 and SEMT4. We also present a layout of the

circuit indicating the highly critical SEMT error sites. The remaining paper is organized as follows. Section 5.2 lists the related works. Section 5.3 describes our proposed framework to analyze SEMTs, identification of error site, SMT modelling of the design, and the SEMT propagation. We present our analysis results in Section 5.4. Finally, we conclude in Section 5.5.

5.2 Related Works

The existing soft error vulnerability estimation techniques could be classified into radiation testing experiments and circuit-level fault propagation techniques. Several of the radiation testing experiments study the error propagation related to SETs and SEMTs. In [11] [12] [13], the radiation experiments were conducted on a chain of inverters at 65-nanometers technology to study the effects of SETs and SEMTs respectively. The radiation based test methods provide the most accurate results as the real radiation environment is being applied to the testing device, the test facilities are not only expensive but also requires a manufactured device prototype. As a result, a low-cost method for calculating a design's vulnerability is required.

Unlike radiation testing, the circuit-level fault propagation techniques do not require a device prototype. In such techniques, simulation-based fault injection methods are commonly employed. In these techniques, the faults will be introduced to the circuit described in hardware description language (HDL) and observe its effects. Using simulation methods, the effect of single event upsets (SEUs) on the LEON3 processor is investigated in [33] and [34]. The impact of SEUs and the time required to recover from faults in LEON3 processor are studied in [35]. The authors in [36] perform a study analyzing the effects of single-event effects in the LEON3 processor under specific loads using simulation techniques. Usually, in simulation techniques, the error sites are extracted from the layout information based on results from radiation experiments [21] [37]. All simulation-based

techniques to study single event or multiple events fault require a large amount of time to run all the possible input test vectors. If the fault propagation is tested for a limited set of inputs, it reduces the accuracy. The formal techniques have a middle ground between computation time and accuracy and exhibit robustness in terms of accuracy compared to the random stimulus simulation techniques. Some of the works that analyze fault propagation using formal techniques are [38] and [28]. In [38] [28], vulnerability of the design is calculated at early stage of the design or register transfer level (RTL) level. However, this RTL level design analysis model was not developed to consider the post-layout information of the circuit design. Thus, we developed a framework to estimate vulnerability of the design by using post-layout information at the gate level.

5.3 Proposed Method

We present a brief overview of the proposed methodology in Fig 5.1. The proposed technique starts by synthesizing the register transfer level (RTL) description of the design to a gate-level netlist using specific technology library files. A layout of the synthesized design is generated and utilized to extract the possible SEMT error sites. An SMT model of the design is built using the synthesized gate-level netlist. The gate level timing details are also extracted for the netlist to include the propagation delays in the SMT model. Finally, the propagation of the possible SEMT error sites is analyzed using SMT solver (Z3).

5.3.1 LEON3 Architecture, Synthesis, and Place & Route

LEON3 is a 32-bit SPARC V8 architecture (IEEE-1754) based processor, available freely under a General Public License (GPL) license for academic research purposes. The intellectual property (IP) core provides several template designs, each for various target implementations [39]. In our analysis, we use the ASIC implementation of LEON3 as

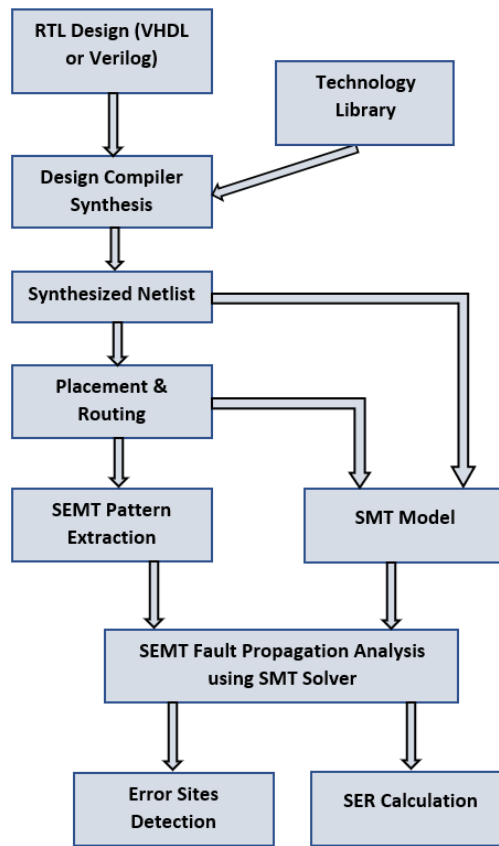


Figure 5.1: The proposed methodology for SEMT modelling and propagation analysis.

it allows us to synthesize and generate the netlist of the processor at a given fabrication technology. During the synthesis step, we retain the component hierarchy by setting the appropriate compiler options. This would allow us to create the SMT model for each of the sub-component in the synthesized LEON3 separately. At the end of the synthesis, the processor has 212 modules, with a total of 633,738 standard cell instantiations. Fig. 5.2 shows the LEON3 processor's hierarchy view seen after the synthesis step. We generate the layout only for the modules in the LEON3 processor core due to the license limitations of the layout tool, which limits us to place no more than 50,000 standard cell instances. Table 5.1 shows the details of submodules in LEON3 core. The core consists of an Integer Unit (IU), 32-bit hardware signed number multiplier (MUL) and a 64 to 32-bit hardware division unit (DIV). During the synthesis configuration of the LEON3 IP, the co-processor, and floating-point unit are set to inactive and are not synthesized.

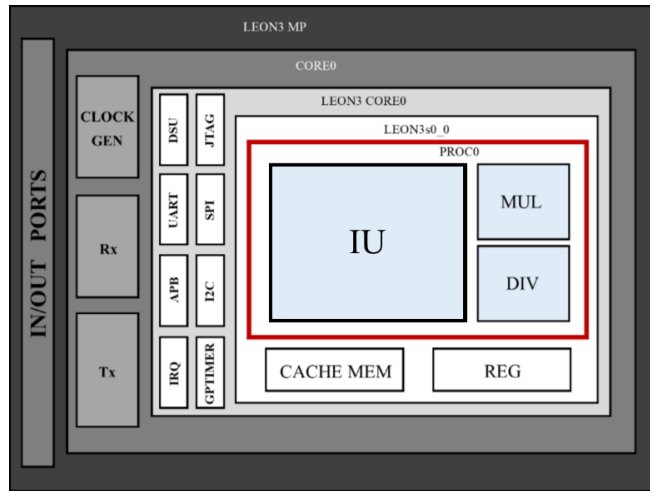


Figure 5.2: LEON3 processor's hierarchy view seen after synthesis.

Table 5.1: Characteristics of sub-modules in the LEON3 core

Module	#Inputs	#Outputs	#DFF	#Logic_Gates
IU	1450	2009	1401	32360
MUL	69	66	90	4958
DIV	107	38	79	1105

5.3.2 Single Event Multiple Transient (SEMT) pattern identification

In our methodology, we follow the results presented in [11] [13] while identifying the SEMT error sites. The results in these papers show that the predominant SEMT patterns comprise of five cells in the horizontal direction and two cells in the vertical direction. This minimal spread in the vertical direction was explained by stating that the charge sharing across wells is exceptionally rare due to the presence of a potential well (power rail) between every vertical layer in the layout of a chip. For an event affecting n cells horizontally, the SEMT pattern will be aliased as SEMT n . We characterize these results and extract the patterns on the LEON3 processor layout.

As an example, consider a high energy particle striking the node 'P' results in the pattern SEMT3 (see Fig. 5.3(A), (B)). Considering the dimensions of the cells (Inverter) in the test chip, area for a specific SEMT pattern can be calculated. Since the test chip used in the radiation experiments was fabricated at 65-nanometer technology, we utilize the TSMC 65 nm general-purpose (GP) Standard Cell Libraries to identify the dimensions of the inverter module and calculate the SEMT pattern area.

We utilize the SEMT event area onto the design layout of LEON3 processor core, and identify the cells encapsulated by that perimeter as a SEMT pattern (see Fig. 5.3(C)). [11] states that the multiple transients generated due to a single ion strike have different pulse widths. These pulses are characterized as primary and secondary transients based on their sizes. In our extracted SEMT patterns, we assume the node **P** as the primary transient site and the remaining nodes as secondary transient sites. During the propagation analysis, we inject appropriate fault widths into the gates according to the pulse width distribution presented in [11].

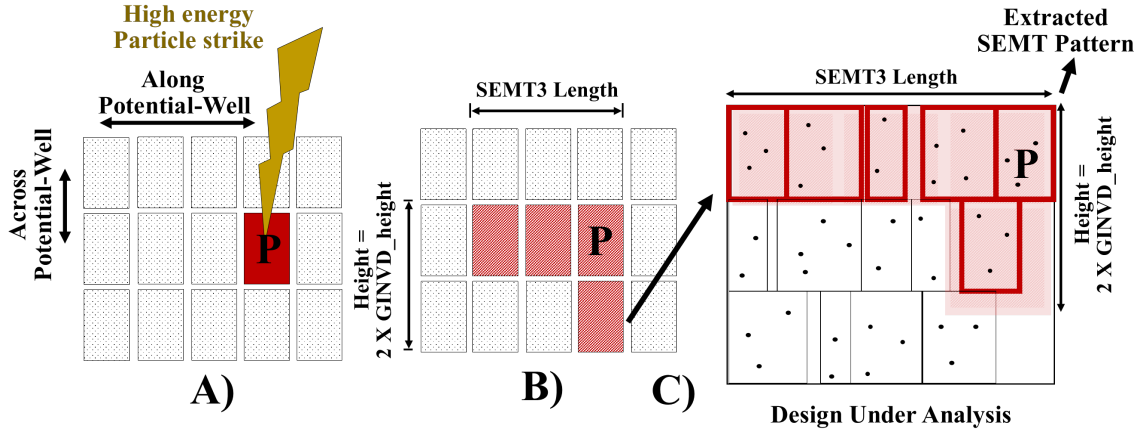


Figure 5.3: Extraction of SEMT Patterns.

5.3.3 SMT Modelling of Logic Gates

The modelling starts by translating all the standard logic gates into their equivalent SMT model. Each standard gate's logic behaviour and its fault propagation mechanism are converted into a symbolic formula using a mix of boolean and arithmetic operators. In addition to Boolean operators $\wedge$ (And), $\vee$ (Or), $\neg$ (not), we use a conditional operator *Ite* and a set of arithmetic operators *GT* (Greater Than), *Min*, *Max*, *Plus*, *Minus* to include electrical masking behaviour while modelling a logic gate. Fig. 5.4 shows an example of a 2-input NAND gate with additional annotations facilitated for SMT modelling.

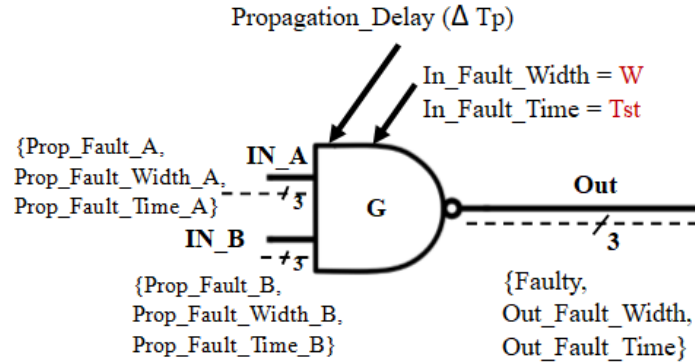


Figure 5.4: An example of 2-Input NAND gate in the proposed SMT modelling.

Each signal is an array of four variables describing the logic value, fault state, the

width and starting time of the fault. The variables IN_A and IN_B represent the logic values of the signals. A signal is faulty if the variable $Prop_Fault$ is true. The variables $Prop_Fault_Width$ and $Prop_Fault_Time$ hold the values of the width and arrival time of the fault, respectively. The polarity of the fault seen at the input is an inverse of the original value of the signal. The polarity of logic value used to select the appropriate propagation delays (rise delay or fall delay) during the propagation of fault through the gate. The fault can reach the output in two different ways. One, if the fault propagated into the input is not electrically and logically masked. The other way is if a fault is injected into a gate. The fault can be injected into a gate by assigning the variable In_Fault_W to a value greater than 0. In this case, the injected fault width and the injected fault starting time are assigned to the variables Out_Fault_Width and Out_Fault_Time .

During the propagation, the fault from different gates may re-converge into the same gate resulting in fault width attenuation or broadening. When two faults converge, the resultant fault width and starting time depend on the converging fault polarities, widths, and arrival times. Fig. 5.5 illustrates an example where two faults were re-converging into a 2-Input NAND gate. The variables W_O , and T_O represent the output fault width and starting time after re-converging into the gate. Based on the incoming fault polarities, different re-converging scenarios have different output fault width and time. The resultant fault re-converging characteristics are shown in Fig. 5.6. An example showing fault re-convergence characterization is shown in Alg. 1.

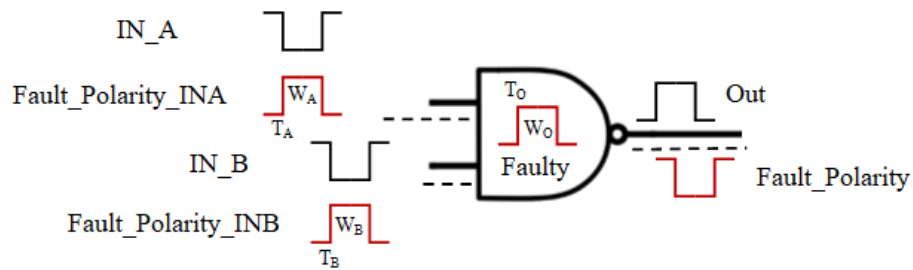


Figure 5.5: Fault re-converging example in 2-input NAND gate.

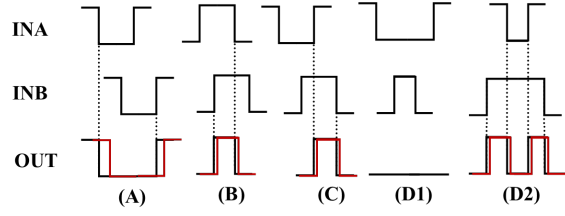


Figure 5.6: Example of a fault re-converging scenarios.

Algorithm 1 An example of fault re-convergence characterization.

Example of re-converging case in Fig. 5.6(A)
if $((Fault_Polarity_INA == False) \wedge (Fault_Polarity_INB == False))$ **then**
 $T_O = \Delta T_P + Min(T_A, T_B)$
 $W_O = Max(T_A + W_A, T_B + W_B) - T_O$
end if

5.3.4 SMT Modelling of LEON3 Processor's Sub-Modules

Though the sub-modules in LEON3 are a part of the same processor core, due to the large size of the design, we model and analyze the SEMT propagation in each sub-module separately. Once the faults due to SEMT traverses the combinational path and latches into a register, its further propagation will be a single or multiple bit upset (MBU) analysis problem. The further propagation of MBUs can be affected only by logical masking effects and not electrical masking effects and therefore will have a different but less complicated modelling than Section 5.3.3. In our analysis approach, the SEMT propagation is analyzed between registers, and calculate the fault propagation probability. This will indicate the probability that a fault will cause an erroneous behavior of the circuit, assuming there is no TMR or ECC implemented for that register.

To facilitate the propagation analysis between the registers, we must first create the SMT functions of the logic gates present in between. Since we use a Python API while creating the SMT functions, the netlist instances need to be topologically sorted before their SMT equivalent function can be declared. Moreover, the circuits also contain feedback paths between the registers and logic gates, introducing cycles or loops in the design. The presence of loops makes it impossible to sort a netlist topologically, and a method is needed to separate the gate instances between the registers while ignoring the feedback paths. Alg.

2 provides an overview of the steps followed when creating an SMT model of the design. We start by parsing the synthesized gate-level netlist instances in the Verilog file. Each signal in the logic gate instantiation is declared as a graph node, and we add a graph edge between the input nodes and the output node. If the netlist instance is a register, the signals are not added to the graph but instead are declared as SMT's symbolic variables. Not adding such paths to the graph essentially prunes the feedback loops present in the design and separates the logic gates present between the registers. Moreover, declaring the register's input and output as a symbol maintains the accuracy while propagating the fault as symbolic variables in SMT's hold any value (True or False) with the same probabilities. The signals connected to the input of a register are aliased as pseudo primary outputs (PPOs), and the output signal of a register is aliased as a pseudo primary input (PPI). The graph is then topologically sorted, and the SMT function for each instance is declared following the sorted order. To help with the graph creation and topological sorting, we have used the graphing tool NetworkX. The NetworkX package is designed to handle data on a scale relevant to modern problems and most of its core algorithms rely on extremely fast legacy code implemented in C and FORTRAN [40]. These characteristics make it suitable while creating and analyzing graphs for large designs.

Algorithm 2 Creating the SMT Model

```

Read instances in Verilog netlist file
if Instance is a register then                                     ▷ Do not add to Graph
    Add instance inputs to pseudo primary outputs (PPOs)
    Add instance outputs to pseudo primary inputs (PPIs)
    Declare PPIs and PPOs as SMT Symbols
else                                                               ▷ Instance is a Logic Gate
    Declare the input and output signals as Graph Nodes
    Add edges between Graph Nodes                                   ▷ Graph.add_edge(in, out)
end if
if Graph is Directed-Acyclic then                                   ▷ is_directed_acyclic(Graph)
    Topologically sort the Graph Nodes                             ▷ topological_sort(Graph)
    Create SMT instances in the sorted order
end if

```

The Integer Unit (IU) sub-module in LEON3 is a well-pipelined design with a total of 1401 pipeline registers. As an example, consider a pipeline stage in the IU shown in Fig. 5.7. Following Alg. 2, the graph of the considered part is constructed, as shown in

Fig. 5.8. It is to be noticed that the path internal to the register (n63396 $\rightarrow$ ICI92) is not added to the graph. This allows the topological sorting of the graph in case there is a feedback path (assume ICI92 $\rightarrow$ n65539). After sorting the nodes, the SMT model of the design is created by converting each gate instance into an SMT function, as shown in Listing 5.1. Then the SEMT propagation is analyzed by checking the fault propagation to the PPIs as well as PPOs.

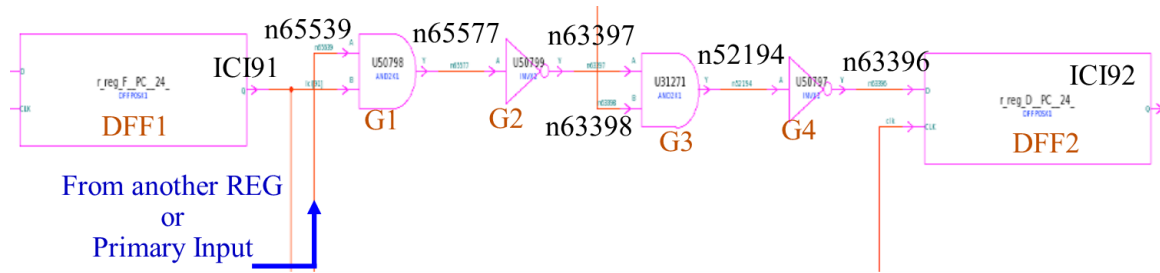


Figure 5.7: A pipeline stage of IU seen in Synopsys DC tool after synthesis.

Listing 5.1: SMT Script for the segment shown in Fig. 5.7.

```
primary_inputs = [Sym(N1), Sym(N2), .....] # Primary Inputs
PPIs = [Sym(ICI91), Sym(ICI92), .....] # Pseudo Primary Inputs
....
n65577 = AND_SMT(ICI91, n65539, G1_Tp, G1_W, G1_T)
n63397 = INV_SMT(n65577, G2_Tp, G2_W, G2_T)
n52194 = AND_SMT(n63397, n63398, G3_Tp, G3_W, G3_T)
n63396 = INV_SMT(n52194, G4_Tp, G4_W, G4_T)
....
primary_outputs = [.....] # Primary Outputs
PPOs = [n63396, .....] # Pseudo Primary Outputs
outputs = Join(primary_outputs, PPOs)
```

The modelling of remaining sub-modules Multiply unit (MUL), and Divide Unit (DIV) remain identical to the method followed in the above example.

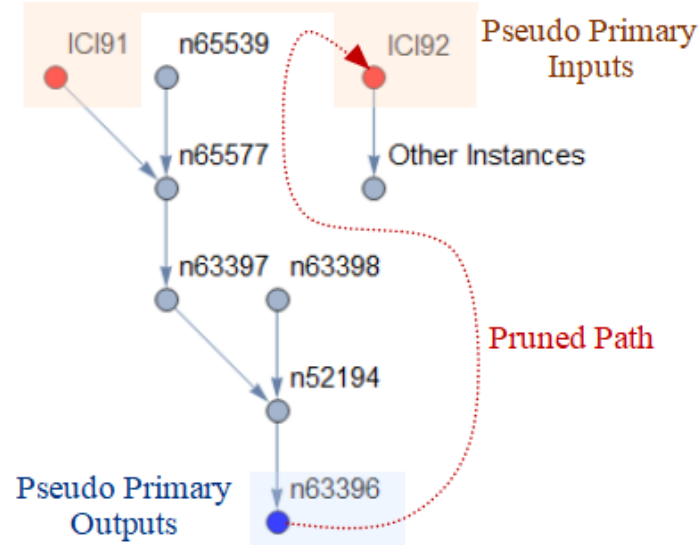


Figure 5.8: Constructed graph of the pipeline stage seen in Fig. 5.7

5.3.5 SEMT Propagation analysis using SMT Solvers

In order for a SEMT fault to result in a soft error, it should either reach the primary outputs of the design or get latched into a flip flop that is internal to design (Pseudo Primary Output). The SEMT pattern is injected into the circuit by adding an assertion specifying which gates in the pattern are faulty. This assertion includes information regarding the injected faults widths and fault injection time. Then fault propagation to the output Op can be verified by adding another assertion specifying the fault state of Op is *True*. Under these assertions, the model is checked for satisfiability condition using an SMT solver. If the model is satisfiable, it implies that the injected SEMT pattern reaches the output Op . Otherwise, the injected SEMT fault is prevented from reaching the output due to the masking effects. The Eq. (3) calculates the probability that the injected SEMT fault propagates to the output. This probability is computed by dividing the total number of satisfying input assignments over the total number of possible input vectors (N).

$$P(SEMT_i) = \frac{\#Satisfying_Input_Assignments}{N} \quad (3)$$

The variable *Satisfying_Input_Assignments* in equation (3) reflects all satisfactory assignments for a specified formula representing an SEMT injection scenario. As shown in Alg. 3, this amount is calculated by generating all input variable assignments in a restricted search space. An important step while calculating the propagation probabilities at an output *Op* is to find its cone of influence (COI). The SEMT propagation to *Op* only depends on the primary inputs (PI) or pseudo primary inputs (PPI) that are present in the COI of *Op*. Therefore, when analyzing the fault propagation, the solver is restricted to finding a satisfying assignment for the input signals that are present in the output cone. The COI for an output can be identified by knowing the variables that constitute its logic state formula. The SMT solver's built-in function `get_free_variables` returns a list of variables that constitutes the SMT formula. As an example, the COI of the Pseudo Primary Output *n63396* in Fig. 5.7 is identified as *ICI91, n65539, n63398*, where *ICI91* is a PPI and *n65539, n63398* are PIs.

Algorithm 3 Calculating the propagation probability of $SEMT_i$ at the output Op

```

procedure MODEL_COUNTING( $Op, SEMT_i, Cone$ )
  Start the Solver
  Add assertion  $Op[1] = True$ 
  Add assertion  $SEMT_i$ 
  while Solver returns SAT do
    Partial_Model = Get_Model(Restricted_Cone)
    Add assertion(Not(And(Partial_Model)))
    Sat_Count = Sat_Count+1
  end while
   $N = len(Restricted\_Cone)$ 
   $P(SEMT_i) = Sat\_Count / 2^N$ 
  return  $P(SEMT_i)$ 
end procedure

```

5.4 Results

The proposed methodology is fully automated using Tcl and python scripting. The RTL to Gate-level synthesis is performed using the tool Synopsys DC tool, and the layout is generated using Cadence Encounter SoC. The LEON3 ASIC is synthesized at TSMC 65 nm General Purpose (GP) Standard Cell Libraries. The SER is estimated for each module in the LEON3 core individually, for the SEMT pattern sizes SEMT3, and SEMT4 as well as SETs. The average propagation probabilities for each pattern size are reported in Table 5.2. We notice that the integer unit (IU) is highly sensitive to the SEMT faults when compared to the Multiply unit (MU) and Division Unit (DIV). This can be explained by the high number of registers in the integer unit and short pipeline paths in the design. Therefore, there are lesser chances for fault re-convergence and logical masking as the faults propagate to the primary output or to a register. The propagation probability almost doubles going from SET to SEMT4. Since the SEMT faults introduce faults into multiple combinational cells at once, there is a higher chance of faults due to SEMT pattern propagates a larger number of outputs when compared to SETs.

Table 5.2: Comparision of average propagation probabilities due to SET and various SEMTs.

Module	Average Propagation Probability		
	SET	SEMT3	SEMT4
IU	0.32	0.33	0.65
MUL	0.24	0.28	0.45
DIV	0.22	0.30	0.51

Due to the nature and semantics of formal verification techniques, a design can be checked for *Fatality* against the SEMT faults. A SEMT fault or a pattern is said to be fatal when there is no input variable assignment that could mask the fault and prevent it from reaching the output or attenuate the fault width during the propagation. Such a pattern could be identified by modifying the first assertion in the Alg. 3 to $Op[1] = False$

and checking for the SMT model satisfiability. If the SMT model is not-satisfiable, it will imply that the fault due to $SEMT_i$ has no logical masking condition or it cannot be entirely attenuated during the propagation. Fig. 5.9 shows the identified fatal SEMT patterns overlayed on the processor's layout when the injected primary fault width is 500ps. Such kind of analysis is only possible through formal methods and is extremely difficult to perform using simulation-based fault injection techniques as the design has to be simulated exhaustively. This critical zone information can be used to design and place the transient filters in the process of radiation hardening.

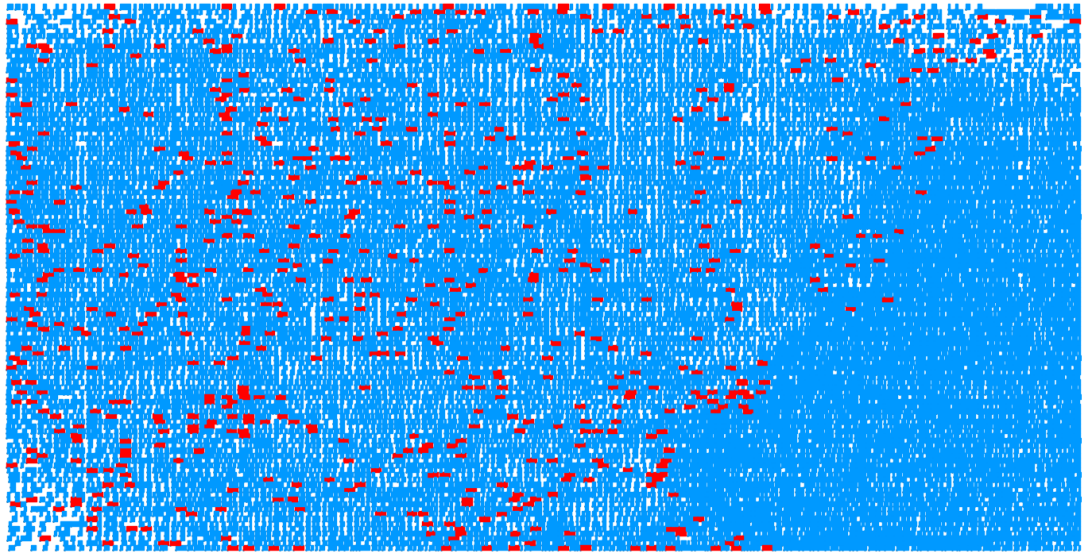


Figure 5.9: Fatal SEMT Patterns marked on the LEON3 core Layout.

Finally, Table 5.3 reports the time consumed to build the SMT formula and to estimate the SER of each module in the LEON3 processor core. All the experiments were conducted on a high performance compute server powered by Intel Xeon processor cluster with 160 CPU cores, each clocked at 2.83 GHz and a system RAM memory of 1 TB. Total time to build and analyze SMT model is approximate 8.5 hours, but it can be minimized utilizing more computation resources like more memory and number of cores.

Table 5.3: CPU Time elapsed to build and analyze the SMT model.

Module	Time (S)	
	Building SMT Model	SMT Analysis
IU	1852	6340
MUL	580	11364
DIV	379	10393

5.5 Conclusions

In this paper, we present a methodology to analyze the effects of SEMT faults on digital circuits using Satisfiability Modulo Theories. The results reveal that the Integer Unit part of the processor's core is highly susceptible to SEMT faults. We observed an overall average SEMT fault propagation probabilities of 30% and 54% due to the effects of SEMT3 and SEMT4 faults on the LEON3 processor core. A key result pointing the most critical zones on the processor's layout has been presented, which highlights the importance of formal techniques in the field of reliability analysis and show the necessity to integrate with the existing vulnerability analysis techniques. To our knowledge, this is the first time such SEMT analysis is done to LEON3 processor core at this level of abstraction.

Chapter 6

Article III: Application-Specific Formal Approach to Estimate Soft Error Vulnerability of Register File using Satisfiability Modulo Theories

Authors: Vivek Bansal, Otmane Ait Mohamed

Abstract: This paper presents an application specific-approach to estimate the soft error failure rate of a circuit due to the effect of external radiations. The proposed approach utilizes the Satisfiability modulo theory (SMT) to evaluate the impact of soft errors on the microprocessor's register file. We investigate the vulnerable factors at the gate level using the layout information of the circuit. A case study is conducted by performing an analysis on the register file of the openMSP430 core for benchmark programs. Exhaustive analyses focused on single and multiple event transients were performed. These analyses have been

used to measure the vulnerability factor of different registers, identifying their application-specific criticality.

Index Terms: soft-error, criticality, transient, register-file

6.1 Introduction

With the advancements in the semiconductor industry and the aggressive downscaling of devices, electronic circuits such as semiconductor memories and microprocessors have become more prone to transient or soft errors [41]. A soft error occurs when a logic value of a logic element is erroneously changed due to the effects of radiation particle bombardment on the device. The radiation particles travel through the substrate of the device's transistor and may cause a transient voltage glitch i.e. single event transient (SET). Due to charge sharing between neighbor transistors, SETs can cause multiple transistors to be affected i.e. single event multiple transients (SEMT) [13].

Several research papers have been published to study SET or SEMT of the digital circuits [42] [43] [36]. Effect of single and multiple event transients [36] is studied based on simulation-based fault injection into the model of the LEON3 processor. Faults are injected into flip-flops, register file, and cache memories and their behavior against transient fault have been studied. Therefore, transient faults in any part of the circuit have become a significant threat to its reliability. The processor is one of the main components in the design of digital circuits, thus, an accurate and efficient model is needed to analyze the transient faults in the processor.

Also, register files in a processor are frequently accessed part throughout the execution of a program, so any soft error occurs in the register file will propagate to the other parts of the processor and will change the original functionality of the program that leading to the incorrect output of the program. Several error handling techniques such as parity or error correction code (ECC) and recovery through exiting copy (REC) [44] [45] are used

to protect the register file against soft errors. However, the addition of parity or error correction bit or copying of the content of the register file becomes an overhead for the processor. Therefore, it is very important to check the soft error vulnerability of the register file that is being used throughout the program.

In this paper, we propose an application-based approach to calculate the soft error vulnerability of the program and its associated register file based on the block-based dependency graph and register data flow graphs. This approach relies on the vulnerability factor of the register file block in the core that is being used throughout the program. The vulnerability of the register file block is calculated based on SETs and SEMTs propagation using the Satisfiability Modulo Theory (SMT). To demonstrate the proposed approach, we present the analysis of the openMSP430 core as a case study.

The remainder of this paper is organized as follows: Section II describes the overview of openMSP430 and section III lists preliminaries for our methodology. The detailed proposed methodology is described in section IV. The experiment on the selected benchmark is presented in section V. Finally, we conclude the analysis based on the experiment in Section VI.

6.2 Architecture

openMSP430 is a synthesizable 16-bit microcontroller core, compatible with Texas Instrument's MSP430 core family [46]. It is based on Von Neumann's architecture. The microcontroller has a single dedicated address space for instruction as well as for data. The core is composed of 16 different registers (R0-R16), each register contains 16-bits. *R0*- *R3* have dedicated functions, *R0* as a program counter (PC), *R1* as a stack pointer (SP), *R2* as status register and *R2* & *R3* as constant generator registers. *R4-R15* are general purpose registers.

As Fig. 6.1 shows, MSP430 architecture incorporates 16-bit RISC CPU, peripherals,

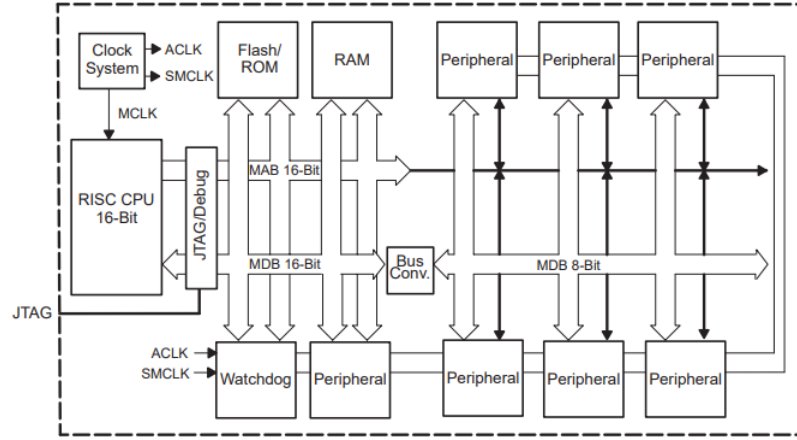


Figure 6.1: MSP430 Architecture

and a flexible clock system that interconnect using a von-Neumann common memory address bus (MAB) and memory data bus (MDB) [47]. In our case study, we use register transfer level (RTL) description of the openMSP430, freely available for academic research purposes [46].

6.3 Preliminaries

A program as shown in Fig. 6.6 having multiple blocks of executable instructions [48] is represented by a block-based dependency graph (BDG). Each block is a set of statements whose execution begins with the first instruction and ends with the last. A Register data flow graph (RDFG) for each block is generated based on its sequence of statements.

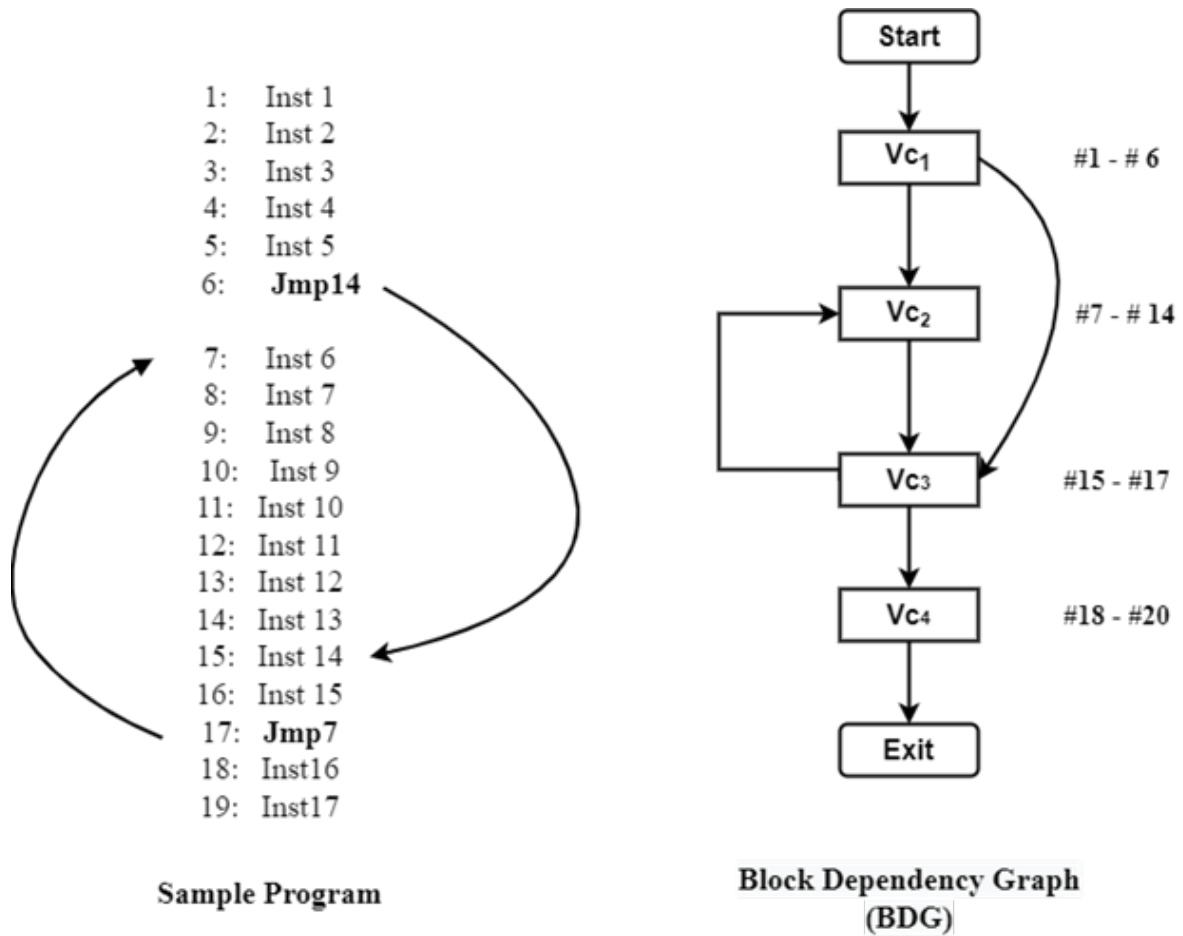


Figure 6.2: A typical Sample Program with its Block-dependency Graph (BDG).

Block-based Dependency Graph (BDG): A program can be defined as a block-based dependency flow graph $G_c = \{V_c, E_c\}$, where V_c represents the set of vertices $\{V_{c1}, V_{c2}, V_{c3}, \dots\}$ denoting each block in the program and E_c represents a set of edges to possible flows of control from one block to another. A basic block terminates at either an instruction that has a branch to another basic block or the instruction is receiving the transfer of control flow from two or more places in the program. Figure 6.2 shows an example of a block-based dependency graph for the example program. It consists of four nodes (V_{c1} , V_{c2} , V_{c3} , V_{c4}) excluding start and exit node. Nodes represent the different blocks of the sample program. Each node has a set of instructions as mentioned in the BDG. For example, Inst

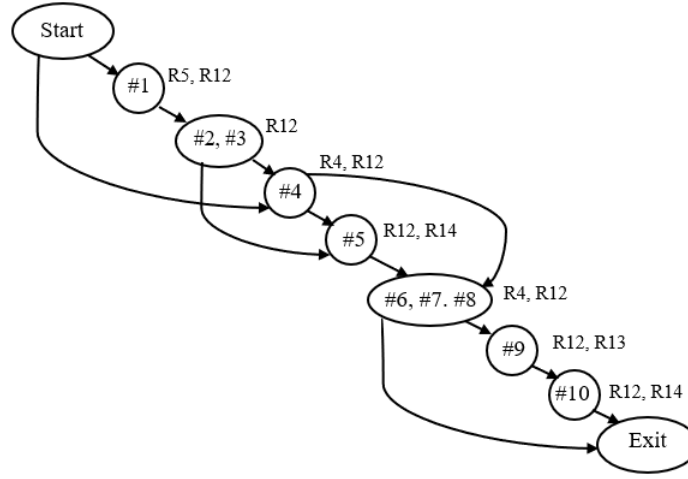


Figure 6.3: Register Data Flow Graph.

1 - Inst5 and Jmp14 are executed in a block denoted as *Vc1*, and instructions Inst 14, Inst 15, and Jmp7 are executed in another block denoted as *Vc3*).

Register Data Flow Graph (RDFG): A RDGF can be defined as $Gd = \{Vd, Ed\}$. Vd is a set of nodes $\{Vd1, Vd2, Vd3, \dots\}$, each node is associated with an operation (s) that is being performed in that block. If execution of a RDFG node $vd1$ has data dependency to another RDFG node $vd2$, then the RDFG has edge between them. Generally, the directed edge represents how the data stored in one register is being used in the execution of the next instruction. An example of a register data flow graph for one block is shown in Fig. 6.3. Each node represents one or more instructions depending upon the register data dependency between them. All the registers associated with the execution of particular instruction (s) are also denoted in the graph.

6.4 Proposed Methodology

We present a brief overview of the proposed methodology in Fig. 6.4. The proposed technique starts by compiling the design program to get assembly language instructions

with the help of the GCC compiler. Based on the assembly language description of the program, the block-based dependency graph and register data flow graphs are obtained based on the mode of operation and data flow throughout the program. Further, the Satisfiability modulo theory (SMT) is applied to each node in every register data flow graph to estimate soft error vulnerability factors. Additionally, we calculate the vulnerability of the register file and program reliability after the complete execution of the program.

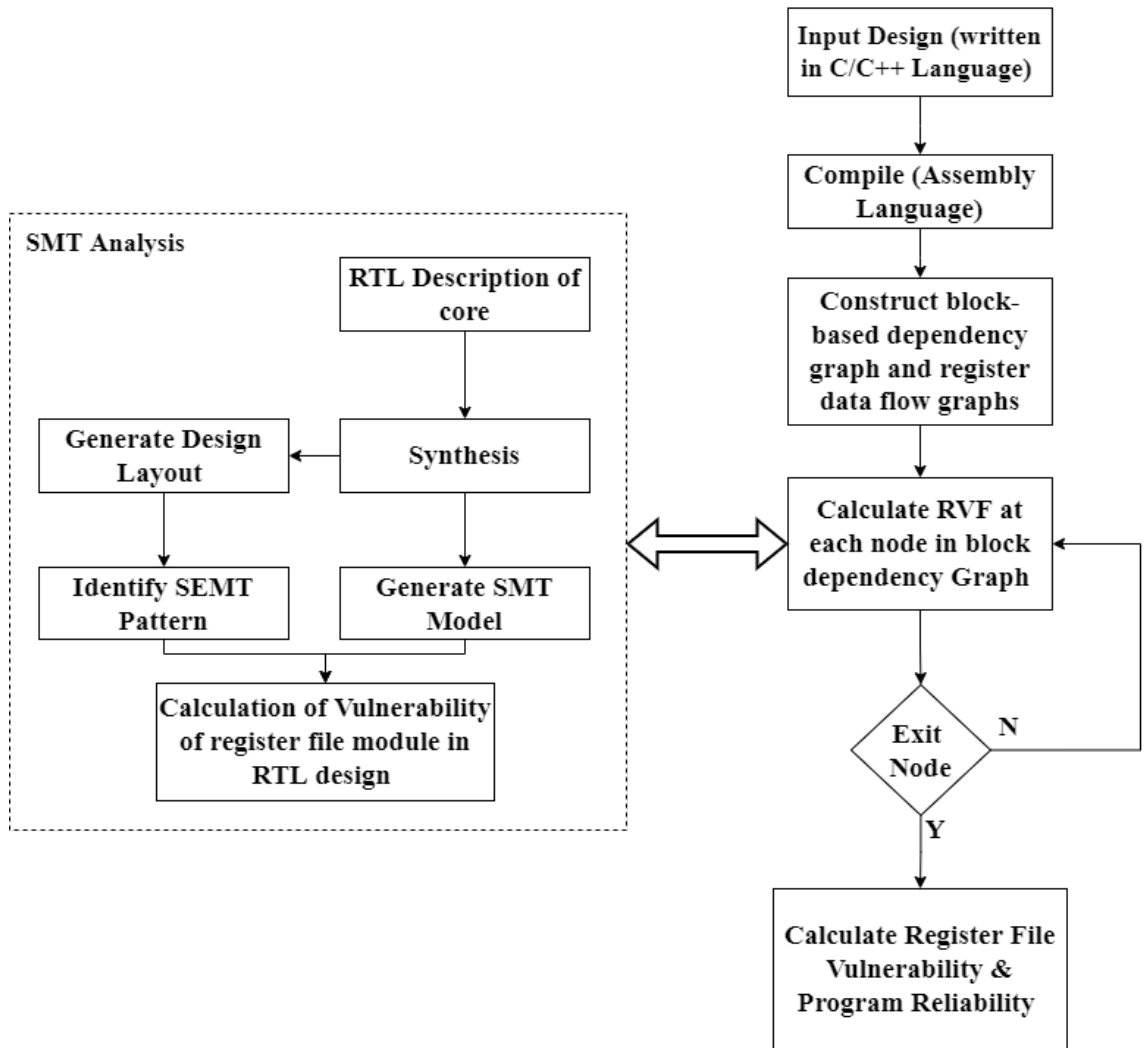


Figure 6.4: Basic Steps of Proposed Methodology.

6.4.1 Satisfiability Modulo Theory (SMT) Analysis

To perform Satisfiability Modulo Theory (SMT) analysis, we use the Register-Transfer Level (RTL) design of openMSP430 core as it is freely available [46]. Using a design synthesis tool (i.e. Synopsys Design Vision), we synthesize the complete RTL design to obtain a gate-level netlist having multiple module declarations. With Python scripting and network graph theory, we obtain a module hierarchy that helps us to create an SMT model for each module in the synthesized openMSP430. Fig. 6.5 shows the openMSP430 processor's hierarchy view, seen after the synthesis step. Additionally, we generate the layout for modules in the core using a layout tool (i.e. Cadence Encounter). After the generation of the layout, we route the design and extract the timing information in the standard delay file (.sdf).

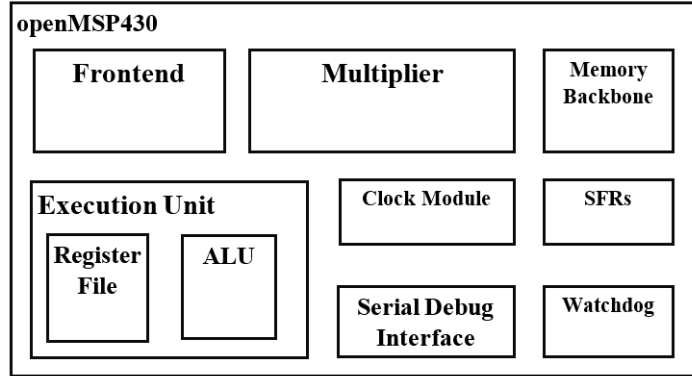


Figure 6.5: Design Hierarchy after synthesis.

To calculate the vulnerability factor of the register file module in the openMSP430 core, we analyze the SEMT propagation at the gate level. An SMT model of the design is built using the synthesized gate-level netlist and the timing reports (.sdf file) available for its generated layout. At the same time, SEMT patterns i.e. error sites due to radiations, are also extracted. Using SEMT patterns and the SMT model of the design, the vulnerability factor (i.e. soft error probability) is calculated for the register file is calculated using an

SMT solver.

6.4.2 Register File Vulnerability

To calculate the soft error rate of the register file, we define a register vulnerability factor (RVF) for each register in the openMSP430 core. The vulnerability factor is defined as the probability that a transient fault or soft error may produce the wrong output. The vulnerability factor of any register is considered in the execution of the program if the register is read by the processor in the next instruction. The vulnerability factor of the register is overwritten when it is loaded with new data. Basically, we measure each register utilization in each operation through run-time tracing of its read and write access.

To compute the RVF of any register during the execution of any instruction, the first step is to identify the type of operation and flow of its execution in the microprocessor. Secondly, we determine all registers that are being used in the execution of the instruction. After that, the vulnerability factor of each register used in the operation is recalculated using the vulnerability factor of the register file. After every instruction of the block, all the values are passed to the next instruction. If the next instruction reads the same register used in a previous instruction, then it is vulnerable that has the vulnerability factor calculated in the previous step. This value will be updated after the execution of the current instruction.

To compute the vulnerability of the register file, the RVF of each register is calculated for the complete execution trace of the program. Then, the vulnerability of the register file is computed as the sum of all registers' RVFs divided by the number of registers (N) in the register file.

$$VulnerabilityofRegisterFile = \frac{\sum_{i=0}^{i=N} RVFi}{N} \quad (4)$$

6.5 Analytical Experiment

To demonstrate the scalability of the proposed method, we have performed several analytical experiments on different benchmark programs written in C language:

- QuickSort Algorithm
- Matrix Multiplication
- Fibonacci Series Number Calculation

Based on the instruction set of MSP430 instruction set, we compile the source code written in C-Language into assembly language using GCC compiler. The execution path of all the programs depends on the input parameters. For each benchmark, we run the program a number of times and extract the path having a higher probability. So, the block-based dependency graph (BDG) is formed for the path to check the flow of the design as described in section 6.4. With python scripting, register data flow graphs (RDFGs) for each node in the BDG are constructed to check how the data is flowing from one register to another register depending on the mode of operation.

```

main:
I1      SUB.W #8, R1
I2      MOV.W #0, @R1
I3      MOV.W #1, 4(R1)
I4      MOV.W @R1, R12
I5      ADD.W 4(R1), R12
I6      MOV.W R12, 2(R1)
I7      MOV.W #3, 6(R1)
I8      BR #.L2
.L3:
I9      MOV.W 4(R1), @R1
I10     MOV.W 2(R1), 4(R1)
I11     MOV.W @R1, R12
I12     ADD.W 4(R1), R12
I13     MOV.W R12, 2(R1)
I14     ADD.W #1, 6(R1)
.L2:
I15     MOV.B #10, R12
I16     CMP.W 6(R1), R12 { JGE .L3
I17     NOP
I18     ADD.W #8, R1
I19     RET

```

Figure 6.6: Glimpse of Fibonacci Series Program

We present a glimpse of the Fibonacci series in Fig. 6.6, showing four different blocks of a program. Each block has a set of instructions that denotes the one node in the BDG. In the Fibonacci series, different kinds of operations like MOV, ADD, PUSH, etc. are utilized. For each kind of operation, the processor has a unique path and it utilizes a set of modules in the processor to execute the instruction. At the first node of BDG, all the registers do not have any vulnerability values. The vulnerability values (RVF) of the register will start to change while the processor is running some operation.

According to the RDFG of any block of the algorithm, the first node "START" receives

RVF values for all registers from the "EXIT" node of the previous RDFG node. If any instruction reads the vulnerable register while its execution and writes the data back to another register then the RVF of the reading register remains the same and the RVF of the write register is overwritten over the previous value. The soft error probability of registers corresponding to the operation is calculated and stored in a list. The list containing sot error probabilities of all registers is passed to the next instruction in the block. At the "EXIT" node of the RDFG, all updated values will be passed to the next node of BDG. Similarly, after execution of all instructions in the program, RVFs of all registers are extracted from "EXIT" node of the BDG to calculate the vulnerability of the register file as per equation 4.

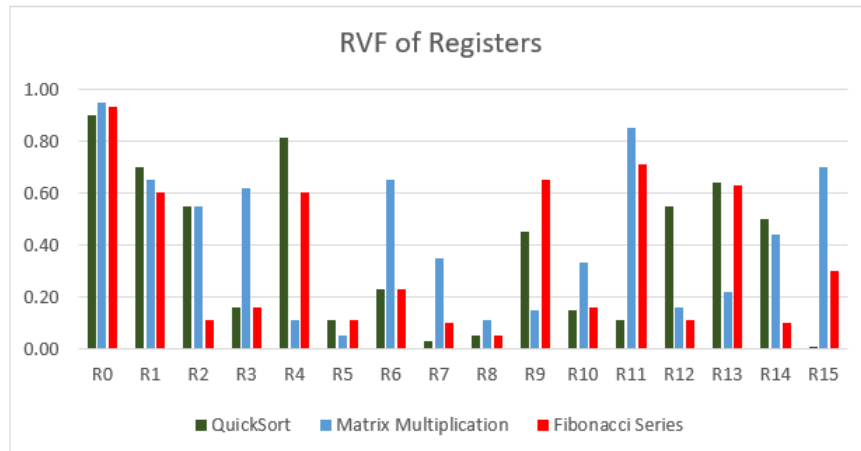


Figure 6.7: Register Vulnerabilities.

We have computed register vulnerability factors for all three mentioned benchmarks. Fig. 6.7 illustrates the comparison of three benchmark programs with their register vulnerability factors. It shows soft error vulnerability factor values of all registers. As seen from the graph, the program counter register ($R0/PC$) has a higher probability of having a soft error because this register is used in the execution of every instruction.

Table 6.1 indicates the vulnerability of the register file for the benchmark programs. The register file for the matrix multiplication algorithm has the maximum vulnerable value.

Table 6.1: Experimental Results

Benchmarks	Vulnerability of Register File
QuickSort	0.3718
Matrix Multiplication	0.4306
Fibonacci Series	0.3386

6.6 Conclusion

In this paper, we have proposed a methodology to estimate the vulnerability factors of registers against transient faults or soft errors based on SMT analysis. We tested our method on various benchmarks with an openMSP430 microcontroller core and extracted the most vulnerable registers after their execution. Our experimental results indicate that program counter (*PC*) and stack pointer registers (*SP*) are vulnerable for all benchmarks circuits. Also, the most commonly used register throughout the execution is more vulnerable as compared to other general-purpose registers.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Soft-error vulnerability is a major concern for radiation-exposed microelectronic devices. In this thesis work, we introduced a formal technique-based electronic design automation (EDA) tool to model and analyze the vulnerability of digital circuits at an early stage of the design cycle. In chapter 4, we have presented our in-house built tool, called Multiple Event Transient Analysis (META). With the help of the tool, we have analyzed the vulnerability of various ISCAS benchmark circuits through the analysis of single event multiple transient faults.

We extended our META tool work in chapter 5 to support processor core fault analysis and presented a case study of the LEON3 processor to check its design vulnerability against multiple transient faults. The main idea behind this work was to check the vulnerability of various modules on the LEON3 processor core. In chapter 6, we introduced a methodology to estimate the soft error vulnerability of register files in the processor core for a particular application. For that, we utilized open source openMSP430 core from the Texas Instrument MSP430 core family and we had run some applications (e.g. QuickSort, Matrix Multiplication, or Fibonacci Series) on it. Based on the block dependency graph

and data flow of the program, we extracted the vulnerability of the register file.

7.2 Future Work

Below are some future directions for this work:

- In order to validate the proposed methodology in a real experiment, future work will include a comparison of results obtained from the in-built tool META to radiation experiment results from the radiation experimental facility.
- Future research could look at more complex processors, as well as different architectures.
- Extend the tool to support n times unrolling of the sequential circuit automatically.
- SEMT fault pattern extraction in our proposed tool is based on results obtained from very few radiation experimental studies, it is possible to explore this area based on the physical implementation of the device like ambipolar diffusion or bipolar amplification effect between transistors can be explored further.
- Our methodology can be extended to identify SEMT fault propagation paths and apply Triple Modular Redundancy (TMR) or gate hardening on the most vulnerable paths.

Appendix A

User Manual: META



Copyright © 2019-2021 Concordia University. All rights reserved.

No part of this software, documentation, or publication may be reproduced, transcribed, stored in a retrieval system, translated into any language, computer language, or transmitted in any form or by any means, electronically, mechanically, magnetically, optically, chemically, photocopied, manually, or otherwise, without prior written permission from Concordia University.

This software is supplied under a license agreement and may only be installed, used, or copied in accordance with that agreement.

Disclaimer

Concordia provides this publication “as is” without warranty of any kind, either expressed or implied.

This publication may contain technical inaccuracies or typographical errors. While every precaution has been taken in the preparation of this document to ensure that it contains accurate and up-to-date information, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained in this document.

Concordia's policy is one of continual development, so the content of this document is periodically subject to be modified without notice. These changes will be incorporated in new editions of the publication. Concordia may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

Concordia may have patents or pending patent applications covering subject matters in this document. The furnishing of this document does not give you any license to these patents.

Latest Update: July 2022

A.1 Introduction

Documentation for META is intended for users of LINUX only. To run the tool on your system, run the set-up file to install the Python packages required for this tool. All packages will be automatically installed as per system and tool requirements.

Figure 5.1 illustrates the structure and general usage flow for analyzing a circuit design with META. This tool has been built to identify the propagation of single event multiple event transients (SEMT) at the gate level using an SMT solver. It models electrical, logical, and latching-window masking effects and verifies the fault propagation based on the design layout.

To perform SEMT analysis using META, we need the following information:

- Synthesized Verilog file (.v)

- Cell placement file (.def)
- Cell Delay file (.sdf)
- Cell timing file (.tdf)

A.2 Running META tool using Graphical User Interface (GUI):

This topic provides an overview of the META Tool functioning. Below are steps to run the tool:

A.2.1 Creation of directory

Create a new directory and copy the design files into it. Also, save layout and timing files in the directory. This directory will store the tool-generated experiment output files that can be utilized for interpretation.

A.2.2 Launch META tool

Open UNIX shell prompt from the directory, source

/nfs/private/a/ait/fvg/META_2.0/sem_t_analysis

As the META tool is stored on the ENCS server under the functional verification group (FVG) directory so we will directly source it from there. NOTE: Make sure you have access to the ENCS server directory.

Upon opening META, you will see the Welcome to Multiple Event Transient Analysis (META) tool dialog box illustrated in [A.1](#)

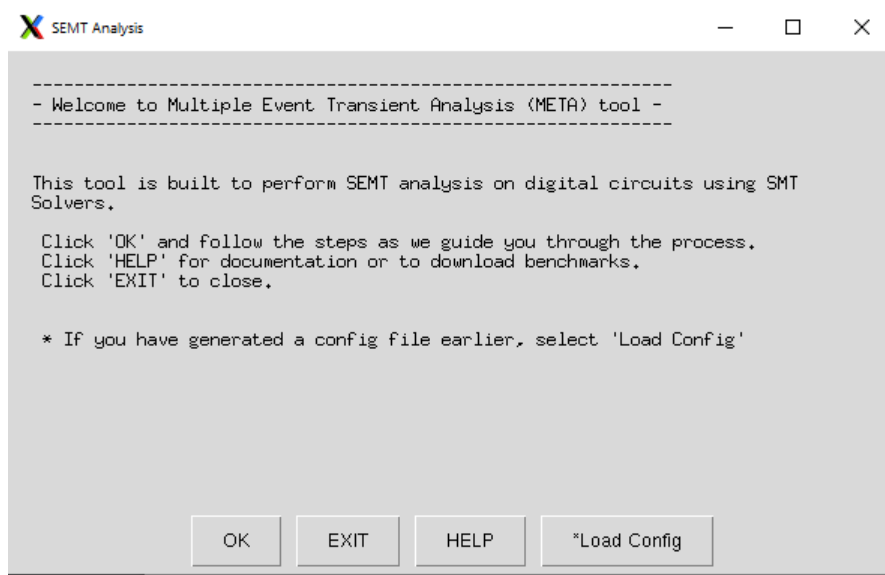


Figure A.1: META Tool: Welcome Page

A.2.3 Load the Design

Click 'OK' on the welcome dialog box and follow the steps:

To perform SEMT analysis, META has two modes [A.2](#):

- **Mode1:** You have Synthesized Verilog file (.v), Cell placement file (.def), Cell Delay file (.sdf) and Cell timing file (.tdf). You can select each file one by one using dialogue boxes as shown in Fig. [A.3](#)
- **Mode2:** You have only circuit design file (.v/.vhd) (e.g. any Benchmark circuit design). The tool automatically generates the synthesized gate-level netlist and layout files.

Synopsys Design Compiler tool will be used to generate the gate level design file (.v)

Cadence Encounter SOC tool will be used to generate layout and delay information files (.DEF, .TDF, .SDF)

Note:

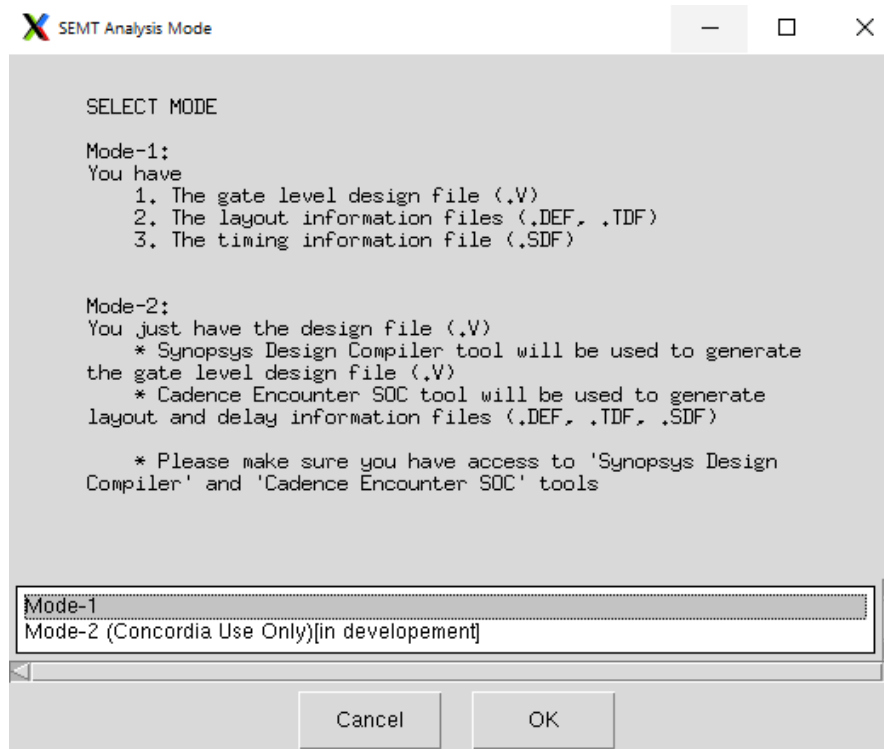


Figure A.2: META Tool: Mode Selection

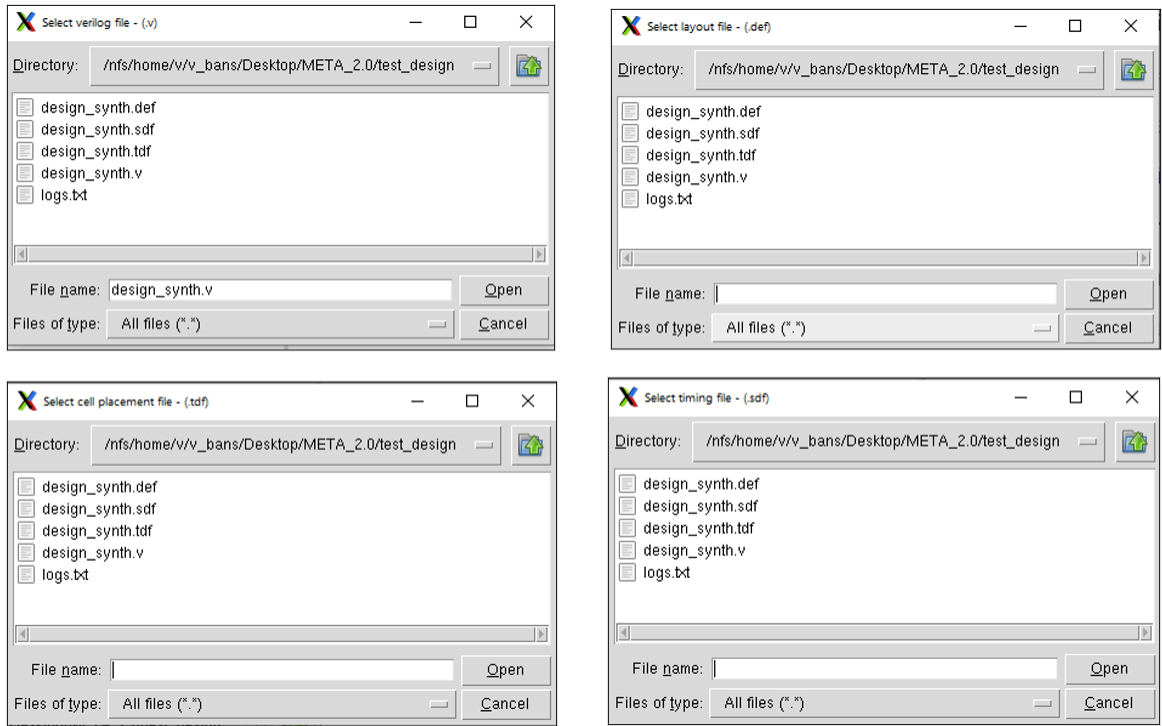


Figure A.3: META Tool: Design layout Files Selection

- Please make sure you have access to 'Synopsys Design Compiler' and 'Cadence Encounter SOC' tools
- Mode2 is in the development phase (will be available soon)

A.2.4 Select SEMT Pattern Parameters

As a single high-energy particle can affect several neighboring cells, which causes a single event multiple transient faults in the circuit. Therefore, the affected area for each size of the SEMT pattern needs to be extracted based on the circuit layout design. The size n of the SEMT pattern tells us how many cells are affected horizontally by a single transient fault. In the META tool, you can select its pattern size e.g. SEMT3 as shown in Fig. A.4. It means that the particle strike can generate transients in three cells along the well and one cell across the well, resulting in the pattern SEMT3.

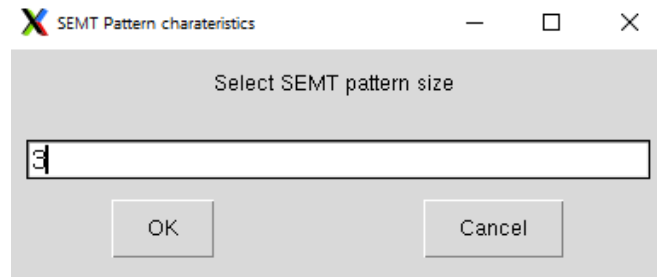


Figure A.4: META Tool: SEMT Pattern Size Selection

Based on the cell dimension and cell placement information extracted from the place routing information file, the tool automatically extracts the SEMT patterns based on their selected size. All the extracted SEMT patterns are written to a file (SEMT Pattern.py) where each pattern is a python SMT formula, specifying the gates and their fault injection widths.

A.2.5 Fault Analysis Evaluation Type

In the META tool, you can perform SEMT fault analysis in four different ways (as shown in fig. A.5 based on fault propagation using SMT solver.

- **Critical Fault Propagation Analysis** In this mode, SEMT fault propagation analysis is performed for each identified SEMT patterns site and the most vulnerable zones or error sites are shown on the design layout
- **One solution Evaluation** In this mode, only one propagation model for each SEMT pattern is generated for which fault is reaching the output of the circuit.
- **n Solution Evaluation** In this mode, you can specify the number of propagation models for which fault is reaching the output of the circuit. For each SEMT pattern, 'n' number of propagation models will be generated.

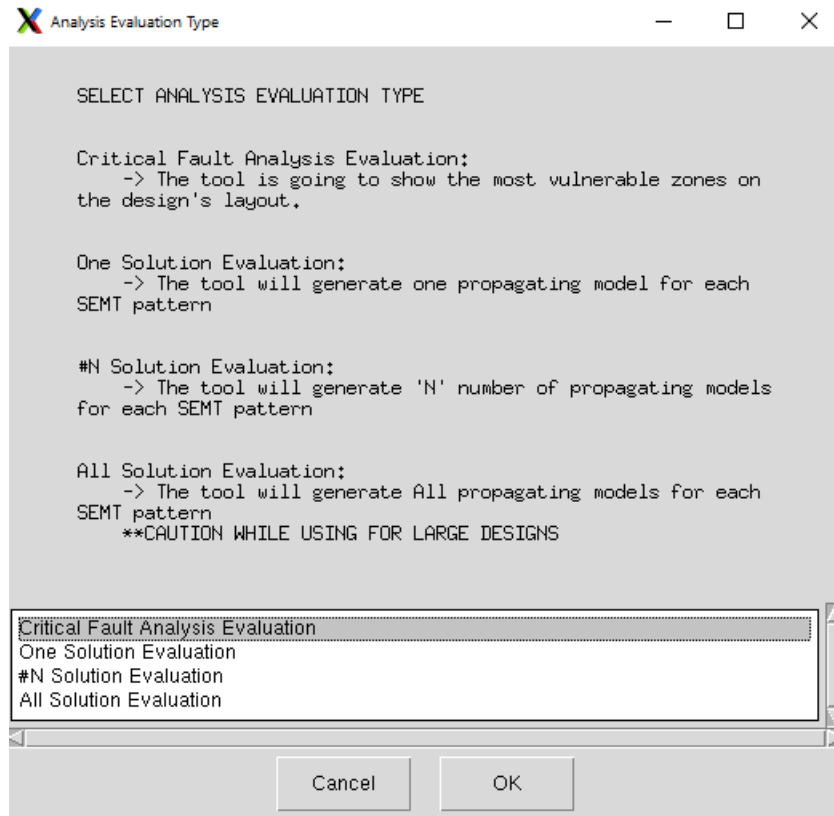


Figure A.5: META Tool: Fault Evaluation Type Selection

- **All Solution Evaluation** In this mode, all propagation models for each SEMT pattern are generated for which fault is reaching the output of the circuit. For each SEMT pattern, the probability that the injected fault propagates to the output is computed. Furthermore, soft error rate (SER) is computed as an average of propagation probabilities of all identified SEMT patterns to any output. Soft Error Rate (SER) is used to indicate the vulnerability of digital circuits due to soft errors.

A.2.6 Testbench Generation

The META tool has inbuilt functionality to generate a Verilog test bench for fault injection simulation run. If you click on “YES” as shown in fig. A.6 then it generate a new directory “verilog_tests” in the current directory. The test-bench file is automatically

created and saved in the current directory.

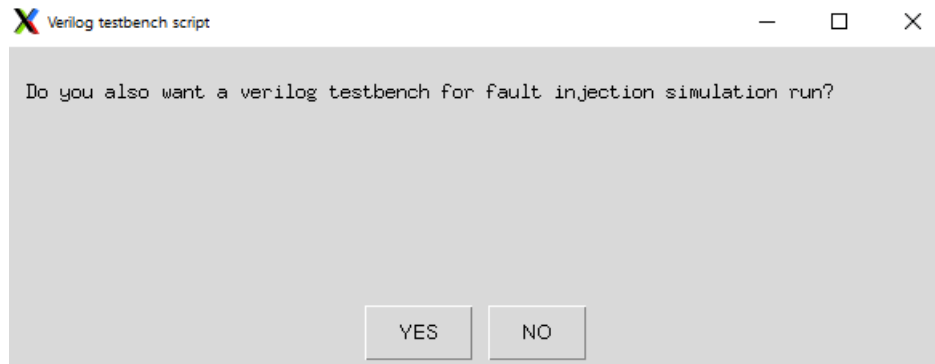


Figure A.6: META Tool – Verilog Test-bench

A.2.7 Configuration File

If you are running the tool first time, the SEMT configuration file (SEMT_config.py) will be generated automatically and saved in the current directory. [A.7](#)

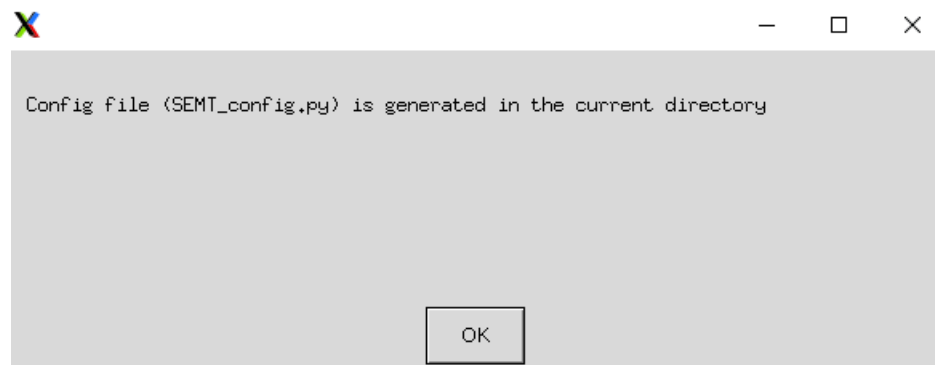


Figure A.7: META Tool - Configuration File

Next time, you can directly load the configuration file (as shown in fig. [A.1](#) instead of loading design and layout files individually. The configuration file contains information about the information for the selection of the design file, layout files, timing delay files, SEMT pattern size, and so on. It also has information about the location of all the files.

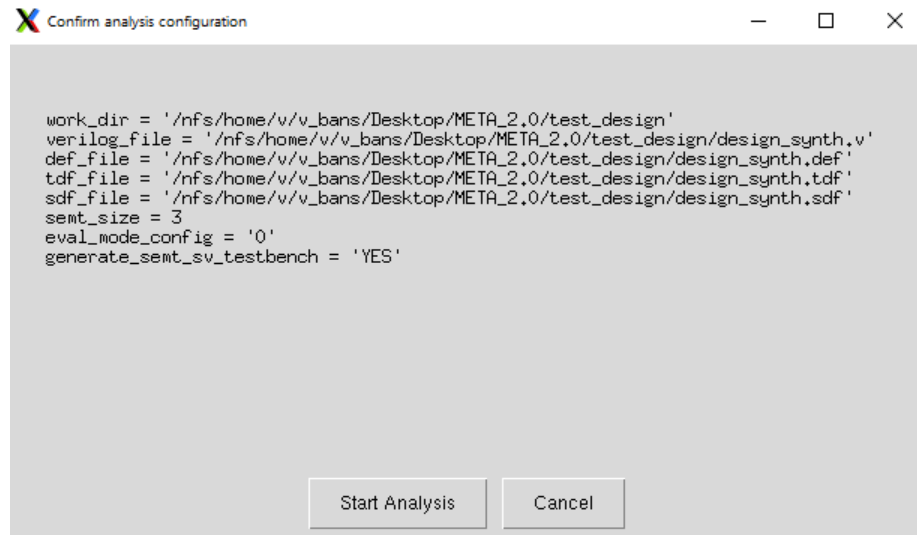


Figure A.8: META Tool - Start Analysis

A.2.8 Run SEMT Analysis

After loading of design files (Verilog file (.v), cell placement file (.def), cell Delay file (.sdf) and cell timing file (.tdf)) or loading the configuration file, the META tool will show display all the information including the path of all files and selected analysis parameters [A.8](#). After confirmation of all information, you can start the analysis. The tool will start SEMT analysis based on the selected evaluation mode.

Next time, you can directly load the configuration file instead of loading the design and layout files individually. The configuration file contains information about the information for the selection of the design file, layout file, timing delay files, SEMT pattern size, and so on. It also has information about the location of all the files.

A.2.9 Output - SEMT Fault Pattern

As each SEMT pattern has its own effects on the circuit's error propagation probability, the analysis for a specific pattern will be unique. Thus, the propagation analysis will be performed for each identified SEMT pattern individually. While checking the propagation

probabilities, each SEMT pattern is also checked for fatality and fatal SEMT patterns will be marked on the design layout.

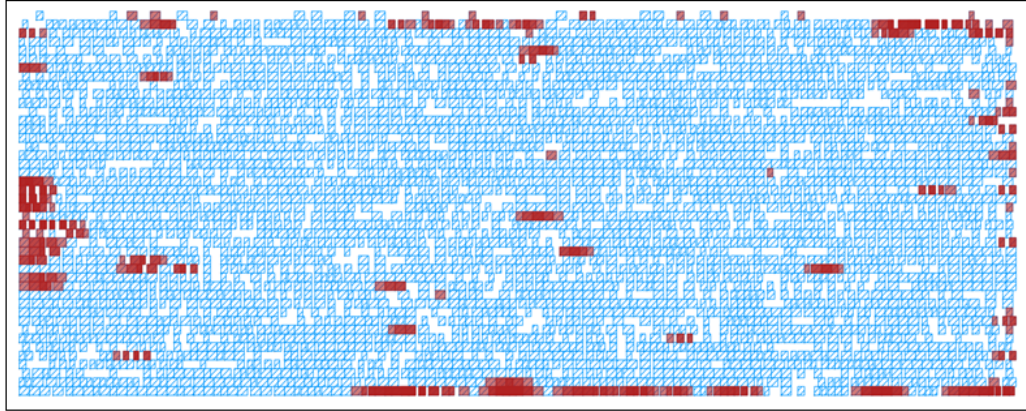


Figure A.9: META Tool Output - SEMT Pattern Error Sites

Figure A.9 illustrates the design layout for a sample benchmark circuit. In this figure, the red portion tells us the most vulnerable sites on the circuit layout. For larger circuit design, we will have thousands of SEMT patterns, so the META tool uses multi-threading to calculate the propagation probability for several SEMT patterns in parallel, executed across multiple CPU cores. All the SEMT patterns for which the fault will be propagated to the final output will be shown on display. After obtaining the propagation probabilities for all the SEMT patterns, the soft error rate (SER) will be calculated.

These results can help the chip designers to develop efficient hardening techniques targeting vulnerable sites or areas with minimum impact on the design constraints.

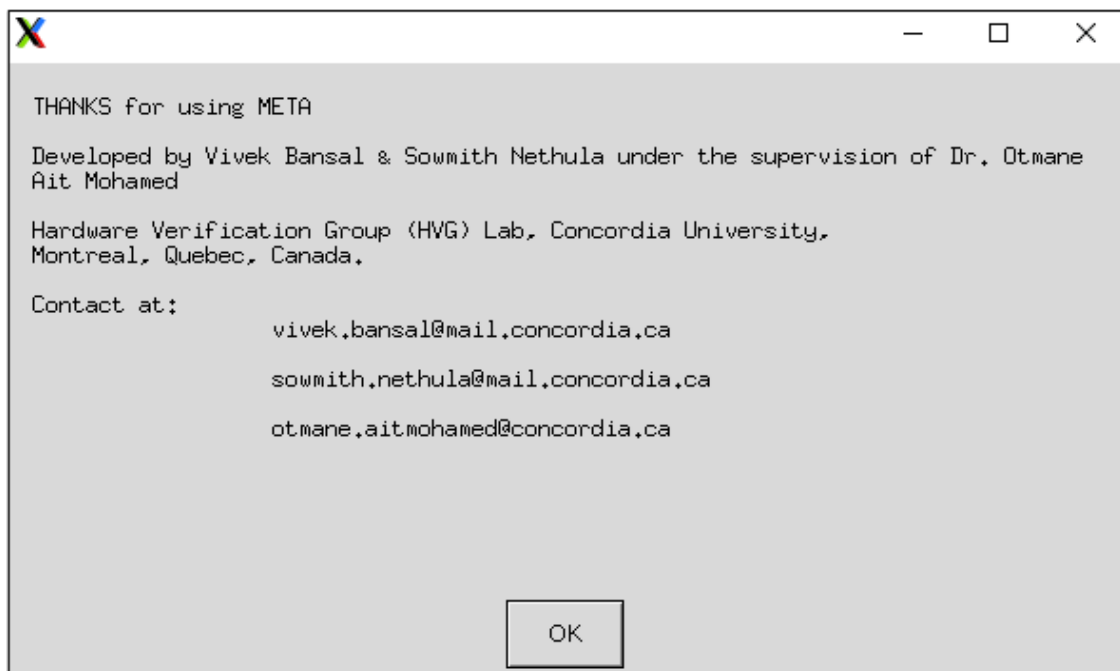


Figure A.10: META Tool: Thank You Note

Bibliography

- [1] G.E. Moore. Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff. *IEEE Solid-State Circuits Society Newsletter*, 11:33–35, Sep. 2006.
- [2] Hossein Asadi and Mehdi B. Tahoori. Soft error modeling and remediation techniques in asic designs. *Microelectron. J.*, 41(8):506–522, aug 2010.
- [3] Ji Li and Jeffrey Draper. Accelerated soft-error-rate (ser) estimation for combinational and sequential circuits. *ACM Trans. Des. Autom. Electron. Syst.*, 22(3), May 2017.
- [4] R. Baumann. Soft errors in advanced computer systems. *IEEE Design Test of Computers*, 22(3):258–266, May 2005.
- [5] Natasa Miskov-Zivanov and Diana Marculescu. Multiple transient faults in combinational and sequential circuits: A systematic approach. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(10):1614–1627, Oct 2010.
- [6] D. Binder, E. C. Smith, and A. B. Holman. Satellite anomalies from galactic cosmic rays. *IEEE Transactions on Nuclear Science*, 22(6):2675–2680, Dec 1975.
- [7] T.C. May and M.H. Woods. Alpha-particle-induced soft errors in dynamic memories. *IEEE Transactions on Electron Devices*, 26(1):2–9, Jan 1979.

- [8] J. F. Ziegler, H. W. Curtis, H. P. Muhlfeld, C. J. Montrose, B. Chin, M. Nicewicz, C. A. Russell, W. Y. Wang, L. B. Freeman, P. Hosier, L. E. LaFave, J. L. Walsh, J. M. Orro, G. J. Unger, J. M. Ross, T. J. O’Gorman, B. Messina, T. D. Sullivan, A. J. Sykes, H. Yourke, T. A. Enger, V. Tolat, T. S. Scott, A. H. Taber, R. J. Sussman, W. A. Klein, and C. W. Wahaus. Ibm experiments in soft fails in computer electronics (1978–1994). *IBM Journal of Research and Development*, 40(1):3–18, Jan 1996.
- [9] Balaji Narasimham, Bharat L. Bhuva, Ronald D. Schrimpf, Lloyd W. Massengill, Matthew J. Gadlage, Oluwale A. Amusan, William Timothy Holman, Arthur F. Witulski, William H. Robinson, Jeffrey D. Black, Joseph M. Benedetto, and Paul H. Eaton. Characterization of digital single event transient pulse-widths in 130-nm and 90-nm cmos technologies. *IEEE Transactions on Nuclear Science*, 54(6):2506–2511, Dec 2007.
- [10] Damien Giot, Philippe Roche, Gilles Gasiot, Jean-Luc Autran, and Reno Harboe-Sorensen. Heavy ion testing and 3-d simulations of multiple cell upset in 65 nm standard srams. *IEEE Transactions on Nuclear Science*, 55(4):2048–2054, Aug 2008.
- [11] Adrian Evans, Maximilien Glorieux, Dan Alexandrescu, Cesar Boatella Polo, and Véronique Ferlet-Cavrois. Single event multiple transient (semt) measurements in 65 nm bulk technology. In *2016 16th European Conference on Radiation and Its Effects on Components and Systems (RADECS)*, pages 1–6, Sep. 2016.
- [12] Chen Jianjun, Chen Shuming, Chi Yaqing, and Liang Bin. Characterization of single-event transient pulse quenching among dummy gate isolated logic nodes in 65 nm twin-well and triple-well cmos technologies. *IEEE Transactions on Nuclear Science*, 62(5):2302–2309, Oct 2015.
- [13] Pengcheng Huang, Shuming Chen, Jianjun Chen, Bin Liang, and Yaqing Chi. Heavy-ion-induced charge sharing measurement with a novel uniform vertical inverter chains

- (univic) semt test structure. *IEEE Transactions on Nuclear Science*, 62(6):3330–3338, Dec 2015.
- [14] Johann Heyszl, Ralph Nyberg, Dietmar Heinz, and Georg Sigl. Enhancing fault emulation of transient faults by separating combinational and sequential fault propagation. In *2016 International Great Lakes Symposium on VLSI (GLSVLSI)*, pages 209–214, May 2016.
- [15] Luis Entrena, Mario García Valderas, Raúl Fernández Cardenal, Marta Portela García, and Celia López Ongil. Set emulation considering electrical masking effects. *IEEE Transactions on Nuclear Science*, 56(4):2021–2025, Aug 2009.
- [16] Xiaoyu Zhang, Bin Liang, and Ruiqiang Song. Circuit-level soft error rate evaluation approach considering single-event multiple transient. In *2019 IEEE 13th International Conference on ASIC (ASICON)*, pages 1–4, Oct 2019.
- [17] Natasa Miskov-Zivanov and Diana Marculescu. A systematic approach to modeling and analysis of transient faults in logic circuits. In *2009 10th International Symposium on Quality Electronic Design*, pages 408–413, March 2009.
- [18] Mahdi Fazeli, Seyed Nematollah Ahmadian, Seyed Ghassem Miremadi, Hossein Asadi, and Mehdi B. Tahoori. Soft error rate estimation of digital circuits in the presence of multiple event transients (mets). pages 1–6, March 2011.
- [19] Mohammad Reza Rohanipoor, Behnam Ghavami, and Mohsen Raji. Improving combinational circuit reliability against multiple event transients via a partition and restructuring approach. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(5):1059–1072, May 2020.
- [20] Georgios Ioannis Paliaroutis, Pelopidas Tsoumanis, Nestor Evmorfopoulos, George Dimitriou, and Georgios I. Stamoulis. A placement-aware soft error rate estimation of

- combinational circuits for multiple transient faults in cmos technology. In *2018 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 1–6, Oct 2018.
- [21] Mojtaba Ebrahimi, Hossein Asadi, and Mehdi B. Tahoori. A layout-based approach for multiple event transient analysis. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2013.
- [22] Mojtaba Ebrahimi, Hossein Asadi, Rajendra Bishnoi, and Mehdi B. Tahoori. Layout-based modeling and mitigation of multiple event transients. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(3):367–379, 2016.
- [23] Yankang Du and Shuming Chen. A novel layout-based single event transient injection approach to evaluate the soft error rate of large combinational circuits in complementary metal-oxide-semiconductor bulk technology. *IEEE Transactions on Reliability*, 65(1):248–255, March 2016.
- [24] Xuebing Cao, Liyi Xiao, Jie Li, Rongsheng Zhang, Shanshan Liu, and Jinxiang Wang. A layout-based soft error vulnerability estimation approach for combinational circuits considering single event multiple transients (semts). *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(6):1109–1122, June 2019.
- [25] Christos Georgakidis, Georgios Ioannis Paliaroutis, Nikolaos Sketopoulos, Pelopidas Tsoumanis, Christos Sotiriou, Nestor Evmorfopoulos, and Georgios Stamoulis. A layout-based soft error rate estimation and mitigation in the presence of multiple transient faults in combinational logic. In *2020 21st International Symposium on Quality Electronic Design (ISQED)*, pages 231–236, March 2020.
- [26] Ghaith Bany Hamad, Otmane Ait Mohamed, and Yvon Savaria. Towards formal abstraction, modeling, and analysis of single event transients at rtl. In *2016 IEEE*

- International Symposium on Circuits and Systems (ISCAS)*, pages 2166–2169, May 2016.
- [27] Ghaith Bany Hamad, Otmane Ait Mohamed, and Yvon Savaria. Efficient multilevel formal analysis and estimation of design vulnerability to single event transients. In *2015 IEEE 21st International On-Line Testing Symposium (IOLTS)*, pages 1–6, July 2015.
- [28] S.Z. Shazli and M.B. Tahoori. Using boolean satisfiability for computing soft error rates in early design stages. *Microelectronics Reliability*, 50(1):149–159, 2010.
- [29] L. W. Massengill, O. A. Amusan, S. Dasgupta, A. L. Sternberg, J. D. Black, A. F. Witulski, B. L. Bhuvu, and M.L. Alles. Soft-error charge-sharing mechanisms at sub-100nm technology nodes. In *2007 IEEE International Conference on Integrated Circuit Design and Technology*, pages 1–4, May 2007.
- [30] G. Hubert, S. Duzellier, F. Bezerra, and R. Ecoffet. Musca sep3 contributions to investigate the direct ionization proton upset in 65nm technology for space, atmospheric and ground applications. pages 179–186, Sep. 2009.
- [31] T. R. Assis, J. S. Kauppila, B. L. Bhuvu, R. D. Schrimpf, L. W. Massengill, R. Wong, and S. Wen. Estimation of single-event transient pulse characteristics for predictive analysis. pages SE–5–1–SE–5–6, April 2016.
- [32] Eishi Ibe, Hitoshi Taniguchi, Yasuo Yahagi, Ken-ichi Shimbo, and Tadanobu Toba. Impact of scaling on neutron-induced soft error in srams from a 250 nm to a 22 nm design rule. *IEEE Transactions on Electron Devices*, 57(7):1527–1538, July 2010.

- [33] Andrea Pellegrini, Kypros Constantinides, Dan Zhang, Shobana Sudhakar, Valeria Bertacco, and Todd Austin. Crashtest: A fast high-fidelity fpga-based resiliency analysis framework. In *2008 IEEE International Conference on Computer Design*, pages 363–370, 2008.
- [34] Antonio da Silva and Sebastian Sanchez. Leon3 vip: A virtual platform with fault injection capabilities. In *2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, pages 813–816, Sep. 2010.
- [35] Hipólito Guzman-Miranda, Miguel A. Aguirre, and J. Tombs. Noninvasive fault classification, robustness and recovery time measurement in microprocessor-type architectures subjected to radiation-induced errors. *IEEE Transactions on Instrumentation and Measurement*, 58(5):1514–1524, May 2009.
- [36] Hamed Abbasitabar, Hamid R. Zarandi, and Ronak Salamat. Susceptibility analysis of leon3 embedded processor against multiple event transients and upsets. In *2012 IEEE 15th International Conference on Computational Science and Engineering*, pages 548–553, Dec 2012.
- [37] Samuel Pagliarini, Fernanda Kastensmidt, Luis Entrena, Almudena Lindoso, and Enrique San Millan. Analyzing the impact of single-event-induced charge sharing in complex circuits. *IEEE Transactions on Nuclear Science*, 58(6):2768–2775, Dec 2011.
- [38] Liang Chen, Mojtaba Ebrahimi, and Mehdi Baradaran Tahoori. Formal quantification of the register vulnerabilities to soft error in rtl control paths. *Journal of Electronic Testing*, 31:193–206, 2015.
- [39] Jiri Gaisler, Sandi Habinc, and Edvin Catovic. Grlib ip core user’s manual. version 1.3. 7-b4144 Gaisler research, <http://gaisler.com/products/grlib/grip.pdf>. 2014.

- [40] Drew Conway. Modeling network evolution using graph motifs. *ArXiv*, abs/1105.0902, 2011.
- [41] I. Chatterjee, B. Narasimham, N. N. Mahatme, B. L. Bhuvu, R. A. Reed, R. D. Schrimpf, J. K. Wang, N. Vedula, B. Bartz, and C. Monzel. Impact of technology scaling on sram soft error rates. *IEEE Transactions on Nuclear Science*, 61(6):3512–3518, Dec 2014.
- [42] D. Munteanu and J.-L. Autran. Modeling and simulation of single-event effects in digital devices and ics. *IEEE Transactions on Nuclear Science*, 55(4):1854–1878, Aug 2008.
- [43] Akira Mochizuki, Naoya Onizawa, Akira Tamakoshi, and Takahiro Hanyu. Multiple-event-transient soft-error gate-level simulator for harsh radiation environments. In *TENCON 2015 - 2015 IEEE Region 10 Conference*, pages 1–6, Nov 2015.
- [44] Hiroyuki Kanbara, Hiroyuki Okuhata, Masanao Ise, Ryota Kinjo, and Yuki Toda. Probability of calculation failures by soft errors in an embedded processor core. In *2009 IEEE 8th International Conference on ASIC*, pages 601–604, Oct 2009.
- [45] Abdulaziz Eker and Oguz Ergin. Exploiting existing copies in register file for soft error correction. *IEEE Computer Architecture Letters*, 15(1):17–20, Jan 2016.
- [46] "OpenMSP430 project", [Online]. Available at "<https://opencores.org/projects/openmsp430>".
- [47] "Texas Instruments.", MSP430. [Online] Available: "<http://www.ti.com/lit/ug/slau049f/slau049f.pdf>".
- [48] Jianli Li, Qingping Tan, and Jianjun Xu. Reconstructing control flow graph for control flow checking. In *2010 IEEE International Conference on Progress in Informatics and Computing*, volume 1, pages 527–531, Dec 2010.