TAL TECH

DOCTORAL THESIS

A Synthetic, Hierarchical Approach for Modelling and Managing Complex Systems' Quality and Reliability

Aneesh Balakrishnan

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY
TALLINN 2022

TALLINN UNIVERSITY OF TECHNOLOGY
DOCTORAL THESIS
11/2022

A Synthetic, Hierarchical Approach for Modelling and Managing Complex Systems' Quality and Reliability

ANEESH BALAKRISHNAN

TAL TECH PRESS

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Computer Systems

The dissertation was accepted for the defence of the degree of Doctor of Philosophy in Computer and Systems Engineering on 08 March 2022

Supervisor: Prof. Maksim Jenihhin,
Department of Computer Systems, Tallinn University of Technology,
<maksim.jenihhin@taltech.ee>,
Tallinn, Estonia

Co-supervisor: Dr. Dan Alexandrescu,
iRoC Technologies,
<dan.alexandrescu@iroctech.com>,
Grenoble, France

Opponents: Prof. Dr. Dimitris Gizopoulos,
University of Athens,
<dgizop@di.uoa.gr>,
Athens, Greece

Prof. Dr. Paolo Rech,
Universitá di Trento,
<paolo.rech@unitn.it>,
Trento, Italy

Defence of the thesis: 03 May 2022, Tallinn

Declaration:
Hereby I declare that this doctoral thesis, my original investigation and achievement, submitted for the doctoral degree at Tallinn University of Technology, has not been submitted for any academic degree elsewhere.

Aneesh Balakrishnan _____
signature

ISSN 2585-6898 (publication)
ISBN 978-9949-83-804-2 (publication)
ISSN 2585-6901 (PDF)
ISBN 978-9949-83-805-9 (PDF)
Printed by Koopia Niini & Rauam

TALLINNA TEHNIKAÜLIKOOL
DOKTORITÖÖ
11/2022

Sünteetiline, hierarhiline lähenemine keerukate süsteemide kvaliteedi ja töökindluse modelleerimiseks ja haldamiseks

ANEESH BALAKRISHNAN

TAL TECH
KIRJASTUS

Contents

List of Publications

The articles corresponding to Roman numbers are disseminated in section Reference with Natural numbers.


I A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, "The validation of graph model-based, gate level low-dimensional feature data for machine learning applications," in *2019 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, IEEE, oct 2019


II A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, "Modeling gate-level abstraction hierarchy using graph convolutional neural networks to predict functional de-rating factors," in *2019 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, IEEE, jul 2019


III A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, "Composing graph theory and deep neural networks to evaluate SEU type soft error effects," in *2020 9th Mediterranean Conference on Embedded Computing (MECO)*, IEEE, jun 2020


IV M. Jenihhin, M. S. Reorda, A. Balakrishnan, and D. Alexandrescu, "Challenges of reliability assessment and enhancement in autonomous systems," in *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, IEEE, oct 2019


V T. Lange, A. Balakrishnan, M. Glorieux, D. Alexandrescu, and L. Sterpone, "On the estimation of complex circuits functional failure rate by machine learning techniques," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks – Supplemental Volume (DSN-S)*, IEEE, June 2019


VI X. Lai, A. Balakrishnan, T. Lange, M. Jenihhin, T. Ghasempouri, J. Raik, and D. Alexandrescu, "Understanding multidimensional verification: Where functional meets non-functional," *Microprocessors and Microsystems*, vol. 71, p. 102867, nov 2019


VII A. Balakrishnan, D. Alexandrescu, M. Jenihhin, T. Lange, and M. Glorieux, "Gate-level graph representation learning: A step towards the improved stuck-at faults analysis," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pp. 24–30, 2021


VIII D. Alexandrescu, A. Balakrishnan, T. Lange, and M. Glorieux, "Enabling cross-layer reliability and functional safety assessment through ML-based compact models," in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, IEEE, jul 2020


IX A. Balakrishnan, G. C. Medeiros, C. C. Gürsoy, S. Hamdioui, M. Jenihhin, and D. Alexandrescu, "Modeling soft-error reliability under variability," in *2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 1–6, ***Award: Outstanding Student Research Paper***, 2021

Other Related Publications

X T. Lange, A. Balakrishnan, M. Glorieux, D. Alexandrescu, and L. Sterpone, "Machine learning to tackle the challenges of transient and soft errors in complex circuits," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, IEEE, jul 2019

XI T. Lange, A. Balakrishnan, M. Glorieux, D. Alexandrescu, and L. Sterpone, "Machine learning clustering techniques for selective mitigation of critical design features," in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 1–7, 2020

XII X. Lai, T. Lange, A. Balakrishnan, D. Alexandrescu, and M. Jenihhin, "On antagonism between side-channel security and soft-error reliability in bnn inference engines," in *2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 1–6, 2021

Author's Contributions to the Publications

I The author, as principal investigator, planned and performed the experiments. The author performed the computer simulations, processed and compared the obtained results with state-of-the-art. The author wrote the paper.

II The author defined the fundamental research problems. The author performed the computer programming and Electronics Design and Automation (EDA) tool simulations. The author processed and compared the results. The author wrote the paper.

III The author who planned and executed the experimental methods and compared the results. The author wrote the paper.

IV The author contributed to writing and revising the paper and took part in the active discussion of proposed scientific approaches.

V The author contributed to writing and discussing the naive models for this publication.

VI The author contributed in writing manuscript and plays a role in the reliability metrics requirement analysis and helps to summarize the facts in taxonomy fashion.

VII The author, as the principal supervisor, defined the experimental methods, executed the EDA and computer simulations, and compared the results. The author wrote the manuscript.

VIII The author contributed to research discussions, execution of computer simulations and paper writing.

IX The author defined the scientific aims and performed the EDA and computer simulations. The author generated and compared the results. The article wins the Outstanding Student Research Paper award.

Acronyms

ADD	Algebraic Decision Diagrams
AI	Artificial Intelligence
ANN	Artificial Neural Network
ASC	Advanced Simulation and Computing
ASIC	Application-Specific Integrated Circuit
BFS	Breadth-First Sampling
BPSG	Borophosphosilicate Glass
BTI	Bias Temperature Instability
CCDs	Charge-coupled devices
CI	Confidence Interval
CNN	Convolutional Neural Networks
CPT	Conditional Probability Table
CUT	Circuit Under Test
DC	Design Compiler
DFS	Depth-First Sampling
DL	Deep Learning
DNN	Deep Neural Network
DRAM	Dynamic Random Access Memory
DVF	Data Vulnerability Factor
EDA	Electronic Design Automation
EDR	Electrical De-rating
EVS	Explained Variance Score
FDR	Functional De-rating
FFC	Functional Fault Coverage
FFR	Functional Failure Rate
GCN	Graph Convolutional Network
GML	Graph Modeling Language
GRF	Geomagnetic Reference Field
HDL	Hardware Description Language
HPC	High-Performance Computing

IBM	International Business Machines
ICD	Implantable Cardioverter Defibrillator
ICs	Integrated Chips
IEC	International Electrotechnical Commission
IP	Intellectual Property
ISO	International Standardization of Organization
LBL	Lawrence Berkeley Laboratory
LDR	Logical De-rating
LSI	Large Scale Integration
LSTM	Long Short Term Memory
MAE	Mean Absolute Error
MAX	Max Absolute Error
ML	Machine Learning
MRF	Markov Random Fields
MSE	Mean Squared Error
PDF	Probability Distribution Function
PGM	Probabilistic Gate Model
PLI	Programming Language Interface
PTM	Probabilistic Transfer Matrices
RAMs	Random Access Memories
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SDF	Standard Delay Format
SEE	Single Event Effect
SEMM	Soft-error Monte Carlo Modeling
SET	Single Event Transient
SETs	Single Event Transients
SEU	Single Event Upset
SEUs	Single Event Upsets
SoC	System on Chip
SOTA	State-of-the-Art
SPICE	Simulation Program with Integrated Circuit Emphasis
SPP	Signal Probability Propagation
SPRA	Signal Probability Reliability Analysis
SRAMs	Static Random Access Memories
SVM	Support Vector Machine
TDR	Temporal De-rating
TI	Texas Instruments
TID	Total Ionising Dose
VPI	Verilog Procedural Interface

Chapter 1

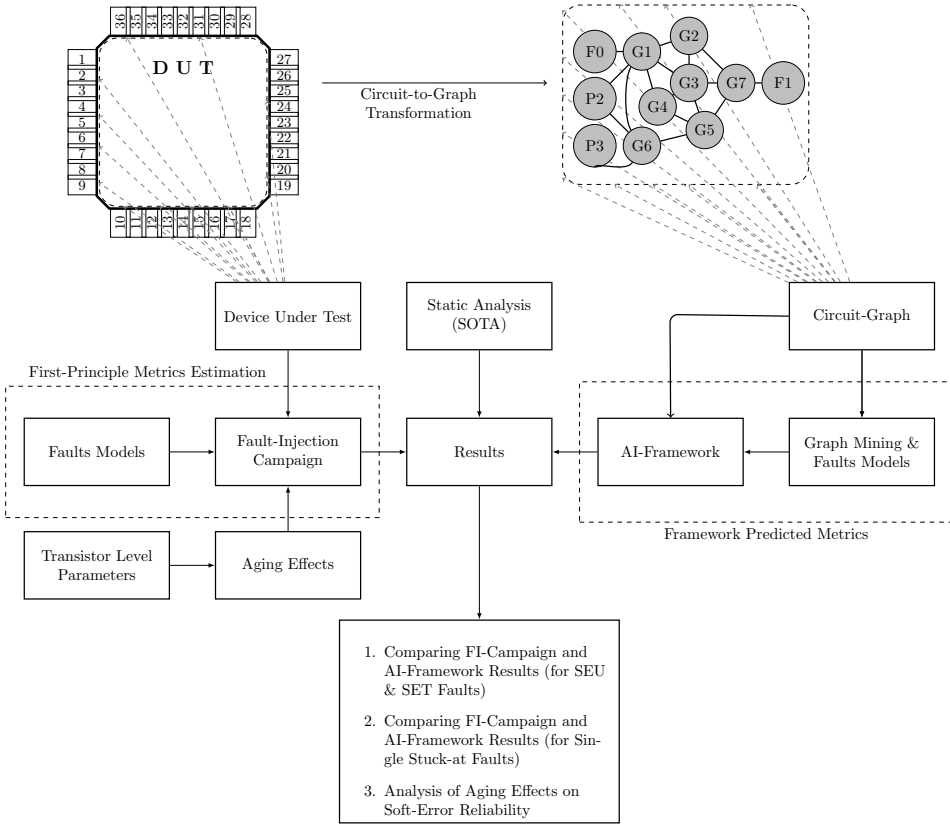
Introduction

The doctoral thesis addresses today's high-performance designs requirements in terms of validation and reliability. The project aims at developing a comprehensive approach that includes EDA modules and tools, design methodologies, and testing practices for modeling and managing the quality of complex designs and systems. The objective of the research is to significantly enhance and develop new statistical and probabilistic methods and algorithms for commercially significant and industrial level internal tools such as TFIT (cell-level SER analysis) and SoCFIT (circuit-level reliability analysis) from IROC Technologies. In addition to the software advancement for Electronic Design Automation (EDA) based error-and-fault evaluation in complex designs: this project also tries to use failure analysis from field data and also tries to improve hardware fault injection through radiation and emulation. The proposed research themes contribute towards the development of an industry-wide reliability framework and set of tools. The tool specifications are established in collaboration with important companies from the networking and automotive applications. IROC industry and academy collaborators both provide test cases.

The circumstance of the doctoral thesis formulation depends on the European Training Network, known as RESCUE [13], which aims in scientifically advanced innovative training for addressing interdependent and dependent challenges in different aspects like Reliability, Security, and Quality regarding the Micro-and-Nanoelectronic System designs. Nanoelectronic systems are at the core of all industry sectors and deployed in life-critical application domains, such as healthcare, transportation, automotive, and security, serving societal needs all over the world.

The core idea of the research is to represent and overcome the challenges in characterization and modeling of the propagation of radiation-induced errors such as Single Event Upsets (SEUs) and Single Event Transients (SETs) as technology advents to the smallest possible feature size. The Ph.D. thesis aims to develop algorithms to actualize relatively complex modeling of radiation-induced error propagation. Alongside that, the research incarnates the developed algorithms to evaluate the effects of the single stuck-at faults. The established principles and methods are cross-validated with fault-injection campaign results as well as the mathematical approaches from State-of-the-Art (SOTA). Moreover, the project expands its research premises to understand the soft-error reliability under the variabilities due to Bias Temperature Instability (BTI) in deep submicron technologies.

Fig. 1.1 provides a schematic visualization of the methods through which the thesis aims to achieve its principal focuses, and the figure point out the principal focuses. The outline in Fig. 1.1 delineates how the thesis research approaches emerge and how those methods are validated.



The Principal Focuses of The Thesis

Figure 1.1 – The Thesis Silhouette

1.1 Motivation

Nano/micro-electronic systems are at the core of all industry sectors and deployed in critical application domains. Such versatile applications demand high-quality standards and metrics for interdependent and dependent challenges in different aspects like Reliability, Security, and Quality. The backbone of the thesis is the problem that is motivated from the perspective of Soft Error Rate (SER) analysis in the functional response of Micro-and-Nanoelectronics systems. As the technology scale-down continues in response to the interest of electronic industries, corporate giants, and needs of high-performance applications, radiation-induced errors and chip-environmental effects are collectively enhancing the critical failures in systems and circuits.

Soft errors in electronic systems are the problems for applications that either require bulk amounts of memories or have very critical reliability requirements. A spectacular example, known as the "Hera" problem of IBM [14]. During 1986, IBM observed an increase in failures of their LSI memories manufactured in the USA. Another company that encountered a major SER issue was Cisco systems in 2003 [15]. Router line cards of the 12000 series, with a selling price of about $200,000, showed failures caused by radiation-induced soft errors. Characterization of single event effects in safe and reliable micro-

electronics for medical applications is challenging problem from the perspective of a high reliability and quality requirements. P. D. Bradley and E. Normand, in [16], delineate the effects of Single event upsets in implantable cardioverter defibrillators. In 2005, St. Jude Medical had identified a particular vendor-supplied memory chip (Static Random Access Memories (SRAMs)) that can be affected at a low-frequency rate by background levels of atmospheric ionizing cosmic radiation ("background cosmic radiation") [17]. This problem had been found in a limited number of St. Jude Medical Implantable Cardioverter Defibrillator (ICD) products. The anomaly can trigger a temporary loss of pacing function and permanent loss of defibrillation support. An aerospace anomaly was reported during the FOTON-2/BIOPAN-5 mission, a space exposure experiment [18]. FOTON M2 was launched from Baikonor Cosmodrome in Kazakhstan on 31 May 2005. Unfortunately, the BIOPAN was failed during the 5th orbit in a total of 253 orbits. The employed commercial microcontroller in BIOPAN is Cygnal C8051F124. The post-flight analysis discovers a high current consumption. Further evaluations confirm a nondestructive latch-up event that occurred on one or more SRAM modules during the mission. A latch-up condition in SRAM was concluded as the possible cause. A perfect example of the backlash due to the single event effects at the earth's surface is the unexpected results in the election on May 18, 2003, in the city of Schaerbeek, marked the biggest day in the history of Brussels. The identified reason is the unanticipated malfunction of voting machines when it interacts with cosmic radiation.

Followed by the reports of real-time critical failure incidents, different authenticated reliability-focused standards have been introduced and reviewed currently. ISO-26262 for the automotive industry and DO-254 for the avionics systems belong to such standards. Based on the knowledge of current reliability-focused standards such as ISO26262 (automotive) or DO-254 (avionics), this thesis is focusing on lead development of the reliability assessment tools and easing the preparation process of reliability reports and safety manuals for high-reliability designs and systems. The requirement of quality digital models is necessary for interpreting current or new types of faults and defects: that are affecting the microelectronic process, technology, standard cell libraries, and complex designs. The scientific expertise that is acquiring through this thesis is relevant in proposing such models.

The problem of the thesis work is motivated from the perspective of soft-error analysis in circuits and systems. The principal aim is to research and develop comprehensive solutions for analyzing and improving the overall quality of devices. To accomplish such a complicated task, strong scientific research is planned to conduct through this thesis. That competitive research aims at addressing: the independent reliability challenges, problems regarding functional safety, and quality, aging, process variability in High-Performance Computing (HPC).

Past literature works [19], [20] [21] [22], and their contributions for the probabilistic modeling of the soft error analysis prove that several factors are prominent to consider in determining the SER of a logic circuit. The first factor is the energy of radiation-induced error (especially for the SET error) because it must have sufficient amplitude to change a signal and propagate as an erroneous signal value through subsequent gates; if not, the fault is electrically masked. The electrical derating factor for SEU error is negligible. The second prominent factor is logical derating because it determines the erroneous value's propagation probability through the logic network and their chances to affect the primary output. The fault must reach a flip-flop during the sensitive portion of a clock cycle, known as the latching window; if not, the fault value is temporally masked. This factor is called temporal derating. Now the thesis importance is to investigate how these derating factor

variations changes with circuit downscaling, temperature, and types of workloads/applications. The complexity of the research becomes more challenging when the probability of a fault to be masked depends on the labyrinth datapath of the error as it encounters on its way to the primary output, i.e., the case of considering path-dependency of faults. Similarly, the propagation delay of the SEU, before reaching a latch or output pins, depends on the gate and interconnect delays along the path in which it traverses. Because different input vectors can sensitize the same sets of datapaths differently, the probability for non-controlling values to present on any traversal path of SEU faults is high. These factors such as the probability of the fault occurring at the input of gates, the vulnerability of error locations at the given logical networks, the error attenuation probability on a given path, and finally, the signal probability at the input of the erroneous gates, are ultimate challenges in the scope of the thesis.

1.2 Problem Formulation

The objective of the research is to significantly enhance and develop new statistical and probabilistic methods and algorithms for gate-level reliability analysis and management. The fundamental challenges are: how to address and tackle the emerging complexity and challenges in estimating the propagation probability of soft-errors as the fabrication of chips becomes more densely packed.

The principal research aims of the thesis and corresponding raised research questions/challenges in the thesis works are the following:

1. The main aim of the thesis is to investigate the radiation-induced uncertainties and failures in logic circuits. The new circuit and chip technologies are more vulnerable to soft-errors due to cosmic radiations, thermal energies, and voltage scaling. In order to limit the exacerbation of the impact caused by soft-errors in the logic circuits, a dedicated software tool is unconditionally required. However, when dealing with today's large complex circuits, traditional approaches such as accelerated fault simulation and radiation facilities require a huge investment of time and resources. ***The followed challenges are solved by automation and scripting, mathematical and statistical modelings, EDA simulations, and metrics verification.***

    - How to apply an algorithm or mathematical models to the components in the gate-level circuit?
    - What are the most significant factors in the propagation of soft-errors in gate-level circuits, and how to overcome the process of modeling such factors?
    - Which type of mathematical algorithms or models are suited to propose a comprehensive probabilistic-based reliability analysis tool?
    - What are the challenges in considering different metrics to verify the proposed models with comprehensive and exhaustive fault-injection methods?
    - How to provide a framework or tool that explains the relatively in-depth science behind the propagation of SET/SEU errors in the circuit?
    - How to assess the proposed framework is significant compared to the state-of-the-art methods, and how the scalability of the framework be verified?

2. A standardization of soft-error reliability is not the ultimate solution to characterize the overall quality of a system. So the proposal of a multidimensional perspective to integrate quality and reliability analysis and expanding the developed algorithms to achieve this goal is the second aim of the thesis. ***Automation and scripting,***

mathematical and statistical modelings, EDA simulations, and metrics verification are those methods that solve the following questions.

- How to incorporate a quality analysis part to the developed reliability analysis framework?
- What are the main faults and factors that need to include for the quality analysis integration?
- What are the additional gains of this multi-dimensional verification tool?

3. Comprehensive research studies provide a description that the soft-error vulnerability becomes more severe as the circuit performance degrades with aging. However, such increased soft-error generation does not necessarily contribute towards circuits' critical functional failures. The third aim of this doctoral thesis is the experimental investigation of soft error propagation at the aged gate-level by considering the different derating factors like Electrical Derating (EDR), Temporal Derating (TDR), Logical Derating (LDR), and Functional Derating (FDR). ***The followed questions are answered by automation and scripting, mathematical and statistical modelings, EDA simulations, and metrics verification.***

   - What is the propagation behavior of radiation-induced errors in aged circuits?
   - How to evaluate the propagation of SEU/SET events in an aged circuit?
   - How effectively will derating factors and their influences change in an aged circuit compared to non-aged circuit parameters?
   - How to model the changes in the propagation behaviors SET and SEU separately?

1.3 Contributions

As an affirmative to the statement that the exhaustive fault injection method is the ultimate reliability assessment method in terms of accuracy, this thesis explores such comprehensive fault-injection campaigns as the first principle method for a realistic comparison of the reliability analysis with the alternately developed frameworks. The comprehensive fault-injection campaigns are very inconvenient in terms of time and EDA licenses. Such drawbacks make this first principle approach infeasible on medium and large-scale circuits. Therefore, new testing and evaluation methodologies based on computational paradigms are inevitable. The contributions of the thesis emphasize the methods and solutions that tackle the challenges in reaching the expected aim.

The expected aim or fundamental idea in providing an alternate solution is to avoid unreasonable test costs of the traditional way of reliability assessment by maintaining good statistical significance in the results of the proposed scope. Research proposals based on Graph Theory and Deep Learning (DL) techniques are more advanced and greatly favored by researchers to learn statistical dependencies of system function based on related parameters. This motivation develops into applying graph embedding methods like node2vec, GCN, and GraphSAGE algorithms and aiming to find the best way to generate relevant feature databases from the gate-level netlist, and subsequently applying to a downstream deep neural network for the functional failure reliability metric assessment.

Chapter 3 briefs this contributions in its own versatility. The published scientific articles [I], [II], [III], [IV] and [V] are the basics of this contributions. The primary steps of developing a reliability framework contribute towards an enhanced compiler and Verilog reader

for the design in the gate-level netlist. The thesis developed a gate-level to graph conversion tool that enables researchers to implement a probabilistic algorithm to infer the error/fault propagation probability. The major thesis turning point is the perception of the gate-level netlist in a graph network. Such graph network is compatible with any latest and highly complex mathematical algorithms (or) more enhanced artificial intelligence-based functions. ***The first part of the thesis's contributions can be summarized as:***

- *A scalable tool for converting the gate-level netlist to a graphical network for simplifying the statistical reliability analysis, is generated.*
- *A very reliable methodology to visualize the netlist and to analyze the component's frequency distributions is integrated to the framework. This methodology enables extraction of the graph nodes' (circuit components') fundamental structural properties.*
- *Applied highly sophisticated graph embedding algorithms (for e.g., node2vec, Graph Convolutional Network (GCN) and graphSAGE) to the circuit-graph and documented the quality of embedded database for the inference of fault propagation probability.*
- *The framework also tested a feed forward deep learning network as well as support vector machine as the inference engine for the embedded graph database.*
- *A comparison of the predicted reliability metrics (including Confidence Intervals, correlation based metrics and error metrics) with that of the random fault-injection campaigns and the exhaustive fault-injection campaigns, is verified and documented.*
- *The developed AI-Framework for the reliability analysis is able to predict functional failure probabilities due to SEUs and SETs as given in* (2.3) *and* (2.5) *respectively. More complex probability factors have to be considered here.*
- *The experiments provide more factual explanations for choosing the framework parameters for analysis of SEU and SET caused functional failures. The whole framework potentially supports the white-box modeling (or) explained artificial intelligence rather than black-box modeling.*
- *Validated and tested the framework for a CPU system model as well as a IP component for different workloads.*

Application of Artificial Intelligence (AI) to extract feature information from the probabilistic network domain [23], has emerged as a prominent tool for graphical node embedding. The process of leveraging a node's features into a vector form is called node embedding. The applications of graph-based neural network algorithms (GCN, DNN, and graphSAGE) for circuit reliability estimation have been proposed in papers [I], [II], and [III], respectively. The papers [I] and [II] proposed algorithms that evaluate the propagation probability of Single Event Upsets (SEUs) in a transductive way. On the other hand, the paper [III] implemented an inductive type framework. The transductive model is not exactly building a predictive model. A completely new unseen data point (a component in the circuit netlist) forces the transductive algorithm to re-run the training phase. But inductive learning builds a predictive model that can also apply to the unseen data.

The thesis contribution is not limited to reliability analysis. In the scientific journal [VI], I, the principal investigator of this thesis, have contributed to establishing a multidimensional perspective to ensure standard quality for a system. This contribution includes describing metrics requirements for the reliability due to radiation-induced error that should

exist in harmony with other reliability aspects and considering various extra-functional domains of the electronic systems' design at the chip design level in nanoscale technology. This motives to extend the developed AI framework to evaluate the effect of single stuck-at faults at the functional level [VII]. The transformed gate-level graph is the source for extracting the features of nets to model the effects of stuck-at faults at the functional level. The proposed framework goes beyond the classical machine learning algorithms (like support vector machine, logistic regression, and linear regression) by replacing the black-box modeling with transparent modeling of the metrics (i.e., white-box modeling [24]). ***The second part of the thesis's contributions can be summarized as follows:***

- *An adapted framework that introduces an edge-to-vertex graph transformation principle to analyze the effects of permanent faults instead of soft-errors.*
- *An algorithm that illustrates: how to predict the propagation of induced stuck-at-1 and stuck-at-0 faults at each net (wire in a netlist).*
- *Better numerical superiority in the fault propagation probability predictions and interestingly reduces the time complexity by 60%.*
- *The characterized failure vulnerabilities result in the reduced set of fault locations for fault diagnosis.*

The circuit variations and repercussions of aging are highly influencing factors in the reliability of a circuit. As per the previous articles in this domain, mainly states that while aging, the occurrence of radiation-induced errors will increase on behalf of the declined critical charge Q_{crit} at the technology cells. The SER is not observed in diverse propagational forms of Single Event Transient (SET) and Single Event Upset (SEU) separately. Contrary to previous findings, the results in this thesis that comprised of research articles [VIII] and [IX], concentrate on the soft-error generation and propagation by considering all the derating factors like Electrical De-rating (EDR), Logical De-rating (LDR), Temporal De-rating (TDR), and Functional De-rating (FDR) associated with an aged Standard Delay Format (SDF) file. The soft-error reliability analysis of aged circuits at higher abstraction (e.g., RTL) is the future vision of this work. ***The third and final part of the relevant contributions of the thesis to the soft-error reliability analysis under aging include the following:***

- *Characterization of threshold voltage degradation (ΔV_{th}) for industrial* 15 nm *technology*
- *Cross-level Modeling of NBTI-induced delay degradation using Artificial Intelligence (AI)*
- *Analysis of derating factors' influence in soft-error propagation*
- *The validation of a signal-processing model to locate exact time-slots of propagating SET generation that have high probability to propagate into the target flip-flop in a critical path*

1.4 Thesis Organizations

The full version of the doctoral thesis is subjectively perceived and arranged as follows:

- **Chapter 1: Introduction**
  The chapter discusses the current and general idea of the researched topic. It states the circumstances in identifying the scope of the research aims and the challenges at that particular subject matter.

- **Chapter-2: History, Background and Tools**
  The introduction chapter paves very strong fundamentals to the reader to understand basic legitimate phenomena and their passive scientific observations in its versatility. This chapter introduces the radiation induced soft-error generation and how the environmental factors accelerates and decelerates its effects on the microelectronic circuits and systems.

- **Chapter-3: An AI-Framework for Soft-Error Reliability**
  The principal objective of the thesis is to research and develop scientific methods to accelerate the reliability estimation due to radiation-induced error propagation in circuits systems and model the corresponding factors that derate the propagation effects. This chapter provides a deep insight into the tools and methods that effectuate the accelerated soft-error reliability estimation.

- **Chapter-4: Integration of Safety, Quality and Reliability**
  System quality and its requirements up to a standard level compete with the system reliability requirement challenges. The chapter intends to provide a multidimensional perspective on required system quality, reliability, and their integration. The core of this chapter relies on an AI-based algorithm that validates the effects of single stuck-at faults at the functional level of system circuits.

- **Chapter-5: Soft-Error Reliability Under Time-Dependent Variability**
  Bias temperature instability (BTI) is one of the dominant reliability challenges in nanoscale CMOS technology. Negative BTI is subject to PMOS devices under negative gate voltages at elevated temperatures. System engineering continuously focuses on aggressive technology scaling and increases the vulnerability of radiation-induced soft errors simultaneously. A close observation of both effects and modeling their combined effects is the principal aim of this chapter.

- **Chapter-6: Conclusion and Perspective**
  This chapter provides a conclusion to the holistic approaches in the thesis.

Chapter 2

History, Background and Tools

This chapter briefly reviews different nomenclature and types of radiation-induced soft errors and faults. The discussion expands to the soft error sensitivity of logic components and its relation with technology scaling and the propagation characteristics of the induced error and collectively prognostics effects of temperature/environmental variabilities in the system. The chapter briefing continues to introduce various methodologies based on the AI methods. To simplify the use case of the word "radiation-induced soft error", a compact term called "soft-error" is used synonymically in the rest of the thesis.

2.1 Preamble of Chapter

The recent rapid expansion in the application of autonomous systems has triggered numerous unprecedented novel businesses services with social benefits. However, the unleashed benefits accompany the unforeseen circumstances for computationally challenging mission- and safety-critical application scenarios. The use of avant-garde computing architectures to employ high-performance nanoelectronic autonomy in the system design proliferate the prodigious complexity and heterogeneity of today's advanced cyber-physical systems and system-of-systems. In the latest centuries, technology advanced to incorporate artificial intelligence in autonomous robotic vehicles/systems. Therefore, system reliability is often an enabling factor for a new product or technology on the way to market.

In the electronic product development phase of the autonomous systems, each integral component should comply with standard functional safety features ranging from the specification to design implementation, integration, verification, validation, and production release. The standard International Standardization of Organization (ISO) 26262 is an adaptation of the Functional Safety standard of International Electrotechnical Commission (IEC) 61508 for Automotive Electric/Electronic Systems. ISO-26262 defines functional safety for automotive equipment applicable throughout the lifecycle of all automotive electronic and electrical safety-related systems.

To ensure standard functional safety to the advanced commercial electronic components and systems, more dedicated and commercially qualified tools are required for the relibilty analysis. In [25] Robert C. Baumann states that the once-ephemeral radiation-induced soft error has become a key threat to advanced commercial electronic components and systems. If the radiation-induced soft error left unchallenged, they become potential sources for inducing the highest failure rate of all other reliability mechanisms combined depending on the applications.

2.2 Historical Perspective of Soft-Error Effects' Analysis

The case of soft-error anomalies are firstly reported in a scientific paper written by Binder in [26]. According to them, "anomalies" in communication satellite operations have been caused by the unexpected triggering of flip-flop circuits, and the authors investigate interactions with galactic cosmic rays as an alternate mechanism for the satellite anomalies. The particular phenomenon that counts as the reason behind the cosmic ray interaction caused electrical undulation, was the charging of the base-emitter capacitance of critical transistors to upset the voltage. The charge is accumulating by the dense ionization track of an energetic, high-Z cosmic ray.

Intel Researchers (May and Wood [27]) presented a new physical mechanism for soft errors in DRAMs in 1978 at the International Reliability Physics Symposium (IRPS). The paper clearly states that the new physical soft error mechanism in dynamic Random Access Memories (RAMs) and Charge-coupled devices (CCDs) is the upset of stored data by the passage of heavily ionizing radiation like alpha through the memory array space. Alpha particles in the radioactive decay of uranium and thorium present in parts-per-million levels in package materials penetrate the die surface and create enough electron-hole pairs near a critical storage node to generate a random, single-bit error.

Ziegler (IBM) and Lanford (Yale) researched at the same time about the generation of secondary particles due to nuclear interactions between cosmic rays and silicon materials that might trigger errors[28]. Also, extends their research to the sea-level flux of cosmic-ray particles [29]. The interaction of each type of particle with silicon is estimated on emphasis with processes that produce bursts of charge. In the work [30], an experiment has been designed to prove the ability of iron-roup cosmic rays to generate such errors by depositing soft-errors in solid-state static RAMs by iron nuclei from the Lawrence Berkeley Laboratory (LBL) Bevalac. Subsequently, various de-lidded device types were experimented with the beams of argon and krypton ions using the LBL 88-inch Cyclotron accelerator at energies near $2\,\mathrm{MeV/nucleon}$. The deduced conclusion from such tests reveals that some cells are essentially immune to bit error while others are quite vulnerable. In the year 1979 [31], Guenzer irradiated dynamic RAMs with neutrons having mean energies of 6.5, 9, and $14\,\mathrm{MeV}$ and with $32\,\mathrm{MeV}$ protons, and have been observed the generation of single event upset.

In 1983, researchers at IBM also commenced publishing their articles regarding the framework for modeling the diffusion and collection of charge induced by energetic particles [32]. By 1996, P. C. Murley [33] from IBM introduced Soft-error Monte Carlo Modeling (SEMM) program. In [14], J. F. Ziegler covers the historical review of IBM experiments in appraising radiation-induced soft fails in Large Scale Integration (LSI) electronics over fifteen years (1978-1994), concentrating on major scientific and technical advances that have not been published before 1994.

The principal author J. F. Dicello of the work [34] in 1983, reported that soft-error rates in a 4K static RAM have been observed for 164 MeV/c pions and 109 MeV/c muons and concludes that effects of pions in cosmic rays at sea level may not be negligible. IBM initiated unaccelerated real-time SER tests in 1983, using a portable tester with several hundreds of chips. Real-time SER testing is also known as field-testing or system SER (SSER) testing. The IBM measurements provided evidence for the significant contributions of cosmic radiation at sea level to the SER, and its impact increases exponentially with altitude and detailed publication of relative change of SER with altitudes are provided in [35] [36].

In the early 1990s [37][38], International Business Machines (IBM) developed an ion micro-beam radiation system that has been used to probe the relative effect of SEUs in individual circuits and nodes of a CMOS Dynamic Random Access Memory (DRAM). With

this system, it is possible to direct an ion beam of diameter as small as 1 µm onto a circuit or test structure with a placement accuracy of 1 µm.

In 1993 C. Lage from Motorola also presents a quantitative model which attributes most soft errors in dense SRAMs not to alpha particles as is commonly accepted, but to cosmic ray events in his work [39].

In 1995, Baumann [25] from Texas Instruments (TI) investigated that boron compounds are a non-negligible source of soft errors. The interaction of cosmic ray neutrons and boron is demonstrated as the dominant source of alpha particles and other radiations in electronic devices utilizing Borophosphosilicate Glass (BPSG).

In 2004, H. H. K. Tang re-emphasized the Monte Carlo program (SEMM tool) through SEMM-2 [40], a new simulation system for radiation-induced single event upsets. Christopher Weaver from Intel in 2004, has proposed in his work [41] the various techniques to Reduce the Soft Error Rate of a High-Performance Microprocessor.

In 2005, Sarah E. Michalak from Los Alamos National Laboratory presented a paper [42] that proposing a statistical method for predicting the number of Fatal Soft Errors in Los Alamos National Laboratory's Advanced Simulation and Computing (ASC) Q Supercomputer.

The historical perspective and the SER trends are summarized by R. Baumann from TI in [43] [44]. Those papers briefly review the types of failure modes for SEs that appear in terrestrial applications and then addresses the sensitivity as a function of technology scaling for various memory and logic devices.

2.3 SEE Production Mechanisms

2.3.1 System Reliability and Soft-Errors

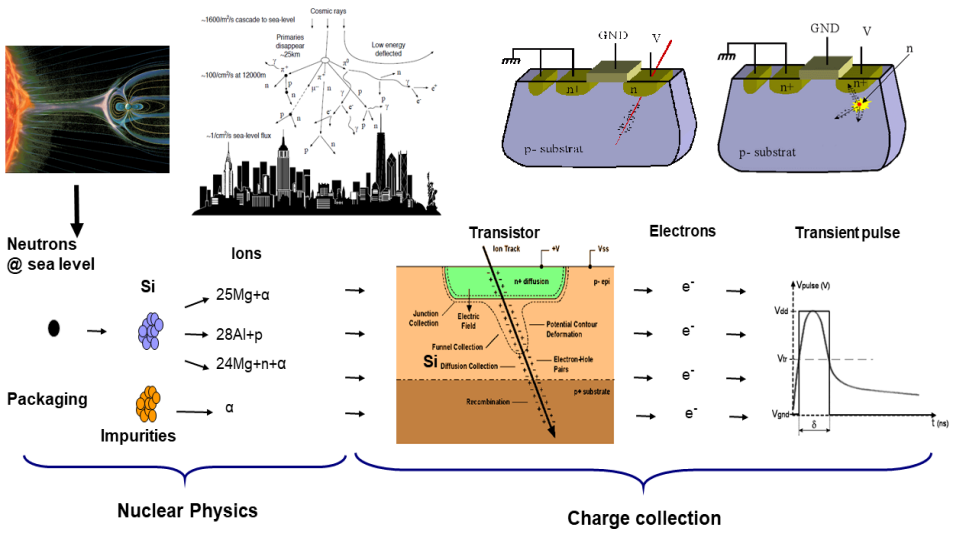


Figure 2.1 – Single Event Production Mechanisms (Courtesy: iROC's Internal Document)

A soft-error in the thesis context is a default usage of an inadvertent error (or) unidentified change in the electrical state of circuit components. The origin of such electrical change unconditionally depends on the environment external to system design. Soft-error is not directly causing permanent damages to the system components intact, but it leads to an unintended variation to classified system behavior based on the dependability of

applications. For real-time applications: automotive vehicular systems and navigation, biological assisting devices, and airborne systems (or) commercial data centers, soft errors remain undetectable and unmitigated, and their damaging influence enhances dangerous consequences of catastrophic failures.

Fig. 2.1 is a classical visualization of the generation of a transient pulse as result of the interaction of radiation particle with substrate elements in an electronic system environment. The transient pulse is explicitly known as Single Event Transient (SET) or implicitly known as Single Event Effect (SEE). The case of this once-ephemeral radiation-induced fault in a flip-flop or latched to flip-flop (or) to a memory element, is scientifically abbreviated as the Single Event Upset (SEU). Such 'Faults' will randomly cause observable 'Errors' and then leads to 'Failures' at the systems' functional level. Reliability is defined according to ISO as the degree to which a system, product, or component performs specified functions under constraints of specified conditions and time. In the holistic approach of this thesis article, the term reliability is more concentrated towards the soft-error-based reliability and issues in a system. Thesis approaches and dedicated analysis are modeling the factors which contribute towards soft-error reliability.

2.3.2 Energetic Particle Ionization

The ionization process and its characterization in semiconductor materials are subdivided into two main categorical processes: the direct ionization by the incident energetic particle and the ionization caused by secondary particles of nuclear reactions.

- **Direct Ionization:**
  In Fig . 2.2 from [25], the creation of the ionization track by the alpha particle is illustrated. In Fig. 2.2, part (a) shows on the onset of an ion passage through the substrate material, a cylindrical track of electron-hole pairs with submicron radius is created. This ionization track generates a high-carrier concentration. The resultant ionization track traverses or forms close to the depletion region, then carriers are rapidly collected by the electric field creating a large transient current/voltage peak at that node. A notable feature of the event is the concurrent distortion of the potential into a funnel shape [45]. The funnel in part (b) greatly enhances the efficiency of the drift current I_{drift} collection by extending the high field depletion region deeper into the substrate. The size of the funnel is a function of substrate doping. The funnel distortion increasing for decreased substrate doping. The drift collection phase is completed within a nanosecond (ns) and followed by a phase where diffusion (I_{diff}) begins to dominate the collection process and it is provided in part (c).

- **Indirect Ionization:**
  When the high-energy proton or neutron penetrates through the semiconductor lattice, a nuclear interaction with a target nucleus probably happens and each interaction can cause the direct liberation of charged particles. Such nuclear reactions include Elastic Interaction, Inelastic Interaction, Inelastic Collision, Nuclear Fission.

2.3.3 Source of Radiation: Natural Space Environment

The exposure of the microelectronics and the Integrated Chips (ICs) to the radiation environment is a fundamental reason for SEEs. The depth of hazards depends on the types of energetic particles, particle energies (or) deposited charge by particles, their fluxes, and fluences (Total Ionising Dose (TID)). A taxonomy of radiation environment types is

chronicled here. Altitude, angle of inclination, recent solar activity, and amount of spacecraft shielding have crucial influences on concentration and type of particles. According to Shwank in [46], the earth's natural space radiation environment include:

- **Trapped Radiation Domains**
  A geomagnetic cavity called the magnetosphere [47] is forming due to the earth's magnetic field. The naturally formed earth's magnetic field above the dense atmospheric space is occupied with trapped electrons, protons, and small amounts of low-energy heavy ions. The gyrated motion of energetic particles along the earth's magnetic field and the back and forth reflection between the pairs of conjugate mirror points: regions of maximum magnetic field strength along their trajectories in opposite hemispheres, cause the formation of highly populated space of electrons and protons. A parallel phenomenon of drifting electrons eastward around the earth, while protons and heavy ions drift westward, is also depicted in Fig.2. Fig. 2 [47] illustrates the spiral, bounce, and drift motion of the trapped particles.

  - **Electrons**: Electrons are subatomic negatively charged particles. The characterization of Energetic Van Allen belt electrons into "inner zone" and "outer zone" populations is provided in Fig. 2.3. The inner zone electrons are less severe compared to the outer zone electrons.
  - **Protons**: A proton is a subatomic particle that is symbolized by P^{+}, with a positive electric charge of +1e elementary charge and a mass slightly less than that of a neutron. Protons holding energies greater than $10\,\mathrm{MeV}$ populate regions 1 and 2 with an approximate trapping boundary placed at L = 3.8 as shown in Fig. 2.4. In contrast to the electrons, the energetic trapped protons (E > 1 MeV) occupy a volume of space which varies inversely and monotonically with their energy [47]. Consequently, these particles cannot be assigned to "inner" and "outer" zones.

- **Galactic Cosmic Rays**
  Galactic cosmic rays always present in the cosmic space of earth that originate from outside of our solar system. Cosmic radiation is entirely composed of galactic radiation in the absence of solar activity. The spectrum of galactic cosmic rays outside our solar system is estimated to be uniform. Its composition as a function of atomic mass is given in [47] [48] [49].

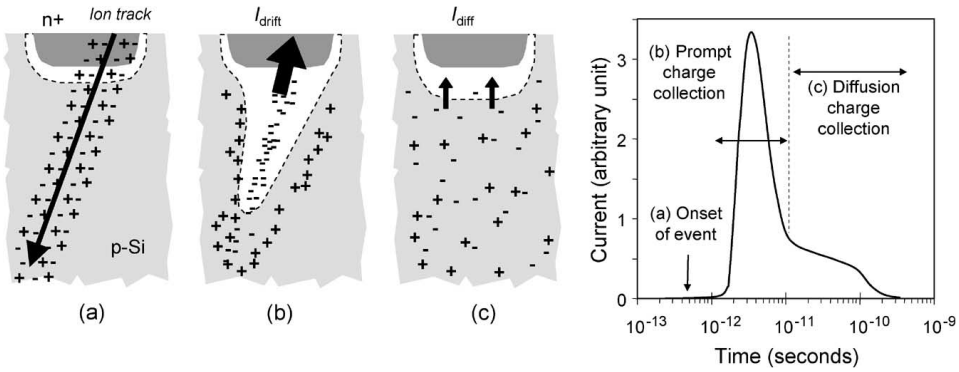


Figure 2.2 – Charge Generation and Collection Phases in a Reverse-biased Junction and the Resultant Current Pulse Caused by the Passage of a High-Energy Ion [25]

- **Solar Cosmic Rays**
  The so-called solar particles which are commonly known as solar flares or appearing due to the coronal mass ejections, are random but follow approximately an 11-year cycle of solar activity. A solar flare is attributed to a process outcome of a tremendous explosion on the sun when trapped energy in 'twisted' magnetic fields is subjected to a sudden release [50].

- **Terrestrial Radiation Environment**
  The relative possibility of cosmic rays particles reaching the atmospheric region of earth naturally depends on their penetration energy as demonstrated in Fig. 2.4. The arrival of cosmic radiation particles and collision with atomic nuclei in the air create cascades of interactions and reaction outcomes like Leptons, Photons, Hadrons, including neutrons, called air shower as depicted in Fig. 2.5.

  - **Thermal Neutrons**:
    A state at which neutrons are in thermal equilibrium with the atmospheric environment is the condition in which thermal neutrons release all their energy. These thermal neutrons possess low energy and probable to initiate a fission reaction, which becomes the source of charged particles [51] [52] and [53].
  - **Muons:**
    The high-energy interaction between the incoming protons and the atmospheric particles produces pions (π). Positive and negative pions are very unstable and almost immediately decay into positive and negative muons, respectively. The predominant particles at sea level is considered to be muons [54] [55]. Muons are charged particles; both negative and positive muons can deposit charges by the ionization process when they penetrates through matter [56]. Ziegler and Lanford in [28] point out the scientific machanism behind the interactions of muons with the matter at relatively low incident primary energies.
  - **Alpha Particle:**
    The main source of alpha particles is the packaging material. Alpha (α) particles are the aftereffect of the emission followed by the decay of unstable isotopes. The unstable isotopes are similar to doubly ionized Helium (He) nuclei and consist of two protons and two neutrons. Three radioactive decay chains that are primarily responsible for the α particles:

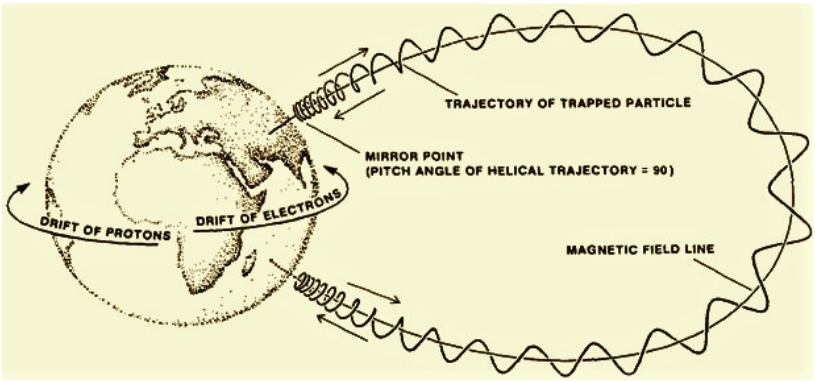


Figure 2.3 – Charged Particle Distribution in the Magnetosphere (After Ref. [46])

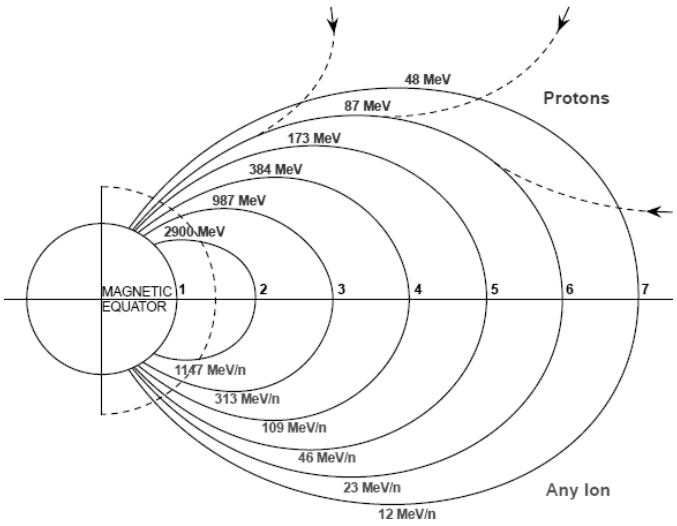


Figure 2.4 – Energy Requirement for Cosmic Rays in Magnetosphere Penetration (After Ref. [46] [47] [50] [57])

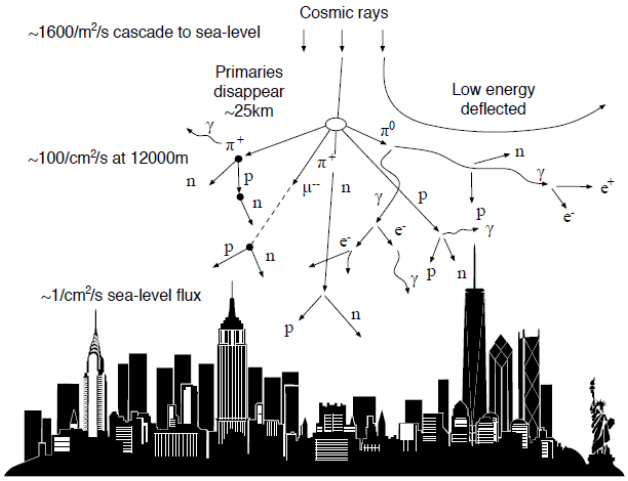


Figure 2.5 – Cosmic Rays Causing Cascades of Particles (After Ref. [57])

2.3.4 Source of Radiation: An Artificial Radiation Environment

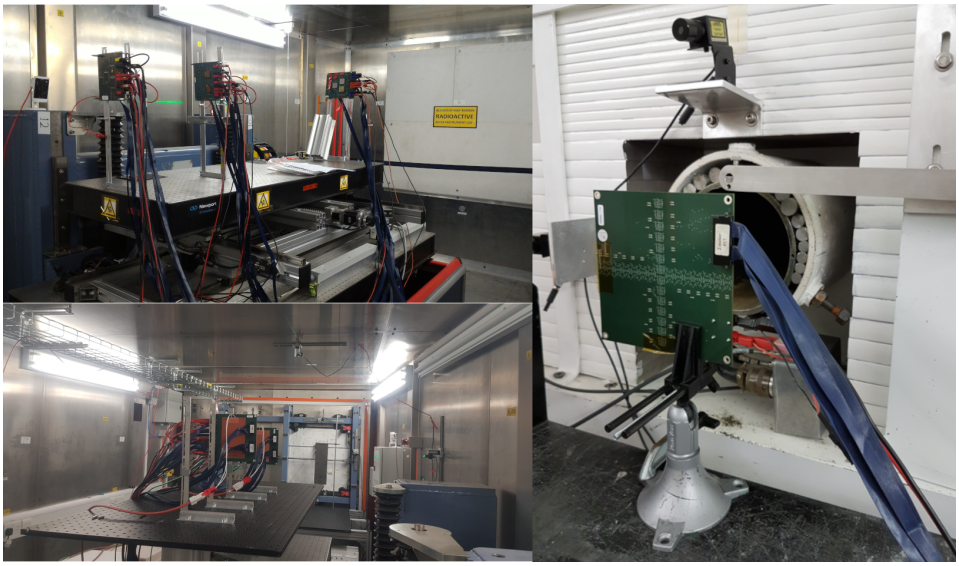


Figure 2.6 – Radiation Laboratories (Courtesy: IROC Testing Department)

A well-apprehended way of characterizing the response of electronic devices in space and nuclear radiation environments is the laboratory sources. A wide range of laboratories is available in Europe and USA for testing the circuit. The radiation sources in the lab are irradiation cells based on radioactive isotopes, sources based on the generation of x-rays, and particle accelerators. The Single Event Effects Particle Accelerators are available for characterizing heavy-ion and proton-induced single-event effects. These sources are versatile regarding ion species, energy, and flux. A sampling visualization of some of the most often-used accelerator facilities is shown in Fig. 2.6.

2.4 Use Case of Faults and Time-Dependent Variability

- **SET**
  The SET is an after-effect of a radioactive event that generates a transient electrical pulse at the combinational cell through a cumulative accumulation of charges at the critical points.

- **SEU**
  SEUs are interpreted as the inversion of stored electrical value in sequential cells such as the flip-flop, the latch, and the memory cells. Their vulnerability to SEUs is due to the interaction of silicon material with the radiation environment. This Ph.D. work explicitly emphasizes the use case of "SEU" only to refer an upset that occurs at the sequential cells. But some authors (e.g., Shuler in [58]) also used the SEU term to refer a latched Single Event Transient (SET) pulse to a down stream sequential cell.

- **Single Stuck-at Faults**
  The Stuck-at fault models are applying for a structural test approach. The test method with single stuck-at models does not include all combinations of 1's and 0's to a VLSI

device, while a reduced set of test vectors executes such production test. Stuck-at-fault Models operate at the logic model of digital circuits. In these fault models, an input or an output can be stuck at logic level zero (S@0) or logic level one (S@1).

- **Aging and Temperature Instability**
  Bias Temperature Instability (BTI) refers to instability that depends on time in transistors and, increasing bias and temperature accelerates such variabilities. As per the BTI test, the absolute threshold voltage of Metal Oxide Semiconductor Field Effect Transistor (MOSFET) increases, while the device is biased in inverse mode. The threshold voltage shift ΔV causes a decrease in drain current in the on-state of the transistor and speed up the reduction of CMOS circuits. Degradation due to the BTI develops during normal transistor operations as time lapses. For p-channel MOSFETS, the Negative Bias Temperature Instability (NBTI) represents BTI effect, whereas, for n-channel MOSFETs, the instability is known as Positive Temperature Instability (PBTI) since the corresponding gate bias conditions are negative and positive, respectively [59].

2.5 Fault Injection Campaign: First Principle Data Gathering

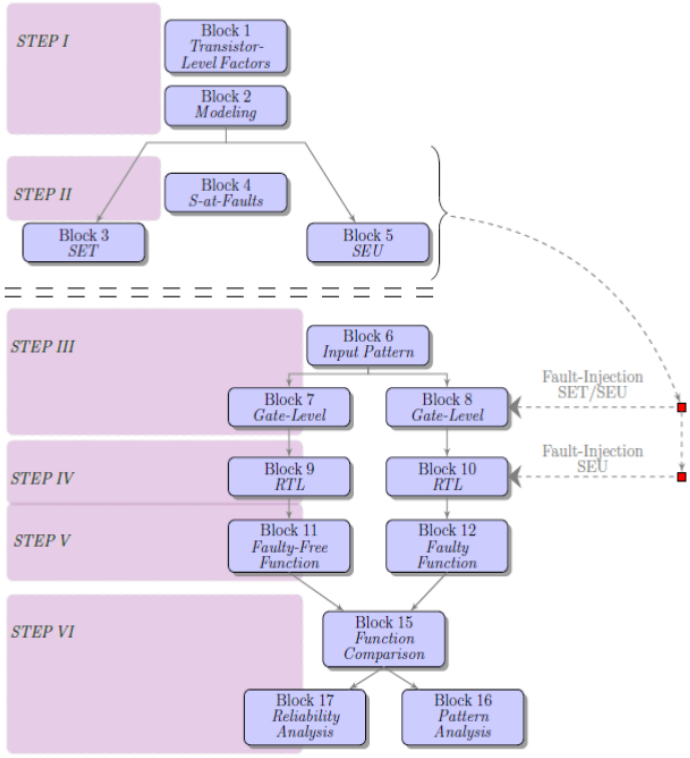


Figure 2.7 – Execution of an Fault-Injection Campaign

Researchers working on fault evaluation techniques in circuits and systems routinely use test cases to prove/evaluate the quality of their fault simulation, evaluation, or acceleration techniques. The validation of the self-developed techniques and corresponding matrices (e.g., fault-propagation probability or circuit reliability) are compared and re-

evaluated through the first-principles methods. The first principle elucidates a standard way to gather the data regarding the standard benchmark circuits by using the commercial simulation tools (Questa/Affirma/VCS) and open-source tool (Varilattor/Icarus Verilog).

Fig. 2.7 is picturing very fundamental steps in the workflow of the fault injection campaign. In Step-1, the principal radiation-induced faults (SEU and SET) at the gate-level follows a digital pulse model. Such a pulse model possesses some characteristics that depend on the transistor level parameters like critical charge Q_{crit}. Sometimes, the environment characterizes the single event effect. For example, one can consider that: a typical neutron environment caused SET pulse width is 50ps, alpha particles caused transient pulse width is 10ps (or) Heavy Ions caused pulse width is 100ps. While SET fault follows a digital pulse of width (w) at the gate component, the SEU follows a model of keeping an erroneous value at the flip-flop from the point of the occurrence until the next clock cycle arrives at the flip-flop. This how step-2 is evolved from step-1 as provided in Fig. 2.7. The third type of fault in the thesis works is the single stuck-at faults, where keeping a value of net in the gate-level circuit as '1' or '0' for the complete cycle of the processing time of the circuit to produce an output. Single stuck-at faults are categorized as manufacturing defects (or) faults that depend on an aging phenomenon in a circuit. In a so-called Fault Injection (FI) campaign, a fault that belongs to a set of SET/SEU/stuck-at models will inject into a circuit component at the given time (t). If it is a gate-level abstraction, then three types of faults are exercised. If it is RTL abstraction, then only SEU fault is exercised. It is shown in Block-8 and Block-10 respectively in Fig. 2.7. Block-7 and Block-9 are responsible for the fault-free simulations. Finally, faulty circuit function and fault-free function are compared and these processes are visualized in blocks 11, 12, and 15.

These comparisons between faulty simulation results and golden simulation (fault-free simulation) results statistically analyze and characterize the reliability of the circuit (or) the Functional Fault Coverage (FFC) of the input patterns. The requirement for building a high-quality design validation environment that injects faults in circuits and evaluates the reliability and quality through standardized first principles (i.e., fault simulation) depends on the following facts:

- The efficient database requirement for researchers
- Quality circuit simulations
- Requirement of standardizing the metrics
- Incorporating the reliability metrics and environmental parameters with the dependencies of applications in simulation

2.6 State of the Art: A Review of the Previous Static Analysis

Probabilistic Transfer Matrices (PTM) [60] is a gate-level approach that accurately assesses the reliability of a combinational circuit. A Conditional Probability Table (CPT) encodes the probability of an output combination to happen for a given input vector, which is the fundamental block in the proposed calculation. Reliability calculation based on PTM involves matrices multiplications. The main drawbacks of PTM are the excessive amount of memory and the required computational complexity in matrix manipulation. For large or medium scaled circuits, the computational complexity grows exponentially with the number of inputs and outputs and turns PTM impractical for applied industrial circuits. Some methods have been proposed to decrease time consumption and memory usage. Some authors proposed an approach to compress the memory space usage based on Algebraic

Decision Diagrams (ADD) [61]. But it is effective only for square matrices, then dealing with non-square matrices requires zero paddings.

Another improvement method is effectively reducing memory usage and time consumption by eliminating some useless but expensive inter-data. Compared with PTM, Signal Probability Reliability Analysis (SPRA)[62] is not based on the conditional probability of an output vector but a perticular signal probability at the output. That probabailty is defined as the probability of the value of the output signal equals to 1. For a fault-prone logic signal, four states exist: signal=correct 0, signal=correct1, signal=incorrect 0 and signal=incorrect 1. SPR calculates the reliability by computing the signal probability from the input to the output. In this process, the accumulative probability of correct 0 and correct 1 is the signal reliability. The advantage of SPR is that its complexity is linear, which can reduce the time consumption and memory usage dramatically with comparison to PTM. The main drawback of this model is the signal correlations, which invalidates the straightforward computation of joint probabilities.Thus, it just attains an approximate result when deals with signal correlations or fanout re-convergence issues in the circuits. Such inappropriate fluctuations in the results are solved by introducing heuristic algorithms (e.g., the Dynamic Weighted AveragingAlgorithm (DWAA) and the multi-pass approach [63]).

Multi-Pass SPR (SPR-MP) [63] is an effective method to solve the signal correlations difficulties by calculating the partial signal reliability at cells and accumulating them to attain circuit reliability. An accurate estimate of the circuit reliability is claimed with the SPR-MP algorithm in [63]. The main drawback of this method is that the time consumption increases exponentially with the number of fanouts. Another analytical method to overcome the impact of signal correlations is the Conditioned Probability Matrix (CPM) [64]. It decorrelates the correlated signals with conditional probabilities and accelerates the estimation process with the direct SPR approach. Its complexity depends not only on the size of the circuit but also on the re-convergence sources, which are defined as the point at which signals correlate. Thus, CPM can also deal with the scalability problem with combinational circuits.

Han et al. presented a Probabilistic Gate Model (PGM) [65]. This model relates the probability of the output node to that of input signals and the error probability of the logic gate. Here, two distinct computational algorithms are proposed with different merits. The approximate algorithm obtains an approximate evaluation of the circuit reliability without considering the signal correlations, and vulnerability to complexity increases linearly with the number of gates. While the accurate algorithm that taking account the signal correlations attains an accurate reliability evaluation methodologies. But the improved techniques of reliability assessment of a circuit has exponential complexity in the worst case.

Sellers et al. [66] introduced the concept of analyzing errors with boolean differences which including the concept of how to use the Boolean Difference (BD) in error-detection and error-correction in logic circuits design. Mohyuddin et al. [67] extended its application to reliability analysis. The authors presented a gate-level probabilistic error propagation model Boolean Difference-based Error Calculator (BDEC) which takes the boolean function of the gate, the signal error probabilities of the gate inputs, and the gate error probability as its input parameters and calculates the error probabilities of the outputs. It can achieve reliability estimation with some accuracy and scalability with linear complexity increment with the number of gates of the circuits.

A probabilistic-based methodology for nanoscale architecture design was proposed by Bahar et al. [68]. The authors discussed a novel nanoscale architecture based on Markov Random Fields (MRF). Later, Lu et al. [69] extended the concept of MRF-based design ar-

chitecture and proposed a probabilistic logic to replace the Boolean logic for nanoscale devices. According to MRF, the conditional probability can be expressed in terms of a function contributed by its neighborhood. With statistical physics, the Probability Density Function (PDF) of a node can be expressed by the energy level contributed by its neighboring nodes. Thus, the probability of the output nodes can be achieved with the integration theory. Rejimon et al. proposed a probabilistic methodology based on Bayesian Network (BN) [70]. The authors proposed a probabilistic error model based on BN and estimated the overall error probability of the output by comparing the fault-prone output to the ideal output. It was proved to be a compact and minimal framework to estimate the reliability of circuits.

But, these algorithms are not suitable for the medium and large-scale circuit-netlists that authors of this paper investigated as the test devices. The main reasons are the exponential rise of mathematical computational complexity (time and memory requirements) and re-convergence issues. Most primarily, for the case of the large circuit, a significant numerical variation in reliability analysis is observed while considering the different signal probabilities due to workload stress and the soft-error susceptibility rates. The trade-off between the time overhead (by the comprehensive fault-injection methods) and the inadequacy in metrics estimation (by synthetic models) motivates the development of high-performance tools with acceptable metrics-estimation error quantile. Ran Xiao and Chunhong Chen briefed in their article [71] a comprehensive survey on the mathematical algorithms for reliability at gate-level abstraction.

Alessandro Vallero in his research works [72] [73], uses a Bayesian network as a probabilistic model that consists of a Directed Acyclic Graph (DAG). This model has been chosen for estimating the cross-layer reliability of a system where nodes are analogous to the hardware architecture level modules (e.g., CPU, register file, and IP cores) and the application software level modules. To make the probabilistic inference on the reliability of each module (or) component in the system, the authors in [72] and [73], have chosen the Noisy-MAX algorithm. However, such methods need a Conditional Probability Table (CPT) as a quantitative model for each module for different input-output combinations to accelerate the marginalization of variables in the process of inferring error propagation probability. Different methods to find the probabilistic inference with Bayesian Networks are explained in [74], [75],[76], [77] and [78], that includes junction tree, sum-product algorithm, message passing algorithm, and beam search algorithm.

2.7 AI/ML for System Reliability: A Recent Literature Survey

The work in [79] supports the statement that with the advent of machine learning techniques, the ability to learn from past behavior to predict future behavior makes it possible to predict an individual component's time until failure much more accurately. Here, the authors explore the predictive abilities of a machine learning technique to improve the ability to predict individual component times until failure in advance of actual failure. Fred Lin from Facebook Inc. in his work [80] presents a machine learning framework that predicts the required remediations for undiagnosed failures, based on similar repair tickets closed in the past. The authors explain the methodology in detail for setting up a machine learning model, deploying it in a production environment, and monitoring its performance with the necessary metrics because, in production, the autonomous system diagnoses hardware failures based on the rules that the subject matter experts put in the system. A common practice for estimating the reliability of electrical systems is to use statistics and probability methods that provide quantitative data with reliability indices from experimentation testing and simulations. George Thiel from Microsoft Corporation in his work [81], proposes a temporal Convolution Neural Network-based model that is

insensitive to the noise in the time dimension and designs a loss function to train the model with extremely imbalanced samples effectively. Nikolaos Georgoulopoulos et al in [82] describe a novel approach for predicting failure rates for components and hardware systems. A physics-of-failure-based methodology is provided to predict the degradation rate of a population using a Monte Carlo approach. The work in [83] provides a survey. In this survey, they mention that machine learning (ML) and deep learning (DL) methods can assist in effectively predicting hardware errors at a sufficient amount of time before they occur. The survey is presented on hardware failure prediction techniques for servers using ML and DL methods, with a focus on HDD, RAM, and CPU issues. These techniques are categorized based on the ML or DL algorithm they use for the prediction process. Hardware failures in cloud data centers may cause substantial losses to cloud providers and cloud users. Therefore, the ability to accurately predict when failures occur is of paramount importance. In the work, [84], the authors present FailureSim, a simulator based on CloudSim that supports failure prediction. The FailureSim obtains performance-related information from the cloud and classifies the status of the hardware using a neural network. The authors from [85], investigated and compared one of the Deep Learning Architecture called Deep Neural Network (DNN) with the classical Random Forest (RF) machine learning algorithm for the malware classification. The authors have studied the performance of the classical RF and DNN with 2, 4 & 7 layers architectures with the four different feature sets, and found that irrespective of the features inputs, the classical RF accuracy outperforms the DNN. In [86] proposes a hybrid prognostic scheme with the capability of uncertainty assessment is proposed in this paper, which combines particle filter (PF) and Relevance Vector Machine (RVM). The Author from the work [87] develops a machine learning model that has been further developed based on the Finite Element Analysis (FEA) simulation results to make reliability predictions for chip package designs with improved computational efficiency.

2.8 Static Analysis and Derating Factors

The thesis work defines a probabilistic static analysis tool based on the methods in SOTA and processed some simulations. The mathematical abstractions in equations (2.3) (2.4) (2.5) and (2.6), elaborates accelerated Soft-Error Rate (SER) estimation in an system. Comparison of the proposed algorithm in the thesis with a probabilistic analysis tool from state-of-the-art is necessary to brief the quality of the proposed method. At the gate-level analysis, individual factors like EDR, TDR, LDR, and FDR contribute to the overall SER. The importance of such a static tool for reliability estimation is to provide early-stage finalization for system designers as redemption from analysis jeopardy. The different derating factors are summarized below.

Logical Derating (LDR)

The Logical Derating (LDR) refers to the probabilistic metric that represents the probability to mask an SET/SEU fault while propagating through the circuit. Depending on the SOTA of static analysis methods in this thesis, a simple way for calculating the Signal Probability Propagation (SPP) of each component is exploiting the truth table for the input and output combinations. For example, the SPP in the case of a AND gate is represented in Fig. 2.8, where input signals 'A' and 'B' have the signal probability 0.5 for having high-signal-state '1'. The output signal (c) has a probability of 0.25 for possessing a high-signal-stats '1'. These models depend on different rules for different gates and they are scientifically documented by the articles [65], [88] and [63]. To avoid repeating the texts and old equations (that are not significant for this thesis), the readers are advised to go through the articles. The LDR calculation in this thesis for SOTA based static analysis method used these SPP

models and the calculated probabilities are compared with reference fault injection simulation results as well as with results by the developed AI-Framework in this thesis.

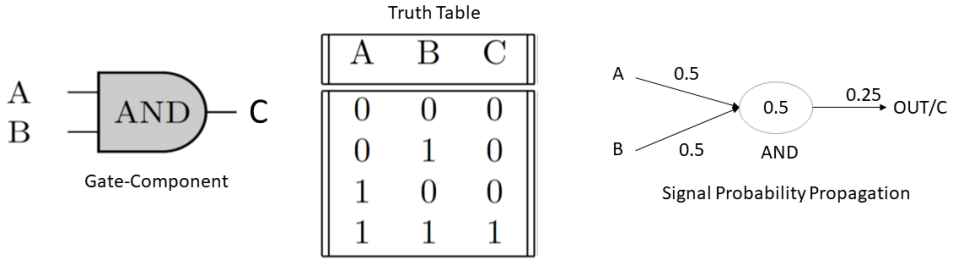


Figure 2.8 – The Fault Signal (SEU/SET) Propagation Model in a AND Gate

Temporal Derating (TDR)
In the probabilistic static analysis tool, the mathematical temporal derating is estimating as the fractional ratio of latching window (or) so-called opportunity window [89] which is approximately indicated by setup (t_{setup}) and hold (t_{hold}) time of the flip-flop. The SET pulse width is defined by the δ. The equations (2.1) and (2.2) are briefing the temporal derating calculation for SEU and SET fault respectively. The detailed scenario of the calculation completely relies on the profound ideas in the works [89] [90] and [91].

$$TDR_{SEU} = 1 - \frac{\text{Delay} + \frac{t_{setup}}{2} - \frac{t_{hold}}{2}}{T_{clk}} \tag{2.1}$$

$$TDR_{SET} = \frac{\delta}{T_{clk}} \tag{2.2}$$

In (2.1), the "Delay" simply represents the propagation delay entitled to the arrival of required data at the input of flip-flop. In short, the time represents the slack (t_{slack}) = T_{clk} - Delay.

Electrical Derating (EDR)
The case study of electrical derating is applicable for SET propagation while considering the fact that SEU fault duration is persist enough to propagate. In the case of electrical derating, a digital logical pulse model is derived to represent the analog transient pulse as defined in the work cited by [89], where the various pulse widths in pico-seconds (ps) are reasonably interesting in the gate-level.

Functional Derating (FDR)
The functional derating is not completely defined in the premises of mathematical calculation. The reason behind that, is the adaptive variation of circuit behavior relative to the application and the environmental dependencies. Modeling of such collective behaviors depends on the complete functional simulation campaign rather than a general mathematical model. Therefore, functional derating in (2.3) and (2.5) are deriving through the high-level fault injection simulation campaign, especially at RTL [92][93] (using Modelsim and failures at block-level (or) like Data Vulnerability Factor (DVF) type failures) or even at architectural level (Gem-5 [94][95] simulation tool).

2.9 Mathematical Modeling of System Soft-Error Reliability

Single Event Effects (SEEs) are the challenging phenomena to predict when the critical parts of the circuit interacted with the radiation particles. The Single Event Upset (SEU) and Single Event Transient (SET) are widely used here as the prominent representatives of SEEs. The use-case for SEU fault mainly implies an inversion of the stored value in a flip-flop during a clock period, while SET represents a transient pulse of arbitrary width in the output-net of a gate. It will propagate through the combinatorial network and latched to the downstream sequential element. Block diagram in Fig. 2.7 shows a structure of the fault-injection campaign that infers the statistical functional failure metrics of SEU/SET events. The mathematical model for Functional Failure Rate ($FFR_{i,seu}$) of a SEU event is described as:

$$FFR_{i,seu} = FIT_{i,seu} \cdot \prod_{j \in T,L,F} DR_{ij} \tag{2.3}$$

$$FFR_{seu} = \sum_{i \in FF} FFR_{i,seu} \tag{2.4}$$

where, DR_{iT}, DR_{iL}, and DR_{iF} represent the fault derating factors [91] such as TDR, LDR and FDR respectively. Similarly, $FIT_{i,seu}$ denotes the rate of soft errors at the flip-flop (i) in Failure-In-Time (FIT) unit. Because of the interdisciplinary nature of functional failure modeling of SEE events, the Functional Failure Rate due to SET ($FFR_{g,set}$) can formulate as:

$$FFR_{g,set} = \int_{w_{min}}^{w_{max}} FIT_g(w) \cdot \prod_{j \in E,T,L,F} DR_{gj}(w)\, dw \tag{2.5}$$

$$FFR_{set} = \sum_{g \in Gate} FFR_{g,set} \tag{2.6}$$

where, $DR_{gE}(w)$ is termed as EDR factor [91]. Both EDR ($DR_{gE}(w)$) and TDR ($DR_{gT}(w)$) in (2.5) depend on SET pulse-width (w). Other factors such as DR_{gL} and DR_{gF} are corresponding to LDR and FDR respectively as explained in (2.3). FIT_g denotes the rate of soft errors at the gate (g) in FIT unit. Readers could refer to the papers [44] and [22] for the deep insights about the radiation-induced soft-errors and their inevitable intrusive nature in the functioning of microelectronic devices in aggressive radiation environments.

2.10 Regression Evaluation Metrics

- **Mean Squared Error**
  If $\hat{y}_i$ is the predicted value and y_i is the true value corresponding to the i^{th} sample, then the Mean Squared Error (MSE) to be estimated over n samples defined as,

  $$MSE(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2 \tag{2.7}$$

  The regression error will become minima as MSE approaches to zero.

- **R-Squared Score** It is also known as the coefficient of determination. If $\hat{y}_i$ is the predicted value of the i^{th} sample, and y_i is the corresponding true value, then the coefficient of determination estimated over n samples defined as,

  $$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2}{\sum_{i=0}^{n-1} (y_i - \bar{y}_i)^2} \tag{2.8}$$

where: $\bar{y}_i = \frac{1}{n}\sum_{i=0}^{n-1} y_i$. Numerical value 1 indicates a good regression fit, while 0 indicates a worse fit.

- **Explained Variance Score** If $\hat{y}$ is the predicted value of the target value y, then Explained Variance Score (EVS) estimated over n samples is defined as,

$$EVS(y,\hat{y}) = 1 - \frac{Var\{y-\hat{y}\}}{Var\{y\}} \tag{2.9}$$

where: Var is the square of the standard deviation. The best possible score is 1 and lower values are worse.

- **95% Confidence Interval**
In this thesis, a metric called confidence interval (CI) for the population-mean (μ) describes a comparison metric. Here population implies sample space. The population-mean, μ is estimated by using a sample mean $\bar{X}$. When the population standard-deviation σ is given, the formula for a confidence interval (CI) for a population mean is:

$$\bar{X} \pm Z.\sigma/\sqrt{n} \tag{2.10}$$

where: $\bar{X}$ is the sample mean, σ is the population standard-deviation, n is the sample size, and Z represents the appropriate Z-value from the standard distribution characteristics for your desired confidence level [96]. Here Z-value is chosen 1.96 for 95% confidence interval.

2.11 Data Mining and Clustering Techniques

Charu C. Aggarwal [97] states that data mining is the study of collecting, cleaning, processing, analyzing, and gaining useful insights from data. In this thesis, we present a naive data representation for a circuit in reliability estimation and modeling. This intermediate representation of the circuit is called a circuit-graph. Therefore, data mining is used as a technique that collects and processes different aspects of data corresponding to the circuit. We exploit different aspects of data mining algorithms for finding similar components by virtue of their structural peculiarities (e.g., number of inputs, number of outputs, and number of neighboring components, etc.). Here two algorithms named k-means and hierarchical clustering performs finding similar components. For detailed reading, please refer the book [97].

K-means Clustering
K-means algorithm is an iterative algorithm that tries to split the dataset into k distinct non-overlapping clusters where each data point belongs to only one group. The partitioned subgroups have possible low intra-cluster data points and possibly high inter-cluster distances. Here the quantifying objective function assigns data points to a cluster by minimizing the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points within that cluster). The applications of k-means algorithms are significantly increased in different fields [98] [99] [100].

Hierarchical Clustering
Hierarchical clustering treats each observation as a separate cluster in the initial step. Then it iteratively executes the following two steps: (1) identify the data points that are closest together, and (2) merge them into most similar distinct clusters. This iterative

process continues until all the clusters are optimized. Hierarchical algorithms typically cluster the data with distances functions such as Euclidean, Manhattan, Canberra, and Minkowski. However, the use of distance functions is not a mandatory solution. Distinct hierarchical algorithms use different clustering methods, such as density- or graph-based methods, as a subroutine for constructing the hierarchy [97]. Hierarchical algorithms generate a taxonomy of data depending on their similarity metric. The applications of hierarchical clustering are also proliferating in diverse Fields [101] [102] [103].

2.12 Artificial Intellegnce

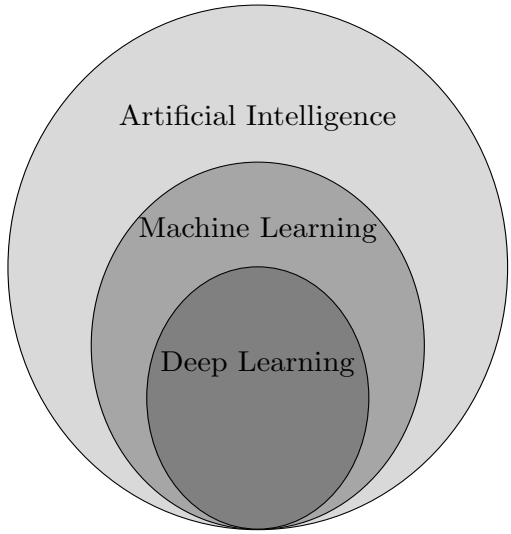


Figure 2.9 – AI Overview (After Ref. [104] [2] [105] [106])

According to John McCarthy ([107] one of the founders of the discipline of artificial intelligence) AI is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable. The picture in Fig. 2.9 (inspired by [104] [2] and [105] [106]) gives a clear idea of the relationship between Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) to the reader. Deep Neural Network (DNN) or Artificial Neural Network (ANN) are considered the category of DL. Deep learning is a subset of machine learning and is derived from the concept of artificial intelligence. In machine learning, data is parsed for the learning phase and then followed by learning a decision rule, while in the case of deep learning algorithms, artificial neurons appear in layers to create the ANN that can learn and make intelligent decisions on its own. Artificial intelligence is a global idea of ML and DL and enabling the machine (e.g. a computer) to attain a given task based on a stipulated set of rules called an algorithm.

As mentioned before, the deep neural networks can make intelligent decisions on its own, the work in this theis mainly based on a neural network, called GCN. Intelligent networks like GCNs is actually different from traditional neural network algorithms and slightly varied from traditional Convolutional Neural Networks (CNN)s. A normal neural network consists of staked hidden layers, where each of the neurons (or) nodes from the current layer receives input from all the nodes from the previous layer, commonly known as dense layers. Then performs a dot product between the data at the input of the neu-

ron and the weights of the neuron. After that, the result is passed through an activation function. These determined values are passing to the successive layers by concatenating the input, hidden, and output layers together. CNN is different from the traditional way of constructing the dense layered neural network. In CNN, the initial input features are convolved with kernel input filters and then down-sampled through a pooling layer. Finally, connected to a fully connected neural network.

2.12.1 Machine Learning and Support Vector Machine

The support vector machine works on the foundation of a good theoritical learning algorithm to solve regression analysis as well as classification type problems. The SVM for regression analysis can be called as SVR in short. It was invented by Vladimir Vapnik and his co-workers, and first introduced at the Computational Learning Theory (COLT) 1992 conference with the paper [108]. SVM characterizes the maximal margin algorithm for supervised learning models. In the maximal margin principle, SVR tries to find the optimal hyperplane which maximizes the margin and minimizes the error. Compared to classification problems, regression analysis outputs a continuous variable. The SVR approach based on a standard epsilon-insensitive hinge loss function defines a margin of tolerance ε where no penalty is given to errors. At the same time, it punishes the wrong estimation with a cost-insensitive symmetric hinge loss function.

An alternate approch in SVR applications is the kernel modification. A kernel which possible to transform the given data set to higher dimensional space to derive a liner decision boundary. A properly chosen Radial Basis Functions (RBF) had employed as a kernel function in this work. RBF is also called the Gaussian Kernel which means that each feature vector of the dataset in the transformed dimensional space influenced by the Gaussian observation.

2.12.2 Deep Learning and Feed Forward Network

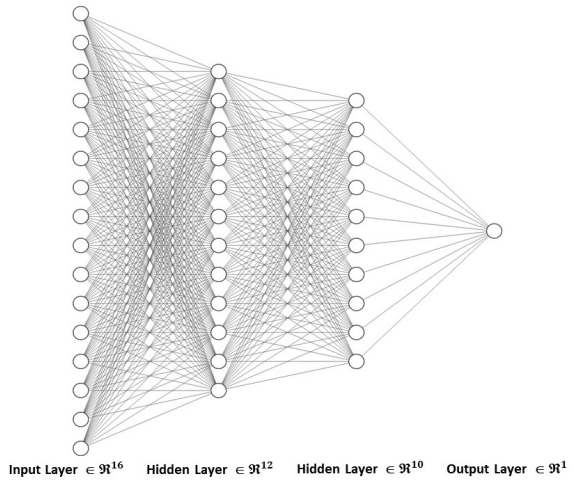


Figure 2.10 – Deep Neural Network

Deep Neural Network is an important step in the machine learning algorithms. Their learning methods are trying to model data with complex architectures and distributions by combining different non-linear transformations. In this work, a general fully connected

DNN is implemented. The other main categories of deep learning methods are Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). The elementary bricks of deep learning are the artificial neurons (perceptrons) which are inspired by biological neurons. An artificial neuron combines the input signals with adaptive weights and uses an activation function to deliver the output to be estimated. An in-depth discussion about the architecture as well as the adopted parameters has given in later sections. A Feed Forward Neural Network is a category of ANNs, and it is called DNN. The connections between nodes are in a feed-forward way and do not form a cycle. The opposite of a feed-forward neural network is a RNN, in which a certain connection of nodes has feedback from past states of the nodes. The feed-forward model or DNN is the simplest form of the neural network as information is processed and inferred in one direction. The data that passes through multiple hidden layers, is traversing in the forward direction and never in backward. The fundamental unit of Feed Forward Neural Network is a single layer perceptron, and the typical DNN structure is shown in 2.10.

2.12.3 Graph Convolutional Neural Network (GCN)

Figure 2.11 provide a architectural view of GCN. The work made a GCN model of two hidden layers as given in Figure 2.11. The first layer in this work contains 4 hidden nodes and the second layer contains 2 hidden nodes. These two hidden layers stacked between the input layer and the output layer. The input layer contains a number of nodes which equivalent to the gate-level netlist elements of the circuits. It varies from circuit to circuit. The model can able to model even for a large number of elements of the circuit by this time. But it is difficult to say a limit now. Both hidden layer's nodes activated by the non-linear function called a hyperbolic tangential function (Tanh). During the training phase, the model is updating at each step and optimized by an adaptive learning rate optimization algorithm called 'Adam' [109]. The dimension of the hidden layers can be chosen by arbitrarily and it depends on the parsed adjacency matrix.

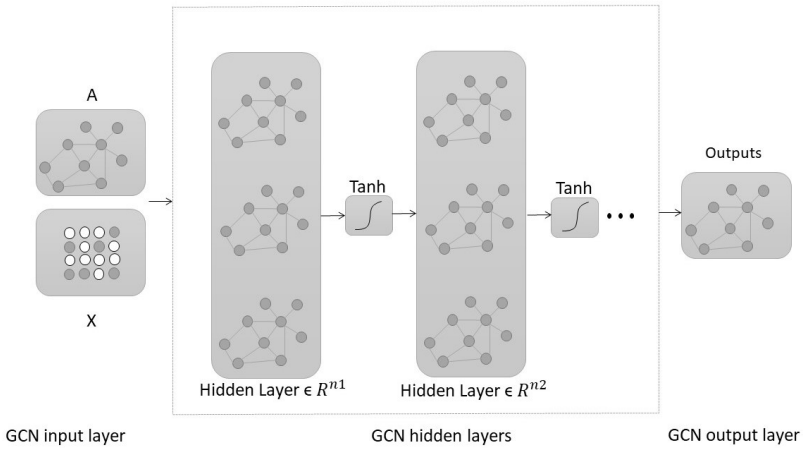


Figure 2.11 – GCN Model [110]

2.12.4 Long-Short Term Memory (LSTM)

Long Short Term Memory (LSTM) networks are an enhanced case of Recurrent Neural Network (RNN) that is capable of learning sequential order dependence in prediction problems. LSTM has a neural network architecture to model temporal sequences, memorize very recent past data and predict long-term dependencies with trained time-series data [111]. An LSTM cell module has three interacting components: Input Gate, Forget Gate, and Output Gate. glslstm can solve many unsolvable time-series tasks by feed-forward networks using fixed-size time windows.

2.13 Applied Electronic Design Automation Tools in Thesis

The basis of this doctoral thesis is to research and develop a software framework for the analysis of reliability due to soft error. The consolidation of developed models and their validation depends on the results of real-time fault-injection simulations. The well-apprehended device simulation in this thesis chose Verilog language and simulation tools like Modelsim, Synopsis VCS, and an open-source tool like Icarus Verilog. Automating the fault-injection campaign that injects numerous faults and corresponding analysis, a framework is developed in the scripting languages called Tcl/Tk. Tcl is a high-level, general-purpose, interpreted, dynamic programming language. Integrated fault-injection functions of the SoCFIT tool linked to the ModelSim simulation framework for fault injections. Verilog Procedural Interface (VPI) functions are explore to generate the fault-injection functions in SoCFIT tool. SoCFIT is a reliability-focused design characterization platform that predicts quickly and accurately the failure rate (FIT) and various derating factors of Application-Specific Integrated Circuit (ASIC) and System on Chip (SoC). Design Compiler (DC) of Synopsis toolset is the technology that synthesizes the gate-level circuit from RTL abstraction of all the test-circuits in the thesis case studies. The timly use of Makefiles in shell scripting made a simple way to organize code compilation.

2.14 Applied Software Languages and Tools in Thesis

A software framework for the static analysis of soft-error reliability is the fundamental aim of this thesis work. The framework is built using Artificial Intelligence AI models, tools, and open-source algorithms. The AI tools are adapted from Python language. The adapted AI libraries are Apache MXNet, Tensorflow, Keras, PyTorch, Scikit-learn, and Theano. A software framework for the static analysis of soft-error reliability is the fundamental aim of this thesis work. The framework is built using Artificial Intelligence AI models, tools, and open-source algorithms. The AI tools are adapted from Python language. The adapted AI libraries are Apache MXNet, Tensorflow, Keras, PyTorch, Scikit-learn, and Theano. The different data mining libraries and clustering algorithms are explored from R-Language. R is a programming language as well as a free software environment for statistical computing and graphics supported by the R Core Team and the R Foundation for Statistical Computing [112]. C/C++ and Java are the principal languages to build some executable functions for fault-injection toolset and a compiler for Verilog scripting programs.

Chapter 3

An AI-Framework for Soft-Error Reliability

The published scientific papers: [I], [II], [III], [IV] and [V] are the backbones of this chapter. [I], [II], and [III] contribute 90% of this chapter, where as [IV] and [V] contribute 9% and 1% respectively. The chapter refers to the paramount and commercially advantageous results of an AI based framework that is an accelerated analytical tool for the inference of functional level failures due to the radiation-induced errors in micro/nano-electronic advanced circuits. In the published article [IV], the necessity of standard reliability evaluation metrics and how its standardization enhance the quality of the system are outlined. Papers [I] and [II], pave the primary steps to solve the labyrinth problem of developing an advanced mathematical solution to the accelerated analysis of soft- errors. Finally, a comprehensive and sophisticated tool/method is developed by exploring the complex features of artificial intelligence techniques and published in [III]. All these methods supports explained artificial intelligence or white-box modeling, while an example for black-box modeling is provided in [V] that is not the prime vision of this chapter.

3.1 Preamble of Chapter

With the advent of small-scaled technologies, the vulnerability of Single Event Effect (SEE) dominates in the radiation response of complex-microelectronics designs. Single Event Effects (SEEs) such as Single Event Upsets (SEUs) and Single Event Transients (SETs) have been characterized as the principal reliability-concerned physical phenomena at gate-level abstraction. Heavy ions, protons, and neutrons induce SEEs that are affecting highly sophisticated electronic designs at their functional level. Therefore, critical functional failures due to SEEs are one of the complex aspects to characterize reliably. An exhaustive Fault-Injection (FI) is a well-apprehended way of assessing the severity of faults by exploring the complete fault-space and providing more accurate reliability metrics. However, this fault-injection strategy turns more cumbersome in terms of execution time and EDA tool licenses. This thesis targets a more advanced systematic framework based on Artificial Intelligence (AI) algorithms to accelerate reliability metrics estimation with an accepted level of numerical approximation error. The behavioral patterns of cell instances, including gates and flip-flops, are extracted/separated/derived from the gate-level in a probabilistic manner, using AI algorithms. The probabilistic patterns are an exploratory source to train and optimize a Deep Neural Network to postulate a model to predict the SER/Functional Failure Rates (FFRs) due to SEUs and SETs.

Exhaustive fault injection-based reliability assessment is infeasible on medium and large-scale circuits in terms of time and EDA licenses. Therefore, new test methodologies based on mathematical and statistical models attract the attention of reliability researchers.

The Probabilistic Transfer Matrices (PTM) is an example of such a model that performs the matrix multiplication for the reliability analysis [60]. Similarly, Algebraic Decision Diagrams (ADD) [61], Signal Probability Reliability Analysis (SPR) [113], Multi-Pass SPR [63] and Probabilistic Gate Model (PGM) [65] [114], are the dedicated algorithms for analysis of the fault signal propagation. But, these algorithms are not suitable for medium or large-scale circuits that were investigated as the test devices in this thesis. The main reasons are the exponential rise of mathematical computational complexity (time and memory requirements), re-convergence issues, and most importantly, a significant numerical variation in reliability analysis while considering the signal probabilities due to workload stress and the susceptibility rate of soft errors.

The scientific problem that motivates this work is developed from the perspective of Soft Error Rate (SER) analysis [19] in the combinational circuits. There are several factors to be considered in determining the SER of a logic circuit. Such factors include electrical, logical, temporal, and functional derating factors and are premised on:

1. The soft-error vulnerability rate of a combinational cell in Failure-in-Time (FIT) units
2. The location of error generation at the given logical networks
3. The probability of error attenuation on a path of propagation
4. The signal probability at the input of sequential and combinational cells

To facilitate a more realistic reliability analysis of a system design, more complex models that encapsulate different derating factors are essential at the logical abstraction level. Depending on the flexibility of those compact models that address difficulties for the logic level SER analysis, more comprehensive failure analysis is possible before the final stage of the system design. This idea is the core principle of the development of the implemented approach in this chapter. The developed method is incorporated into an EDA tool which embeds the models to facilitate more accurate reliability analysis. The exhaustive fault injection method is the ultimate reliability assessment method in terms of accuracy but its inconveniences in terms of time and EDA licenses make this approach infeasible on the size of circuits from the scalability perspective. So a new testing methodology (or) tool is the ultimate aim of this thesis.

3.2 Reliability and Functional Safety Requirements

3.2.1 Understanding Reliability Standards in Autonomous System

Autonomous systems will enable huge societal changes (and possibly progress). As expected, stringent safety and reliability expectations and requirements are firmly set in international standards, implicit customer expectations and, not unexpectedly, insurance policies. Autonomous systems are also an emerging industrial field and are very likely to stay with us for a very long time. Accordingly, it is very probable that many successive, evolutionary or revolutionary standards will be issued to govern them. International standards are the clearest and most authoritative prescribers regarding reliability and safety. The list of current or under-development standards in this field includes:

- EC 61508 [115] (Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems) is aimed at all industrial fields and is the template for many application-specific standards;

- One of the most well-known derivatives of the previous is ISO 26262 [116], which addresses the functional safety of automotive systems;

- IEC 62279 is an adaptation of [115] for railway applications;

- ISO 13849 [117] is a safety standard which applies to parts of machinery control systems that are assigned to provide safety functions;

- AC 25.1309-1A [118] (System Design and Analysis) provides background for important concepts and issues within airplane system design and analysis;

- RTCA/DO-254 [119] (Design Assurance Guidance for Airborne Electronic Hardware) provides guidance for the development of airborne electronic hardware.

- Since change is a permanent feature of the industrial progress, expectations and requirements constantly evolve. While intended to be robust and durable, standards are not safe from being prone to latest fashions and currents in the industry or from being influenced by companies and organizations looking to promote their own position and offering.

Particularly, the terminology and dictionary of any standard is a faithful snapshot of the particular context at the time of the writing and often suffers from updates, changes of signification, meaning overcharges and obsoleteness during the expected lifetime of a standard and even more so when a new standard is devised. The goal of this Section is to pinpoint some basic topics that are common to the different standards and faced by most of them. They are summarized in Table 3.1. Many standards include a part related to Terminology. In this category, the signification and a clear definition of the key terms shall be presented and elaborated. However, the specific meaning can hide behind an ordinary word, requiring a more in-depth discussion and explanation and investing the simple term with a fundamental weight. The "Terminology" category would thus benefit from a Concepts sub-category. As soon as the key terms and concepts have been introduced, the standards are fast to move to the explanation of their core methodology, framework and principles. The Methodology category covers these aspects. In their various proposed methodologies, many of the standards address "risks" to the safety of the intended applications and set a mix of quantitative and qualitative requirements and expectations for these risks. These objectives and goals will be captured in the Requirements category. The reliability and safety of any application will have to be checked against the applicable requirements and improved until its behavior fulfils the expectations of the intended standard. Accordingly, the taxonomy will have to include the Assessment and Management categories. Lastly, any application is designed to work safely and reliably in a given setting. The Environment category would capture the entirety of electrical, thermal, mechanical, radiative conditions to which the application will be subjected. In practice, the concerns about reliability may mix together with those about feasibility (especially when target features are particularly challenging). For this reason, independently on standards and regulations, general safety praxis can be utilized e.g. by using the ALARP method ("as low as reasonably practicable") and providing justification for benefits of the society against the involved risks.

Table 3.1 – MAIN TOPICS IN THE RELIABILITY AND FUNCTIONAL SAFETY STANDARDS [4]

Terminology	Methodology	Requirements	Assessment	Management	Environment
Vocabulary	Development	Hazards and Risks	Models	Online	Electrical
Concepts	System-level	Classification	Probabilistic	Offline	Thermal
	Hardware-level	Event Rates	Simulation	Diagnostic	Mechanical
	Software-level	Mitigation		Maintenance	Radiation

3.2.2 Key Concepts in Reliability and Functional Safety for Autonomous System

For autonomous systems, but not only, the notions of "reliability" and "safety" comprise as many significations as engineers from different industries want to invest in them. Loosely, reliability represents the probability of a system to fail, i.e. higher reliability means less failures, while safety generally means that the system fails in a safe way. A reliable system can be unsafe while a safe system can be unreliable. Furthermore, systems can be made arbitrarily safe and reliable with a corresponding investment of resources and time. Requirements for reliability and safety can be quantitatively and qualitatively very different but standards are often aggressive in setting high requirements for both safety and reliability. The most straightforward approach to address both reliability and safety is to rank risks and hazards according to their impact (safety) and to expect that the probability of risks (reliability) decreases inversely to their impact. An aggregated event rate (often measured in terms of Failure in Time, or FIT), may be associated to the system and/or component according to their role but with an underlying understanding of the risks that make up the "Failure" key term.

In this way, quantity and quality, safety and reliability are harmoniously integrated. However, reliability engineers will find that this task is relatively difficult as two opposing concepts still need to be conciliated: objective versus subjective. The qualificative of "Objective" can be applied to any physical measurements. As an example, technology fault rates can be expressed accurately; a "Soft Error Rate" is an objective measurement of the susceptibility of a technological process under radiations. Faults propagate through the circuit and system and can become Failures. Various methods, such as static and dynamic ones, can accurately and undisputedly (thus objectively) predict the fact that a fault occurring in a deeply-embedded logic cell instance can propagate and affect a primary system output. The question that the reliability engineers and their design colleagues must answer now is whether this fault consequence represents a failure or not, what are the actual consequences and, more importantly, where exactly in terms of risk levels the failure needs to be classified. This is the "Subjective" part and standards try to address this by a prescriptive, function-based assessment. However, in practice, the whole procedure provides some freedom and margins to reliability engineers that can argue for a less critical classification of possible fault outcomes.

Probabilistic risk evaluation and management is a core concept of many reliability assessments. Only a fraction of technological faults will propagate through the circuit and become errors, i.e., erroneous data or values stored instead of correct information. Only a percentage of errors will become failures causing observable deviations of the system behaviour. Furthermore, failures can be classified in criticality classes. If error detection/-correction/management features are implemented, they can address faults, errors and failures at any design level and can reduce the percentage of events graduating from one level to the upper one (see Fig. 3.1).

A first, fundamental contributor to the quality of an autonomous system is the quality

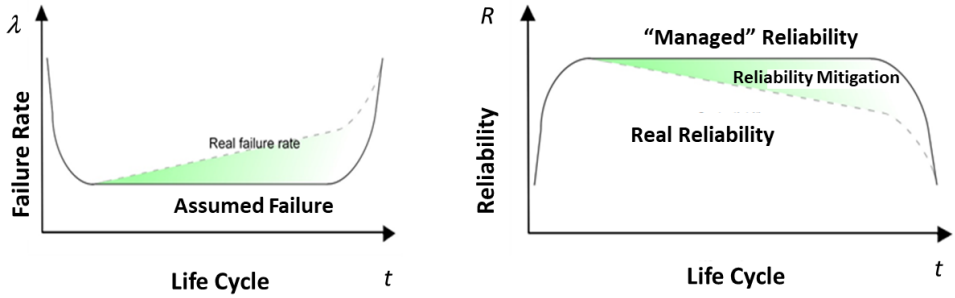


Figure 3.1 – Faults, Errors and Failures in a System [4]

of the underlying implementation technology. The manufacturing process must present a well-characterized, preferably low intrinsic defect and fault rate, resiliency to environmental challenges and a good, well known aging and degradation performance. Moreover, the technology providers (foundries) must offer their customers a full ecosystem with the tools, IPs and solutions for reliable and safe circuit design. A second contributor lies in integrating into the system some solutions for lifetime performance assurance. The classical bathtub curve is no longer an evidence and the reliability of the system must be managed during the expected lifetime through online and offline monitoring, embedded sensors, test instruments and safety mechanisms (see Fig. 3.2).

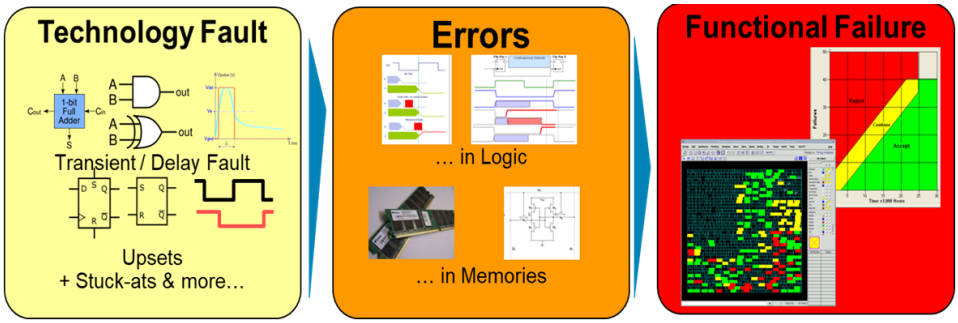


Figure 3.2 – Managed Lifetime Reliability (Courtesy of the RESIST Project) [4]

Configurability and Adaptability, to environment and workload challenges, as well as to intrinsic degradation and aging, are important for today's autonomous systems running dynamic applications in diverse environments. Lastly, the evolution to "Self-" Everything (self-monitoring, self-calibration, self-adaptation, self-configuration, etc.) is an important industry trend and goal that can provide solutions for more reliable and safer autonomous systems.

3.3 Circuit-Graph: Representation and Visualization

The standardization of Verilog-Hardware Description Language (HDL) in 1995 as IEEE Std 1364-1995 provides a simple, intuitive, and effective system design in a standard textual format at multiple levels of abstraction for a variety of design tools, including verification simulation, timing analysis, test analysis, and synthesis. The Verilog language is extensible via the Programming Language Interface (PLI) and the Verilog procedural Interface (VPI) routines [120]. VPI is the dual representation of the PLI 2.0 version that replaces

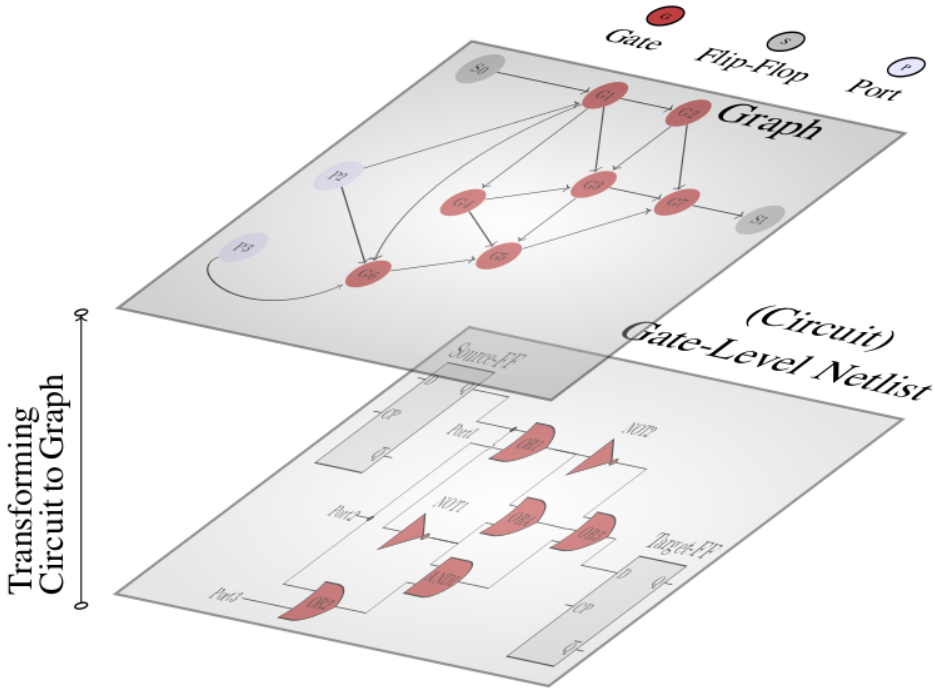


Figure 3.3 – Gate-Level Netlist to Graph

the deprecated PLI version 1. VPI is the third generation interface to the Verilog-HDL. VPI consists of a set of access and utility routines. A standard C/C++ programming language function invokes these routines to establish consistent and object-oriented access to facilitates dynamic interaction with the complete information from the design's Verilog-HDL description. VPI applications consist of connecting the Verilog-HDL simulator with computer-assisted design (CAD) systems, customized debugging tasks, delay calculators, and annotators.

The very significant primary achievement of the thesis is the development of a transformation tool that transforms the gate-level netlist to graph structure with nodes and edges that represents gate-level components and connections (nets) between the gate-level entities, respectively. Fig. 3.3 delineates a primary example, where the lower layer is the gate-level circuit to be transformed, and the upper layer shows the transformed graph. Henceforward, such transformed graph is referred to *circuit-graph*.

Algorithm 1 represents a standard program that is written C/C++ language and linked to the Modelsim Verilog simulation. Through this user-defined algorithm, the Verilog-supported VPI routines collect all the information regarding the components from the gate-level netlist. The coordination of extracted information results in graph-oriented structure and writing it to a Graph Modeling Language (GML) format. GML is a text file format supporting network data with a very easy syntax [121]. As an annex to the Algorithm 1, the end part describes a final GML output format. In the outcome of Algorithm 1, the nodes and edges are attributed to the netlist components and nets, respectively.

As we can see that the input to the Algorithm 1 is a gate-level netlist, and the output is a graph network in GML format, so it is possible to visualize this graph network/ circuit-graph. Gephi is the open-source tool that can visualize the generated circuit-graph. Gephi

Algorithm 1: Transformation of IEEE standard netlist to Directed Cycle Graph (DCG)

Result: **Graphical delegation of gate-level netlist**
Input: gate-level netlist

```
1  class class_name
2      public:
3      string Net_name;
4      string Drivers[size], Loads[size] ;
5  end
6  itr = vpi_iterate (vpiNet, Module) ;
7  while (net =vpi_scan(itr)) do
8      net_name = vpi_get_str(vpiName, net) ;
9      net_class net_name ;
10     net_name.Net_name = net_name ;
11     drivers = vpi_iterate(vpiDriver, net) ;
12     while (driver_h =vpi_scan(drivers)) do
13         pri_h = vpi_handle (vpiPrimitive, driver_h);
14         driver = vpi_get_str(vpiFullName, pri_h);
15         net_name.Drivers[i] = driver;
16     end
17     loads = vpi_iterate(vpiLoad, net) ;
18     while (loads_h =vpi_scan(loads)) do
19         prim_h = vpi_handle (vpiPrimitive, loads_h);
20         load = vpi_get_str(vpiFullName, prim_h);
21         net_name.Loads[i] = load;
22     end
23     Save the each class in a file.txt
24 end
25 Reload the class objects and attributes from file.txt
26 Save the attributes of class object in .gml format
```

Output: *Circuit-Graph*

```
Graph [
        Graph Header     // E.g., Directed, Multigraph etc.
                         //
        node [           // Nodes are gate-level components
                id:      // components' names/labels
            ]
        edge [           // Edges are nets
            label:       // Edges' names
            source:      // Source Node
            target:      // Target Node
            ]            // Source -> Target (Edge-Direction)
        ]
```

[122] is a visualization tool and exploration software for graphs and networks. The Algorithm 1 symbolizes an example of transforming a circuit (openMSP430 16bit microcontroller core written in Verilog) to the graph. Lines 7-10 indicate the pseudocode for col-

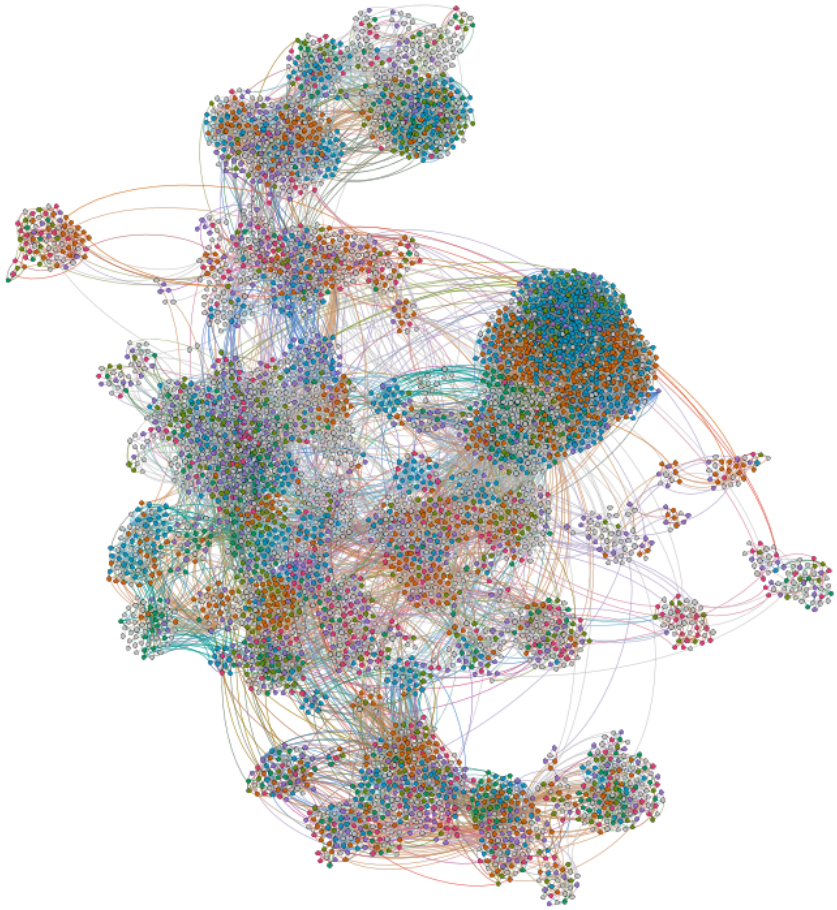


Figure 3.4 – Graph Network Model (Circuit-Graph) of openMSP430 (4995 nodes)

lecting all the names of the nets. For each net (circuit connection), the algorithm search for the single/multiple driver component/entity (lines 12-16) while lines 17-22 search the single/multiple loads of net. Finally, it stores the information in class (C++ object-oriented entity) objective (form lines 1 to 4). Lines 25 and 26 transform all the reserved information to a graph in GML format as specified in the annexure of algorithm 1. Fig. 3.4 visualizes the resultant circuit-graph. Fig. 3.5 represents the corresponding component distribution where the color of the bars indicates the nodes in the network in Fig. 3.4.

Now, the circuit structural information in an intermediate graph format eases the application of relevant mathematical and statistical algorithms for information processing on gate-level netlist. Once the intermediate graph structure is available, there are very dedicated libraries in R-language to extract structural information at lightning speed. For example, Fig. 3.5 represents a bar graph that elucidates the frequency of used library cells to synthesis the openMSP430 circuit, and it is generated by using the R/igraph [123] library package in R-language. The bar graphs in Fig. 3.5 represent the distribution of the components in the circuit-graph, and it exactly matches with that in the gate-level netlist.

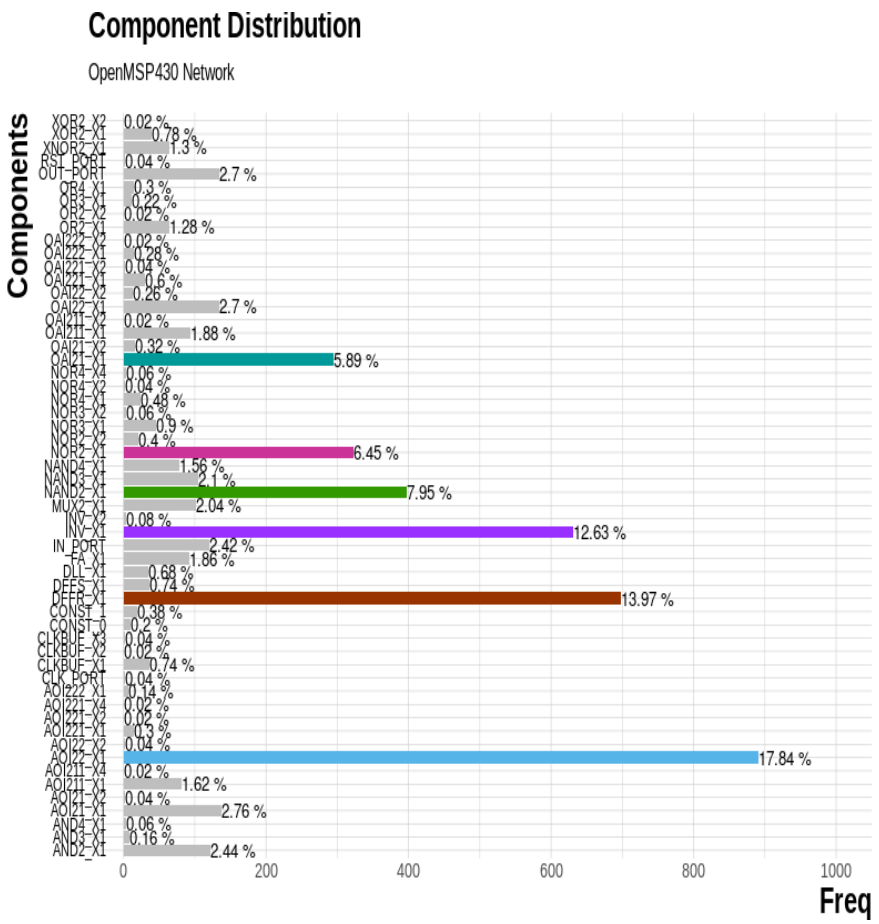


Figure 3.5 – Circuit Component Distribution of OpenMSP430 16bit-μC

3.4 Node2Vec Graph Embedding Methodology

3.4.1 Overview of the Methodology

Fig. 3.6 portrays a better overview of the work based on this method. Section 3.3 is briefing the way of mapping gate-level netlist into the probabilistic graph model and gives an insight into the work. When the structural information of gate-level netlist is embedded into the probabilistic graph, the statistical properties of a graph node are conventionally equivalent to that of a sequential (flip-flop) (or) logic (gate) elements of the circuit. To execute this preliminary aim of this work-flow multiple user-defined VPI functions had written in C/C++ and, it is applied to extract all the relevant details of the gate-level netlist and formatted into a probabilistic graph model through GML graph attributes. In the successive stage of this preliminary work, an SVM-Regressor (SVR) and fully connected DNN have been adopted as the learning frameworks of the features from the probabilistic graph. SVR represents a standard machine learning algorithm whereas DNN is based on the deep learning algorithm.

Node2vec algorithm is a probabilistic-based algorithm that uses the random walk method to generate the feature matrix X for the implemented learning frameworks. This algorithmic framework provides the feature dataset corresponding to the Ethernet MAC circuit

in the desired dimension within fractions of seconds. The random walk method gives a feature vector corresponding to a node by preserving the neighborhood structure. The feature vector is mainly based on transition probabilities from source to target nodes in the neighborhood area and the degree of nodes.

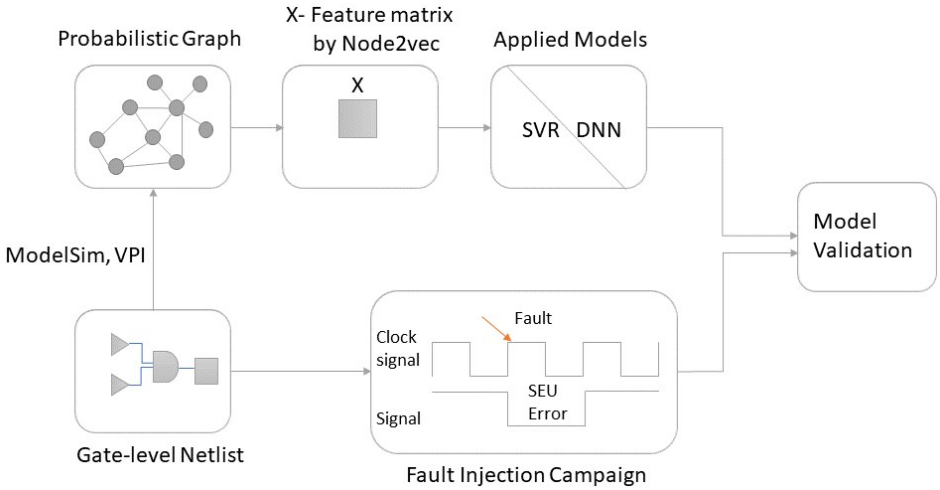


Figure 3.6 – Systematic Work Flow of Node2vec Application [1]

By choosing the fault injection process as the principal method to compare with the observations from the implemented models, more stringent validation of the expected aims become possible. As observed from figure 3.6, the fault-injection-based ground truth data is shuffled and has split with a test size of 40% and a Training size of 60%. After training the learning Machine Learning (ML)/Deep Learning (DL) models, predicted FDR values of flip-flops have been compared with the test vectors from the fault injection campaign FDR data. The ML/DL algorithms are implemented in python with the help of Keras and Scikit-learn libraries which are available as open-source machine learning libraries for Python programming language.

3.4.2 Scalable Feature Learning on Graphs

The node2vec algorithm by Aditya Grover in [109] is endowed here in its novelty. Node2vec algorithm is a framework for learning continuous feature representations in the graph network. It maps the nodes of the graph into the desired dimensional feature space and maximizes the likelihood of preserving the network neighborhood of nodes. Node2vec algorithm can apply to any given directed or undirected, weighted or unweighted edge networks.

Nowadays, representing a dataset in a graphical domain becomes a practically advantageous software tool. We use this approach for predicting and visualizing the probability factors over nodes and edges. The netlist from the gate-level abstraction of the circuits is successfully represented in the graph network domain. For performing a prediction analysis, a careful effort is required to develop a feature vector space that is suitable for different learning algorithms. This requirement has been achieved with the node2vec algorithm.

The feature learning framework of the node2vec algorithm has been formulated as a maximum likelihood optimization problem. The given network can be represented as

$G = (v, \varepsilon)$, where v represents vertices or nodes, and ε represents the edges between the vertices. $f : V \rightarrow \mathbb{R}^d$ is the mapping function from a node to d dimensional feature space, where V stands for a whole set of vertices. f is a matrix with size of $|V| \times d$. A neighborhood sampling strategy S defines a network neighborhood as $N_s(u)$ of a source node u. The framework optimizes the objective function f by maximizing the log-probability for observing a network neighborhood $N_s(u)$ for a node u, conditioned on its feature representation. The objective function is given by:

$$\max_f \sum_{u \varepsilon V} logPr(N_s(u)|f(u)). \tag{3.1}$$

The sampling strategy developed for node2vec is a flexible random walk that interpolates two important sampling strategies termed Breadth-First Sampling (BFS) and Depth-First Sampling (DFS). In BFS, the sampling nodes are the very immediate neighbors of the source node whereas, in DFS the neighbors have been obtained by sampling sequentially at increasing distance from a source node. The two important factors in the node2vec algorithm are flexible biased random walk and search bias α. Let consider a source node u and a random walk length l and c_i denote the i^{th} node in the walk from source node $c_0 = u$. The probability of c_i given c_{i-1} is generated by:

$$P(c_i = x \mid c_{i-1} = v) = \left\{ \begin{array}{cc} \frac{\pi_{vx}}{Z} & \text{if} \quad (v,x) \in E \\ 0 & \text{Otherwise} \end{array} \right\} \tag{3.2}$$

Where: π_{vx} is the unnormalized transition probability between nodes v and x, and Z is the normalizing constant. The search bias factor α is a major factor in calculating π_{vx}. Consider a random walk that just traversed the edge (t,v) and resides on node v. As a next step in the random walk, an unnormalized transition probability π_{vx} on the edge (v,x) leading from v, is estimating. The unnormalized transition probability is set to $\pi_{vx} = \alpha_{pq}(t,x).w_{vx}$, where:

$$\alpha_{pq}(t,x) = \left\{ \begin{array}{ll} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{array} \right\} \tag{3.3}$$

and w_{vx} is the weight of the edge. In the case of unweighted edge, $w_{vx} = 1$. The d_{tx} is the shortest path between t and x. Parameter p is called the Return Parameter and it controls the likelihood of immediately revisiting node in the walk. q is called an In-Out parameter which allows the search to differentiate between inward and outward nodes. Here, feature space with dimension 8 is extracted. The feature vectors of three arbitrary flip-flops have been plotted in Fig 3.7 for giving an illustration of the vector's statistical variance.

3.4.3 Device for Test

To test the applicability of the node2vec generated features for the machine learning frameworks in system engineering, a validation effort is performed on the 10-Gigabit Ethernet MAC IEEE 802.3 standard circuit. Experimenting with independent fault injection at each flip-flop and documenting how probable the faults affect the overall function of the circuit provides the ground truth dataset for the model validations. About one thousand two hundred and two (1202) flip-flops have been included in the evaluation of prediction models. The circuit is accessible at OpenCores [124] as a 10-Gigabit Ethernet project.

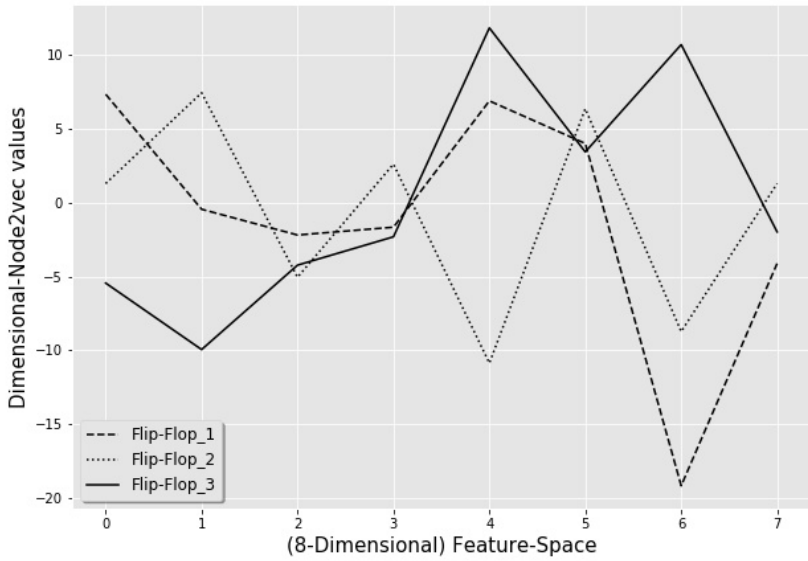


Figure 3.7 – Feature Vector of Three Arbitrary Flip-Flops [1]

3.4.4 SVR Inference on Node2vec Database

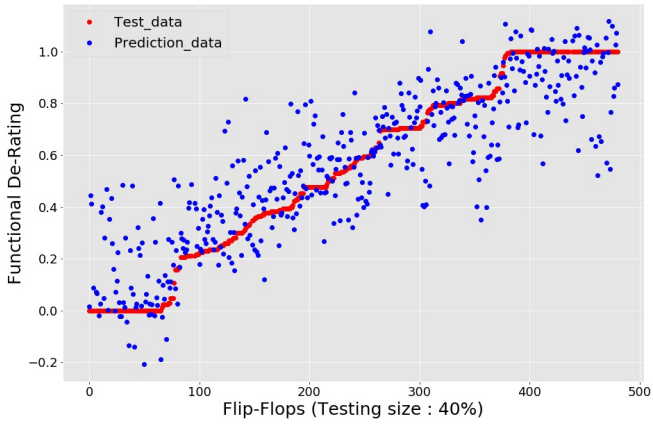
The prediction result of Support Vector Machine (SVM) Regression provided in figure 3.8a and jointly plotted their scatter plot in figure 3.8b respectively. The corresponding evaluation metrics have been tabled in Table 3.4. In SVR, we use the RBF kernel function which is described as,

$$K(X, X') = \exp\left(-\gamma \left\| X - X' \right\|^2\right). \tag{3.4}$$

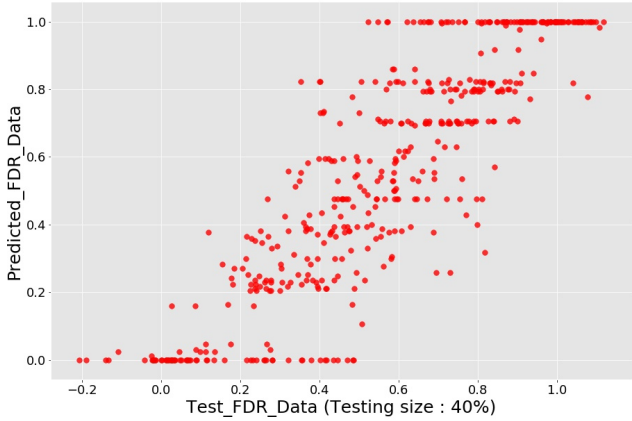
The X and X' are the two data points in vector form. The kernel K maps them to higher dimensional vector space. γ is called the spread of the kernel function and, it tuned to γ = 0.01. The other important parameter is epsilon ε which, responsible for error tolerance and set to ε = 0.0125. The parameter C is the regularization scheme and, proper value is chosen for the penalty factor C. Here C = 10. A grid-search cross-validation method tunes the parameter values. From the prediction diagram Fig. 3.8a, the predicted values approximating the original values which, sorted in ascending order by values. The scatter plot in Fig. 3.8b indicating a good correlation between predicted and original test data. But there is still a space for improvement because the scatter plot having a variance between the axial components. The metrics R^2 from Table 3.4 is indicating the good regression fit of prediction with original data. It is almost 69%. If the predicted values approximate more likely to the tested data, the R^2 will tend to the numerical value 1. In the same way, the metric MSE form Table 3.4 is equal to 0.027 and, it will close to 0 when the approximation becomes better. The metric EVS also mentioned in Table 3.4.

3.4.5 DNN Inference on Node2vec Database

The DNN architecture has been chosen according to Table 3.2. The input layer is nothing but the feature vectors. The Dense_1, Dense_2, Dense_3, and Dense_4 are the hidden layers. Dense_5 is called the output layer which, outputs the estimated regression values. Each hidden layer is a fully-connected dense layer where the number of inputs to each



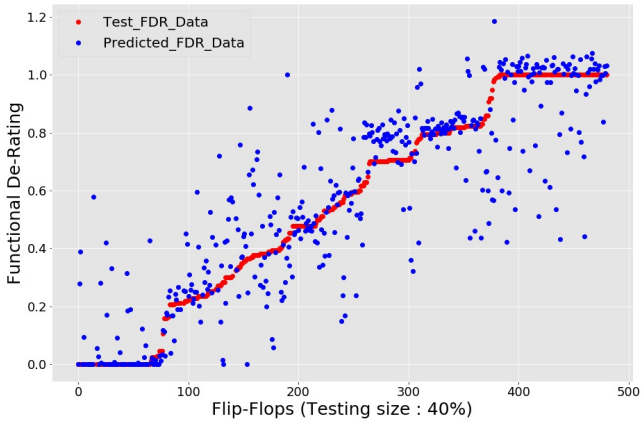
(a) Prediction Over 40% Test Data



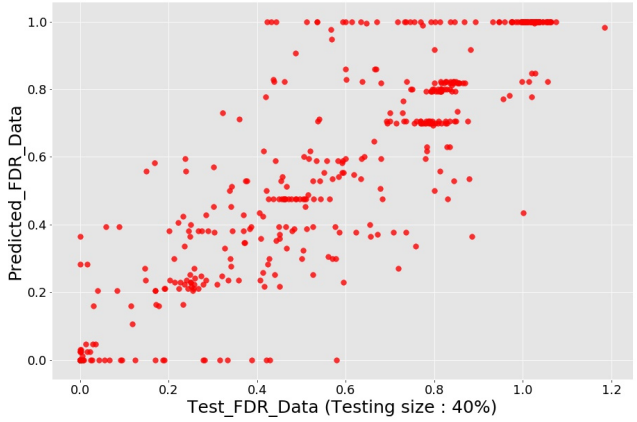
(b) Scatter Plot Between Prediction and True Value

Figure 3.8 – Regression by SVR Model [1]

neuron is equal to the output size of the previous layer. The weights of neuron inputs and the bias factor are the parameters that need to be optimized. The hyper-parameters such as loss = 'Mean Squared Error', optimizer = 'Adam' and batch_size = 10 are chosen according to cross-validation method. The Dense_1 layer has a shape of 126 neurons. With the input feature vector of dimension 8, DNN training for acquiring a good prediction accuracy becomes difficult. So the Dense_1 layer will map the low dimensional input vectors to a high dimensional space. DNN will show significant performance with a higher dataset dimension. The prediction from Fig. 3.9a shows that almost a good approximation made by the DNN. More prediction values are stick to the true values, which indicates a good R^2 value. From Table 3.4, it is given that R^2 = 0.77. The MSE value is 0.0259, which indicating that the mean error is also low. Fig. 3.9b providing a scatter plot. It provides information regarding the correlation between original test values and predictions. Here also, we can see the variance between the two axis components.



(a) Prediction over 40% Test data



(b) Scatter plot between prediction and true value

Figure 3.9 – Regression by DNN Model [1]

Table 3.2 – DNN ARCHITECTURE [1]

Layer	Output shape	Parameters
Dense_1	126	25326
Dense_2	64	8128
Dense_3	36	2340
Dense_4	12	444
Dense_5	1	13

3.4.6 A Comparison: DNN Vs SVR

Metrics form Table 3.4 indicating a dominant performance of DNN in terms of R^2, EVS, and MSE. The score EVS is used to measure the discrepancy between model-driven values and actual data. The high value near 1 shows the model is providing a valuable prediction. In that sense also, the DNN model looks good compared to SVR. But here, some more facts need to be compared. In Table 3.3, the time required to execute different models has been compared. The fault injection campaign over 1202 flip-flops of the Ethernet-MAC circuit took nearly five days per ModelSim software. SVR seems to be very fast because the DNN needs to optimize a comparatively large set of parameters, as explained in Table 3.2. But, when compared to traditional fault injection methods, an important drawback is that ML/DL models depend on 60% true detests. 60% of the whole dataset for the training process has been generated by the fault injection method. So comparatively, half of the fault injection time needs to train the prediction models. So, we can say that ML/DL models are almost 40-50 percent faster than the traditional ones.

Table 3.3 – TIME COMPARISON

Model	Time
Fault Injection (1 Modelsim)	5 days
Fault Injection (7 Modelsim)	17 hours
SVR	$<$ 1 minute
DNN	6 minutes

Table 3.4 – METRIC COMPARISON FOR DIFFERENT REGRESSION MODELS (TRAINING SIZE = 60 %)

Model	MSE	EVS	R^2
DNN	0.025995	0.770322	0.770169
SVR	0.027359	0.690909	0.689758

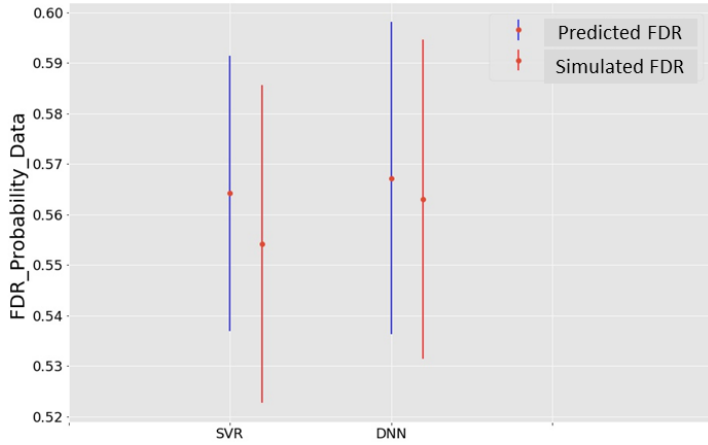


Figure 3.10 – CI Comparison: SVR Vs DNN [1]

Finally, the DNN and SVR are compared using 95% Confidence Interval (CI) and Mean values between predicted and original values. Fig: 3.10 showed this comparison. Here DNN performs comparatively better because the difference between the means of the respective predicted and the target values is small compared to that of SVR.

3.5 Graph Convolutional Embedding Methodology

3.5.1 Literature Overview

Different prodigious research works have been introduced in the past decade in generalizing a conventionally established neural network (e.g., RNN and CNN) for working on

arbitrarily structured graphs, even though it is a challenging problem. This work is centered on GCN [110] neural network. A spectral approach is the principal core of GCN, and a similar approach has been introduced in [125]. By GCN, it exemplifies the spectral rule approach in the graphical learning process, and it achieves significantly faster training times with higher predictive accuracy and also reaches state-of-the-art classification results on several benchmark graph datasets.

3.5.2 Model Definition

The Graph Convolutional Network is a synthetic neural network architecture for machine learning on graphs. Following paper [110], reveals the fact that most of the graph neural networks have been addressing a common architecture in general, which lead to the name called Graph Convolutional Neural Networks. The convolution name comes after using the filter parameters shared across all locations of the graph. The created probabilistic graph model of the gate-level netlist is embedded into the GCN network with the intention of learning the function of features in the graph. The graph is described as a $G = (v, \varepsilon)$, where v represents vertices or nodes and ε represents the edges between the vertices. The graph is characterized as,

1. Every nodes i is attributed with feature vector x_i of dimension D. So for N nodes, we have feature matrix $X: N \times D$.

2. Another important parameter is the adjacency matrix A, which indicates the graph structure.

3. The propagation rule will produce a node-level output of $Z: N \times F$, where the F represents a feature vector of each output node.

$$H^{(l+1)} = f(H^{(l)}, A) \tag{3.5}$$

where: $H^{(l+1)}$ represents the any hidden layer node matrix at $(l+1)^{th}$ level and it equivalent to the function of previous hidden layer node matrix H^l at l^{th} level and the adjacency matrix A. H can be taken as the feature matrix X at initial level, ie $H^{(0)} = X$ and Z at final level. Z represents the graph level output.

3.5.3 Model Propagation Rule

In this whole paper, an exact propagation model for the Graph Neural Network is adapted to tackle the prediction problem. A simple form of the layer-wise propagation rule abbreviated as:

$$f(H^l, A) = \sigma(AH^lW^l) \tag{3.6}$$

where: W^l is the l^{th} neural network weight matrix and $\sigma()$ is the activation function like Rectified Linear Unit (ReLU), while this work utilizes a hyperbolic tangent activation function (Tanh). While the above propagation rule (3.6) seems to be very simple, it is proven to be a dynamical algorithm. The disadvantages of this kind of model: the adjacency matrix (A), which is not normalized, so the multiplication of A with feature matrix (X) will change the scale of the X completely. The author mentioned a second problem that the model does not consider the self-features of a node itself. And the problem is completely taken away by providing an identity matrix for the nodes.

The major problem is overcome by normalizing the matrix A. Normalization of A is achieved by an inverse diagonal node degree matrix D, such that the rows of $D^{-1}A$ sum to 1. So the multiplication becomes more similar in taking the average of neighboring nodes. It leads to symmetric normalization i.e., $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, and it is more than just a mere averaging of neighboring features. These combined methods are used in this work as a propagation rule which is similar to the way implemented in paper[110] and the final layer-wise propagation rule provided as:

$$f(H^l, A) = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^l W^l) \tag{3.7}$$

where, $\hat{A} = A + I$; with I defined as identity matrix and $\hat{D}$ is the diagonal degree node matrix of $\hat{A}$.

3.5.4 Input Feature Matrix

Before applying the GCN model, we can use a node2vec algorithm as implemented in ArticleNo2II to generate an initial feature matrix corresponding to the nodes in the probabilistic graph. Node2vec [109] is an algorithmic framework for learning continuous feature representations of nodes in networks. The algorithm generates a low-dimensional feature space dataset that maximizes the likelihood of nodes that preserve similar structural network neighborhoods. Here, the objective function is optimized through the stimulated biased random walks. It holds a spectrum of equivalences from homophily to overall structural equivalence by anticipating a balanced exploration-exploitation trade-off.

3.5.5 Work-Flow of the Method

Fig. 3.11 depicts a better overview of the work. Before stepping into the detailed structure of the whole approach, it is very relevant to brief the importance of mapping the gate-level netlist into the probabilistic graph model. The more the mapping achieves accuracy, the more the model delivers a valid result because the graph structure maintains the required statical information. ModelSim Linked VPI functions extract all the relevant details of the gate-level netlist and format them into a probabilistic graph model through GML attributes. A detailed review is provided in section 3.3. As stepping forward into the successor stage of this work, GCN adopted a model in (3.6) in order to learn the whole designed probabilistic graph. The more comprehensively explained hierarchical architecture of GCN is updated in section 2.12.3.

The netlist representation in graph domains is subsequently used to extract the adjacency matrix, which is represented by A in Fig. 3.11. Correspondingly, a feature matrix X is also obtained by the random walk method using the node2vec algorithm. The random walk method gives a feature vector corresponding to a node with respect to the formation of its neighboring nodes. The feature vector is mainly based on transition probabilities from source to target nodes and also the degree of nodes. These are the two main inputs given to the GCN model. Then, GCN is commenced learning the whole netlist as a probabilistic graph. As soon as it processes the adjacency matrix and feature matrix, a circuit-graph model of the netlist is delivered. After that, this model is used for the training phase and testing phase for accomplishing the FDR prediction goal. Finally, the predicted data is compared with the fault injection campaign of FDR data. The whole deep learning framework is implemented in MXNet [126].

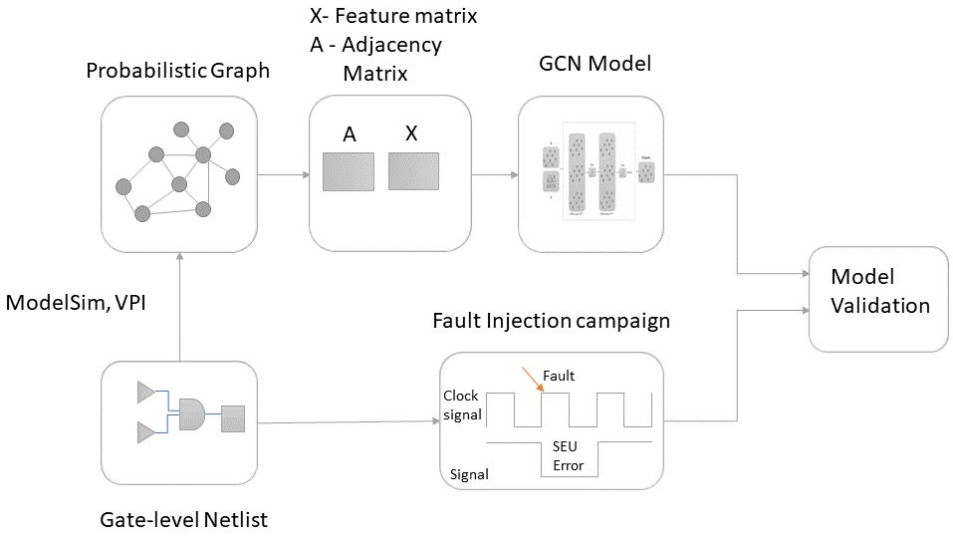


Figure 3.11 – Systematic Block Diagram of the Scientific Work [2]

3.5.6 Test Devices for Modeling and Validations

The model tested with two circuits. The very first one is the double precision floating point adder which is extracted from the double precision floating point core as a sub-module, and meets the IEEE 754 standard and available in the OpenCores [127] website. The second circuit is also accessible from OpenCores as 10-Gigabit Ethernet [124] project, where Management Data Input/Output (MDIO) function of this module designed to meet 10-Gigabit Ethernet IEEE 802.3 standard. In MAC design (based on the Xilinx LogiCORE 10-Gigabit Ethernet MAC), the transmitter and the receiver incorporate the reconciliation layer. Therefore the receive engine, as well as transmit engine, are specifically designed to interface the client and the physical layer.

3.5.7 Results and Discussions

Double Precision Floating Point Adder

Fig. 3.12, Fig. 3.13 and Fig. 3.14 pictorially represent the results for double precision floating point adder. Figure 3.12 represents the 95% confidence interval comparison of the predicted Functional Derating (FDR) data of flip-flops with the generated FDR data from a random fault injection campaign. The Confidence Interval (CI) calculated in python, by finding the mean of the flip-flop's FDR distribution and their FDR distribution error for 95% confidence. There are no electrical features extracted from the circuit's gate-level netlist to train the upholding convolutional neural network model. The training has been done with less than 10 flip-flops' FDR. The overall comparison indicates the prediction almost following the simulated FDR data of SEU faults. As observed from the histogram graph depicted in figure 3.13, the prediction of the FDR Probability Distribution Function (PDF) of the flip-flops comparatively very close to the original PDF of the flip-flops. Fig. 3.14 compares the sorted FDR value of simulated and predicted data. This sorted FDR plot only shows how an overall functional derating curve behaves for flip-flops.

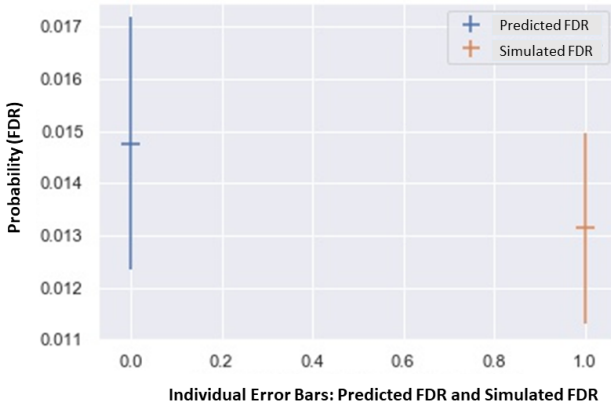


Figure 3.12 – Confidence Interval(CI) Comparison [2]

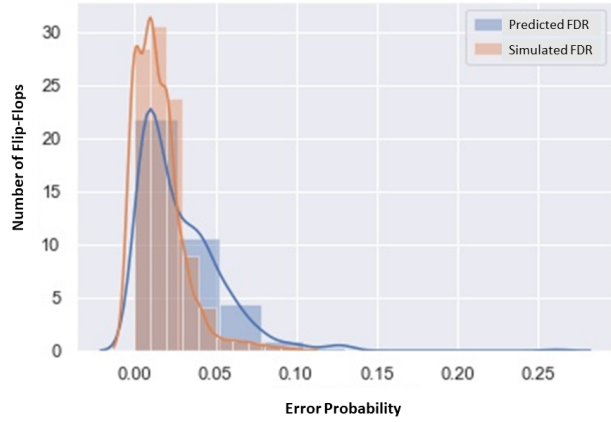


Figure 3.13 – Histogram Comparison [2]

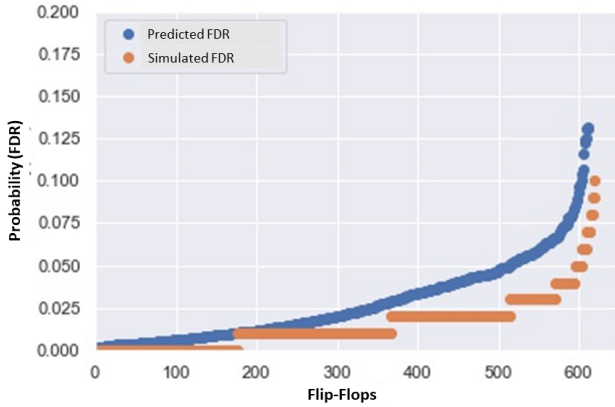


Figure 3.14 – Sorted FDR Probability Comparison [2]

Ethernet MAC

Here the modeling tries to validate on Ethernet MAC circuit. The results reveal that the proposed algorithm is effective for predicting a completely different histogram with a training sequence of 5 flip-flops (i.e., less than 1% of the overall flip-flop number). Fig.

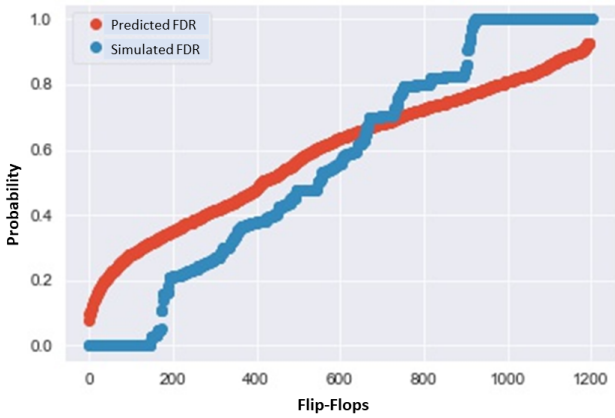


Figure 3.15 – Sorted FDR Probability Graph [2]

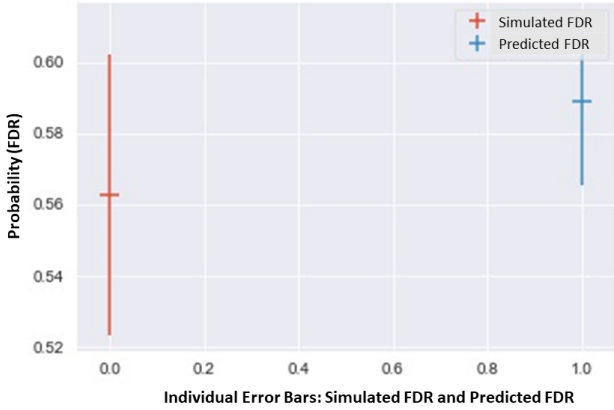


Figure 3.16 – Representation of CI Comparison [2]

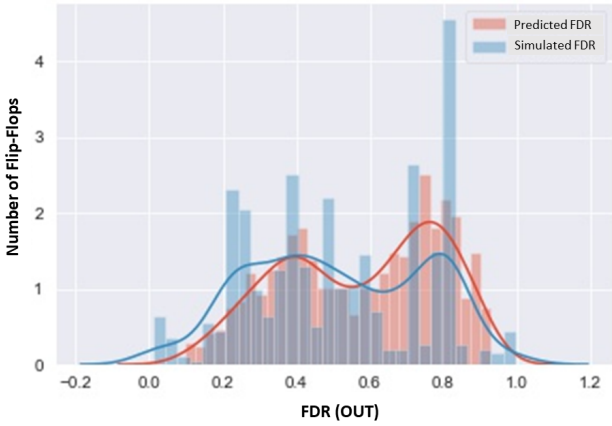


Figure 3.17 – Histogram Comparison (Filtered Some Outliers) [2]

3.15 does not provide any individual flip-flip comparison. The next aim of this work is to predict individual flip-flops' FDR. This comparison provides intuition to the reader that the model can get a reasonable approximation for the flip-flops' independent structural

information. Figure 3.16 represents the confidence interval comparison of the predicted FDR data with FDR data from fault injection campaign on the sequential elements in each clock-cycle independently. Figure 3.17 represents the PDF where some of the data points are filtered out. The filtered out points are considered as outliers within the data space and plotted the remaining data.

3.5.8 Model Drawback

Even though GCN models are achieving their accuracy within a reasonable time duration, the stability in providing good results appears to be degraded if we increase the number of hidden layers of graph convolutional neural networks beyond a certain number. This fact is very important in terms of the prediction-scalability if we consider very-large circuits. But some researchers coming with new optimization methods to overcome the challenges faced by GCN.

3.6 Inductive Representation Learning Methodology

3.6.1 Methodological Overview

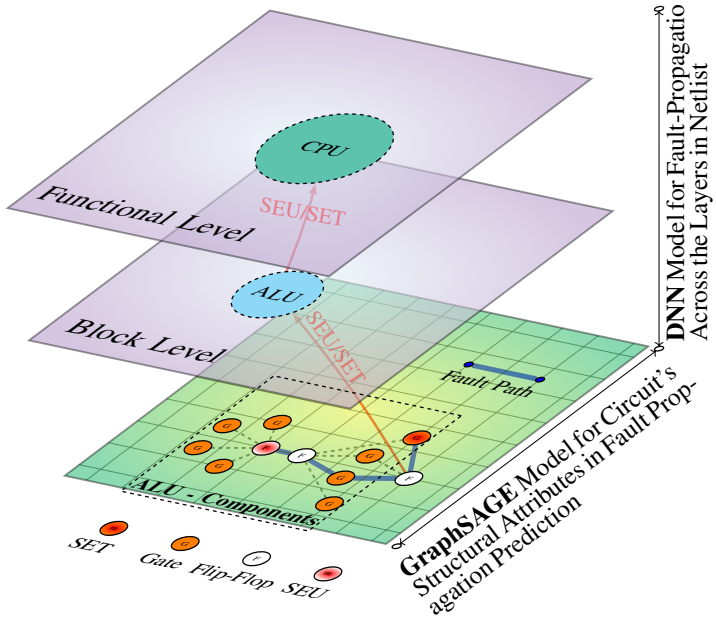


Figure 3.18 – The Fault (SEU/SET) Propagation Model in CPU

Fig. 3.18 demonstrates the role of each neural network structures in the modeling of SEU and SET caused fault propagation. In this context, the gate-level abstraction of the circuit has been transformed into a graph network where vertices (v) analogous to the flip-flops and gates, and the directed edges (ε) represent the connection between them from circuits' input ports to output ports direction. For a virtual example, the graph network of the CPU's ALU part is illustrated in the lower layer of Fig. 3.18. Fig. 3.18 represents the fault propagation through the hierarchical cross-layers in the gate-level netlist and does not mean the fault-propagation across different hierarchical abstraction levels such as RTL and Architectural level. In Fig. 3.18, the superimposed layers in gate-level are disseminated

virtually. The graph-function G of the transformed network is given as:

$$G(v, \varepsilon) \tag{3.8}$$

where: v and ε represents the nodes and edges respectively. The graphSAGE [128], is a general inductive framework which leverages node's (here, flip-flop's or gate's) feature information to efficiently generate node embeddings. The graphSAGE could be also explained as a graph based neural network with sampler and aggregator functions. The graphSAGE framework represents the function that generates a feature database corresponding to the flip-flops and gates by sampling and aggregating features from the node's local neighborhood. The deducted feature database embeds the netlist's structural attributes that determine the soft-error propagation probabilities from a flip-flop (or) a gate to the functional level of the circuit. Fig. 3.18 points out the role of graphSAGE in information sharing between nodes. This embedding part provides not only the local role of nodes in the graph but also their global positional role. Approaches like GCN and node2vec in sections 3.5 and 3.4 are inherently transductive and generally unable to postulate a learning function to unseen nodes. But an inductive node embedding algorithm like graphSAGE meant to be an optimized generalization across the graph. Therefore, not all the nodes need to be available in the training phase of the graphSAGE algorithm and it also reduces the required training size of the downstream DNN algorithm that predicts the fault propagation across the layers. The sampler function in graphSAGE defines the node's neighborhood definition through a uniform sampling of a fixed number of nodes instead of sampling the entire neighborhood space at each depth-wise iteration. It will result in boosting the optimal usage of memory and reduce run-time complexity. The basic idea of the graphSAGE is simplified and explained in Fig. 3.28. From the state-of-art of the graphSAGE framework, numerous aggregator functions are available like Mean aggregator, Long short-term Memory (LSTM) aggregator, Pooling aggregator and Graph Convoultional Network (GCN) based aggregator. Here we implemented a Pooling aggregator with help of a python neural network libraries. Deep Neural Networks (DNNs) [129, 130] is also a dominant model in this work. DNNs are employed to model the fault propagation path across the layers, as mentioned in Fig. 3.18. When a SEU fault at flip-flop (or) a SET fault at the gate (as in Fig. 3.18) propagates to the functional level (upper layer in Fig. 3.18) of the circuit, it traces a fault path. The DNN will learn that path based on the observed fault propagation probabilities from the fault injection process. In this work, a general fully connected DNN has been implemented.

3.6.2 The Principal Goal

The equation (3.9) represents a formulation of the principal aim of this method. The raw-database as prescribed in Table 3.5 and Table 3.6 are referenced in (3.9) as raw-database (Graph,circuit). The graphSAGE performs and optimizes the convolution process between raw-database and circuit-graph ($G(v, \varepsilon)$). These optimized features-set serves the role of an input to the DNN which draws a function proportional to the Functional Failure Rate (FFR).

$$\text{DNN}\{\text{graphSAGE}\{\text{raw-database (Graph,circuit)} \circledast G(v, \varepsilon)\}\} \propto FFR_{SEU/SET} \tag{3.9}$$

3.6.3 Extraction of Raw-Database: Graph Data Mining

The graph network is the source for extracting relevant local as well as global information about components in the circuit. In general, the reliability modeling is complexly

dependent on the black-box modeling because: in most cases, input-output relations are experimentally derived and fitted with poly-parameter dependency. Bringing a detailed and transparent explanation for circuit-fault propagation in reliability modeling poses a high degree of difficulty. Modeling reliability with such a point of view is called white-box modeling. In [131], James Ledoux states that the white-box (or structural) point of view is an alternative approach of modeling in which the structure of the system is explicitly taken into account. A structure-based approach allows analyzing the sensitivity of the circuit system reliability concerning the reliability of its components. A detailed explanation of vital information of components in the circuit, are itemized as per their importance in the Tables 3.5 and 3.6. The mathematical formulation of the listed properties in 3.6 of each component in circuit-graph are provided from equation (3.10) to (3.19). Table 3.5 provides that structural properties of components from the gate-level netlist. These are the real factors which contributes to the soft-error propagation to the output of the circuits. The Design Compiler from Synopsis is the commercial tool to extract such peculiarities from gate-level (or) the scripting in the python programming language is another way to access the factors in Table 3.5 from the generated garph. But, the factors in Table 3.6 is strictly generated from the circuit-graph. Very much dedicated libraries in R language facilitate the generation of the database as provided in Table 3.6. However, Table 3.5 and Table 3.6 are both useful as initial raw-database, the whole work simply stick to the Table 3.6 for the initial raw-database.

Table 3.5 – STRUCTURAL PROPERTIES FROM CIRCUIT

No.	Symbol	Quantity	Properties and Explanation
1	N_I	Fan-in connection	Number of inputs to a cell (Sequential/Gate)
2	N_O	Fan-out connections	Number of outputs to a cell (Sequential/Gate)
3	N_{P_i}	Distance to primary I/p pins	The number of cells (Sequential and Combinational) in the design between input pins and a given cell (Sequential/Gate)
4	N_{P_o}	Distance to primary Out-pins	The number of cells (Sequential and Combinational) in the design between primary output pins and a given cell (Sequential/Gate)
5	N_{TP_l}	Longest timing path	The number of combinational cells in the longest path from a given cell (Sequential and Combinational) to a flip-flop in the signal flow direction
6	N_{FF_s}	Shortest timing path	The number of combinational cells in the shortest path from a given cell (Sequential and Combinational) to a first flip-flop in the signal flow direction

Fig. 3.4 represents the graph-network of a openMSP430 circuit that includes 4995 nodes/components. The graph is generated by using open source software called Gephi [122]. The Gephi software develops the network of openMSP430 micro-controller (μC) from the GML file that is described in Algorithm 1. Fig. 3.5 represents the distribution of NanGate open-source digital cell library components that are used to synthesis the circuit. The colors of bar graphs in Fig. 3.5 indicate their presence in the graph. It also confirms the high-level efficiency of Algorithm 1 in representing the circuit. In Fig. 3.4 and 3.5, the grey colour is assigned to the components, those having their contribution less than 5%. Fig. 3.19, Fig. 3.21 and Fig. 3.23 are standing for how the listed component features in the Table 3.6 is relevant in the scenario of modeling fault propagation. How the listed features in Table 3.6 for each component are relevant in the scenario of modeling fault propagation is going to explain further. For example, consider the three features eccen-

Table 3.6 – STRUCTURAL PROPERTIES FROM GRAPH

No.	Symbol	Quantity	Properties and Explanation
1	$\eta(i)$	Clustering Coefficient (Transitivity)	The local clustering coefficient defines a node's degree of closeness to its neighboring nodes to form a clique (or) complete/closed subgraph. In (3.10), $S_i \subseteq N$ is the set of nodes connected to node $i \in N$ in the network G of N nodes and A edges. There are $\binom{n_i}{2}$ possible edges between nodes in S_i, where n_i is the cordiality of S_i.
2	$C_D(i)$	Degree Centrality	Let's define the degree of a node 'i' (Deg(i)) be the cardinality of the set of adjacent neighbors to the node 'i'. Then, the degree of centrality of a node 'i' is the degree of the node 'i' divided by the maximum possible degree of a node in the network. The maximum possible degree of a node is the number that is one less than the total nodes (n) in the network.
3	$P_D(i)$	Degree prestige	In contrast to the degree centrality of a node, degree prestige is readily related to a network's node by its in-degree rather than its degree as represented in (3.12). The high in-degree represents the popularity of the node in the network or simply represents the high prestige node.
4	$G_D(i)$	Gregariousness	The gregariousness of a node quantifies a node's propensity characteristics to pass the information simultaneously to others. Therefore, it generally depends on the out-degree (number of edges going out) of the node 'i'.
5	$C_C(i)$	Closeness Centrality	The position of a node in a network explicitly expresses the node's importance in the network. The average of shortest paths from the node (i) to all other nodes is represented in (3.14) as AvgDist(i). In (3.15), $\sum_{j=1}^{n} D(i,j)$ is the sum of weights of edges between i and j. Closeness centrality is defined as the multiplicative inverse of AvgDist(i).
6	$P_P(i)$	Proximity Prestige	The proximity prestige is bounded especially to directed networks and depends on the average of shortest paths to node 'i' from the nodes in the influence of node 'i'. The influence of node (i) is a set of all the nodes that reach directly to the node 'i' through a directed path. The proximity prestige is defined in (3.16). The influence fraction is the ratio of the cardinality of the influence set of node (i) to one less than the total number of nodes.
7	$B_C(i)$	Betweenness Centrality	The fraction of pairs that pass through node 'i' is given by $f_{jk}(i) = q_{jk}(i)/q_{jk}$ in (3.17), where $q_{j.k}(i)$ is the number of pairs that pass through node 'i'. Intuitively, $f_{jk}(i)$ represents the level of control that node 'i' possesses over nodes j and k in terms of regulating the flow of information between them. The betweenness centrality $B_C(i)$ is the average value of $f_{jk}(i)$ over all $\binom{n}{2}$ pairs of nodes. The $B_C(i)$ in (3.17) is the number of shortest paths passing through the node (i) while calculating the shortest paths between other pairs of nodes in the graph.
8	$J_M(i,j)$	Jaccard Measure	Jaccard Measure is a common neighbor-hood-based measure that estimates the similarity between the nodes i and j with their neighbor sets S_i and S_j. It is computed as provided in (3.18), $\lvert S_i \cap S_j \rvert$ represents the common neighbours.
9	$M_I(i)$	Morgan Index	The Morgan Index represents K^{th} degree of a node. It simply denotes the number of nodes reachable from the node(i) to a distance K.
10	$E_c(v)$	Eccentricity Index	The Eccentricity Index of a node (v) represents the maximum among the shortest path between node v and all other nodes in the graph.

tricity index, closeness centrality and in-degree of each node in the circuit-graph (Fig. 3.4) of OpenMSP430.

$$\eta(i) = \frac{|(j,k) \in A : j \in S_i, k \in S_i|}{\binom{n_i}{2}} \tag{3.10}$$

$$C_D(i) = \frac{Deg(i)}{n-1} \tag{3.11}$$

$$P_D(i) = \frac{InDeg(i)}{n-1} \tag{3.12}$$

$$G_D(i) = \frac{OutDeg(i)}{n-1} \tag{3.13}$$

$$C_C(i) = \frac{1}{AvgDist(i)} \tag{3.14}$$

$$AvgDist(i) = \frac{\sum_{j=1}^{n} D(i,j)}{n-1} \tag{3.15}$$

$$P_P(i) = \frac{\text{Influence Fraction(i)}}{\text{AvgDist(i) in Influence}} \tag{3.16}$$

$$B_c(i) = \frac{\sum_{j<k} f_{jk}(i)}{\binom{n}{2}}; \; f_{j,k}(i) = \frac{q_{j,k}(i)}{q_{j,k}} \tag{3.17}$$

$$J_M(i,j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} \tag{3.18}$$

$$E_c(v) = \frac{1}{\max\{dist(u,v)\} : \forall u \in V} \tag{3.19}$$

Fig. 3.19 shows the histogram distribution of eccentricity index of nodes. Eccentricity of a node v is the method of finding the maximum among the shortest path between node v and all other nodes in the graph.By highlighting such property of a node 'v', the information regarding the closeness of node 'v' to neighboring nodes and how far away from the node 'u' as provided in (3.19) are revealed. Most likely, 'u' nodes are falling within the subset of input or output ports of the given circuit. It also provides the hint of the location of the farthest node 'u' corresponding to each node v in the graph. In terms of modeling fault-propagation, this property approximates the probability of a fault at node v in propagating to output ports of the circuit. The circuit top views in Fig. 3.20, Fig. 3.22 and Fig. 3.24 exhibit various properties of the nodes in the graph (or) various properties of the component in the circuit. In Fig. 3.20, it is clear that the eccentricity varies across the nodes, and it justifies the histogram while comparing both figures Fig. 3.20 and Fig. 3.19. The high eccentricity is shown by very dark green points and the low eccentricity is shown by white points. Both are less in Fig. 3.20. From the histogram distribution in Fig. 3.19, the count of nodes with eccentricity index between 20 and 30 are high and such distribution is provided in Fig. 3.20 as points with the intensity of moderate green color and they are high in number. Similarly, the distribution of closeness centrality of each node is given in Fig. 3.21 and Fig. 3.22 provides the corresponding variation across nodes. The nodes with low closeness centrality are high in number in Fig. 3.21, and they are shown as white color points in Fig. 3.22. Similarly, the nodes with high closeness centrality are less in number (Fig. 3.21), and Fig. 3.22 picturized those nodes as digitized points in high-intensity green color. In the same way, the histogram in 3.12 shows the distribution of In-degree, which indicates the number of incoming edges to a node.

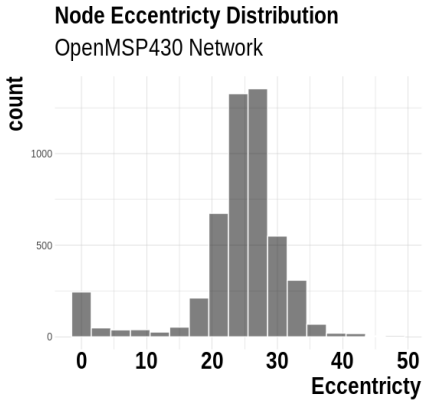


Figure 3.19 – Eccentricity Distribution

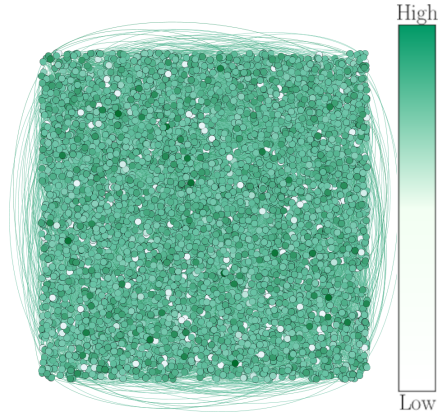


Figure 3.20 – Eccentricity Variation: Top View of Circuit-Graph

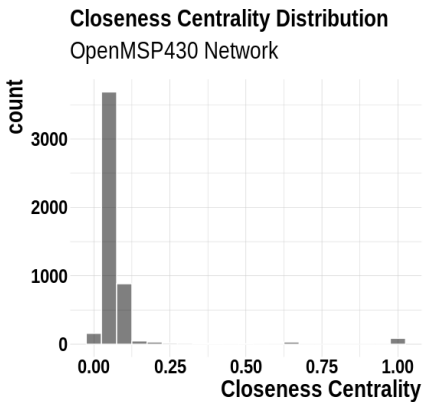


Figure 3.21 – Closeness Centrality Distribution

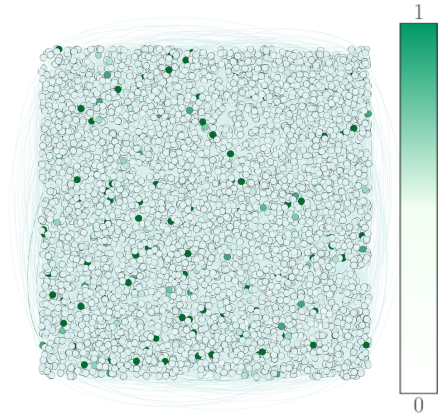


Figure 3.22 – Closeness Variation: Top View of Circuit-Graph

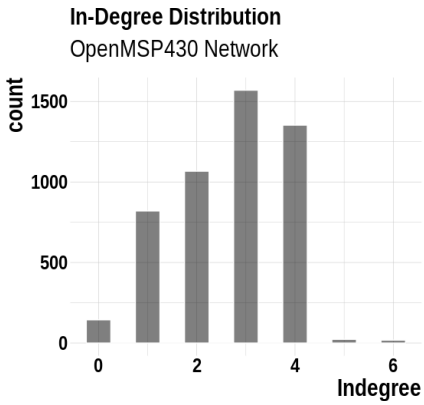


Figure 3.23 – In-degree Distribution

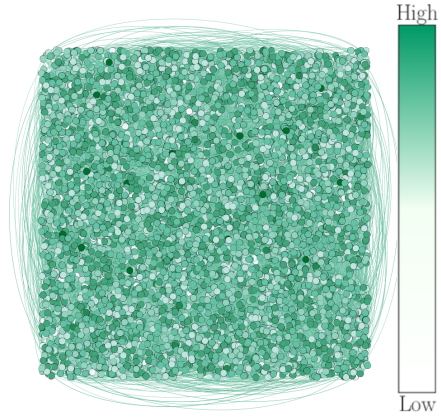


Figure 3.24 – In-degree Variation: Top View of Circuit-Graph

The distribution in Fig. 3.23 is more close to uniform in nature and, that kind of distribution is visualized in Fig. 3.24. From the visualization of Fig. 3.20, Fig. 3.22 and Fig. 3.24, it is way more clear that each node in graph (or each component in a circuit) are more distinguished by the listed characteristics in Table 3.5 and Table 3.6. These tabulated node properties facilitate the developed algorithm in this thesis in modeling the fault-propagation probability at the gate-level circuit. The node properties in Table 3.6 (or) from both Tables 3.5 and 3.6 can apply as initialized input vectors to the graphSAGE algorithm.

3.6.4 Experimental Workflow

The whole approach in Fig. 3.25 has been detailed through two successive work-phases. The first phase of the work is the Fault-Injection campaign, and the second phase is the applied research approach.

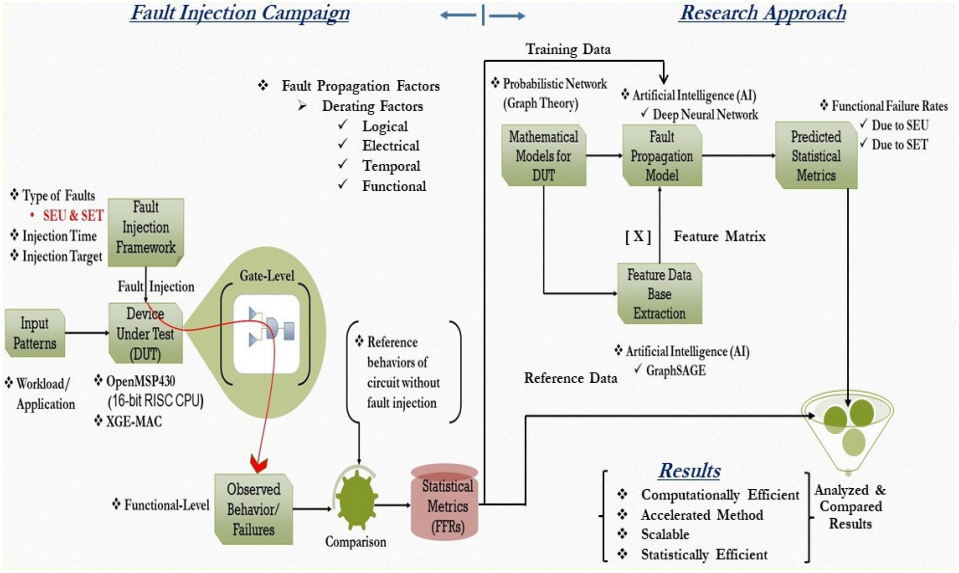


Figure 3.25 – A Systematic Workflow of the Implemented Scientific Work

Phase I: Fault-Injection Campaign

The Fault-Injection (FI) database for the fault-propagation analysis is generated as the main preparatory task of this workflow. The fault-injection database includes the Functional Failure Rates (FFRs) due to SEU and SET type fault injection at the Device Under Test (DUT). Fig. 3.25 provides FI-framework and the investigated fault propagation factors. The AI approach will subsequently use this database as the raw data for testing and training the applied neural networks.

As provided in Fig. 3.25, two different devices act as test circuits. The devices are an openMSP430 micro-controller and XGE-MAC IP component. The fault-injection campaign and its automation exploit the shell scripting and ModelSim simulation tool and linked VPI function to develop a fault-injection campaign. The workload in Fig. 3.25 mentions the applied input pattern. For SET fault, a digital pulse of an arbitrary width (depending on the technology) is applied to gate components. For SEU fault, an erroneous state is applied to the flip-flop until the next clock-cycle comes and changes the state of the flip-flop. The fault simulation at each clock cycle (exhaustive simulation) is stored in a text file (or) in

any other readable format. Finally, golden simulation (fault-free simulation) results and fault simulation results are compared to generate a database for training the inference engine called DNN.

Phase II: Research Approach

The second phase of Fig. 3.25 is nothing but the working-flow diagram of the AI-Framework. A detailed outline of AI-Framework in Fig. 3.26 provides the data flow between the applied algorithms as well as the data flow inside the algorithms. We can see that what are the sub-optimization algorithms and mathematical functions that serve inside each algorithmic structure. The blue-colored arrows in Fig. 3.26 mainly indicate the data flow between the algorithmic blocks, while grey-colored arrows represent the data flow inside an algorithm. To explain each step in Fig. 3.26, more detailed views are provided after this big picture.

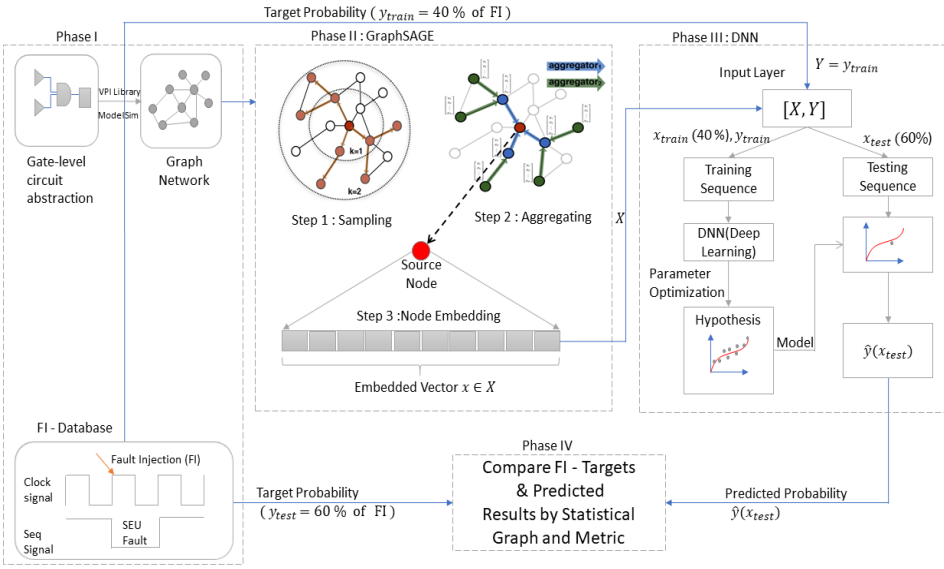


Figure 3.26 – Algorithmic Workflow Diagram of the Implemented Scientific Work [3]

- **Graph Generation: A Mathematical Model for Gate-Level**
  The research approach mainly consists of two types of neural network structures: graphSAGE and DNN. Before applying the neural networks, the gate-level circuit has been mapped to a Probabilistic Bayesian Graph (PGB). A Verilog Procedural Interface (VPI) library function links to a standard simulation tool (ModelSim/open-source) to map the gate-level netlist to a probabilistic network that represents the gate-level components as well as the connections between them. This process is represented in Fig.3.27 The graphSAGE and DNN algorithms explore this probabilistic graph further.

- **GraphSAGE: A Graph Embedding Algorithm**
  In this stage of the work-flow, a feature matrix (X) corresponding to probabilistic graph-nodes is extracted using the graphSAGE algorithm. The graphSAGE algorithm includes two principal steps. The first step is the sampler algorithm. The sampler

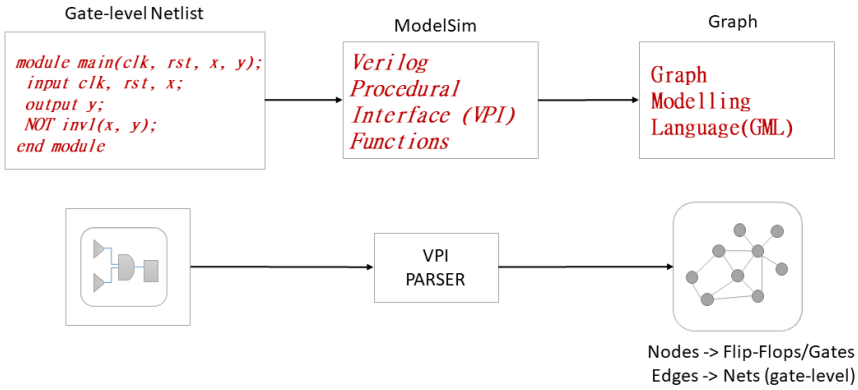


Figure 3.27 – Generation of Circuit-Graph

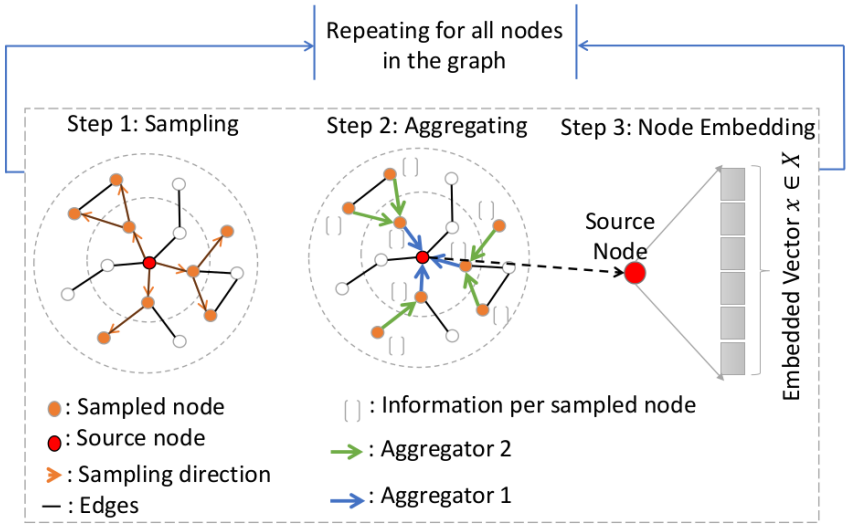


Figure 3.28 – GraphSAGE Algorithm [3]

algorithm defines the neighborhood space of a source node as provided in Fig. 3.28. In this scenario, we defined the parameter $K = 2$, which means that the sampler will sample up to the depth of 2 neighbor spaces. In the second step of the graphSAGE algorithm, an aggregator has been implemented at each depth ($1 \leq k \leq K$). In Fig. 3.28, blue and green arrows indicate the aggregator functions at depth k=1 and k=2 respectively. Here, a max-pooling aggregator is implemented using python libraries. The mathematical abstraction of the pooling aggregator [128] is formulated as:

$$AGGRE_k^{pool} = max(\{\sigma(W_{pool} h_{u_i}^k + b), \forall u_i \in N_k(v)\}), \tag{3.20}$$

where: (3.20) represents the aggregator function at depth k and it basically a neural network with parameters W_{pool} and b. Those parameters are optimized through unsupervised learning. $N_k(v)$ represents k^{th}-neighbourhood of vertex v and $h_{u_i}^k$ indicates the aggregated neighborhood vector and, σ is the activation function of the neural network. In this way, we could represent the whole graphSAGE algorithm as a

graph-based neural network. As a last step, the aggregated information is reformed into a vector as given in Fig. 3.28. The vector contains embedded information about the neighbors' number of inputs (fan-in), the number of outputs (fan-out), distances to the circuit output node, distances from the circuit input node, and probabilities of fault signal propagation from the source node to the neighbor nodes. Also, the embedded vector encompasses the source node's characteristics like the number of inputs and outputs (fan-in and fan-out), distance from the circuit input node, and the distance to the circuit output node. The embedded vector dimension is 50 for this experiment. A hidden convolutional process in the graphSAGE algorithm performs the information sharing between nodes. Any required dimension could be deduced here depending on the applied aggregator/convolutional kernel size. The three steps in Fig. 3.28 are repeated for every node in the probabilistic graph as a source node. At the end of this stage, the algorithm provides a matrix representation (X) for the circuit, where each row vector is the embedded features of the node.

- **DNN Inference Engine**
  Fig. 3.29 outlines the steps of DNN implementation that predicts the fault propaga-

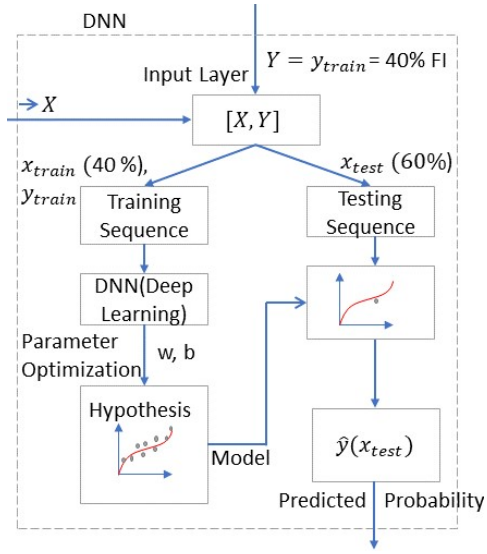


Figure 3.29 – DNN Algorithm [3]

tion across different layers as modeled in Fig. 3.18. There are two parts included in this stage. The first part is the training part of DNN, and the second one is the testing part of DNN. In the training part, 40 % of the feature matrix (X) that was deduced by graphSAGE and corresponding fault propagation probability metric from the FI-database are taken to postulate a hypothesis that best describes the target probability distributions ($FFR_{i,seu}$ and $FFR_{g,set}$). The optimized parameters: weights (w) and bias (b) of DNN that best fit the target distributions are provided as the trained model parameters. In the testing part, the proposed model is applied to an unknown input vector and predicts the target probability metric. The DNN architecture consists of 5 dense layers, including the input and the output layers.

- **Comparison and Analysis**
  The final part of the workflow includes a comparison between the predicted and reference fault propagation probability metrics. The compared results are plotted and analyzed graphically.

3.7 Results and Discussions: Inductive Methodology

In sections 3.5 and 3.4, the delineated methodologies are the initial researches in this framework development. The framework becomes a comprehensive tool when the inductive embedding is incorporated. So that, the results in this section present the main observations of this chapter. The case studies were conducted on gate-level circuits of the 10-Gigabit Ethernet MAC [124] and the openMSP430 [132] cores. OpenMSP430 is a 16-bit microcontroller core compatible with MSP430 family of Texas Instruments (TI) [133]. Gate-level circuits of both cores synthesized with a 45nm NanGate Open Cell Library.

3.7.1 Prediction of SEUs Caused FFRs - 10GE MAC

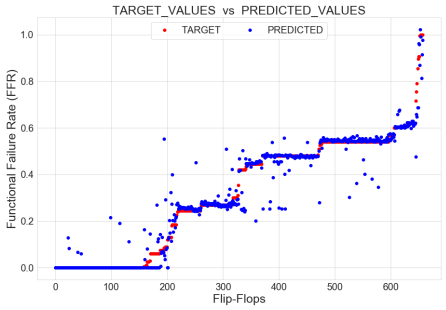


Figure 3.30 – Mean Absolute Error (MAE)= 0.0191 and R^2= 0.95 with a test size of 60% [3]

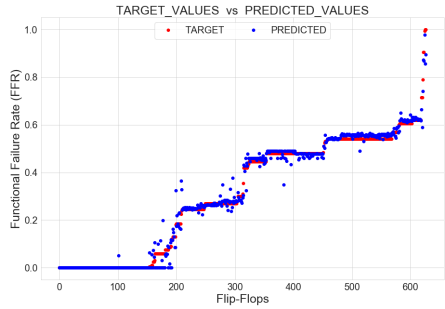


Figure 3.31 – Role of Graph Mining Generated Initial Raw-Database in Prediction (R^2 increases to 97%)

In Fig. 3.30, the red color represents the Functional Failure Rates (FFRs) as given in (2.3), which were empirically derived by the exhaustive fault-injection campaign. The corresponding predicted FFRs were shown in blue color. The graphical comparison visualizes how well the prediction replicates the observed reference database. In this case, the DNN prediction achieves the coefficient of determination (R^2) [134] value of approximately 0.95, where the best model fit value of R^2 metric is 1, and the worst value is 0. In statistics, the R-squared (R^2) value is the measure of goodness-of-fit of a regression model and, the projected R-squared value (0.95) able to explain most of the variations in the reference data. Table 3.7 outlines the impacts of accelerated predictions in simulation time requirements. GraphSAGE and DNN based ensemble algorithm provides a significant reduction in the required test resources without compromising the quality of modeling. However, the implemented algorithm depends on 40% of the fault-injection database for training the downstream DNN. But, it is quite impressive to note that the test and training phase of the whole algorithm takes only less than 10 minutes. Fig. 3.30 plotted 60% (660) of total flip-flops (1100) against their functional failure rates.

The initial framework in the prior work [3] uses node2vec (Random walk method) generated raw data as an input to the graphSAGE algorithm. However, in the improved stage of this work, the graph mining technique provides initial data to the graphSAGE algorithm. This improved adaptation saves unintended graph processing time and the leads to an in-

Table 3.7 – IMPACTS OF SEU PREDICTION IN SIMULATION RESOURCES [3]

Model	Time	Tool	Model Fit (R^2)
Exhaustive - FI	17 hours	7 Modelsim	Target Model
Exhaustive - FI	$\approx$ 5 days	1 Modelsim	Target Model
GraphSAGE + DNN (Node2vec Based Initial Raw Data)	$<$ 10 minutes + (40% FI-Time)	1 Modelsim	0.95
GraphSAGE + DNN (Graph Mining Based Initial Raw Data)	$<$ 5 minutes + (40% FI-Time)	1 Modelsim	0.97

crement in R^2 metric from 95% to 97% (Fig. 3.31 and Table 3.7). The transductive processing nature of the node2vec algorithm is the reason for the unintended graph processing time. The previous work relies on the node2vec algorithm for initial raw data because initial input data is an inevitable factor for the optimization process in the graphSAGE algorithm. Such an approach makes the previous work's processing timing heavily dependent on the complexity and size of the circuit-garph. But here, more meaningful initial data is given to the graphSAGE algorithm through the graph mining process. Table 3.6 points out its importance,and equations from (3.10) to (3.19) brief the way that calculates the initial data.

3.7.2 Importance of 40% Training data and Initial Raw Data

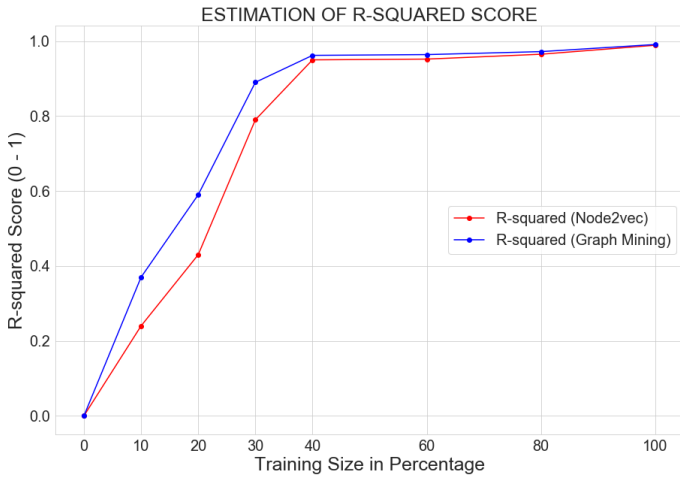


Figure 3.32 – Variation of R-Squared Score with Different Training Sizes

The experiments with different training sizes have led to the conclusion that the training size is better to be 40% for predictions. Fig. 3.32 plotted that observations. The works in [135] exemplified the potential of the algorithm (graphSAGE + DNN) to achieve high accuracy at small training sizes for different data-sets from different fields. Here, the training size of 40% provides a significantly higher correlation between predicted and reference values, compared to other training sizes of less than 40%. The correlation property is

slightly improving with above 40% training sizes. Note, the test packets that were transmitted and received by XGE MAC were chosen randomly for this experiment to imitate the realistic scenario. The input packets were not optimized for the best fault coverage. The Fig.3.32 points out the significant contributions of the initial raw data by the graph/-data mining process. There are two lines where line in red indicate the initialized vector for each node by a node2vec algorithm and apply to the graphSAGE algorithm and, the blue line represents the initial raw data by graph mining to the graphSAGE as explained in 3.6. From Fig 3.32, it is clear that experiments with the graph mining generated initial raw data, achieved more precision in prediction at smaller training sizes. Fig 3.32 explicitly indicates that the choice of initial raw data by graph mining is more relevant in failure prediction when compared to the node2vec algorithm (or) other random initial vectors.

3.7.3 Inference Quality Inconsistency

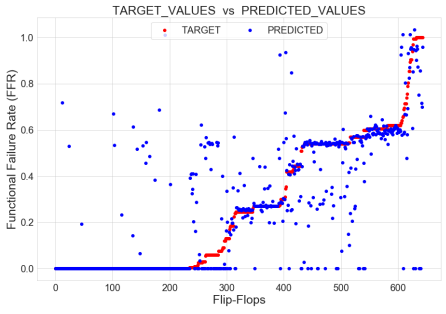


Figure 3.33 – Chance of a 40% Training data in Prediction Quality Deterioration (R^2 Drops to 85%)

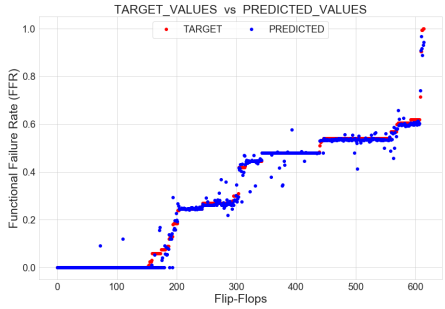


Figure 3.34 – Role of Cluster Based 40% Training Data in Prediction (R^2 remians at 97%)

DNN is the relevant part of the proposed algorithm. From the perspective of the algorithmic level, the 40% training data has a crucial role in predicting functional failure probability due to SEU/SET faults. In Fig.3.30, DNN prediction is shown for carefully chosen training data, which implicitly means that the flip-flops are chosen with failure probability range from 0 to 1 and almost covering the discreet parts of the overall failure probability distribution of the Ethernet MAC circuit. The decision about how to choose the training data stands as the prevailing challenge in modeling complex distributions like failure vulnerability of flip-flops. Because a random selection of 40% from failure probability data after fault injection campaign is not necessarily fetching the required flip-flops for modeling the circuit's overall failure probability. An example is demonstrated in Fig. 3.33, where the training data does not include enough flip-flops with high failure probabilities. This causes a drop in the prediction quality as provided in Fig.3.33, where the predicated failure probabilities are more or less scattered around the reference model (in red color).

The clustering method (Algorithm 2) is a procuring way to group the flip-flops such that the flip-flop group covers the complete failure potability distribution due to SEU/SET faults in the circuit. After that, samples from each group are used for training purposes. The result of such an approach in prediction is delineated in Fig. 3.34. Here, a combination of two algorithms, Hierarchical and, K-means produce valid clusters for this application. Here, the Hierarchical algorithms typically cluster the data with distances and generate a taxonomy of the data. This taxonomy is nothing but clusters of embedded data after the graphSAGE algorithm. From these clusters, we find initial clusters and mean values

that the k-mean algorithm starts with. In the k-means algorithm, the quantified objective function depends on the sum of the squares of the Euclidean distances of data points to their closest representatives as provided in (3.21).

Algorithm 2: Node Clustering Algorithm

Result: Clusters of Graph Nodes
Input: GraphSAGE Embedded Vectors
Format the nodes' embedded vectors as a matrix ;
Initialize number of clusters (N) (e.g., N=10);
for $i \leftarrow 1$ **to** N **do**
    Evaluate: Silhouette Coefficient for i number of clusters
    Update: Silhouette Coefficient
end
Estimate: The optimal number of clusters with large Silhouette Coefficient;
Apply: The Hierarchical (Agglomerative merging) Algorithm for optimal clusters;
Find the centres of clusters;
Initialize the cluster centres as the initial points for K-means algorithm;
Apply: The K-means algorithm for optimal clusters

$$Dist(X,Y) = \|X - Y\|_{L_2}^2 \tag{3.21}$$

where: X and Y are two data points and L_2 represents the Euclidean distances between them. Fig. 3.37 and Fig. 3.38 illustrates the resultant clusters of data. In Fig. 3.35, a classical metric is presented to achieve the clustering method on the 50-dimensional data that is generated by the graphSAGE algorithm. The metric is called the silhouette coefficient that calculates the goodness of the clustering technique as provided in (3.22).

$$\text{Silhouette Coefficient} = \frac{I_{er} - I_{ra}}{max\,(I_{er}, I_{ra})} \tag{3.22}$$

where, I_{er} is the average inter-cluster distance and I_{ra} is the average intra-cluster distance. Intra-cluster distance stands for the diatnce between the points within the cluster while Inter-cluster distance stands for the distance between the points from different clusters. The Silhouette Coefficient values ranging from -1 to 1, where 1 means the clusters are well distinguished, 0 means the clusters are not distinguishable and -1 means the points in the clusters are wrongly assigned. Fig. 3.35 is the plot of Silhouette Coefficient that is calculated for different number of clusters on the data after graphSAGE algorithm. This repeated experiment concludes that the optimal number of distinguishable clusters in the data is 4. Because a maximum value of 0.84 for the Silhouette Coefficient is attained corresponding to cluster number 4. It is picturized in Fig. 3.35. The 4 optimal and distinguishable clusters are plotted in Fig. 3.37. Fig: 3.37 is a 2 dimensional plot of clustered data in 50 dimension. The x-axis (Dim1) and y-axis (Dim2) in Fig. 3.37 are the two principal components (PC1 and PC2) as a result of Principal Component Analysis (PCA) [97] of the data after the graphSAGE algorithm. Even though 2 dimensional interpretation of the clustered data is not 100% in detail. For example, the clusters 3 and 1 are not well separated in Fig. 3.37, but at the same time, the clusters 3 and 1 are well separated in the 3D view in Fig. 3.38. For plotting Fig. 3.38, three principal components (PC1, PC2 and PC3) are specifically used.

Fig. 3.36 is a visual explanation for the necessity of clustreing and how the clustering affects the training phase of the DNN part of the prediction algorithm. The Confidence

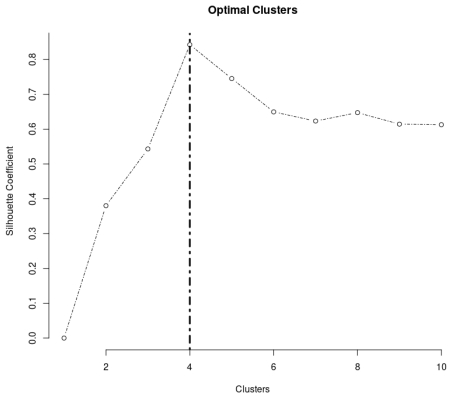


Figure 3.35 – Silhouette Analysis for Optimal Clusters

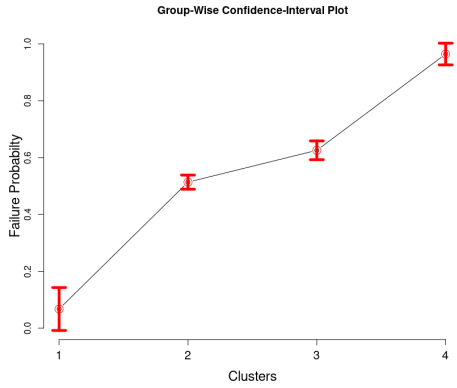


Figure 3.36 – Confidence Interval (CI) Plot of Clusters

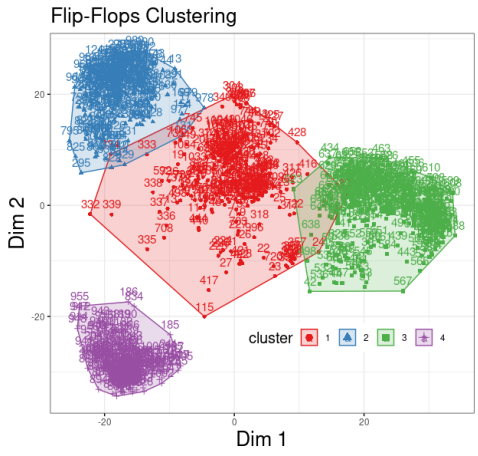


Figure 3.37 – 2D-plot of Clustering

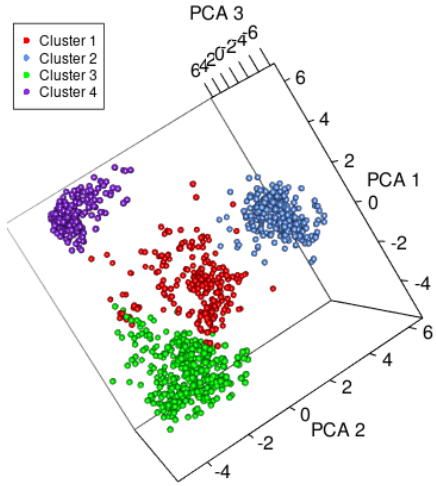


Figure 3.38 – 3D-plot of Clustering

Interval (CI) of the functional failure probability of the flip-flops in the cluster are plotted in Fig. 3.36. The vertical lines in red color represent CI, where the middle part is the mean FFR_{seu} of the respective clusters. Similarly, the upper part is the positive standard deviation, and the lower part is the negative standard deviation from the mean. In this plot, it is trying to say that clustering covers the overall failure probability distribution (Target Model in Fig. 3.30) that is obtained by the fault injection campaign. If we sample 40% data from the FI-campaign for the training phase of DNN, the sampled domain contains flip-flops of versatile soft-error vulnerability. This information dominate in the prediction accuracy of the DNN after graphSAGE algorithm. The clustering quality is devastated if the clusters show considerable overlap between Standard Deviations (SDs) (red color symbols in Fig. 3.36) along the y-axis. Contrary to that, flip-flop's clustering generates the clusters with low Standard Deviation (SD) in failure probabilities in this work. It is a requisite observation from Fig. 3.36 because both the input data features from Table 3.6, and then the graphSAGE algorithm together substantiated the accomplishment of the principal aim of this work. The principal aim is provided in (3.9).

3.7.4 Random Fault-Injection Vs Prediction - 10GE MAC

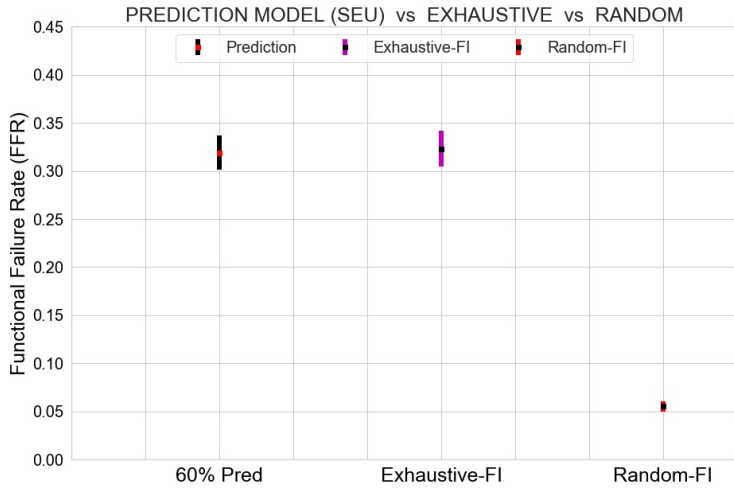


Figure 3.39 – 95% Confidence Interval Comparison: Prediction = 0.318 ± 0.0176, *Exhaustive* = 0.323 ± 0.018 *and Random* = 0.055 ± 0.005

Fig. 3.39, represents the mean value and 95% confidence interval of the random fault-injection results, where the fault was injected randomly at 40 % of the total number of clock cycles in the circuit operation. The random fault-injection data failed to model the reference data, but it reduces the fault injection time by 60% compared to the exhaustive one. In random fault injection, there are a lot of things to consider. Among them, choosing clock cycles for the fault injection is the main concern. A standardized way for performing the random fault-injection method is not documented well yet. Here, the clock cycles for random fault injection are chosen randomly. When we compare the results between the prediction model of SEU and the random fault-injection model, it is quite evident that the prediction model achieves better and relevant statistical accuracy. Fig. 3.39 emphasizes the importance of prediction and provides a visual comparison between the prediction, exhaustive-FI, and random-FI models for 95% confidence intervals.

3.7.5 Inference of SETs Caused FFRs - 10GE MAC

The Functional Failure Rates (FFRs) due to SET events (2.5) were also empirically derived by an exhaustive fault-injection campaign. Here, a transient digital pulse of a width 200 pico-seconds models a SET event in the digital circuit. All the timing factors have been simulated with the Standard Delay Format (SDF) file. The chance of a SET fault to propagate and affects the functional behavior of the circuit is low compared to that of the SEU event. In this case study, a large number of gates appear with zero failure probabilities (approximately 84% of total gates). Only 16% of the total gates in number caused the functional failure of the circuit. Fig. 3.40 plotted a prediction over 60% (1951) of total number of gates (3252). However, in Fig. 3.40, out of 1951 data points, 1606 data points are shown as hidden because those hidden points are corresponding to zero functional failures and also want to enlarge the interestingly noticeable part of the graph. But, the R^2

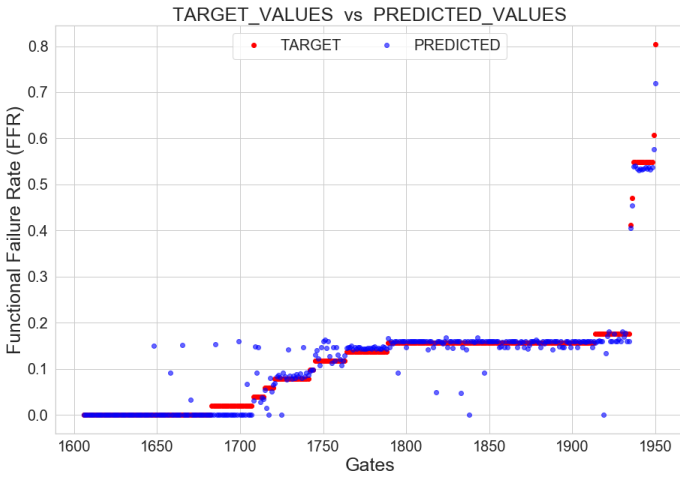


Figure 3.40 – SET - Functional Failure Rate prediction (Test-size = 60%)

value is calculated for the entire predicted FFRs of gates without omitting any gates. Here, the algorithm is trying to model a complex mathematical equation that was given in (2.5). Also, the number of handling data points is increased from 1100 flip-flops to 3252 gates in the prediction process. It does require additional time, but it is insignificant because the required algorithmic running time raises only by 2 minutes compared to SEU-caused failure rates prediction (from 5 to 7 minutes). Such characteristic predominantly outlines the scalable property of the algorithm. The Prediction achieves a very significant accuracy of R^2 = 0.941. Table 3.8 outlines the impact of SET-failure rate prediction that reduces the simulation time by 60%.

Table 3.8 – IMPACTS OF SET PREDICTION IN SIMULATION RESOURCES

Model	Time	Tool	Model Fit (R^2)
Exhaustive - FI	55 hours	7 Modelsim	Target Model
Exhaustive - FI	$\approx$ 16 days	1 Modelsim	Target Model
GraphSAGE + DNN (Test + Training)	$<$ 7 minutes + 40% FI-Time	1 Modelsim	0.941

3.7.6 Prediction of SEUs Caused FFRs - openMSP430

In this section, we have presented the second case study. Here, the proposed framework estimates the SEU based functional failure rates for the 16-bit RISC CPU model with the different applications as workloads. Fig. 3.41 and 3.42 provide the comparison between predicted and reference functional failure data for two applications. The first application (Fig. 3.41) was written in assembly language to test 'MOV' instruction with all addressing modes. The second application (Fig. 3.42) is the 'sandbox' benchmark that was written in C/C++ language. The sandbox represents a testing environment to perform some tasks. Both Fig. 3.41 and 3.42, provide a prediction over 60% (447) of the total number of flip-

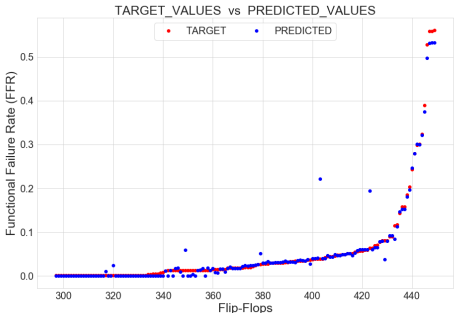


Figure 3.41 – SEU Caused FFR Prediction (MOV Application)

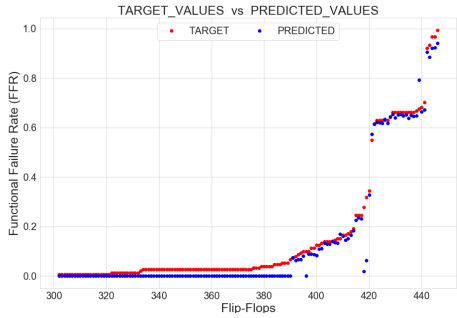


Figure 3.42 – SEU Caused FFR Prediction (Sandbox Application)

flops (745). For highlighting the important observations in predictions, 297 data points with zero failure probabilities are hidden in Fig. 3.41, and 302 data points with zero failure probabilities are hidden in Fig. 3.42. But, the entire 60% predictions used for calculating the R^2 value in both cases. The prediction for the assembly language application achieves R^2 = 0.965 and that for the C/C++ language application achieves R^2 = 0.97. The 40 % training size was chosen based on a experiment of repeating the training validation for different training sizes. Fig. 3.32 explains a very similar experiment. The point which matters here is the capability of framework to predict the SEU based functional failures for different applications with a training size of 40%. This clearly outlines that required fault-injection time of openMSP430 micro-controller is reduced by 60% with an accepted level of variations in statistical results. Table 3.9 shows the impacts of prediction and also list the results of an additional application that tests the performance of multiplier hardware unit.

Table 3.9 – SEU PREDICTION FOR DIFFERENT APPLICATIONS

	Applications		
Parameters	MOV	Sandbox	multiplier
R^2	0.965	0.97	0.96
Training & prediction	$<$ 5 mins	$<$ 5 mins	$<$ 5 mins
Training Data (40%)	$\approx$ 18 hrs	2 days	$\approx$ 18 hrs
Full Database (FI)	$\approx$ 44 hrs	5 days	$\approx$ 44 hrs

3.7.7 Probabilistic Static Reasoning and Comparisons

In all the above results and discussions, the assessment of prediction quality of the AI-Framework exploits the fault-injection simulation environment for the detailed analysis. But the static analysis, which comprises the different derating factors (LDR, TDR, EDR and FDR) from the SOTA also contributes simulation time overhead significantly. However, the metric inference quality of static analysis is comparatively in doubt and it is important to check the quality of framework prediction with static analysis. Fig. 3.43, Fig. 3.44, and Fig. 3.45 provide such types of comparisons for a given test circuit, 10-Gigabit Ethernet MAC. Figures provide the analysis over Functional Failure Rate FFR for 903 flip-flops of the test circuit. The static analysis estimates the LDR,EDR, FDR and TDR independently over all the given flip-flops and sums according to (2.3) and (2.4).

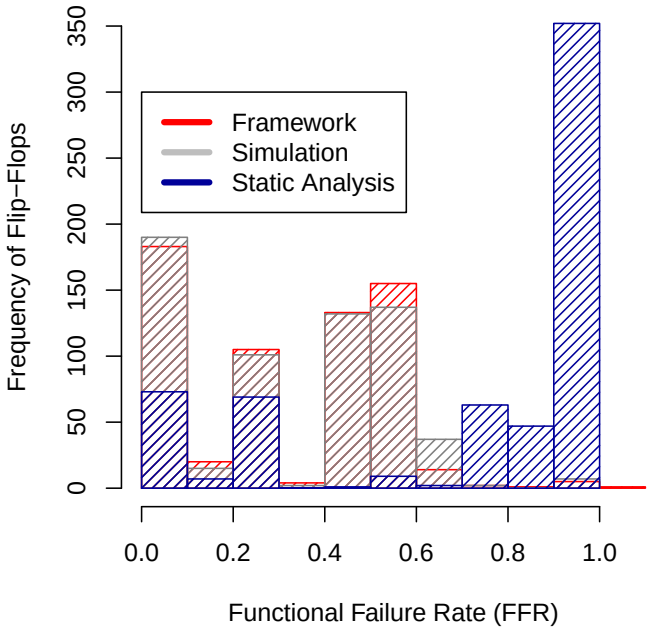


Figure 3.43 – Histogram Distribution

The method of static analysis is detailed in section 2.8. Fig. 3.43 compares the estimated FFR of flip-flops from AI-Framework and from static analysis with fault-injection simulation in terms of Histograms. It quite conveniently concludes that the static analysis method lags far behind the developed framework for metric estimation quality. Similarly Fig. 3.44 and Fig. 3.45 shows flip-flops' frequency density metrics and box-plot metric respectively. From both figures (Fig. 3.44 and Fig. 3.45), it is clear that static analysis finds a large group of flip-flops corresponding to a high failure venerability rate, indeed. These findings indicate that the mathematical models from SOTA fail to model the different de-

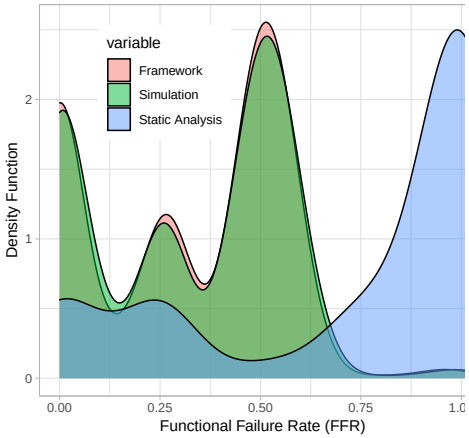


Figure 3.44 – Density Function Plot

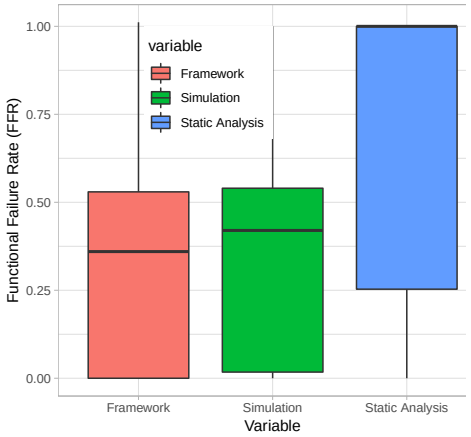


Figure 3.45 – Box Plot

rating factors as in real-time operation. The insignificant SOTA models for fault propagation probability and incapability to hold sufficient scalability presses the significance of a framework that the thesis aims to develop. All these comparisons and reasoning conclude that AI-Framework outperforms the static analysis method in quality of metrics estimation and provides a much superior inference that is as close to the metrics of real-time fault-injection campaigns.

3.8 Chapter Conclusions

An accelerated testing methodology that is scalable and very cost-effective in simulation resource handling has been developed for medium and large scale circuits. The comprehensive results that were explicated here provide a plausible prediction of functional failures due to SEU and SET events. Both case studies contain a graph with less than 5000 nodes (flip-flops and gates). This algorithm can process a graph with approximately 10,000 nodes without any additional running time on a CPU machine with 16GB RAM. In the future, the computational limitations of the algorithm in this scope will be investigated with further experiments. However, the ratio between the number of failure-vulnerable nodes to that of non-failure-vulnerable nodes is an important factor. If this ratio is too small, then training data will be selected such that the algorithm can learn the overall distribution of target probabilities. The selection of training data and its corresponding concerns is solved to an acceptable extent by using clustering techniques. The holistic approach provides a new dimension to the EDA frameworks through Artificial Intelligence.

Chapter 4

Integration of Safety, Quality and Reliability

The key content of this chapter is based on the publications [VI] and [VII]. The chapter intends to provide a concrete idea about the importance of ensuring system quality to a standard level. The core of this chapter relies on an AI-based algorithm which is published in [VII]. The article [VII] covers 90% of this chapter, while article [VI] covers 10% of this chapter. The research in this thesis extends the already developed EDA tool/algorithm to validate the effects of single stuck-at faults at the functional level of system circuits. A fundamental contributor to the quality of an autonomous system is the quality of the underlying implementation technology. The manufacturing process must present a well-characterized, preferably low intrinsic defect and fault rate, and a good resiliency to environmental challenges like well-known aging and degradation performance. In this way, quantity and quality, safety, and reliability are plausible to integrate harmoniously.

4.1 Preamble of Chapter

Fig. 4.1 delineates the importance of testability and yield of a system design for manufacturability. The importance of a comprehensive tool that can used to evaluate both the soft-error reliability as well as the quality of a system from the perspective of testabilty, is highly emphasized in the work [6]. These facts lead the researchers in the reliability and testing fields to develop mathematical algorithms to reduce the time overhead of fault-simulations. Also, in the last decade of years, there is no much sophisticated-literature that combines the simple stuck-at model and advanced mathematical algorithms for fast inference of functional failures due to single stuck-at faults. All these key factors contribute to the extension of the developed Artificial Intelligence (AI) based framework (focused on soft-error reliability) to evaluate the single stuck-at fault's effects at the functional level.

The application of AI to extract feature information from the circuit-graph experiments different graph node embeddings. Algorithms like GCN and graphSAGE leverage a node's features into a vector form. From the journal publication [6], it is motivated to develop the multi-dimensional functional verification potential to the existing framework for soft-error reliability analysis.

The transformed gate-level graph is the source for extracting the features of nets to model the effects of stuck-at faults at the functional level. The proposed framework goes beyond the classical machine learning algorithms (like support vector machine, logistic

regression, and linear regression) by replacing the black-box modeling with transparent modeling of the metrics (i.e., white-box modeling [24]). The argument of white-box modeling is valid in the sense that the AI neural networks are applying to real circuit models.

4.2 Quality Testing for Manufacturabilty

Advanced high-quality chips endorse challenging scenarios in developing industrial specifications by including standard quality metrics. A lower DPPM (Defective Parts per Million) level is an example of a highly demanded test metric by the industrial end-users. Such requirements imply elaborated production tests that generally include single stuck-at faults, transitional faults, path delays, bridging effects, and cross-talks. The functional fault coverage is applied as a standard metric for expressing the quality of test patterns. To characterize the stuck-at fault coverage, a valid set of test patterns, and an efficient fault simulation approach are needed [136].

The fault diagnosis approaches locate the faults that cause functional failures in the circuit. The prior works [137], [138], [139], and [140] significantly use the single stuck-at faults models to perform the fault diagnosis. The fault simulation method has been chosen in these papers to infer a statical observation by comparing the performance between the faulty circuit and fault-free circuit. The single stuck-at fault diagnosis efforts also extended scientifically to solve the problems of multiple stuck-at faults with limitations. The works that have been explicated in [141], [142], [143], and [144] tried to reduce the fault-space for multiple suck-at fault diagnosis through different concepts. Effect-cause analysis and guided probing belong to such fault space deduction approaches. Electron-beam probing as a fault diagnosis approach was applied in work [145], but electron-beam probing is a time-consuming and laborious task. Research papers [146] and [147] deduced the suspected faults by algorithmically generated sensitizing input pairs without probing internal nets. The works that have been provided in [148], [149], and [150], tried to utilize the stuck-at fault model for the analysis of bridging faults. Also, in work [151], both single and multiple fault simulations were applied together for the fault space reduction technique to ease the problems of fault diagnosis. In all these proposed approaches, fault simulations and its running time are inevitable factors. The single stuck-at fault model and its simulation within their premises have been explored in fault diagnosis with different perceptions. The fault-space reduction algorithms in the cited works depend heavily on the stuck-at fault simulations. As a result, the aggressive chip density scaling jeopardized the simulation-based fault diagnosis methodologies.

4.3 Reliability Perspective of Stuck-at Faults

The dominant threats for reliability are, first, random hardware faults such as transient faults by radiation-induced single event effects or soft errors [152], i.e. a subject for Soft-Error Reliability (SER). Second, these are extreme operating conditions, electronic interference and intermittent to permanent faults by process or time-dependent variations, such as aging induced by Bias Temperature Instability (BTI) [153], and it falls in the subject for Life-Time Reliability (LTR). So an extended and scalable testing algorithm scheme that addresses stuck-at faults in a Circuit Under Test (CUT) links to improve yield and reliability. The requirement of an online algorithm that detects possible failure probability due to the stuck-at faults that remain unidentified in the manufacturing phase (or) reincarnated due to harsh environmental properties is inevitable.

4.4 Understanding Multidimensional Functional Verification

The main contribution of the paper [VI] is a taxonomy for multidimensional hardware verification aspects, a state-of-the-art survey of related research works and trends. In [VI],

X. Lai, A. Balakrishnan, and T. Lange et al. articulate the importance of considering various extra-functional aspects of the electronic systems' design at the chip design level as a result of trends of giga-scale integration at nanoscale technology nodes and multi-/many processor-based systems-on-chip architectures. The discussions include the aspects of security, reliability, timing, power consumption, etc. In the hardware part, these are design errors (bugs), manufacturing defects and variations, reliability issues, such as soft errors and aging faults, or malicious faults, such as security attacks.

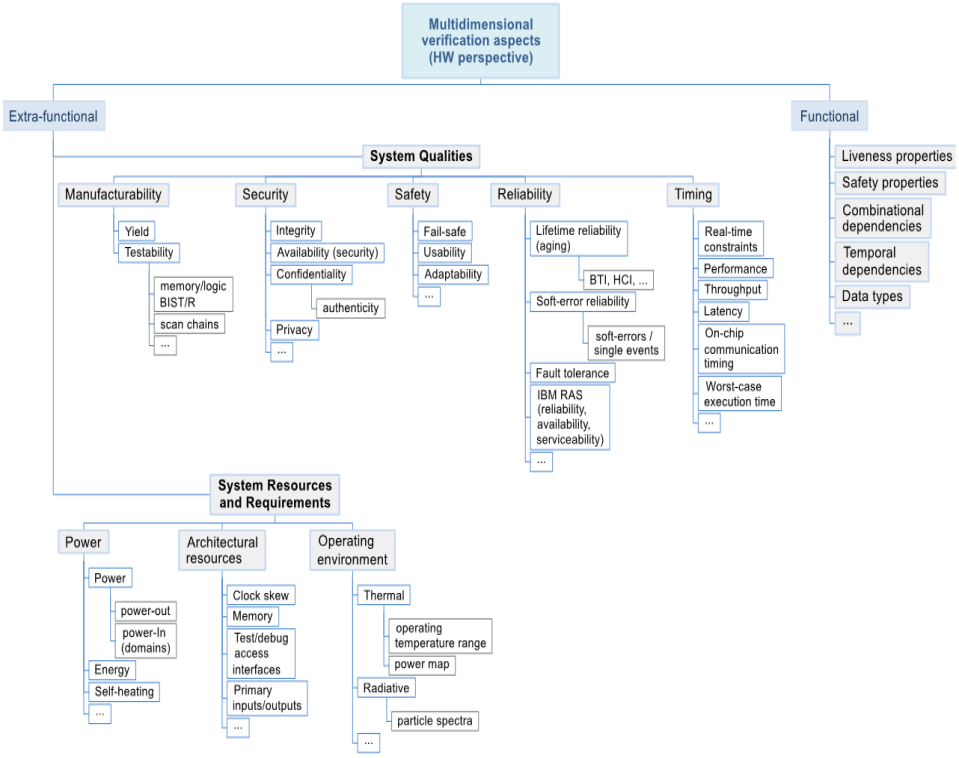


Figure 4.1 – Taxonomy of Multidimensional Verification Aspects [6]

Fig. 4.1 provides the details of different functional and extra-functional (interchangeably referred to as non-functional) aspects of an electronic system under verification. Hardware design model verification detects design errors affecting both functional and extra-functional aspects. The principal task of extra-functional verification of a design model is limited to detecting deviations that cause violation of extra-functional requirements. In practice, it often intersects with the task of functional verification [154], [155], thus establishing a multidimensional space for verification. A "grey area" in the distinction between functional and extra-functional requirements may appear when an extra-functional requirement is a part of the design's main functionality. E.g., security requirements for some HW designs can be split into extra-functional and functional sets if both design's purpose and specified functionality are falls in the system's security aspect. A secure cryptoprocessor is an example of this.

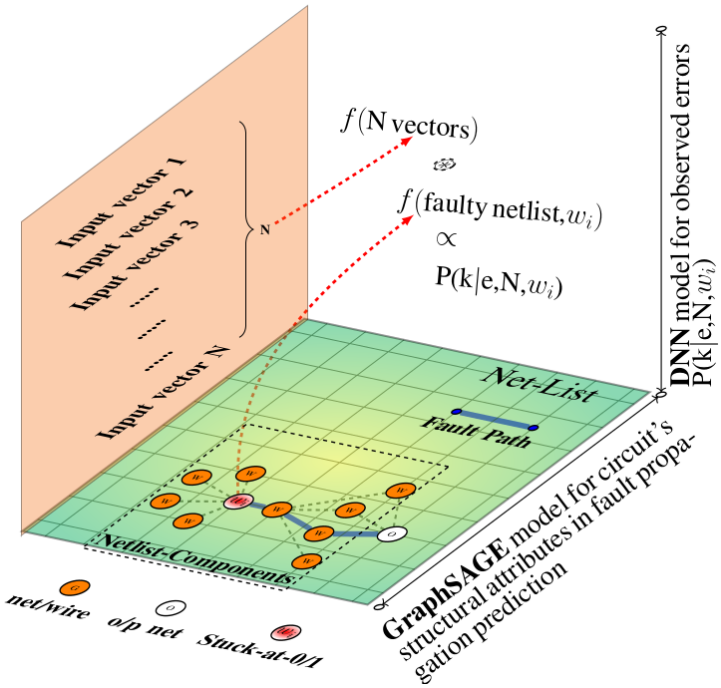


Figure 4.2 – Modeling of Functional Failure Probability [7]

4.5 Principle Aim and Contexts

Fig. 4.2 provides a visualization of different algorithms that were placed to model the error-observation probability at the functional level. The graphSAGE algorithm is the introduced graph embedding neural network that deduces the hidden information corresponding to a net (w_i) from the netlist, where the netlist is a probabilistic graph of nets and components as vertices and edges. The explicated information about a net (w_i) is the structural peculiarities that decide the chance of an injected fault at w_i to cause functional failures. Simultaneously, the injected fault at w_i for each input vector leads to a binomial type failure probability distribution [156]. P(k|e,N,w_i) explicitly models that failure distribution for N input vectors, where k represents total failure outcomes (e) in number for N injected stuck-at faults at w_i. The whole modelling-approch in Fig. 4.2 can be explained as the convolution ($\circledast$) between $f(N\text{vectors})$ and $f(\text{faulty netlist}, w_i)$, and that convolution is proportional to P(k|e,n,w_i) as given in (4.1).

$$f(N\text{vectors}) \circledast f(\text{faulty netlist}, w_i) \propto P(k|e,n,w_1) \tag{4.1}$$

4.5.1 Graph Transformation Principle

The line graph LG is an edge-to-vertex (or) edge-to-node dual representation of graph G. The graph transformation is obtained by associating the vertex of LG with the edge of the graph G. The two vertices of LG are connected with an edge if and only if the corresponding G-edges of that two vertices have a common vertex [157]. A simple and classical example of graph transformation of G into a line graph LG is illustrated in Fig.4.3. When G represents a gate-level circuit as in (3.8), then LG epitomizes the dual form of the gate-level, where nets are the nodes.

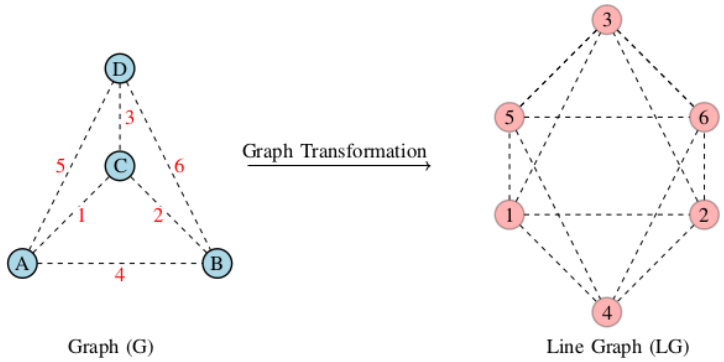


Figure 4.3 – Edge-to-Vertex Transformation of a Graph (G) [7]

4.6 Methodology

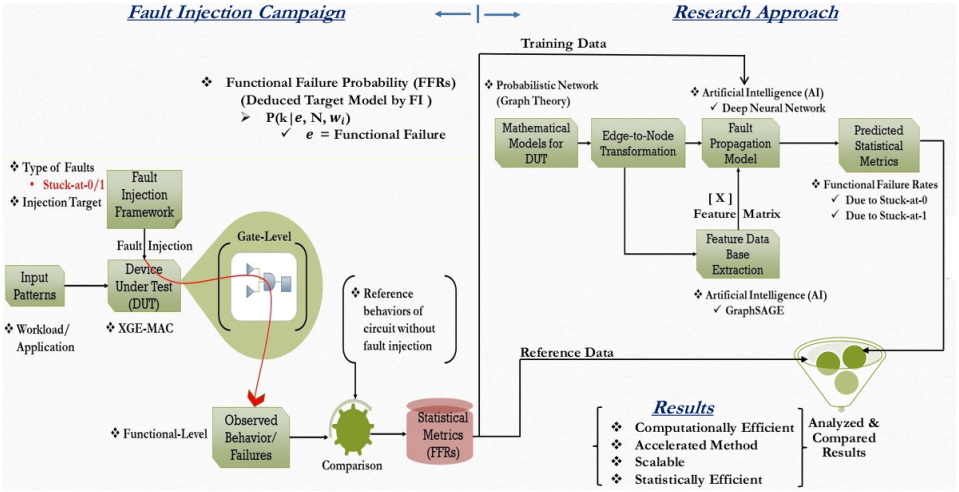


Figure 4.4 – A Systematic Workflow Diagram [7]

Fig. 4.4 chronicled the whole research approach in block-diagrams, and it includes two parts: the fault-injection campaign and the research approach. In the first part (fault-injection campaign), the raw database (FFRs) for the fault propagation analysis is generated. The Fault-Injection campaign and its details are acknowledged below. In the second subsection, more detailed view of the AI model is provided.

4.6.1 Fault-Injection Simulation

At the gate-level, the functional failure modeling of stuck-at faults is further advancing to accelerate the failure evaluation effectively. We have statistically simulated the fault injections and, empirically derived the Functional Failure Rate (FFR) factor of each net (gate-level wire) for stuck-at-1 and suck-at-0 faults. Finally, the fault coverage of the given set of 64 input test vectors (64 transmitted packets) is calculated from the observed results of fault injection campaign. The proposed algorithm is used to predict the empirically derived metrics. In this scenario, stuck-at-1 and stuck-at-0 fault injections at gate-level

have been achieved through modifying the data of each net to logic-1 and logic-0, respectively. The injected value acts as a permanent fault for the entire operation. In total, 3437 nets from different blocks of the circuit (such as TX, RX, Wishbone Interface, Fault State-machine, and Sync_clk), were tested. A high-level representation of the campaign has been shown in Fig.4.5. Fig.4.5 clearly shows that, at the start of simulation (at time t_1), an single stuck-at fault (stuck-at-0/stuck-at-1) applies to the net that is represented as 1. And, repeated the process with N input vectors. An example in Fig.4.5 depicts that if the circuit failed to deliver the required function for input vectors out of N, then the FFR metric is $\frac{2}{N}$. Equations (4.2) and (4.3) express the Functional Failure Rate factor of each net with respect to stuck-at-0 and stuck-at-1 fault, respectively.

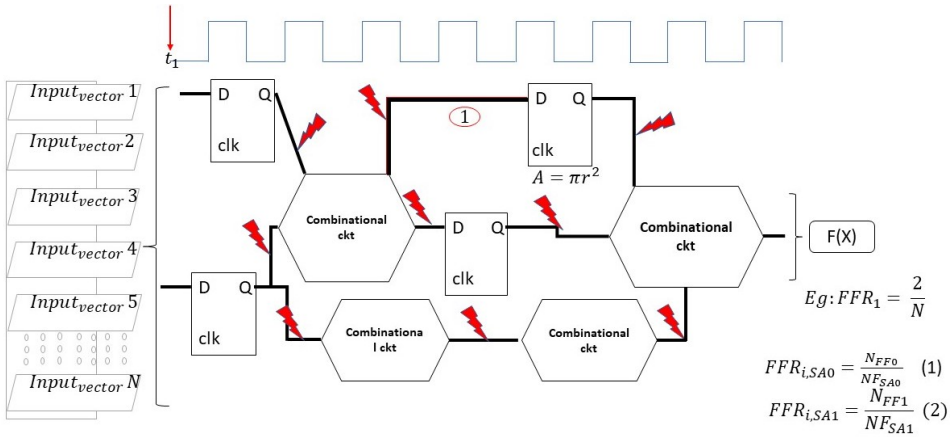


Figure 4.5 – Single Stuck-at-Fault Injection Campaign

$$FFR_{i,SA_0} = \frac{N_{FF_0}}{NF_{SA_0}} \tag{4.2}$$

$$FFR_{i,SA_1} = \frac{N_{FF_1}}{NF_{SA_1}} \tag{4.3}$$

In (4.2) and (4.3), i indicates each net. N_{FF_0} (or) N_{FF_1} represent the number of functional failures due to stuck-at-0 (or) stuck-at-1 faults. Similarly, NF_{SA_0} and NF_{SA_1} counts the maximum number of injected stuck-at-0 and stuck-at-1 faults per net. The maximum of NF_{SA_0} (or) NF_{SA_1} is simply equivalent to the number of input test patterns (i.e., 64 packets). The significance in evaluating the fault coverage of the test patterns is inevitable [158]. The equation (4.4) defines the Functional Fault Coverage (FFC) metric.

$$FFC = \frac{\text{Total Faults Detected as Functional Failures}}{\text{Total Injected Faults}} \tag{4.4}$$

The FFC is a metric that decides the quality of test pattern in producing a prominent subset of all possible functional failures without concerning about 100% possible failures. With the advent of small-scale technology integration in the electronics-chips, a more reliable test pattern according to single stuck-at fault models becomes a desideratum. In this experiment, the quality of the algorithm in estimating the FFC metric is also evaluated.

4.6.2 Research Approach: The AI Model

In the second part (Research Approach), the graph theory and Deep Learning (DL) are applied to estimate the fault-injection inference. The research approach fundamentally exploits FI database to characterize the fault propagation model. The research approach part can be viewed as four phases.

- **The Probabilistic Network Generation**
  After the fault-injection campaign, a probabilistic graph G is generated, which represents the gate-level netlist. A Verilog Procedural Interface (VPI) library function was linked to a standard simulation tool (ModelSim) to design the graph G. The graph G includes vertices that imply the gates and flip-flops in the netlist. Then, an edge-to-vertex transformation was applied to graph G. The resulting line graph LG contains the vertices that analogous to the nets/wires in the netlist. Fig. 4.6 shows those steps in order. The line graph LG was used in the successive algorithm (graphSAGE) to generate the feature matrix of nets for a downstream DNN prediction.

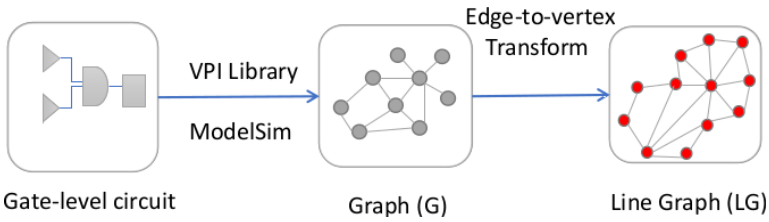


Figure 4.6 – Development of a Line Graph from a Netlist [7]

- **GraphSAGE Embedding Algorithm**
  In the second phase, the grpahSAGE algorithm extracts a feature matrix (X) corresponding to the line graph LG nodes. The graphSAGE includes two principal steps. The first step is the sampler algorithm. The sampler algorithm defines the neighborhood space of a source node. In this scenario, we defined the parameter $K = 2$, which means that the sampler samples up to the depth of the 2 neighbor space. In the second step of the graphSAGE algorithm, an aggregator is implemented at each depth ($1 \leq k \leq K$), that aggregates features from the local neighborhood of the source node. The aggregators were visualized in Fig. 4.7, where blue and green lines indicate the aggregators at depth $k = 1$ and $k = 2$ respectively. Here, the max-pooling aggregators were implemented. Equation ((4.5)) formulates the mathematical abstraction of the max-pooling aggregator [128].

$$AGGRE_k^{pool} = max(\{\sigma(W_{pool} h_{u_i}^k + b), \forall u_i \in N_k(v)\}) \tag{4.5}$$

  where (4.5) represents the aggregator function at depth k, which is a graph neural network with parameters W_{pool} and b. An unsupervised learning method has optimized those parameters. $N_k(v)$ represents k-neighborhood of vertex v and $h_{u_i}^k$ indicates the aggregated neighborhood vector and, σ is the activation function of the neural network.

  In the third step, each node is reformed into a vector. All three steps repeat for all the nodes in the line graph and produce a matrix representation (X) of the circuit. Fig. 4.7 illustrates the node-to-vector mapping (node embedding).

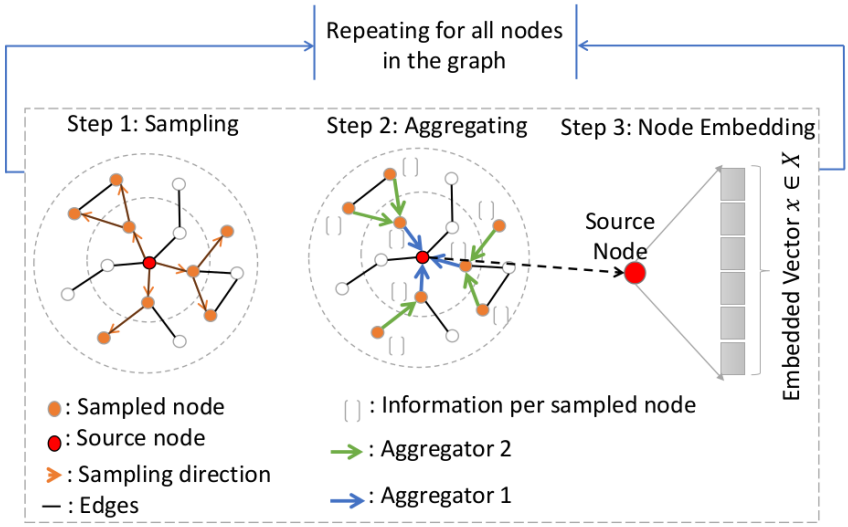


Figure 4.7 – GraphSAGE Algorithm [7]

- **DNN Inference Method**

Fig. 4.8 outlines the DNN algorithm that was exercised for prediction purposes. There are two parts in Fig. 4.8. The first part is the training part of DNN, and the second one is the testing part of DNN. In the training part, 40 % from the feature database X and corresponding target probability metric from FI-database have been randomly selected as x_{train} and y_{train}. The chosen resources postulate a hypothesis that best describes the target probability distributions (FFR_{i,SA_0} and FFR_{i,SA_1}) through the supervised learning method. The optimized parameters (Weights

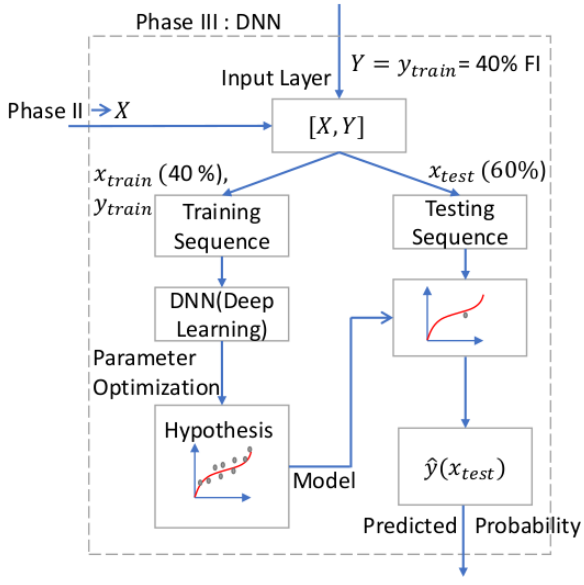


Figure 4.8 – DNN Algorithm [7]

and Bias) of the best-fit provided the model parameters. In the testing part, the proposed model is applied to an unknown input vector from x_{test} and predicts the target probability metric (FFRs) as provided in Fig. 4.8. The DNN architecture consists of 5 dense layers, including the input and the output layers. The input layer consists of 50 nodes. The input vector to DNN is a 50-dimensional vector per net, as provided by the graphSAGE algorithm. The hidden layers are fully connected feed-forward neural networks, each having nodes 64 (1^{st}-hidden), 32 (2^{nd}-hidden), and 12 (3^{rd}-hidden) respectively. The output layer has only one node that performs the prediction. A Rectified Linear-Unit (ReLU) is the activation function that is benefitted in the last layer for prediction while the hidden layers take the advantage of the Hyperbolic Tangent (Tanh) as the activation function. Adam [159] is an adaptive learning rate optimization algorithm that has been used here as an optimizer, and finally, the loss function is updated with the Mean Squared Error (MSE) function.

- **Result Analysis**
  The final phase includes a comparison between the predicted and reference probability metrics, as given in Fig. 4.4. The compared results are plotted in Fig. 4.9, 4.10, 4.11, and 4.12.

4.7 Case Study and Results

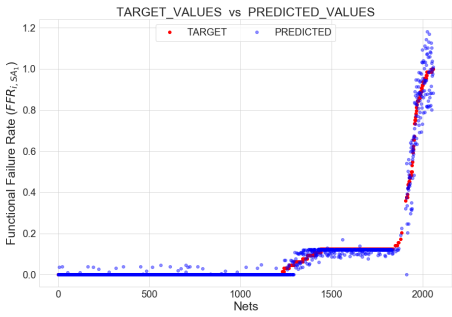


Figure 4.9 – Stuck-at-1: R^2 = 0.93 (40% Training Size) [7]

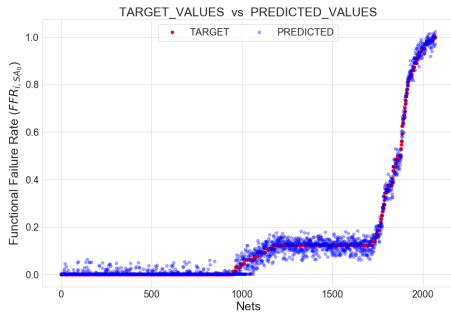


Figure 4.10 – Stuck-at-0: R^2 = 0.94 (40% Training Size) [7]

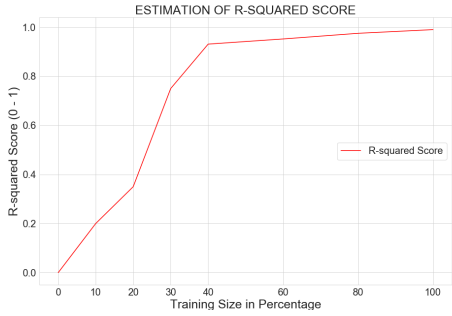


Figure 4.11 – Stuck-at-1: R^2 and Training Size (in %) Relation [7]

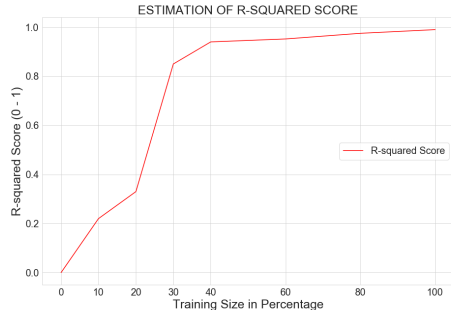


Figure 4.12 – Stuck-at-0: R^2 and Training Size (in %) Relation [7]

The case study experiment was conducted with the gate-level circuit of the 10-Gigabit Ethernet MAC [124]. Among the synthesized nets, the simulation-campaign injected faults on 3437 nets, where avoided nets are redundant in their functions. The redundant nets

includes the parallel connections also. The overall simulation paradigm was summarized in Fig. 4.4. Fig. 4.9 and 4.10 graphically represent the functional failure rates of nets due to stuck-at-1 and stuck-at-0 faults, respectively. The red color legend (in Fig. 4.9 and 4.10) represents reference values, and the blue color legend represents corresponding predicted probabilities. Both graphs were obtained for 40% random training size, which explicitly indicates that only 60% (2062 nets) of total nets (3437) are plotted in Fig. 4.9 and 4.10. The experiments with different training sizes have led to the conclusion that the training size is better to be 40% for both predictions. The 40% training size provides a significantly higher correlation between predicted and reference values, compared to other training sizes of less than 40%. The correlation property is slightly improving with above 40% training size. Such observations are plotted in Fig. 4.11 and 4.12. The graphical visualizations in Fig. 4.9 and 4.10 demonstrate how well the prediction replicates the observed reference values. In both cases, the regressions have achieved adequate values for the coefficient-of-determination (R^2) metric (approximately 0.93 for stuck-at-1 and 0.94 for stuck-at-0). Generally, the best model fit value of R^2 metric is 1, and the worst value is 0. In statistics, the R-squared (R^2) [134] value is the measure of goodness-of-fit of a regression model.

From Fig. 4.9 and 4.10, we can characterize the nets based on their functional-failure vulnerabilities with an acceptable prediction error. When the failure rates of nets numerically close to 1, then nets are more critically characterized. This information provides a more reliable approach for minimizing the cluster size of critical fault-locations in the fault diagnosis. The implemented algorithm depends on 40% of the fault-injection database. However, it is quite impressive to note that the test and training phase of the whole algorithm takes only less than 10 minutes. The holistic algorithm (graphSAGE + DNN) can process a netlist of 10,000 components without any additional running time on a CPU machine of 16GB RAM. The trade-off between the netlist size and the processing time will be investigated in the future with further experiments.

Table 4.1 provides the impacts of prediction in terms of required time and also provides the Functional Fault Coverage (FFC) metric (4.4) of test patterns and its corresponding predicted value. In Table 4.1, the full database is generated by a Fault-Injection (FI) campaign with 7 modelsim. The FFC value is predominant in the circuit diagnosis to evaluate the effectiveness of the test pattern. The fault coverage close to 90% will adequately explain the observability of single stuck-at faults. Note, here, the test patterns/packets were chosen randomly for this experiment, and were not optimized for the best. The low FFC value (27.5%) of the given test patterns is approximately predicted by the framework (28.2%). If fault-injection simulation chooses the test pattern with a higher FFC value, then the probabilistic failure observation by fault-injection per net will also increase. This increased failure observation provides more information to the algorithm and subsequently helps the algorithm to estimate the stuck-at fault failure probability as discussed in Fig. 4.9 and Fig. 4.10 with high accuracy. The random test patterns are chosen to model the real-time scenarios where it is not possible to expect the high possibility of large fault-coverage inputs to the device under test.

In Table 4.1, the Functional Fault Coverage (FFC) of the used test pattern in this experiment is not high. It is also possible to choose a test pattern with high FFC for this experiment. This experiment also proves that prediction helps in estimating FFC of the test pattern. In general, testing circuit quality always demands a high FFC test patterns. Here, this requirement can be achieved by developed framework. This way, the framework that the thesis aims to develop for the soft-error reliability analysis attains a capability to analyze the system quality.

Table 4.1 – STUCK-AT FAULTS CAUSED FAILURES AND METRICS ESTIMATION [7]

Parameters	Stuck-at-1	Stuck-at-0
R^2	0.93	0.94
Training & prediction	$<$ 10 mins	$<$ 10 mins
Training Data (40%)	$\approx$ 1.2 hrs	$\approx$ 1.2 hrs
Full Database (Fault-Injection (FI) Campaign - 7 modelsim)	$\approx$ 3 hrs	$\approx$ 3 hrs
	FFC Calculation	
Empirically calculated FFC	= 0.275	
Predicted FFC	= 0.282	

4.8 Chapter Conclusions

The chapter has validated a naive artificial intelligence approach to predict the functional failure vulnerability of each net at the gate-level abstraction. Also, test the effectiveness of the test patterns. The proposed technique succeeds in reducing the time-complexity of exhaustive fault-injection by 60%. The developed framework depends on the gate-level of a test circuit so that the extracted features predict the realistic fault propagation probabilities. For future work, the developed method combines the single stuck-at model and advanced mathematical algorithms to cluster the nets into critical and non-critical groups depending on the applications. The results open a new possible dimension to a scalable and very cost-effective simulation tool for producing dependable micro-electronics systems.

Chapter 5

Soft-Error Reliability Under Time-Dependent Variability

The conference publications [VIII] and [IX] are contributed to this chapter. The article [IX] stamps its significance by quality results and observations among competent researchers in one of the top conferences and the paper is awarded as Outstanding Student Research Paper. The theoretical hypothesis and experimental case studies in [IX] cover 80% of this chapter, and a workout example from the article [VIII] supports the remaining 20%.

5.1 Preamble of Chapter

System engineering continuously focuses on the aggressive technology scaling which increases the vulnerability of radiation-induced soft-errors [19] [160]. Simultaneously, the time-dependent variabilities like the effects of Bias Temperature Instability (BTI) expedite the reliability assessment of high-performance Cyber-Physical Systems (CPS) into an increasingly challenging job. Scientific efforts significantly highlighted the relevance of experiments that account for the impact of aging on soft-error reliability. The works [161], [162] and [163] have proposed their methods and chronicled their observations on soft-error susceptibility under circuit aging. A. Gebregiorgis in [164] has presented a cross-layer reliability analysis in the presence of soft-errors, aging, and process variation effects. Similarly, the work [165] provides ramification of aging and temperature on Propagation Induced Pulse Broadening (PIPB) effect of Single Event Transient (SET) pulse for nano-scale CMOS. In [166], F. L. Kastensmidt shows that aging and voltage scaling enhance the Soft Error Rate (SER) susceptibility of SRAM-based FPGAs by two times.

The main motivation of the work is to investigate the soft-error reliability challenges at low-scaled (e.g., 15-nm) technologies with time-dependent voltage variabilities. The downscaled technology results in clock-frequency maximization and provides soft-error propagation a high sensitivity to the path delay variations in the design due to aging. In this work, aging effects are limited to the NBTI. The Positive Bias Temperature Instability (PBTI) remains unconsidered throughout the research based on the facts: the PBTI effect for the NFET transistors at small scale technologies for the High Performance (HP) applications is comparatively low compared to the NBTI effect for PFET transistors [167]. Also, the PBTI effect's dependence on the quality of gate-oxide materials [168] and the complexities in the efficacy to model PBTI's relative voltage degradation, are more than the intended research focus. However, the deduced conclusions for NBTI can extend to the combined effect of NBTI and PBTI. The research aim of the chapter in this thesis is not to concentrate

on the soft-error generation but to provide a deeper understanding into their propagation characteristics by considering the derating factors like Electrical Derating (EDR), Temporal Derating (TDR), Logical Derating (LDR), and Functional Derating (FDR). This thesis also unlocks an option to enhance the soft-error reality analysis of the aged circuits at a higher abstraction level (e.g., RTL).

5.2 An Introductory Example of ML as Compact Models

Intellectual Property (IP) and component providers use compact models to provide simplified views of their products to their users. These models can express design qualities and features useful for simulation, integration, testing, and many other purposes. Current and upcoming standards mandate the use of reliability and functional safety metrics, requiring the elaboration of the appropriate models. This section introduces a machine-learning-based approach to integrate the many individual models of a subsystem's elements in a single compact model that can be re-used and assembled further up in the hierarchy. The compact models provide consistency, accuracy, and confidentiality.

This example addresses the calculation of a propagation rate of transient faults in the system and expresses in Soft Error Rate (SER) metric. For the demonstration, we utilize naive equations for modeling the error rates of each element. Finally, aggregating the contribution of the various components. Firstly, let's introduce the following functions for the calculation of the different types of Soft Error Rates (indicated in FITs/MegaBit or MegaCell):

$$RawSER_{seq}(V_{dd}) = 100 \cdot (1 + 1.2 - V_{dd}) \tag{5.1}$$

$$RawSER_{comb}(V_{dd}, PW) = 50 \cdot (1 + (1.2 - V_{dd})) \cdot \frac{V_{dd}}{PW} \tag{5.2}$$

where: a Flip-Flop has 100 FITs/Mb at the nominal 1.2V supply voltage. SER decreases at higher voltages and increases at lower. V_{dd} means supply voltage. A combinatorial cell can exhibit a spectrum of Single Event Transients with declining event rates for longer-duration events. We will limit the minimal Pulse Width (PW) at 10 ps which is the shortest transient that can propagate in the selected technological process. PW means Pulse Width (in ps).

The Alpha particles (emitted by impurities in the packaging materials) caused Single Event Effect (SEE)s dominates in natural working environments. While neutrons caused SEEs dominate in the atmosphere where the interaction of high-energy particles is high. Both contributions depend on numerous parameters, and it is challenging to provide a model that integrates the effect of all the parameters. As an example, the neutron-induced SEE rate depends on many factors, including physical location, altitude, solar activity, shielding, and so on. Following the approach from [169], let's focus on the altitude and cutoff (dependent on location), ignoring solar modulation. In this case, (5.3) expresses the actual Neutron Flux (NF) at a given geographic position.

$$NF = NF_{ref} \cdot GRF \cdot e^{-(\frac{A - A_{ref}}{L})} \tag{5.3}$$

where: NF_{ref} is the the neutron flux at the reference location (e.g., New York = 14 n/sq. cm/h). L is the flux attenuation length for neutrons in the atmosphere (148 g/cm2). Finally, A is the areal density of the location of interest, A_{ref} is the areal density of the reference location.

$$A = 1033 \cdot e^{-0.03813(\frac{a}{1000}) - 0.00014(\frac{a}{1000}) + 6.4 \cdot 10^{-}7(\frac{a}{1000})^3} \tag{5.4}$$

While these factors can be described analytically and integrated into the various models, the GRF represents the Geomagnetic Reference Field (GRF) and varies according to the geographical positions. As an example, by using the values of geomagnetic vertical cutoff rigidity (provided by the Aerospace Medical Research Division of the Federal Aviation Administration's Civil Aerospace Medical Institute), the relative neutron flux is calculated. The cutoff data were generated by M.A. Shea and D.F. Smart using the international GRF for 1995 [170] [171]. Therefore, the actual GRF values can only be provided as a table of data indexed according to the longitude and latitude, and cause many issues to integrate the tabular data in a compact model. These issues include the need for interpolation between the available sparse and low granularity data.

Machine-Learning Models cope very efficiently with these difficulties. Firstly, the training phase of ML model is independent of any type of data (analytical and tabular (or) linear and non-linear). It is an adequate model to interpolate GRF data between the locations and allows the approximate calculation of the GRF for any geographic position. The neutron-induced error rate is provided as the base value for the reference setting (New York, sea-level) that needs to be multiplied by an acceleration factor calculated according to the actual location and altitude. As an example, the following Table. 5.1 shows the acceleration factors for a selection of altitudes and locations. The final acceleration factor can be calculated by multiplying the appropriate location factor with the desired altitude factor.

Table 5.1 – ALTITUDE DEPENDENT NEUTRON DATA [8]

Altitude	Altitude	Neutron Flux (n/cm^2h)	Neutron Flux (n/cm^2s))	Neutron Flux (relative to sea level)
0	0	14.0	0.003889	1.0
1000	304.8	18.2	0.005056	1.3
2000	609.6	23.4	0.0065	1.7
5000	1524	47.6	0.013222	3.4
10000	3048	134.6	0.037389	9.6
20000	6096	668.5	0.185694	47.8
30000	9144	2001.1	0.555861	142.9
35000	10668	2993.2	0.831444	213.8
40000	12192	4147.0	1.151944	296.2

Table 5.2 – LOCATION DEPENDENT NEUTRON DATA [8]

Location	Neutron Flux
Colorado Springs	4.42
Bangalore	1.02
Beijing	0.72
Grenoble, France	1.24

In the following phase, an assumed simple de-rating approach calculates the overall Error Rate of an ASIC with 1 Mbit of Flip-Flops and 10 Mbits of Combinatorial cells. The overall Error Rate is the summation over the computed de-rated contribution of each element:

$$SER_{ASIC} = \sum_{Element} RawSER_{S|C} \cdot \prod_{\text{TDR,LDR,FDR}} DR \tag{5.5}$$

$$TDR_{seq}(Freq) = \frac{Slack}{ClockPeriod} \tag{5.6}$$

$$= 1 - \frac{Freq(H_z)}{2e^6} \tag{5.7}$$

$$TDR_{comb}(PW, Freq) = \frac{PW}{ClockPeriod} \tag{5.8}$$

$$= \frac{PW}{Freq} \tag{5.9}$$

$$LDR = 0.25 \tag{5.10}$$

$$FDR = 0.25 \tag{5.11}$$

Finally, the overall SER of the ASIC can be described as follows:

$$\begin{aligned} SER(PW, Freq, Vdd) = (&1Mbit \cdot RawSER_{seq} \cdot TDR_{seq} \\ &+ 10Mbit \cdot RawSER_{comb} \cdot TDR_{comb}) \\ &\cdot LDR \cdot FDR \end{aligned} \tag{5.12}$$

$$\begin{aligned} SER(PW, Freq, Vdd) = \Big(&1Mb \cdot 100\frac{FIT}{Mb} \cdot (1 + (1.2 - V_{dd}) \cdot 1 - \frac{Freq(H_z)}{2e^6} \\ &+ 10Mb \cdot 50\frac{FIT}{Mb} \cdot (1 + (1.2 - V_{dd})) \cdot PW \cdot Freq\Big) \\ &\cdot 0.25 \cdot 0.25 \end{aligned} \tag{5.13}$$

where: the considered de-ratings are Logic De-Rating, the Temporal De-Rating (TDR), and Functional De-Rating (FDR). The Fault/Error/Failure dichotomy and the usage of various De-Rating factors depend on the scientific nomenclatures as presented in [172].

Final customization can be made to clarify the value or the value range for the Pulse Width parameter which is a technology attribute and may not make sense to or be fillable by the final user. Therefore, the ASIC provider, in agreement with the technology provider, may specify values for the PW according to the working environment. In this example we consider a neutron environment (ground applications) with a typical PW of 50 ps, Alpha particles- 10 ps and Heavy Ions-100 ps. The overall SER equation can disclose unit technology data (e.g., FIT rates per Mb, typical pulse widths), design structure (1Mb of FF and 10Mb of combinatorial cells), or knowledge (such as calculation of de-rating factors). The next step in this example consists of a compact ML model elaboration in exercising the overall SER equation over the range of valid values for the environment, frequency, and voltage.

The final phase of the work continues with the training of the ML model. In this work, we are interested in the applicability of different ML models for the intended usage. Accordingly, several ML regression methods have been evaluated: Linear Models (Linear and Ridge), Kernel Ridge Regressor (with Linear, Polynomial, RBF, Sigmoid Kernels), Decision Tree Models, Neighbors-based Models (K-Nearest and Radius), Support Vector Machine Models (Linear SVR, SVR with various kernels, NuSVR), Multilayer Perceptron Neural Networks.

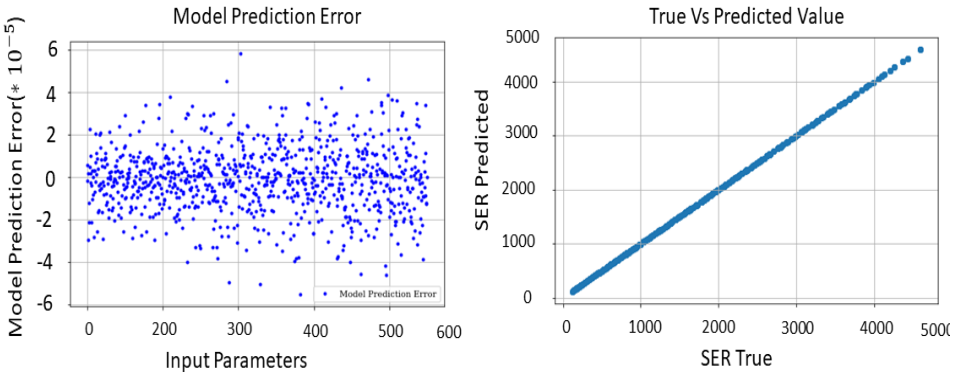


Figure 5.1 – Compact Modeling of Soft-Error Rate Using Ridge Regression (Polynomial Kernel) [8]

All models have been trained with 60% of the data set, the train data set. After the training, the models are exercised over the full permitted parameters range. This allows, on the one hand, to measure the accuracy of the model to recall values that were already in the training data set. On the other hand, testing the model with data not used for training evaluates model's capacity to interpolate and extrapolate the given data. The performance of the models is measured by comparing the predicted values against the reference values and calculating the following metrics: MAE, Max Absolute Error (MAX), Root Mean Squared Error (RMSE), Explained Variance Score (EVS) and Coefficient of Determination (R^2). The following Table 5.4 presents the results for the most accurate and promising models:

- Ridge Regression with RBF Kernel
- Ridge Regression with Sigmoid Kernel
- k-Nearest Neighbors Regression
- Support Vector Regression with Polynomial Kernel

Fig. 5.1 shows the graphs representing the model prediction error for the train and test data set, as well as the correlation between the actual and predicted values when the Ridge Regression with polynomial Kernel is used. In comparison Fig. 2 the results for the k-Nearest Neighbors regression are shown. The graphs show as well very clearly, that the

Table 5.3 – MODELLING OF ALTITUDE DEPENDENT NEUTRON DATA [8]

ML Model	Dataset	ML Model Error/Correlation Measurements				
		MAE	MAX	RMSE	EV	R^2
Ridge Regression (Polynomial Kernel)	Train	1.688e-06	1.199e-05	2.119e-06	1	1
	Test	1.695e-06	1.331e-05	2.106e-06	1	1
Ridge Regression (RBF Kernel)	Train	2.716e-05	2.545e-04	3.530e-05	1	1
	Test	2.907e-05	6.609e-04	4.421e-05	1	1
k-Nearest Neighborhood Regression	Train	0	0	0	1	1
	Test	21.91	195.2	30.98	0.99	0.99
Support Vector Regression (Polynomial Kernel)	Train	1.787e-02	6.830e-02	2.233e-02	1	1
	Test	1.756e-02	6.243e-02	2.195e-02	1	1

k-Nearest Neighbors regression is perfectly able to recall the training data but less efficient when predicting the test data.

5.3 Modeling NBTI Induced Voltage Degradation

The primary work of this research activity is to find a mathematical abstraction to estimate the changes in threshold voltage ΔV_{th} due to the NBTI process in the aged circuits, and completely revised from the original papers [173] and [174]. The paper [173] has presented a predictive model for the Negative Bias Temperature Instability (NBTI) of PMOS under both short-term and long-term operations. This model has comprehensively

Table 5.4 – PARAMETERS AND UNITS FOR NBTI PREDICTIVE MODELING [9]

Symbol	Quantity	Numerical value in SI unit
K_v	Technology Constant	6.1773e-11
C	Temperature Dependence	2.2987e-12
T_0	Constant	10^{-8} (s/nm^2) $\rightarrow 10^{10}$ (s/m^2)
E_0	Technology-Independent	0.08 V$/$nm $\rightarrow$ 0.08e9 V$/$m
E_a	Technology Independent	0.13 eV $\rightarrow$ 2.08260e-20 J
k	Boltzmann Constant	1.38e-23 m^2 kg s^{-2} K^{-1}
$1/n$	H_2 Diffusion Model	1/6
t_{ox}	Oxide Thickness	0.9 * 10e-9 m
ξ_1	Back Diffusion Const.	0.9
ε_{ox}	Oxide Permittivity	$3.9 * 8.854 * 10^{-12}$ F/m
K_1	Model Constant	7.5e22.5 $C^{-0.5}\, m^{-2.5}$
E_{ox}	Electric Field (gate oxide)	27.7e-7 V$/$m

Unit Abbreviations: K = Kelvin, F = Farad; V = Volt, s = Second, m = Meter, nm = Nano-meter, J = Joule, kg = Kilogram.

apprehended NBTI dependence on the key transistor-design parameters, based on the reaction-diffusion (R-D) mechanisms. The work in [173] presented a quality model accuracy verification with 90-nm technology while [174] reassured that with industrial 65-nm technology. A characterization of threshold voltage (ΔV_{th}) degradation due to NBTI for 15-nm technology cells, has been presented through this chapter, where the models from [173] and [174] are integrated with 15-nm technology parameters as in Table 5.4. A model for the $\Delta V_{th,t}$ considering the long-term effect of NBTI is provided in [173]. At high frequencies, the threshold voltage degradation of long-term evaluation is independent of the frequency [173] [175]. So the ΔV_{th} at time 't' can be written as [173]:

$$\Delta V_{th,t} \approx \left(\frac{n^2 K_v \alpha C t_1 t}{\xi_1^2 t_{ox}^2 (1-\alpha)} \right)^n \tag{5.14}$$

The parameter α in (5.14) is the duty-cycle that defines the signal probability in a clock period. Equation (5.14) refers to the dynamic NBTI, $\forall\, \alpha \in [0,1]$. As $\alpha \rightarrow 1$, (5.14) is attributed to the static NBTI that corresponds to the case of PMOS under constant stress. A graph between ΔV_{th} and α, at $t = 1$ year is given in Fig. 5.2. The upper-limit of the degradation model (5.14) is estimated by a power law model [176] as privided in (5.15). The static ΔV_{th}

calculation at t = 1 year gives 7mV that fits the trends of static ΔV_{th} of various technology-nodes as given in PTM models [177] [178].

$$|\Delta V_{th,t}| = (K_v{}^2 t)^n \tag{5.15}$$

The formula in (5.14) is deduced into a simpler form by substituting the parameters of 15nm technology and follows the form of (5.16), where relation of technology specific factors is represented as separate form.

$$|\Delta V_{th,t}| \simeq \underbrace{\left(\frac{n^2 K_v C t_1}{\xi_1^2 t_{ox}^2}\right)^n}_{\text{Technology dependent}} t^n \left(\frac{\alpha}{1-\alpha}\right)^n \tag{5.16}$$

5.4 An AI Revolution for NBTI Predictive Model

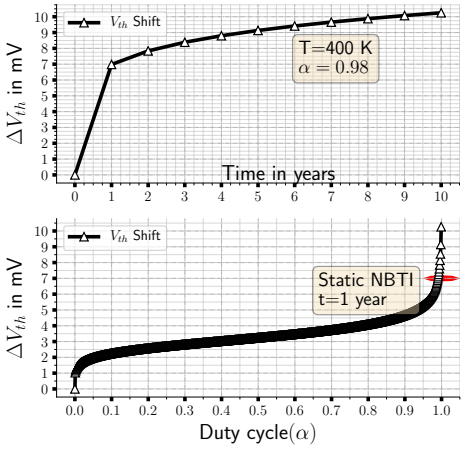


Figure 5.2 – Mathematical Analysis [9]

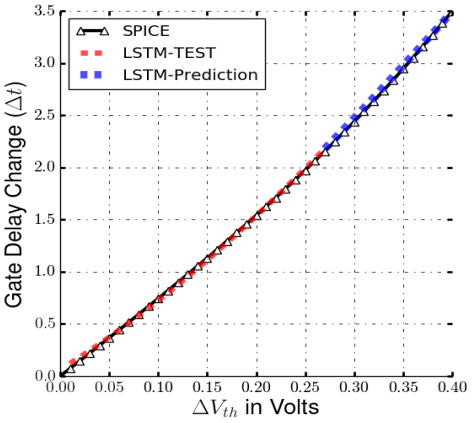


Figure 5.3 – LSTM Model (Inverter) [9]

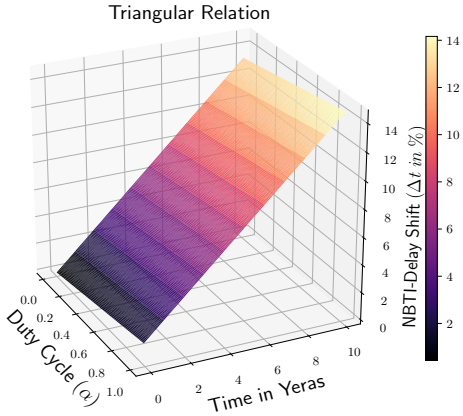


Figure 5.4 – SVM-Regression Model (Inverter) [9]

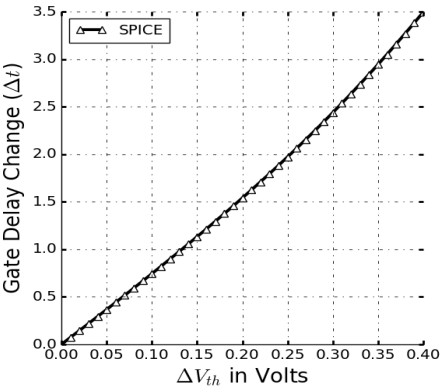


Figure 5.5 – SPICE Simulation of 15 nm Inverter [9]

The gate-delay degradations of 15-nm technology library cells are characterized through the SPICE simulation at the transistor level. A neural network called Long Short-Term

Memory (LSTM) is a perfect method to predict the future delays from past delay (Δt) samples even without an input like ΔV_{th}. Fig. 5.3 depicts the complete results of LSTM modeling, where the predicted values from previous samples are shown in blue. However, the LSTM model is not a completely dependable one here for generating a delay Δt for the current input ΔV_{th} in a timely important simulation environment. To simplify the exhaustive fault-injection experiment, a Machine Learning (ML) model called Support Vector Machine (SVM) is used to automate the simulation data. Previous scientific literatures [179] and [153] presented a simple polynomial function for modeling variation between Δt and ΔV_{th}. In this work, SVM can perform as a comprehensive model generator for a triangular relation between α, time 't', and Δt, where ΔV_{th} is hidden in the SVM model. The AI revolution is implicitly referring to such complex and compact modeling of indirectly related multi variables. Fig. 5.4 delegates a SVM model for an inverter. Once SVM is trained offline, it will become a fast online computing model generator of Δt for the inputs α and time 't'. This compact ML model is very much straightforward in exporting the variability from the transistor level to the gate level. Another important factor to choose SVM over LSTM is the low static ΔV_{th} (7mV at t=1 year) by (5.16), because a large simulation data needs to train LSTM as provided in Fig. 5.3, where ΔV_{th} ranges from 0 to 0.4V, while ΔV_{th} in Fig. 5.4 perfectly follows the ΔV_{th} values as provided in Fig. 5.2. Fig. 5.5 depicts a glsspice simulation of the NOT/Inverter gate.

5.5 Methodology

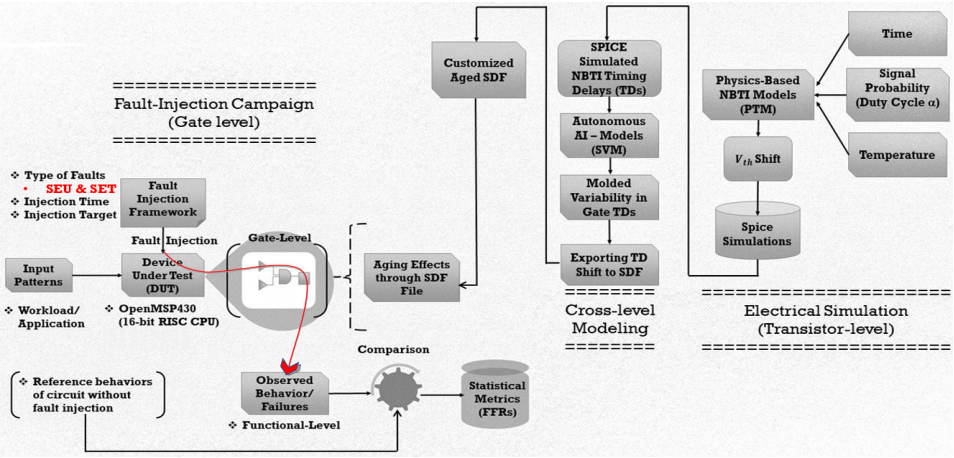


Figure 5.6 – Delineation of Aging Aware Fault-Injection Framework [9]

5.5.1 Phase I: Fault-Injection Campaign

Single Event Effects (SEEs) are the consequences of interactions between the circuit and the radiation particles. SEUs and SETs are widely used here as the prominent representatives of SEEs. The use-case for SEU fault mainly implies an inversion of the stored value in a flip-flop until the clock period changes. The SET represents a transient pulse of arbitrary width at the gate and having the probability to propagates and latches to the downstream sequential element. Fig. 5.6 shows a block diagram of the fault-injection campaign that infers the statistical functional failure metrics of SEU/SET events. The mathematical model for Functional Failure Rate ($FFR_{i,seu}$) of an SEU event is described in (5.17) and the Func-

tional Failure Rate due to SET ($FFR_{i,set}$) is formulated in (5.18).

$$FFR_{i,seu} = FIT \cdot TDR \cdot LDR \cdot FDR \tag{5.17}$$

$$FFR_{i,set} = FIT \cdot EDR \cdot TDR \cdot LDR \cdot FDR \tag{5.18}$$

where FIT denotes the rate of soft errors at the i^{th} flip-flop/gate in the Failure-In-Time (FIT) unit. Electrical Derating (EDR), Temporal Derating (TDR), Logical Derating (LDR), and Functional Derating (FDR) are more sophistically portrayed in Fig. 5.9. The classical explanations for each derating factors are available in [19].

5.5.2 Phase II: Aging Aware Gate-Level Circuit

Fig. 5.6 represents how an injected fault propagates through the aged circuits. Circuit aging is modeled through degraded propagation delay as the result of the NBTI process at PMOS transistors. Sophisticated cross-layer modeling is also demonstrated in Fig. 5.6. A way more experimentally proved and scientifically adapted physics-based NBTI models (5.16) are employed to generate the V_{th} shifts.

Electrical simulations have been carried out using SPICE. Voltage sources have been injected in the gate of PMOS transistors to model the V_{th} shift caused by aging. The voltage introduced by the source was then swept from 0V to 0.4V to represent the effects of different levels of aging. The impact of aging on each logic gate is estimated by measuring the input-to-output time delay, i.e., the time delay from when the input is in $V_{DD}/2$ to when the output rises from '0' to '1' at the $V_{DD}/2$ mark. The spice simulation is conducted for 23 gates from a 15-nm Nangate library which including inverter (NOT), AND, OR, NAND, NOR, XOR, XNOR with input combinations ranging from 2 to 4. The transistor model (SPICE Model Card) for 15-nm Nangate library is customized from HP Predictive Technology Models (PTM) [178]. An Simulation Program with Integrated Circuit Emphasis (SPICE) simulation of 3 input NOR and 2 input NOR are shown in Fig. 5.7 and Fig. 5.8 respectively.

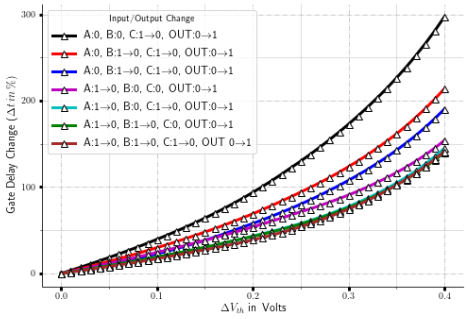


Figure 5.7 – SPICE Simulation of 3 Input NOR Gate

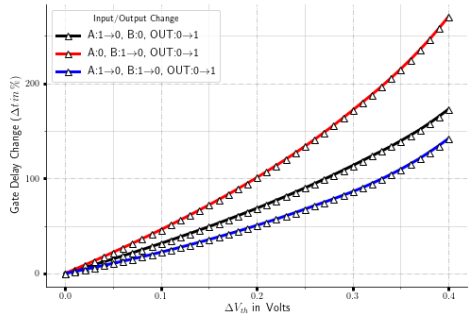


Figure 5.8 – SPICE Simulation of 2 Input NOR Gate

After the simulation process, the change in propagation delays of combinational cells is submitting to AI models. Fig. 5.6 depicts the significance of AI models in cross-layer modeling. A versatile model by SVM is used to generate the propagation delay change Δt as a function of duty cycle (α) and time in years. This model is very easy to port to gate-level or even to a higher hierarchical level, and adequately producing Δt values by

concurrently changing time and signal probability. Those model-driven values are numerically very close to the original simulation Δt and the test phase of SVM confirms that the Mean Squared Error (MSE) is less than 2% for the given test data.

After testing the compatibility and quality of the model, the model outputs (Δt) are exported to the SDF file. A customized SDF file introduces time-dependent variability at the gate level. The customization of the SDF file means altering the value under the keyword entry called 'IOPATH' that indicates the gates' Input-Output path delay. Before changing the gates' 'IOPATH' values, corresponding signal probabilities and signal transitions at each terminal are pre-estimated through signal-activity analysis through Verilog Procedural Interface (VPI) functions and Value Change Dump (VCD) files by Modelsim. So that, the changed values should coincide with the real-time input and output transitions between '0' and '1'. To process the above-explained cumbersome work for large IEEE standard SDF files, then a dedicated algorithm is written in python, and the corresponding pseudo-code is presented in Algorithm 3. The algorithm generates an IEEE standard customized aged SDF file that provides an aged behavior while injecting SEUs/SETs in a simulation environment.

Algorithm 3: IEEE Standard Aged SDF File

Result: Customized Aged SDF File

Estimate ΔV_{th} for 10 years $\forall$ used gates $\in$ Tech.lib;
for *Each used Tech.lib gate* **do**
  for *Each 'i'* $\in$ *SVM model Training* **do**
    SVM : input vector $[\alpha_i, \text{time}_i, \Delta V_{th_i}] \mapsto \Delta t_i$;
  end
  Test SVM model and Save in Python script;
end
Read the SDF file in IEEE standard format;
for *Each cell instance in SDF File* **do**
  Identify the Tech. library cell name ;
  Calculate signal probability (α) from VCD file ;
  if *[*α_i*, time*$_i$*] given* **then**
    Calculate ΔV_{th_i} by (5.16) ;
    SVM.gate = Load trained SVM.gate kernel;
    Δt_i = SVM.gate.predict($[\alpha_i, \text{time}_i, \Delta V_{th_i}]$) ;
    Update the instance's IOPATH in SDF file;
  end
end

5.6 Results and Discussions

5.6.1 Device Under Test

The case studies are conducted by gate-level circuits of the openMSP430 cores, a 16-bit microcontroller (μC) which is synthesized by the 15-nm NanGate Open Cell Library. As per the static timing analysis of the test case circuit, the critical paths contain a maximum path delay of 250ps. The CPU unit in the MSP430 μC is executing the application 'sandbox' by sourcing a clock period of 2ns. The Fig. 5.9 demystifies the propagation of SEU/SET faults to the circuit function.

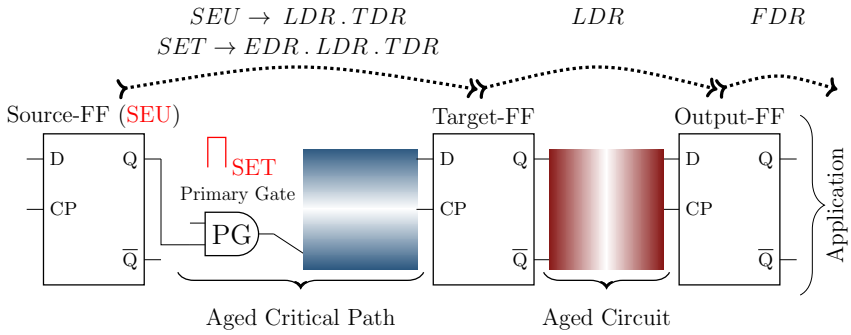


Figure 5.9 – From SEU/SET Fault to Failure [9]

5.6.2 Aging Impact on SEU Fault Propagation

The NBTI effect is modeled in terms of propagation delay Δt of logic gates. The setup and hold time constraints of flip-flops dictate the maximum and minimum delays of the logic gates between them. The maximum delay constraint limits the number of consecutive gates on the critical path of a high-speed circuit. In this section, the results are proving that the propagation probabilities of SEU faults are dropping according to the changed delay of the signal path. These results are explaining in Fig. 5.10 and Fig. 5.11. In Fig. 5.10 a term called Polarization Window (PW) is introduced in red color. This is because the time width of the PW completely masks the SEU faults that have been generated inside the window. From Fig. 5.9, it is clear that induced-SEU fault at source flip-flop (Source-

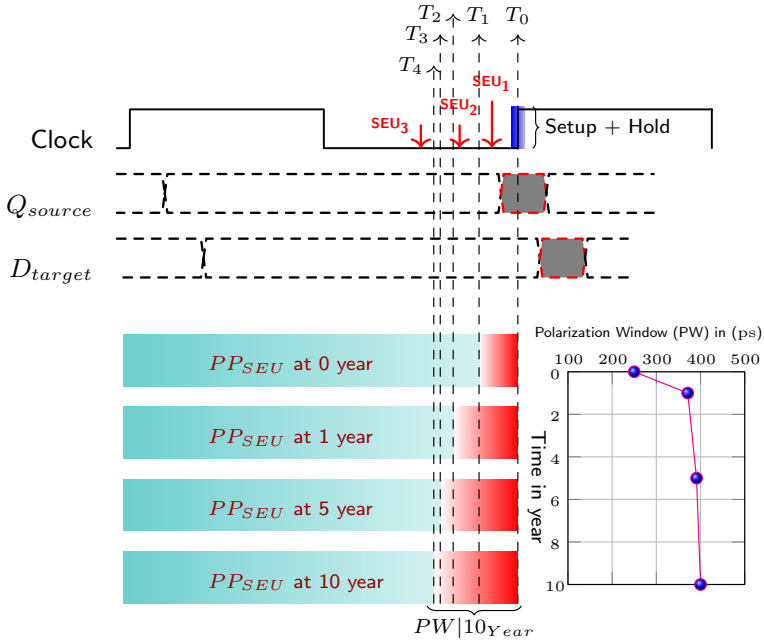


Figure 5.10 – SEU-Faults' Polarization Window (PW) Elongation [9]

FF) reaches the target flip-flop (Target-FF) after a path delay $t_{D_{set}}$ between them. So that, a fault SEU_1 at source-FF between T_0 and T_1 as provided in Fig. 5.10 not overlaps the

latching window (SETUP+HOLD time) when it reaches the target-FF, and masks the SEU_1 fault propagation. But the fault SEU_2 is propagated and latched to the target flip-flop because it is stable during the SETUP and HOLD time of target-FF. After 1 year, the PW is prolonged to $T_0 - T_2$ due to NBTI effect and clearly masks the fault SEU_2 as demonstrated in Fig. 5.10. Similarly, after 10 years, the faults SEU_1 and SEU_2 are masked and the fault SEU_3 is still propagating to target-FF. These conclusions are statistically derived from the fault-injection campaign of 1582 SEU faults at the path of longest delay as given in Fig. 5.11, where the upper-row justifies that SEU_1 is masked without aging, and the lower-row justifies that SEU_1 and SEU_2 are masked after 10 years.

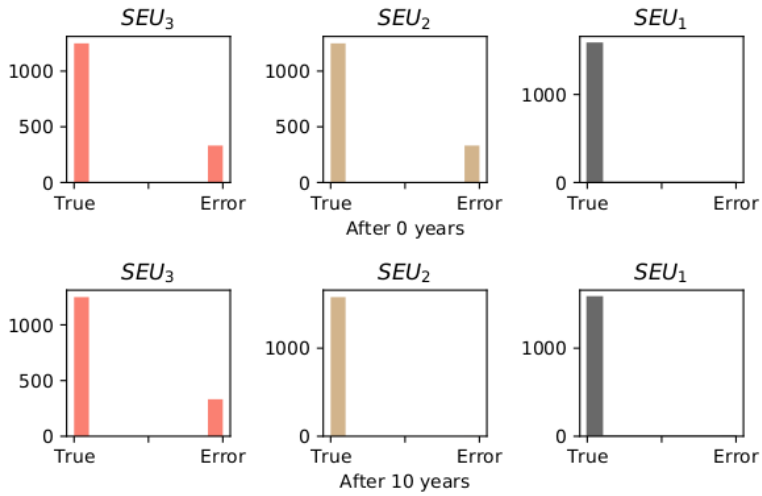


Figure 5.11 – SEU-Fault Injection Result (After 0-10 years) [9]

5.6.3 Impact of Aging in SEU Caused Circuit-Functional Failures

In Fig. 5.10, the Propagation Probability (PP_{SEU}) indicates the chances of generated SEUs to propagate through the downstream circuit within an arbitrary clock period T_{clk} and is modeled as an uniform distribution as specified in (5.19).

$$PP_{SEU} = \begin{cases} \frac{1}{T_{clk}}(T_{clk} - PW|_{\tau_{d_0}}) & \forall \text{Non-aged } \tau_{d_0} \\ \frac{1}{T_{clk}}(T_{clk} - PW|_{\tau_{d_i}}) & \forall \text{Aged } \tau_{d_i} \end{cases} \tag{5.19}$$

The effect of elongated polarization window PW due to aged delays τ_{d_i} in (5.19) models the increased masking of SEU faults and confirms it through an exhaustive Fault-Injection (FI) campaign as shown in Fig. 5.12. Color C-2 represents a FI campaign of 74000 injected faults at 188 flip-flops (FFs) of the non-aged circuit, which produce a total of 1181 functional failures as in (2.3). The 188 FFs are explained as high failure vulnerability FF-instances out of 720 FFs. A 10 year aged model of the same circuit is simulated with an increased FI rate by 50% (color C-1 in Fig. 5.12). This increment is performed based on the previous studies that the rate of SEU generation increases due to decreased critical charges Q_{crit} while aging.

Even though the FI rate is boosted by 50% for the aged circuit, the number of detected functional failures per flip-flop and the cumulative failures (568), are comparatively less contrary to the case in color C-2 in Fig. 5.12, which shows high-level masking of SEU faults.

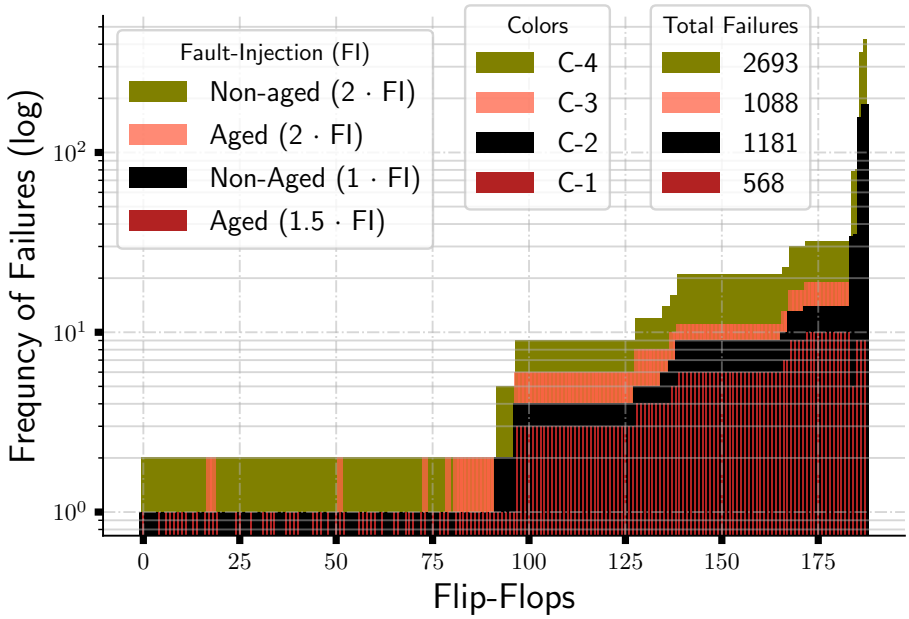


Figure 5.12 – Functional Failures (2.3) *of SEU-Fault Injection [9]*

As an extension of the analysis, a FI campaign consists of 148000 faults (almost double in number) is performed at the 10-year aged circuit. So that a total of 1088 failures are observed (color C-3 in Fig. 5.12) and that plot contrasts with a reference bar-graph of FI of 148000 faults at the non-aged circuit (color C-4). FI in C-3 leads to more failures per FF compared to the case in C-2, but still, the overall failures in color C-3 are fewer. This proved that aging causes a masking effect in SEU fault propagation.

5.6.4 Aging Impact on SET Fault Propagation

As the circuit ages, considerable propagation-probability deteriorations for SEU faults have been notified. Nevertheless, SETs are not attenuated when traversing the critical paths in response to elongation of the polarization window in Fig. 5.10. The time-slot at which the generated SETs are latching to target-FF (Fig. 5.9) is shifting advance in time due to the delay changes in the traversal path of SET. These observations are delineated in Fig. 5.13, where the propagation delay $t_{D_{set}}$ of SET_1 at gate PG in Fig. 5.9 varies from 248 ps to 398 ps, which causes a shift in the time slot of propagating SET from T_1 to T_4. The observations are simulated by a SET fault of 20ps in width. Below the timing diagram in Fig. 5.13, the results of 100 SET fault injections over a second-half cycle of an arbitrary fault-latching clock are provided.

5.6.5 Modeling of Aging Impact on SET Fault Propagation

The shift in time-spots of propagating SET as provided in Fig. 5.13, is modeled by the signal processing method. The SET pulse is assigned by a digital pulse model of width 'w' in gate-level, which possess same electrical boundary properties of a transient pulse at transistor-level. Similarly, the SETUP and HOLD time-periods of a target-FF in Fig. 5.9, are modeled as a single digital pulse $S_{SH}(t - T_0)$ of width equal to the sum of SETUP and

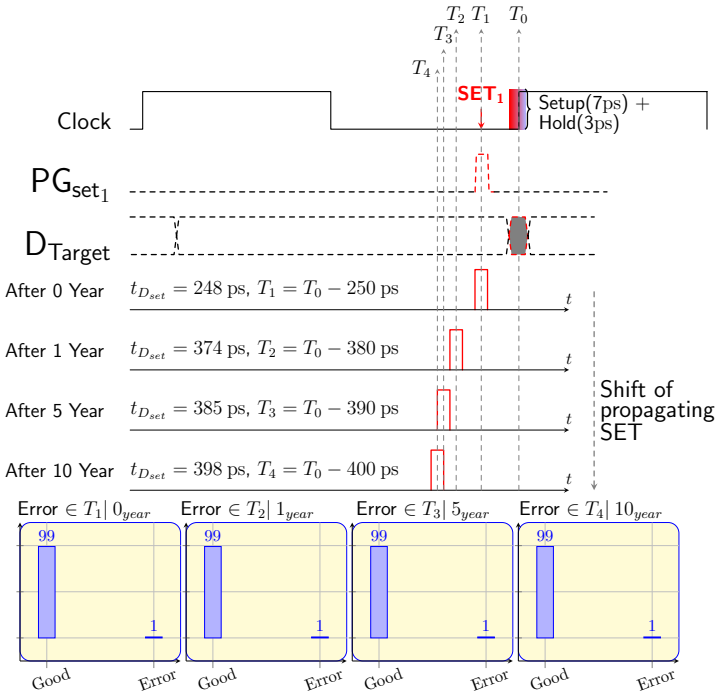


Figure 5.13 – Shift in Propagating Spot of 20ps-*SET [9]*

HOLD, and the amplitude is normalized by $1/w$. The delayed impulse function $\delta(t - t_{D_{set}})$ in step-1 of Fig. 5.14 convolves with induced SETs (G_{set_1} and G_{set_2}), and produces delayed SETs $G_{set_1}(t - (T_1 + t_{D_{set}}))$ and $G_{set_2}(t - (T_2 + t_{D_{set}}))$ that are analogous to propagated SET pulses through a path delay $t_{D_{set}}$ and reach target-FF as in a real-time scenario of Fig. 5.9. The convolution and generation of delayed SETs is represented by step-1 and step-2 in Fig. 5.14. Similarly, step-4 indicates a valid-convolution ($\circledast_v$)[180] between $S_{SH}(t - T_0)$ in step-3 and propagated SETs in step-2, where convolution product is only given for points at which the signals overlap completely. The valid-convolution results in the generation of two impulse functions $\delta(t - (T_2 + t_{D_{set}}))$ in blue and $\delta(t - T_0)$ in black with amplitudes $A(\delta)$=0 and $A(\delta)$=1 respectively, where $A(\delta)$=1 represents the non-masked fault as given in (5.20). The case of $A(\delta) < 1$ represents the masked fault and various cases of step-4 are presented in Fig. 5.14.

$$\underbrace{G_{set}(t) \circledast \delta(t - t_{D_{set}}) \circledast_v \frac{1}{w} S_{SH}(t)}_{FPP_{i,set}} = \delta(t) * \begin{cases} A(\delta) = 1 \\ 0 < A(\delta) < 1 \end{cases} \tag{5.20}$$

where, $FPP_{i,set}$ is the Fault Propagating Probability of SET pulse at the i^{th} gate. So that, the complex equation (2.5) can be simplified as:

$$FFR_{i,set} = FIT \cdot FPP_i \cdot LDR_i \cdot FDR_i \tag{5.21}$$

$FPP_{i,set}$ generate a model which characterize TDR_i and EDR_i in (2.5). For the case of aging, the path delay $t_{D_{set}}$ in (5.20) models NBTI caused time-slot shift of propagating SETs and $\frac{1}{w}$ factor in (5.20) models the width of propagating SETs.

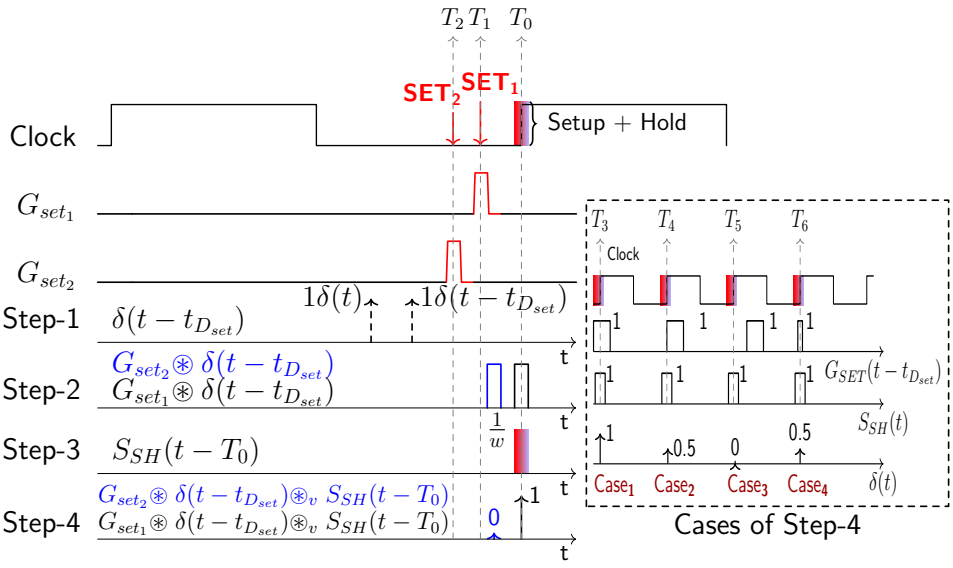


Figure 5.14 – SET Propagating Model [9]

5.7 Chapter Conclusions

Based on the observed results of this research, it has been concluded that voltage variability due to aging shows significant masking property on SEU fault propagation and reduces the effect of increased soft-error susceptibility while aging. Furthermore, this work explicitly reveals that aging shifts the SETs' propagating spots within the clock period. Besides, the latching probability of a SET fault to a downstream flip-flop is successfully modeled by a signal-processing method. The revised AI models and SDF-based aged technology data from this work are considering in the future to emulate aging-aware timing models at the system's RTL abstraction, which enables an early aging analysis in the design process.

Chapter 6

Conclusion and Perspective

6.1 Conclusions

The industrial-level scientific researches and innovative pedagogical methods in this doctoral dissertation: aim to develop effective mathematical (or) statistical strategies for accelerated soft-error reliability analysis and focus on characterizing fault propagation probabilities under time-dependent circuit variabilities. The thesis develops an Artificial Intelligence (AI) framework that performs as a tool for soft-error reliability analysis as well as a tool for quality analysis in predicting the single stuck-at faults effects and estimating fault coverage metrics of input test patterns. The framework is scalable and efficient in resource handling and ultimately reduces the required simulation time overhead by 60%. The thesis is successful in contributing innovative approaches to draw significant relationships between Bias Temperature Instability BTI and soft-error analysis. Based on a quantitative and qualitative analysis of the aged circuit's Functional Failure data in response to radiation-induced errors from the fault-injection simulations, it can be concluded that derating factors such as LDR, EDR, TDR and FDR are significantly sensitive to the time-dependent voltage variabilities and these are important factors to consider when designing and targeting long term reliable systems.

The thesis itself is a challenging journey, and at each point, it provides significant contributions. An intermediate circuit-graph representation of gate-level netlist and its non-complex delegation in GML format boosts the chances of testing different efficient statistical algorithms on the circuit directly. Such computationally relevant solutions tackle the challenges of applying the complex mathematical models and algorithms explicitly on the large circuit's netlist instead of testing on small circuit test benches from the state-of-the-art articles. The testing medium and large circuits are the principal advancements of the developed framework in this thesis compared to previous case study results.

The main disadvantage of the prevalently used old statistical algorithms is the inconsistency in producing a globally optimized solution in closed-loop problems. Such drawbacks do not greatly favor solving the error propagation probability in convergence or re-convergence circuit paths. Due to these scientific facts, the framework adopted the latest graph-based convolutional neural network algorithms to process the circuit graph. An unsupervised training phase of the neural-network algorithm learns the probabilistic inference in the closed-loop (or) re-convergence circuit path. Node2vec, GCN and graphSAGE are the applied graph embedding neural networks. The application such graph embedding algorithm generates a feature dataset that applies to downstream feed-forward DNN. After the statical analysis, it is concluded that a remarkable numerical superiority

is found in the fault propagation probability inference and reduces the simulation time-overhead by 60%. The estimated FFR metrics' accuracy is ranging from 94% to 98%.

The radiation-induced error characterization and their effect analysis are challenging problems in the system design flow. However, AI-Framework in the thesis extends its functional verification to multi-dimensional aspects by incorporating quality analysis for system design. The extension includes a graph transformation tool. The thesis framework validates an artificial intelligence approach to predict the Functional Failure Rate (FFR) of each Net (connections between components) at the gate-level abstraction and the effectiveness of the test patterns. The proposed technique succeeds in reducing the time-complexity of exhaustive fault-injection by 60%. The accuracy of FFR estimation ranges from 93% to 94% and estimates a very close value to the FFC of test pattern with an error less than 2%. This extended AI-Framework is a scalable and very cost-effective simulation tool for the production of dependable micro-electronics systems.

The inter-dependency between the radiation-induced errors and corresponding propagation probability parameters fluctuates with the time-dependent circuit variabilities. The thesis contributes a significant conclusion that aging caused voltage variability shows influential masking properties on SEU fault propagations and reduces the effect of increased soft-error susceptibility while aging. Furthermore, the experimental studies in this part of the thesis explicitly reveal that aging shifts the SETs' propagating spots within the clock period. Besides that, the latching probability of a SET fault to a downstream flip-flop is successfully modeled by a signal-processing method. And a new metric called 'Polarization Window' is introduced to account for the SEU masking characteristics of the aged circuit.

The holistic approach in the thesis predominantly contributes to an artificial intelligence based tool that is scalable and very effective in resource management for radiation-induced error reliability analysis. The framework is competent for predicting single stuck-at faults' effects and estimating fault coverage of test-patters in quality analysis. The adequate experimental results and their decisive and extensive analysis provide groundbreaking facts about the relations of soft-error propagation and aging caused circuit variabilities.

6.2 Future Work

The future perspective of the doctoral thesis is realizing a vision to accomplish the development of a synthetic hierarchical abstraction tool from gate-level to architectural-level. The thesis work exports aging aware time models across the cross-levels from transistor-level to gate-level. At the gate-level, a sophisticated soft-error reliability analysis tool predicts the effects of soft-error propagation. Now, the framework in this thesis has the opportunity to export these error-propagation characteristics and aging-aware timing models at gate-level and transistor-level to the RTL and architectural level of a system design. This enables the researchers and scientists to perform an early-stage reliability analysis in the system design flow. The embedded characteristics of the circuit graph can be easily exploited to RTL levels to analyze the SEU fault generation at flip-flops and its propagation through the circuit. For time-dependent soft-error analysis, the propagation timing paths and its models between flip-flops (present parallelly in both gate-level and RTL) pass on to the RTL level. To compete with the traditional fault-injection methods at the architectural-level compact AI algorithms like Reinforcement Learning Method is planned to adapt. In this way, the holistic approach in this thesis generates a way more effective and design favorable accelerated analysis tool across the hierarchical abstraction layers. A very detailed version of future visions and perspectives is provided in Fig. 6.1.

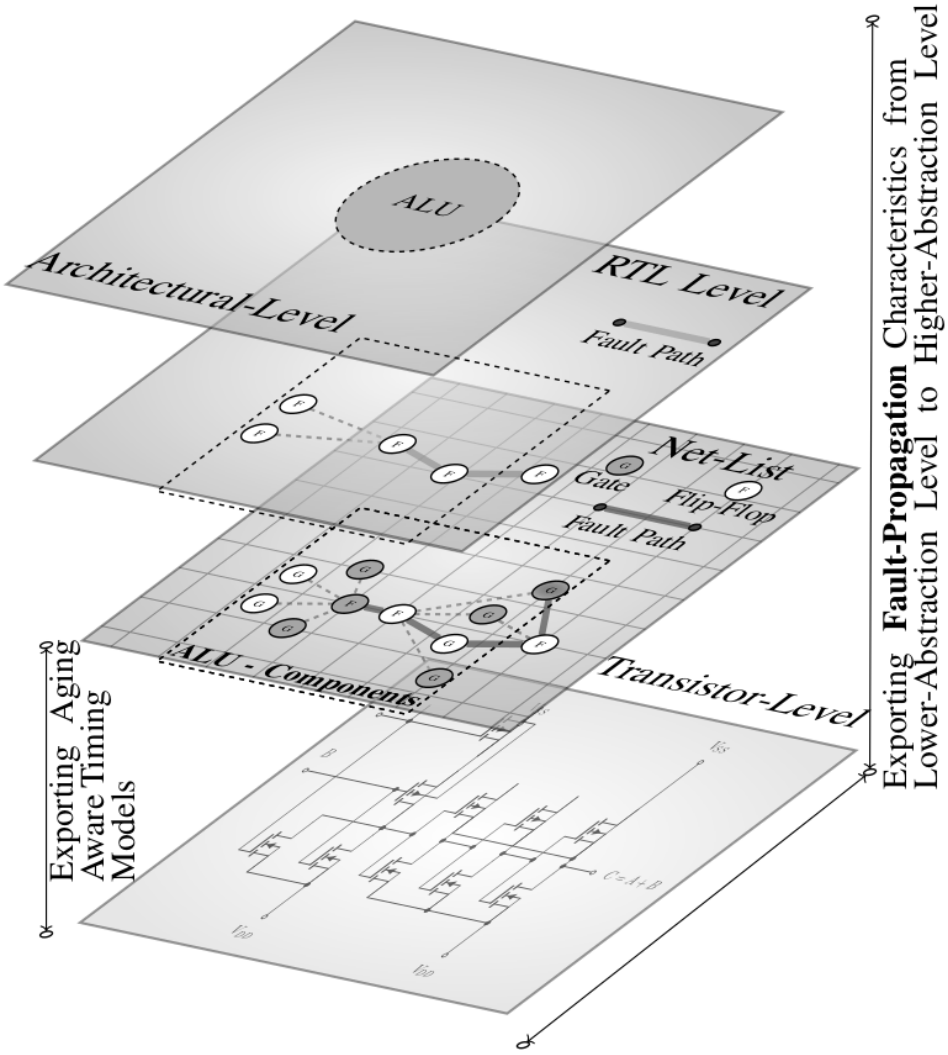


Figure 6.1 – Reliability Modeling Across Hierarchical Abstraction Levels

List of Figures

List of Tables

References

[1] A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, "The validation of graph model-based, gate level low-dimensional feature data for machine learning applications," in *2019 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, IEEE, oct 2019.

[2] A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, "Modeling gate-level abstraction hierarchy using graph convolutional neural networks to predict functional de-rating factors," in *2019 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, IEEE, jul 2019.

[3] A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, "Composing graph theory and deep neural networks to evaluate SEU type soft error effects," in *2020 9th Mediterranean Conference on Embedded Computing (MECO)*, IEEE, jun 2020.

[4] M. Jenihhin, M. S. Reorda, A. Balakrishnan, and D. Alexandrescu, "Challenges of reliability assessment and enhancement in autonomous systems," in *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, IEEE, oct 2019.

[5] T. Lange, A. Balakrishnan, M. Glorieux, D. Alexandrescu, and L. Sterpone, "On the estimation of complex circuits functional failure rate by machine learning techniques," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks – Supplemental Volume (DSN-S)*, IEEE, June 2019.

[6] X. Lai, A. Balakrishnan, T. Lange, M. Jenihhin, T. Ghasempouri, J. Raik, and D. Alexandrescu, "Understanding multidimensional verification: Where functional meets non-functional," *Microprocessors and Microsystems*, vol. 71, p. 102867, nov 2019.

[7] A. Balakrishnan, D. Alexandrescu, M. Jenihhin, T. Lange, and M. Glorieux, "Gate-level graph representation learning: A step towards the improved stuck-at faults analysis," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pp. 24–30, 2021.

[8] D. Alexandrescu, A. Balakrishnan, T. Lange, and M. Glorieux, "Enabling cross-layer reliability and functional safety assessment through ML-based compact models," in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, IEEE, jul 2020.

[9] A. Balakrishnan, G. C. Medeiros, C. C. Gürsoy, S. Hamdioui, M. Jenihhin, and D. Alexandrescu, "Modeling soft-error reliability under variability," in *2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 1–6, ***Award: Outstanding Student Research Paper***, 2021.

[10] T. Lange, A. Balakrishnan, M. Glorieux, D. Alexandrescu, and L. Sterpone, "Machine learning to tackle the challenges of transient and soft errors in complex circuits," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, IEEE, jul 2019.

[11] T. Lange, A. Balakrishnan, M. Glorieux, D. Alexandrescu, and L. Sterpone, "Machine learning clustering techniques for selective mitigation of critical design features,"

in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 1–7, 2020.

[12] X. Lai, T. Lange, A. Balakrishnan, D. Alexandrescu, and M. Jenihhin, "On antagonism between side-channel security and soft-error reliability in bnn inference engines," in *2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 1–6, 2021.

[13] M. Jenihhin, S. Hamdioui, M. S. Reorda, M. Krstic, P. Langendörfer, C. Sauer, A. Klotz, M. Huebner, J. Nolte, H. T. Vierhaus, G. Selimis, D. Alexandrescu, M. Taouil, G. J. Schrijen, J. Raik, L. Sterpone, G. Squillero, and Z. Dyka, "Rescue: Interdependent challenges of reliability, security and quality in nanoelectronic systems," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 388–393, 2020.

[14] J. F. Ziegler, H. W. Curtis, H. P. Muhlfeld, C. J. Montrose, and B. Chin, "Ibm experiments in soft fails in computer electronics (1978–1994)," *IBM J. Res. Dev.*, vol. 40, p. 3–18, Jan. 1996.

[15] CISCO, "12000 single event upset failures overview and work around summary," *Cisco Open Circuit Cases*, April 15, 2003.

[16] P. Bradley and E. Normand, "Single event upsets in implantable cardioverter defibrillators," *IEEE Transactions on Nuclear Science*, vol. 45, no. 6, pp. 2929–2940, 1998.

[17] Govt. of Canada, "Notice - important safety information on certain st. jude medical implantable cardiac defibrillators - for health professional." `https://www.healthycanadians.gc.ca/recall-alert-rappel-avis/hc-sc/2005/14362a-eng.php`, posted: October 6 2005.

[18] R. Harboe-Sørensen, "Radiation effects in spacecraft electronics," *5th LHC RADIATION WORKSHOP-CERN*, Nov 29, 2005.

[19] M. Nicolaidis, *Soft Errors in Modern Electronic Systems*. Springer, Boston, MA, 2011.

[20] D. Alexandrescu and E. Costenaro, "Towards optimized functional evaluation of see-induced failures in complex designs," in *2012 IEEE 18th International On-Line Testing Symposium (IOLTS)*, pp. 182–187, 2012.

[21] D. Alexandrescu, E. Costenaro, and M. Nicolaidis, "A practical approach to single event transients analysis for highly complex designs," in *2011 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, pp. 155–163, 2011.

[22] M. Ebrahimi, A. Evans, M. B. Tahoori, E. Costenaro, D. Alexandrescu, V. Chandra, and R. Seyyedi, "Comprehensive analysis of sequential and combinational soft errors in an embedded processor," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1586–1599, 2015.

[23] J. Pearl, "Chapter 3 - markov and bayesian networks: Two graphical representations of probabilistic knowledge," in *Probabilistic Reasoning in Intelligent Systems* (J. Pearl, ed.), pp. 77 – 141, San Francisco (CA): Morgan Kaufmann, 1988.

[24] O. Loyola-González, "Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view," *IEEE Access*, vol. 7, pp. 154096–154113, 2019.

[25] R. Baumann, T. Hossain, S. Murata, and H. Kitagawa, "Boron compounds as a dominant source of alpha particles in semiconductor devices," in *Proceedings of 1995 IEEE International Reliability Physics Symposium*, pp. 297–302, 1995.

[26] D. Binder, E. C. Smith, and A. B. Holman, "Satellite anomalies from galactic cosmic rays," *IEEE Transactions on Nuclear Science*, vol. 22, no. 6, pp. 2675–2680, 1975.

[27] T. C. May and M. H. Woods, "A new physical mechanism for soft errors in dynamic memories," in *16th International Reliability Physics Symposium*, pp. 33–40, 1978.

[28] J. F., W. A. Ziegler, and Lanford, "Effect of cosmic rays on computer memories," *American Association for the Advancement of Science*, 1979.

[29] J. F. Ziegler and W. A. Lanford, "The effect of sea level cosmic rays on electronic devices," *Journal of Applied Physics*, vol. 52, no. 6, pp. 4305–4312, 1981.

[30] W. A. Kolasinski, J. B. Blake, J. K. Anthony, W. E. Price, and E. C. Smith, "Simulation of cosmic-ray induced soft errors and latchup in integrated-circuit computer memories," *IEEE Transactions on Nuclear Science*, vol. 26, no. 6, pp. 5087–5091, 1979.

[31] C. S. Guenzer, E. A. Wolicki, and R. G. Allas, "Single event upset of dynamic rams by neutrons and protons," *IEEE Transactions on Nuclear Science*, vol. 26, no. 6, pp. 5048–5052, 1979.

[32] G. Sai-Halasz, "Cosmic ray induced soft error rate in vlsi circuits," *IEEE Electron Device Letters*, vol. 4, no. 6, pp. 172–174, 1983.

[33] P. C. Murley and G. R. Srinivasan, "Soft-error monte carlo modeling program, semm," *IBM Journal of Research and Development*, vol. 40, no. 1, pp. 109–118, 1996.

[34] J. F. Dicello, C. W. McCabe, J. D. Doss, and M. Paciotti, "The relative efficiency of soft-error induction in 4k static rams by muons and pions," *IEEE Transactions on Nuclear Science*, vol. 30, no. 6, pp. 4613–4615, 1983.

[35] T. O'Gorman, "The effect of cosmic rays on the soft error rate of a dram at ground level," *IEEE Transactions on Electron Devices*, vol. 41, no. 4, pp. 553–557, 1994.

[36] T. J. O'Gorman, J. M. Ross, A. H. Taber, J. F. Ziegler, H. P. Muhlfeld, C. J. Montrose, H. W. Curtis, and J. L. Walsh, "Field testing for cosmic ray soft errors in semiconductor memories," *IBM Journal of Research and Development*, vol. 40, no. 1, pp. 41–50, 1996.

[37] L. Geppert, U. Bapst, D. Heidel, and K. Jenkins, "Ion microbeam probing of sense amplifiers to analyze single event upsets in a cmos dram," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 2, pp. 132–134, 1991.

[38] D. Heidel, U. Bapst, K. Jenkins, L. Geppert, and T. Zabel, "Ion microbeam radiation system," *IEEE Transactions on Nuclear Science*, vol. 40, no. 2, pp. 127–134, 1993.

[39] C. Lage, D. Burnett, T. McNelly, K. Baker, A. Bormann, D. Dreier, and V. Soorholtz, "Soft error rate and stored charge requirements in advanced high-density srams," in *Proceedings of IEEE International Electron Devices Meeting*, pp. 821–824, 1993.

[40] H. Tang and E. Cannon, "Semm-2: a modeling system for single event upset analysis," *IEEE Transactions on Nuclear Science*, vol. 51, no. 6, pp. 3342–3348, 2004.

[41] C. Weaver, J. Emer, S. Mukherjee, and S. Reinhardt, "Techniques to reduce the soft error rate of a high-performance microprocessor," in *ACM SIGARCH Computer Architecture News*, vol. 32, pp. 264– 275, 07 2004.

[42] S. Michalak, K. Harris, N. Hengartner, B. Takala, and S. Wender, "Predicting the number of fatal soft errors in los alamos national laboratory's asc q supercomputer," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 329–335, 2005.

[43] R. Baumann, "The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction," in *Digest. International Electron Devices Meeting,*, pp. 329–332, 2002.

[44] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, 2005.

[45] C. Hsieh, P. Murley, and R. O'Brien, "A field-funneling effect on the collection of alpha-particle-generated carriers in silicon devices," *IEEE Electron Device Letters*, vol. 2, no. 4, pp. 103–105, 1981.

[46] J. R. Schwank, M. R. Shaneyfelt, and P. E. Dodd, "Radiation hardness assurance testing of microelectronic devices and integrated circuits: Radiation environments, physical mechanisms, and foundations for hardness assurance," *IEEE Transactions on Nuclear Science*, vol. 60, no. 3, pp. 2074–2100, 2013.

[47] E. Stassinopoulos and J. Raymond, "The space radiation environment for electronics," *Proceedings of the IEEE*, vol. 76, no. 11, pp. 1423–1442, 1988.

[48] P. Meyer, R. Ramaty, and W. R. Webber, "Cosmic rays - astronomy with energetic particles," *Physics Today*, vol. 27, no. 10, pp. 23–32, 1974.

[49] F. W. Sexton, "Measurement of single event phenomena in devices and ics," *IEEE NSREC Short Course*, pp. pp III–1 – III–55, 1992.

[50] J. L. Barth, "Modeling space radiation environments," *IEEE NSREC Short Course, Snowmass,CO*, July 1997.

[51] Y.-P. Fang and A. S. Oates, "Thermal neutron-induced soft errors in advanced memory and logic devices," *IEEE Transactions on Device and Materials Reliability*, vol. 14, no. 1, pp. 583–586, 2014.

[52] S.-J. Wen, S. Pai, R. Wong, M. Romain, and N. Tam, "B10 finding and correlation to thermal neutron soft error rate sensitivity for srams in the sub-micron technology," in *2010 IEEE International Integrated Reliability Workshop Final Report*, pp. 31–33, 2010.

[53] S. Wen, R. Wong, M. Romain, and N. Tam, "Thermal neutron soft error rate for srams in the 90nm–45nm technology range," in *2010 IEEE International Reliability Physics Symposium*, pp. 1036–1039, 2010.

[54] Keran and O'Brien, "The natural radiation environment, report n 720805-p1," *Rapport technique. United States Department of Energy*, 1971.

[55] Keran and O'Brien, "Luin: A code for the calculation of cosmic ray propagation in the atmosphere, report n eml-338," *United States Department of Energy*, 1978.

[56] J.-L. Autran, "Effects of low energy muons on electronics: Physical insights and geant4 simulation," 13^{th} *European Conference on Radiation and its Effects on Components and Systems (RADECS 2012)*, 09 2012.

[57] E. COSTENARO, "Techniques for the evaluation and the improvement of emergent technologies' behavior facing random errors," Master's thesis, L'UNIVERSITÉ DE GRENOBLE ALPES, Augest 2006.

[58] R. Shuler, B. Bhuva, and B. Narasimham, "Pulse distortion in set measurements from layout and adjacent signals," in *NASA*, 2009.

[59] T. Grasser, *Bias Temperature Instability for Devices and Circuits*. Springer, New York, NY, 22nd February 2008.

[60] S. Krishnaswamy, G. Viamontes, I. Markov, and J. Hayes, "Accurate reliability evaluation and enhancement via probabilistic transfer matrices," in *Design, Automation and Test in Europe*, IEEE, 2005.

[61] R. Bahar, E. Frohm, C. Gaona, G. Hachtel, E. Macii, A. Pardo, and F. Somenzi, "Algebraic decision diagrams and their applications," in *Proceedings of 1993 International Conference on Computer Aided Design (ICCAD)*, IEEE Comput. Soc. Press, 1993.

[62] D. T. Franco, M. C. Vasconcelos, L. Naviner, and J. Naviner, "Reliability of logic circuits under multiple simultaneous faults," in *2008 51st Midwest Symposium on Circuits and Systems*, pp. 265–268, 2008.

[63] D. T. Franco, M. C. Vasconcelos, L. Naviner, and J. Naviner, "Signal probability for reliability evaluation of logic circuits," *Microelectronics Reliability*, vol. 48, pp. 1586–1591, 2008.

[64] J. T. Flaquer, J. Daveau, L. Naviner, and P. Roche, "Fast reliability analysis of combinatorial logic circuits using conditional probabilities," *Microelectronics Reliability*, vol. 50, pp. 1215–1218, sep 2010.

[65] J. Han, H. Chen, E. Boykin, and J. Fortes, "Reliability evaluation of logic circuits using probabilistic gate models," *Microelectronics Reliability*, vol. 51, pp. 468–476, feb 2011.

[66] F. F. Sellers, M. Y. Hsiao, and L. W. Bearnson, "Analyzing errors with the boolean difference," *IEEE Transactions on Computers*, vol. C-17, no. 7, pp. 676–683, 1968.

[67] N. Mohyuddin, E. Pakbaznia, and M. Pedram, "Probabilistic error propagation in logic circuits using the boolean difference calculus," in *2008 IEEE International Conference on Computer Design*, pp. 7–13, 2008.

[68] R. I. Bahar, J. Mundy, and Jie Chen, "A probabilistic-based design methodology for nanoscale computation," in *ICCAD-2003. International Conference on Computer Aided Design (IEEE Cat. No.03CH37486)*, pp. 480–486, 2003.

[69] X. Lu and X. Song, "A probabilistic logic for nanoscale devices," in *2007 IEEE International Conference on Integrated Circuit Design and Technology*, pp. 1–4, 2007.

[70] T. Rejimon, K. Lingasubramanian, and S. Bhanja, "Probabilistic error modeling for nano-domain logic circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 1, pp. 55–65, 2009.

[71] R. Xiao and C. Chen, "Gate-level circuit reliability analysis: A survey," *VLSI Des.*, vol. 2014, jan 2014.

[72] A. Vallero, A. Savino, A. Chatzidimitriou, M. Kaliorakis, M. Kooli, M. Riera, M. Anglada, G. Di Natale, A. Bosio, R. Canal, A. Gonzalez, D. Gizopoulos, R. Mariani, and S. Di Carlo, "Syra: Early system reliability analysis for cross-layer soft errors resilience in memory arrays of microprocessor systems," *IEEE Transactions on Computers*, vol. 68, no. 5, pp. 765–783, 2019.

[73] A. Vallero, A. Savino, G. Politano, S. Di Carlo, A. Chatzidimitriou, S. Tselonis, M. Kaliorakis, D. Gizopoulos, M. Riera, R. Canal, A. Gonzalez, M. Kooli, A. Bosio, and G. Di Natale, "Cross-layer system reliability assessment framework for hardware faults," in *2016 IEEE International Test Conference (ITC)*, (Fort Worth, TX, USA), pp. 1–10, IEEE, 2016.

[74] M. Cox and B. De Vries, "Robust expectation propagation in factor graphs involving both continuous and binary variables," in *2018 26th European Signal Processing Conference (EUSIPCO)*, pp. 2583–2587, 2018.

[75] P. Wijayatunga, "Probabilistic graphical models and their inferences (tutorial)," in *2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, pp. 251–252, 2019.

[76] K. Yoon, R. Liao, Y. Xiong, L. Zhang, E. Fetaya, R. Urtasun, R. Zemel, and X. Pitkow, "Inference in probabilistic graphical models by graph neural networks," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pp. 868–875, 2019.

[77] C. Chen, Y. Liu, X. Zhang, and S. Xie, "Scalable explanation of inferences on large graphs," in *2019 IEEE International Conference on Data Mining (ICDM)*, pp. 982–987, 2019.

[78] M. Stender, J. Graßhoff, T. Braun, R. Möller, and P. Rostalski, "A hybrid factor graph model for biomedical activity detection," in *2021 IEEE EMBS International Conference on Biomedical and Health Informatics (BHI)*, pp. 1–4, 2021.

[79] A. Chigurupati, R. Thibaux, and N. Lassar, "Predicting hardware failure using machine learning," in *2016 Annual Reliability and Maintainability Symposium (RAMS)*, pp. 1–6, 2016.

[80] F. Lin, A. Davoli, I. Akbar, S. Kalmanje, L. Silva, J. Stamford, Y. Golany, J. Piazza, and S. Sankar, "Predicting remediations for hardware failures in large-scale datacenters," in *2020 50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*, pp. 13–16, 2020.

[81] X. Sun, K. Chakrabarty, R. Huang, Y. Chen, B. Zhao, H. Cao, Y. Han, X. Liang, and L. Jiang, "System-level hardware failure prediction using deep learning," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2019.

[82] G. Thiel and F. Griggio, "Novel cumulative degradation approach to predict components failure rates," in *2019 IEEE International Reliability Physics Symposium (IRPS)*, pp. 1–7, 2019.

[83] N. Georgoulopoulos, A. Hatzopoulos, K. Karamitsios, I. M. Tabakis, K. Kotrotsios, and A. I. Metsai, "A survey on hardware failure prediction of servers using machine learning and deep learning," in *2021 10th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pp. 1–5, 2021.

[84] N. A. Davis, A. Rezgui, H. Soliman, S. Manzanares, and M. Coates, "Failuresim: A system for predicting hardware failures in cloud data centers using neural networks," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pp. 544–551, 2017.

[85] M. Sewak, S. K. Sahay, and H. Rathore, "Comparison of deep learning and the classical machine learning algorithm for the malware detection," in *2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pp. 293–296, 2018.

[86] Y. Chang and H. Fang, "A hybrid prognostic method for system degradation based on particle filter and relevance vector machine," *Reliability Engineering & System Safety*, vol. 186, pp. 51–63, 2019.

[87] L. Jiahe, "Machine learning aided design optimization for micro-chip reliability improvement," in *2020 3rd World Conference on Mechanical Engineering and Intelligent Manufacturing (WCMEIM)*, pp. 131–135, 2020.

[88] G. Asadi and M. Tahoori, "An accurate ser estimation method based on propagation probability [soft error rate]," in *Design, Automation and Test in Europe*, pp. 306–307 Vol. 1, 2005.

[89] D. Alexandrescu, E. Costenaro, and M. Nicolaidis, "A practical approach to single event transients analysis for highly complex designs," in *2011 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, pp. 155–163, 2011.

[90] H. Nakamura, K. Tanaka, T. Uemura, K. Takeuchi, T. Fukuda, and S. Kumashiro, "Measurement of neutron-induced single event transient pulse width narrower than 100ps," in *2010 IEEE International Reliability Physics Symposium*, pp. 694–697, 2010.

[91] M. Nicolaidis, *Soft Errors in Modern Electronic Systems*. Springer Publishing Company, Incorporated, 1st ed., 2010.

[92] H. Schirmeier and M. Breddemann, "Quantitative cross-layer evaluation of transient-fault injection techniques for algorithm comparison," in *2019 15th European Dependable Computing Conference (EDCC)*, pp. 15–22, 2019.

[93] J. Roux, V. Beroulle, K. Morin-Allory, R. Leveugle, L. Bossuet, F. Cézilly, F. Berthoz, G. Genévrier, and F. Cerisier, "High level fault injection method for evaluating critical system parameter ranges," in *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 1–4, 2020.

[94] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, p. 1–7, aug 2011.

[95] S. Harini and A. Ravikumar, "Effect of parallel workload on dynamic voltage frequency scaling for dark silicon ameliorating," in *2020 International Conference on Smart Electronics and Communication (ICOSEC)*, pp. 1012–1017, 2020.

[96] R. Walrath, *Standard Scores*. Boston, MA: Springer US, 2011.

[97] C. C. Aggarwal, *Data Mining: The Textbook*. Cham: Springer, 2015.

[98] N. Paulino, J. C. Ferreira, and J. M. P. Cardoso, "Optimizing opencl code for performance on fpga: k-means case study with integer data sets," *IEEE Access*, vol. 8, pp. 152286–152304, 2020.

[99] E. Bayram and V. Nabiyev, "Image segmentation by using k-means clustering algorithm in euclidean and mahalanobis distance calculation in camouflage images," in *2020 28th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2020.

[100] K. Venkatachalam, V. P. Reddy, M. Amudhan, A. Raguraman, and E. Mohan, "An implementation of k-means clustering for efficient image segmentation," in *2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT)*, pp. 224–229, 2021.

[101] N. Pang, J. Zhang, C. Zhang, and X. Qin, "Parallel hierarchical subspace clustering of categorical data," *IEEE Transactions on Computers*, vol. 68, no. 4, pp. 542–555, 2019.

[102] S. Park and Y. B. Park, "Photovoltaic power data analysis using hierarchical clustering," in *2018 International Conference on Information Networking (ICOIN)*, pp. 727–731, 2018.

[103] Y. Rong and Y. Liu, "Staged text clustering algorithm based on k-means and hierarchical agglomeration clustering," in *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, pp. 124–127, 2020.

[104] H. Ji, O. Alfarraj, and A. Tolba, "Artificial intelligence-empowered edge of vehicles: Architecture, enabling technologies, and applications," *IEEE Access*, vol. 8, pp. 61020–61034, 2020.

[105] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, 2018.

[106] S. Bhattacharyya, V. Snasel, A. Hassanien, S. Saha, and B. Tripathy, *Deep Learning: Research and Applications*. De Gruyter Frontiers in Computational Intelligence, De Gruyter, 2020.

[107] J. McCarthy, "What is artificial intelligence? technical report," *Stanford University*, pp. Accessed on Jan, 2022, 2007.

[108] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer New York, 1995.

[109] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," *CoRR*, vol. abs/1607.00653, 2016.

[110] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *CoRR*, vol. abs/1609.02907, 2016.

[111] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pp. 338–342, 01 2014.

[112] R. C. Team, "R: A language and environment for statistical computing," *R Foundation for Statistical Computing, Vienna, Austria*, 2016.

[113] D. T. Franco, M. C. Vasconcelos, L. Naviner, and J. Naviner, "Reliability of logic circuits under multiple simultaneous faults," in *2008 51st Midwest Symposium on Circuits and Systems*, pp. 265–268, 2008.

[114] R. I. Bahar, J. Mundy, and Jie Chen, "A probabilistic-based design methodology for nanoscale computation," in *ICCAD-2003. International Conference on Computer Aided Design (IEEE Cat. No.03CH37486)*, pp. 480–486, 2003.

[115] I. 61508-1:2010, "Functional safety of electrical/electronic/programmable electronic safety-related systems - part 1: General requirements," *International Electrotechnical Commission*, 2010.

[116] I. 26262-1:2011/2018, "Functional safety of electrical/electronic/programmable electronic safety-related systems - part 1: General requirements," *International Standardization Organization*, 2018.

[117] IEC, "En iso 13849: safety-related control systems for machinery," *http://www.iec.ch/search/?q=13849*, 2010.

[118] G. Aleksandrowicz, E. Arbel, R. Bloem, T. D. ter Braak, S. Devadze, G. Fey, M. Jenihhin, A. Jutman, H. G. Kerkhoff, R. Könighofer, S. Koyfman, J. Malburg, S. Moran, J. Raik, G. Rauwerda, H. Riener, F. Röck, K. Shibin, K. Sunesen, J. Wan, and Y. Zhao, "Designing reliable cyber-physical systems," in *Lecture Notes in Electrical Engineering*, pp. 15–38, Springer International Publishing, nov 2017.

[119] RTCA, "Do-254," *Advisory Circular (AC) 20-152 Design Assurance Guidance for Airborne Electronic Hardware*, 19 April 2000.

[120] C. Dawson, S. Pattanam, and D. Roberts, "The verilog procedural interface for the verilog hardware description language," in *Proceedings. IEEE International Verilog HDL Conference*, pp. 17–23, 1996.

[121] M. Himsolt, "Gml:," in *A portable Graph File Format*, 2010.

[122] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," in *ICWSM*, 2009.

[123] G. Csardi and T. Nepusz, "The igraph software package for complex network research," *InterJournal*, vol. Complex Systems, p. 1695, 2006.

[124] A. Tanguay, "10ge mac core specification," *OpenCores.org*, 2008.

[125] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *CoRR*, vol. abs/1606.09375, 2016.

[126] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," 2015.

[127] OpenCores.org, "Opencores." `https://opencores.org/`. Last visited on Nov. 2021.

[128] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *CoRR*, vol. abs/1706.02216, 2017.

[129] R. J. Schalkoff, *Artificial Neural Networks*. McGraw-Hill Higher Education, 1st ed., 1997.

[130] R. Hecht-Nielsen, "Neurocomputing: picking the human brain," *IEEE Spectrum*, vol. 25, no. 3, pp. 36–41, 1988.

[131] J. Ledoux, "Software Reliability Modeling," in *Handbook of Reliability Engineering* (H. Pham, ed.), Reliability, pp. 213–234, Springer-Verlag London, 2003.

[132] O. Girard, "openmsp430," *OpenCores.org*, November 6, 2017.

[133] Texas Instruments, "User's guide." `https://www.ti.com/lit/ug/slau049f/slau049f.pdf?ts=1600178024619`, 2016. Last visited on Nov. 2021.

[134] Scikit-learn.org, "r2_score." `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html`. Late visited on Nov. 2021.

[135] C. Data61, "Stellargraph machine learning library." `https://github.com/stellargraph/stellargraph`, 2018. Last visited on Nov. 2021.

[136] A. Veneris and I. N. Hajj, "Design error diagnosis and correction via test vector simulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 12, pp. 1803–1816, 1999.

[137] S. Venkataraman, W. K. Fuchs, and J. H. Patel, "Diagnostic simulation of sequential circuits using fault sampling," in *Proceedings Eleventh International Conference on VLSI Design*, pp. 476–481, 1998.

[138] P. G. Ryan, S. Rawat, and W. K. Fuchs, "Two-stage fault location," in *1991, Proceedings. International Test Conference*, pp. 963–, 1991.

[139] J. A. Waicukauski and E. Lindbloom, "Failure diagnosis of structured vlsi," *IEEE Design Test of Computers*, vol. 6, no. 4, pp. 49–60, 1989.

[140] J. A. Waicukauski, V. P. Gupta, and S. T. Patel, "Diagnosis of bist failures by ppsfp simulation," in *18th IEEE International Test Conference*, p. 480–484, Sep.1987.

[141] M. Abramovici, "A maximal resolution guided-probe testing algorithm," in *Proceedings of the 18th Design Automation Conference*, DAC '81, p. 189–195, IEEE Press, 1981.

[142] Abramovici and Breuer, "Multiple fault diagnosis in combinational circuits based on an effect-cause analysis," *IEEE Transactions on Computers*, vol. C-29, no. 6, pp. 451–460, 1980.

[143] N. Kuji and T. Tamama, "Integrating an electron-beam system into vlsi fault diagnosis," *IEEE Design & Test of Computers*, vol. 3, pp. 23–29, jul 1986.

[144] T. Yamada, S. Hamada, T. Matsumoto, T. Takahashi, and T. Nakayama, "Method of diagnosing multiple stuck-at-faults in combinational circuits," *Systems and Computers in Japan*, vol. 22, no. 11, pp. 21–30, 1991.

[145] N. Itazaki, T. Sumioka, S. Kajihara, and K. Kinoshita, "Automatic fault location using e-beam and lsi testers," in *Proceedings of 1993 IEEE 2nd Asian Test Symposium (ATS)*, pp. 255–260, 1993.

[146] H. Takahashi, N. Yanagida, and Y. Takamatsu, "Multiple stuck-at fault diagnosis in combinational circuits based on restricted single sensitized paths," in *Proceedings of 1993 IEEE 2nd Asian Test Symposium (ATS)*, pp. 185–190, 1993.

[147] N. Yanagida, H. Takahashi, and Y. Takamatsu, "Multiple fault diagnosis by sensitizing input pairs," *IEEE Design Test of Computers*, vol. 12, no. 3, pp. 44–, 1995.

[148] S. Millman, E. McCluskey, and J. Acken, "Diagnosing cmos bridging faults with stuck-at fault dictionaries," in *1990 International Test Conference*, (Los Alamitos, CA, USA), IEEE Computer Society, sep 1990.

[149] I. Pomeranz and S. M. Reddy, "Location of stuck-at faults and bridging faults based on circuit partitioning," *IEEE Transactions on Computers*, vol. 47, no. 10, pp. 1124–1135, 1998.

[150] Jue Wu and E. M. Rudnick, "Bridge fault diagnosis using stuck-at fault simulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 4, pp. 489–495, 2000.

[151] H. Takahashi, K. O. Boateng, K. K. Saluja, and Y. Takamatsu, "On diagnosing multiple stuck-at faults using multiple and single fault simulation in combinational circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 3, pp. 362–368, 2002.

[152] S. Mukherjee, *Architecture Design for Soft Errors*. Morgan Kaufmann, 22nd February 2008.

[153] M. Jenihhin, G. Squillero, T. S. Copetti, V. Tihhomirov, S. Kostin, M. Gaudesi, F. Vargas, J. Raik, M. Sonza Reorda, L. Bolzani Poehls, R. Ubar, and G. C. Medeiros, "Identification and rejuvenation of nbti-critical logic paths in nanoscale circuits," *Journal of Electronic Testing*, vol. 32, p. 273–289, 2016.

[154] W. Chen, S. Ray, J. Bhadra, M. Abadir, and L.-C. Wang, "Challenges and trends in modern soc design verification," *IEEE Design Test*, vol. 34, no. 5, pp. 7–22, 2017.

[155] J. Bhadra, M. Abadir, L. C. Wang, and S. Ray, "A survey of hybrid techniques for functional verification," *IEEE Design & Test of Computers*, vol. 24, pp. 112–122, 04 2007.

[156] Springer, *Binomial Distribution*, pp. 44–45. New York, NY: Springer, 2008.

[157] J. L. Gross and J. Yellen, *Graph Theory and Its Applications, 2nd ed.* Boca Raton, FL: CRC Press, 2006.

[158] J. Schat, "On the relationship between stuck-at fault coverage and transition fault coverage," in *2009 Design, Automation Test in Europe Conference Exhibition*, pp. 1218–1221, 2009.

[159] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2014.

[160] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, 2005.

[161] A. P. Shah and P. Girard, "Impact of aging on soft error susceptibility in cmos circuits," in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 1–4, 2020.

[162] M. Omaña, D. Rossi, T. Edara, and C. Metra, "Impact of aging phenomena on latches' robustness," *IEEE Transactions on Nanotechnology*, 2016.

[163] D. Rossi, M. Omaña, C. Metra, and A. Paccagnella, "Impact of aging phenomena on soft error susceptibility," in *IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, pp. 18–24, 2011.

[164] A. Gebregiorgis, S. Kiamehr, F. Oboril, R. Bishnoi, and M. B. Tahoori, "A cross-layer analysis of soft error, aging and process variation in near threshold computing," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2016.

[165] A. Yan, Y. Ling, K. Yang, Z. Chen, and M. Yi, "Aging-temperature-and-propagation induced pulse-broadening aware soft error rate estimation for nano-scale cmos," in *2018 IEEE 27th Asian Test Symposium (ATS)*, pp. 86–91, 2018.

[166] F. L. Kastensmidt, J. Tonfat, T. Both, P. Rech, G. Wirth, R. Reis, F. Bruguier, P. Benoit, L. Torres, and C. Frost, "Aging and voltage scaling impacts under neutron-induced soft error rate in sram-based fpgas," in *2014 19th IEEE European Test Symposium (ETS)*, pp. 1–2, 2014.

[167] C. Auth, C. Allen, A. Blattner, D. Bergstrom, M. Brazier, M. Bost, M. Buehler, V. Chikarmane, T. Ghani, T. Glassman, R. Grover, W. Han, D. Hanken, M. Hattendorf, P. Hentges, R. Heussner, J. Hicks, D. Ingerly, P. Jain, S. Jaloviar, R. James, D. Jones, J. Jopling, S. Joshi, C. Kenyon, H. Liu, R. McFadden, B. McIntyre, J. Neirynck, C. Parker, L. Pipes, I. Post, S. Pradhan, M. Prince, S. Ramey, T. Reynolds, J. Roesler, J. Sandford, J. Seiple, P. Smith, C. Thomas, D. Towner, T. Troeger, C. Weber, P. Yashar, K. Zawadzki, and K. Mistry, "A 22nm high performance and low-power cmos technology featuring fully-depleted tri-gate transistors, self-aligned contacts and high density mim capacitors," in *Symposium on VLSI Technology (VLSIT)*, pp. 131–132, 2012.

[168] S. Sinha, B. Cline, G. Yeric, V. Chandra, and Y. Cao, "Design benchmarking to 7nm with finfet predictive technology models," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, 2012.

[169] JESD89-3B, "Test method for beam accelerated soft error rate," jedec test standard no. 89-3b, JEDEC Solid State Technology Association, 2007.

[170] C. E. Barton, R. T. Baldwin, D. R. Barraclough, S. Bushati, M. Chiappini, Y. Cohen, R. Coleman, G. Hulot, P. Kotze, V. P. Golovkov, A. Jackson, R. A. Langel, F. J. Lowes, D. J. McKnight, S. Macmillan, L. R. Newitt, N. W. Peddie, J. M. Quinn, and T. J. Sabaka, "International geomagnetic reference field, 1995 revision: International association of geomagnetism and aeronomy (iaga) division v, working group 8," *Geophysical Journal International*, vol. 125, no. 1, pp. 318–321, 1996.

[171] J. Wilkinson, "Flux calculation," Last Update: 2019.

[172] M. Vilchis, R. Venkatraman, E. Costenaro, and D. Alexandrescu, "A real-case application of a synergetic design-flow-oriented ser analysis," in *2012 IEEE 18th International On-Line Testing Symposium (IOLTS)*, pp. 43–48, 2012.

[173] S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, and S. Vrudhula, "Predictive modeling of the nbti effect for reliable design," in *IEEE Custom Integrated Circuits Conference*, 2006.

[174] W. Wang, V. Reddy, A. T. Krishnan, R. Vattikonda, S. Krishnan, and Y. Cao, "Compact modeling and simulation of circuit reliability for 65-nm cmos technology," *IEEE Transactions on Device and Materials Reliability*, vol. 7, no. 4, pp. 509–517, 2007.

[175] G. Chen, K. Y. Chuah, M. F. Li, D. S. H. Chan, C. H. Ang, J. Z. Zheng, Y. Jin, and D. L. Kwong, "Dynamic nbti of pmos transistors and its impact on device lifetime," in *IEEE International Reliability Physics Symposium Proceedings*, 2003.

[176] J. B. Bernstein, "Chapter 3 - failure mechanisms," in *Reliability Prediction from Burn-In Data Fit to Reliability Models* (J. B. Bernstein, ed.), pp. ix–x, Oxford: Academic Press, 2014.

[177] R. Vattikonda, W. Wang, and Y. Cao, "Modeling and minimization of pmos nbti effect for robust nanometer design," in *2006 43rd ACM/IEEE Design Automation Conference*, pp. 1047–1052, 2006.

[178] PTM, "Spice models of predictive technology model (ptm)," in *Website: ptm.asu.edu*, 2012. Last visited on Nov. 2021.

[179] S. Kostin, J. Raik, R. Ubar, M. Jenihhin, T. Copetti, F. Vargas, and L. B. Poehls, "Spice-inspired fast gate-level computation of nbti-induced delays in nanoscale logic," in *IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits Systems*, 2015.

[180] Numpy, "Valid-convolution," in *Website: numpy.org/doc/stable/reference /generated/numpy.convolve.html*, 2021. Last visited on Nov. 2021.

Acknowledgements

First and foremost, I am immensely grateful by expressing my sincere gratitude towards my principal supervisors from both academic and industrial environments: Prof. Maksim Jenihhin and Dr. Dan Alexandrescu, for mentoring and guiding me to complete my doctoral thesis with triumphant aspiration. The completion of the doctoral dissertation could not have been possible without the expertise of my beloved supervisors. My gratitude extends to the RESCUE European Training Network (ETN) project that has received funding from the European Union's Horizon 2020 Programme under the Marie Skłodowska-Curie actions for research, technological development, and demonstration.

I profoundly thank the following people for stretching a helping hand in difficult situations:

- Representatives and research fellows from the European Innovative Doctoral Training project called RESCUE/H2020-MSC-ITN-ETN (2017-2021), for their willingness to impart their knowledge.

- My colleagues: Maximilien Glorieux, Thomas Lange, and Aduong In, from IROC Technologies, Grenoble, France, for sharing their experience, support and for offering invaluable encouragement in the journey of thesis completion. The insight and knowledge from the industrial environment into the thesis subject topics steered me through scientific research and gave me an upper hand in publications.

- The Scientific staffs Prof. Jaan Raik and Dr. Kristina Vassiljeva, and research fellows from the Department of Computer Systems, Tallinn University of Technology, Estonia, for their friendly conversations at the end of meetings and personal support in my academic endeavors.

The meetings and conversations throughout the thesis project were vital and inspiring me to think outside the box from multiple perspectives to form comprehensive and objective critics.

Last but not least, I am humbly expressing my gratitude for all the emotional support from my father, Ambattayil Balakrishnan, and my mother, Ponnath Seethlaksmi. Finally, I could not have completed this dissertation without my lovable siblings: Abhilash Balakrishnan and Anju Bency, and the sister-in-law Anitha Abhilash, and their wise counsel and sympathetic ear.

Abstract

A Synthetic, Hierarchical Approach for Modelling and Managing Complex Systems' Quality and Reliability

Automated electronics systems and their applications advance to the stage of a decision-making process in the life-critical domains such as healthcare, transportation, automotive and security, and medical. The inevitable application of electronic devices proliferates the commercial business markets and economies of countries. So that, integrating standard reliability, safety, and quality into the function of electronic devices is the principal objective of electronics system manufacturers and vendors.

With the advent of semiconductor process technology into the Very Deep Sub-Micron (VDSM) level, more invigorated system performance for the integrated functionalities boosts the prevalent usage of electronics systems. However, such technology enhancements and their applications pose reliability challenges in the system design flow and sometimes generate catastrophic failure threats depending upon the criticality in application domains. These scientific conclusions commercially mandate the requirement of new Electronic Design Automation (EDA) tools and paradigms to ensure reliable cyber-physical systems and the effectiveness in resource (e.g., cost, area, and energy) management.

The aggressive technology scaling fosters the requirements of reliability standards and metrics and jeopardizes the process of system design. Well-defined EDA tools and methodologies are currently available for the automation of system design-flow. Apart from the current state-of-the-art of reliability analysis methodologies, the novelty of this thesis is that exploring and incorporating the emerging computing technologies from Artificial Intelligence (AI) area.

The ultimate aim of chronicled experimental case studies in this doctoral research is to develop a synthetic approach for modeling and managing the quality and reliability of complex systems. In the reliability analysis in the deep sub-micron level, soft-errors (Single Event Transient (SET) and Single Event Upset (SEU)) are characterized as a principal concern of complex system manufacturers. The Soft errors are radiation-induced single event effects that cause transient errors and erroneous latched values in circuits. Even though soft-errors do not distress the underlying semiconductor devices, the induced erroneous states in the circuit may corrupt computing data, which is hard to recover later. As the technology scaling advances, reduced nodal capacitances and supply voltages and increased chip frequency concomitant to densely packed chips increase the rate of soft errors. Besides that, the time-dependent voltage variations in the aged circuit also contribute towards an increased soft-error generation. To compromise with the effects of such errors due to harsh environmental repercussions, well apprehended characterization and modeling methods need to be added to the system design flow. Exhaustive fault injection campaigns are the traditionally accepted first principal reliability analysis methods. But these approaches are very cumbersome in terms of required simulation resources such as time, simulation tool licenses, and costs. A significant challenge in reliability analysis and automation tool development is the reduction of resource costs by ensuring good scalability and quality in estimating commercially standard metrics. This study is trying to solve this problem throughout the thesis.

The thesis's first challenge is to characterize an intermediate representation for gate-level circuits to apply the different statistical (or) mathematical algorithms. A graph network called circuit-graph throughout this thesis, is a comprehensive structural representation of the gate-level netlist. This intermediate representation allows the application of different statical algorithmic inferences. After tackling this primary challenge, sophisti-

cated graph embedding approaches such as node2vec, Graph Convolutional Neural Network (GCN) and the graphSAGE generate a database to apply an inference algorithm based on feed-forward Deep Neural Network (DNN). On the analysis of that inference, results lead to a conclusion that 60% percentage of fault-injection simulation time is saved by providing an accuracy in-between 94% - 95% in estimating the real-time Functional De-Rating (FDR) factor. However, the training phase of the developed framework depends on the 40% fault-injection campaign results and time. So how to choose this training dataset remains another challenge. The data mining approach and its applications solve this problem to a large extend. The clustering algorithms from the data mining area find the clusters of flip-flops with similar structural peculiarities that cause functional failures of the system. A combination of k-means and hierarchical clustering approaches produces the sets of flip-flops correlated in their feature sets. From these clusters, choosing 40% of training data becomes a straightforward process. Through this method, the prediction accuracy is improved to 97%.

Many scientific articles contribute towards multidimensional hardware verification and articulate the importance of considering various extra-functional aspects of the electronic system design. In the hardware part, different design errors, manufacturing defects, variations, and reliability issues, (or) malicious faults need significant consideration while standardizing the required reliability matrices. A quality analysis tool is integrated into the developed framework for analyzing the effects of single stuck-at faults (as manufacturing defects (or) variational defects) by implementing a graph transformation approach. Based on this tool, the developed AI-Framework predicts the effects of single stuck-at faults at the functional level of circuits and reduces the traditional simulation time-overhead by 60% with estimated metrics accuracy ranging from 93% to 94%. This method also contribute to predict the Fault Coverage Metric (FFC) of input pattern in circuit testing process.

Scientific efforts highlight the relevance of case studies that account for the time variation impingement due to aging on soft-error reliability. In the final part of the thesis work, contribution of Machine Learning (ML)/ Deep Learning (DL) is explored to investigate the soft-error reliability challenges at low-scaled (e.g., 5-nanometer) technologies with time-dependent voltage variabilities. The aggressive downscaling of technology nodes results in clock-frequency maximization and provides soft-error propagation a high sensitivity to the path delay variations in the aged design. In this part of the work, aging effects are limited to the Negative Bias Temperature Instability (NBTI). The observed results conclude that voltage variability due to aging shows significant masking characteristics on SEU fault propagation and reduces the effect of increased soft-error susceptibility while aging. Furthermore, this work explicitly proves that aging shifts the propagating spots of SETs' within the clock period.

The scientific efforts in this doctoral thesis develop an accelerated automation tool for soft-error reliability analysis. The automation tool is scalable and very cost-effective in simulation resource handling for medium and large-scale circuits. The comprehensive results explicate a plausible prediction of functional failures due to SEU and SET events. The framework's objective adds a multidimensional functional verification and additionally serves as a tool to analyze the effects of single-stuck faults at the functional level. Alongside that, very significant scientific observations and facts are provided regarding the effects of aging in soft-error propagation.

Kokkuvõte

Sünteetiline, hierarhiline lähenemine keerukate süsteemide kvaliteedi ja töökindluse modelleerimiseks ja haldamiseks

Automatiseeritud elektroonikasüsteemid ja nende rakendused liiguvad elutähtsate valdkondade, nagu tervishoid, transport, autotööstus ja turvalisus ning meditsiin, otsustusprotsessi etappi. Elektrooniliste seadmete vältimatu kasutuselevõtt tõukab tagant vastavate riikide äriturge ja majandust. Seega standardse töökindluse, ohutuse ja kvaliteedi integreerimine elektroonikaseadmete funktsionaalsusesse on elektroonikasüsteemide tootjate ja müüjate peamine eesmärk.

Seoses pooljuhttehnoloogia tulekuga väga sügavale alammikronilisele (VDSM) tasemele suurendab integreeritud funktsioonide süsteemi jõudluse kasv elektroonikasüsteemide kasutamist. Sellised tehnoloogilised täiustused ja nende rakendused tekitavad aga süsteemi projekteerimise voos usaldusväärsuse probleeme ja mõnikord tekitavad katastroofilisi rikkeohte olenevalt rakendusvaldkondade kriitilisusest. Need asjaolud nõuavad uusi elektroonilise disaini automatiseerimise (EDA) tööriistu ja paradigmasid, et tagada usaldusväärset küberfüüsikaliste süsteemide ja ressursside (nt kulu, pindala ja energia) haldamist.

Agressiivne tehnoloogia skaleerimine nõuab usaldusväärsuse standardite ja mõõdikute väljatöötamist ning seab ohtu süsteemi vigadeta projekteerimise. Süsteemi projekteerimisvoo automatiseerimiseks on praegu saadaval täpselt määratletud EDA tööriistad ja metoodikad. Lisaks praegustele usaldusväärsusanalüüsi metoodikatele on selle lõputöö uudsuseks tehisintellekti (AI) kasutavate arvutustehnoloogiate uurimine ja kaasamine.

Selle doktoritöö eksperimentaalsete juhtumiuuringute lõppeesmärk on töötada välja sünteetiline lähenemine keerukate süsteemide kvaliteedi ja töökindluse modelleerimiseks ning juhtimiseks. Usaldusväärsuse analüüsis VDSM tasemel iseloomustatakse nn pehmeid vigu (Ühe sündmuse mööduv (SET) ja ühe sündmuse häired (SEU)) keerukate süsteemide tootjate peamise probleemina. Pehmed vead on kiirgusest põhjustatud üksikute sündmuste tulemid, mis põhjustavad vooluahelates mööduvaid vigu ja ekslikke lukustusväärtusi. Kuigi pehmed vead ei riku aluseks olevaid pooljuhtseadmeid, võivad ahelas esile kutsutud vigased olekud rikkuda arvutusandmeid, mida on hiljem raske taastada. Tehnoloogia skaleerimise edenedes suurendavad sõlmede vähenenud mahtuvused ja toitepinged ning kiibi suurenenud sagedus, mis kaasneb tihedalt pakitud kiipidega, pehmete vigade esinemissagedust. Peale selle aitavad ajast sõltuvad pinge kõikumised vananenud vooluringis kaasa ka pehmete vigade suurenenud tekkele. Selliste karmide keskkonnamõjude tõttu tekkivate vigade mõju vähendamiseks tuleb süsteemi projekteerimisvoogu lisada hästi mõistetavad iseloomustus- ja modelleerimismeetodid. Põhjalikud rikete sisestamise kampaaniad on traditsiooniliselt aktsepteeritud esimesed peamised töökindlusanalüüsi meetodid. Kuid need lähenemisviisid on nõutavate simulatsiooniressursside, nagu aeg, simulatsioonitööriistade litsentsid ja kulud, osas väga probleemsed. Usaldusväärsuse analüüsi ja automatiseerimistööriistade arenduse oluliseks väljakutseks on ressursikulude vähendamine, tagades standardsete mõõdikute hindamisel hea mastaapsuse ja kvaliteedi. Käesolev uuring püüab nimetatud probleemi lahendada.

Lõputöö esimene väljakutse on iseloomustada loogikalülituste taseme ahelate esitust, et rakendada erinevaid statistilisi (või) matemaatilisi algoritme. Graafikuvõrk, mida lõputöös nimetatakse skeemigraafiks, on loogikalülituste taseme võrguloendi terviklik struktuurne esitus. See esitusviis võimaldab rakendada erinevaid staatilisi algoritmilisi järeldusi. Pärast selle peamise väljakutsega tegelemist loovad keerukad graafiku manustamise lähenemisviisid, nagu node2vec, graafiline konvolutsiooniline närvivõrk (GCN) ja graphSAGE, andmebaasi, et rakendada edasisuunalisel süvanärvivõrgul (DNN) põhinevat järeldusalgo-

ritmi. Analüüsi tulemusena jõuti järeldusele, et 60% rikke sisestamise simulatsiooniajast säästetakse, tagades reaalajas funktsionaalse amortisatsiooniteguri (FDR) hindamise täpsuse vahemikus 94% - 95%. Väljatöötatud raamistiku õpetusfaas sõltub aga 40% ulatuses rikke sisestamise kampaania tulemustest ja ajast. Seega jääb närvivõrgu õpetamise andmestiku valimine teiseks väljakutseks. Andmekaevandamise lähenemisviis ja selle rakendused lahendavad selle probleemi suures osas. Andmekaeve valdkonnast pärit klasterdamisalgoritmid leiavad sarnaste struktuuriliste iseärasustega klastrite klastrid, mis põhjustavad süsteemi funktsionaalseid rikkeid. K-keskmiste ja hierarhiliste rühmitamise lähenemisviiside kombinatsioon loob nende funktsioonikomplektidega korrelatsioonis olevad trigerid. Nende klastrite tõttu muutub 40% treeningandmete valimine lihtsaks protsessiks. Nimetatud meetodi abil paraneb ka ennustustäpsus 97%-ni.

Paljud teadusartiklid aitavad kaasa mitmemõõtmelisele riistvara verifitseerimisele ja rõhutavad elektroonilise süsteemi disaini erinevate funktsionaalsete aspektide arvestamise tähtsust. Riistvara osas tuleb nõutavate töökindlusmaatriksite standardiseerimisel märkimisväärselt arvesse võtta erinevaid projekteerimisvigu, tootmisdefekte, variatsioone ja töökindlusprobleeme või (või) pahatahtlikke tõrkeid. Kvaliteedianalüüsi tööriist on integreeritud väljatöötatud raamistikku üksikute konstantrikete (tootmisdefektidena (või variatsioonidefektidena)) mõjude analüüsimiseks, rakendades graafikute teisendamise lähenemisviisi. Selle tööriista põhjal ennustab väljatöötatud tehisintellekti raamistik üksikute rikete mõju ahelate funktsionaalsel tasemel ja vähendab traditsioonilist simulatsiooni ajakulu 60% ja mõõdikute hinnanguline täpsus jääb vahemikku 93% kuni 94%. See meetod aitab ennustada ka sisendandmete katvuse mõõdikut (FFC) ahela testimise protsessis.

Teaduslikud jõupingutused rõhutavad juhtumiuuringute asjakohasust, mis võtavad arvesse vananemisest tingitud ajamuutuste mõju pehmete vigade usaldusväärsusele. Lõputöö viimases osas uuritakse masinõppe (ML) / süvaõppe (DL) panust, et uurida pehmete vigade usaldusväärsuse väljakutseid peente (nt 5-nanomeetriliste) tehnoloogiate puhul, millel on ajast sõltuvad pingemuutused. Tehnoloogiliste sõlmede agressiivne vähendamine toob kaasa taktsageduse maksimeerimise ja tagab pehme vea levimise kõrge tundlikkuse vananenud skeemi ahela viivituse muutuste suhtes. Selles töö osas on vananemismõjud piiratud negatiivse nihke temperatuuri ebastabiilsusega (NBTI). Vaadeldud tulemused järeldavad, et vananemisest tingitud pinge varieeruvus näitab SEU rikke levimisel olulisi maskeerimisomadusi ja vähendab vananemise ajal suurenenud pehmete vigade vastuvõtlikkuse mõju. Lisaks tõestab see töö selgesõnaliselt, et vananemine nihutab SET rikete levimispunkte taktiperioodi jooksul.

Doktoritöö teaduslikud jõupingutused viisid pehmete vigade usaldusväärsuse analüüsi kiirendatud automatiseerimistööriistani. Automatiseerimistööriist on skaleeritav ja väga kuluefektiivne keskmise ja suuremahuliste ahelate simulatsiooniressursside haldamisel. Põhjalikud tulemused selgitavad SEU ja SET sündmustest tingitud funktsionaalsete rikete usutavat prognoosi. Raamistiku eesmärk lisab mitmemõõtmelise funktsionaalse kontrolli ja lisaks toimib vahendina üksikute konstantrikete mõju analüüsimiseks funktsionaalsel tasandil. Lisaks on esitatud väga olulised teaduslikud tähelepanekud ja faktid vananemise mõjude kohta pehmete vigade levimisel.

Appendix 1

I

A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, “The validation of graph model-based, gate level low-dimensional feature data for machine learning applications,” in *2019 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, IEEE, oct 2019

The Validation of Graph Model-Based, Gate Level Low-Dimensional Feature Data for Machine Learning Applications

Aneesh Balakrishnan*†, Thomas Lange*‡, Maximilien Glorieux*, Dan Alexandrescu*, Maksim Jenihhin†
*iRoC Technologies, Grenoble, France
†Department of Computer Systems, Tallinn University of Technology, Tallinn, Estonia
‡Dipartimento di Informatica e Automatica, Politecnico di Torino, Torino, Italy
{aneesh.balakrishnan, thomas.lange, maximilien.glorieux, dan.alexandrescu}@iroctech.com maksim.jenihhin@taltech.ee

***Abstract*—As an alternative to traditional fault injection-based methodologies and to explore the applicability of modern machine learning algorithms in the field of reliability engineering, this paper proposes a systemic framework that explores gate-level netlist circuit abstractions to extract and exploit relevant feature representations in a low-dimensional vector space. A scalable feature learning method on a graphical domain called node2vec algorithm [6] had been utilized for efficiently extracting structural features of the netlist, providing a valuable database to exercise a selection of machine learning (ML) or deep learning (DL) algorithms aiming at predicting fault propagation metrics. The current work proposes to model the gate-level netlist as a Probabilistic Bayesian Graph (PGB) in the form of a Graph Modeling Language (GML) format. To accomplish this goal, a Verilog Procedural Interface (VPI) library linked to standard simulation tools has been built to map gate-level netlist into the graph model. The extracted features have used for predicting the Functional Derating (FDR) factors of individual flip-flops of a given circuit through Support Vector Machine (SVM) and Deep Neural Network (DNN) algorithms. The results of the approach have been compared against data obtained through first-principles approaches. The whole experiment implemented on the features extracted from the 10-Gigabit Ethernet MAC IEEE 802.3 standard circuit.**

***Index Terms*—Probabilistic Graph Model, Deep learning, Machine Learning, Functional Derating, Single-Event Upset (SEU), Gate-Level Netlist, Graph Modeling Language.**

I. Introduction

System engineering focuses on the integration of new small-scale technologies, constantly advancing the state of the art. The costly and difficult implementation of micro- and nano-scale devices highlights the challenges faced by all the partners from the design and manufacturing flow and, always aiming at improving their technological competitiveness. Current quality requirements, from end users or industrial standards, motivate designers and reliability engineers to dedicate significant effort and resources to reliability and functional safety aspects. Particularly, issues due to radiation based effects - Single Event Effects (SEEs) impact reliability metrics and are challenging to evaluate. A valuable approach to tackle these effects is the fault injection and simulation principle that provides precise and accurate information about circuit behaviour under stress, allowing the calculation of actual circuit-level reliability metrics.

This work was supported by the RESCUE ETN project. The RESCUE ETN project has received funding from the European Union's Horizon 2020 Programme under the Marie Skłodowska-Curie actions for research, technological development and demonstration, under grant No. 722325

978-1-7281-2769-9/19/$31.00 ©2019 IEEE

A. Motivation

Nowadays, increased user expectations or actual factual requirements formulated by the reliability and functional safety standards in high dependability applications make reliability modeling and assessment increasingly relevant. The reliability assessment process is usually accomplished with the different types of fault injection methods like exhaustive and random. The exhaustive fault injection method is obviously the ultimate reliability assessment method in terms of accuracy but very cumbersome in terms of resources, time, EDA licenses and so on, making this approach unfeasible on medium and large circuits. The random fault injection provides a solution to avoid unreasonable costs while allowing for accuracy (or statistical significance) on the proposed scope. Research proposals based on mathematical and statistical methods are always put forward by the research scientists. Nowadays, ML/DL techniques are more advanced and greatly favoured by researchers to learn statistical and functional dependencies between the feature representations of different systems. This is the main motivation for the idea of getting different algorithms and trying to find the best ways to develop relevant feature databases in the field of reliability assessment.

B. Organization of the Paper

The paper includes five sections in total. Section I summarized the State-Of-The-Art in the field of reliability engineering before presenting the motivation of the current work and the organizational structure of the paper. Section II gives a background introduction to Support Vector Machine, Deep Neural Networks and, different reliability factors of the microelectronic systems. In Section III, the main methodological implementation overview has given. Also, it explains the node2vec algorithm and different regression metrics. Section

IV illustrates the results and their validations in terms of different regression metrics and diagrams. As future progress of this work, a deep learning algorithm with a complex architecture called GCN has introduced and, its recent progress has briefed in section V. In VI, the whole work and its holistic approaches have concluded.

II. Background

A. Interpretation of Standard Reliability Based Terms

1) ***Single Event Effects****:* As the term suggests, a single event effect results from a Single Event - the interaction of a energetic particle with the device. The main effects are classified in two categories, destructive and non-destructive. This work is mainly contributing to the derating analysis of Single Event Upsets in sequential elements. The quantitative analysis of single event effects is based on different derating factors, called functional derating, logical derating, temporal derating, and electrical derating.

2) ***Electrical Derating****:* The Electrical Derating evaluates the propagational probability of the analog Single Event Transient (SET) pulse generated by the particle interaction. Based on their electrical pulse width and electrical amplitude range, it defined how well a transient error obstructs the standard signal propagation in the given circuit.

3) ***Temporal Derating****:* Temporal (or time) derating represents the opportunity window of an event (SET or SEU) and it's probability to be latched to the downstream sequential elements like flip-flop, latch and memory.

4) ***Logical Derating****:* The logical vulnerability of the SEE within the combinational (or) sequential cell networks based on their logical boolean functions is quantified with masking effect probability, termed as logical derating factor.

5) ***Soft Error****:* The fault - the primary consequence of the Single Event (SET/SEU) can be dropped or blocked in the circuit. If the fault propagates to and is memorized in state element (flip-flop, latch, memory), then it becomes a Soft Error. Please note that Bit/Cell Upsets (Single or Multiple) in memory instances are also Soft Errors.

6) ***Functional Derating****:* Functional Derating evaluates how likely is the Soft Error to cause an observable impact (Functional Failure) on the functioning of the circuits or systems.

B. The Machine learning and Deep Learning Algorithms

Giving an introductive subsection helps the reader to develop a clear idea of the relationship between Artificial Intelligence (AI), Machine Learning and Deep Learning. Deep Learning or Deep Neural Networks (DNN) or Artificial Neural Networks (ANN) are commonly considered as a subset of machine learning which in turn is derived from the concept of Artificial Intelligence. In machine learning, a database including the labeling vector of each class is parsed during the learning process and then exploit the learned dependencies between feature and class labels for deriving a decision margin, whereas, in case of deep learning algorithms, it appears in layers that can learn and make intelligent decisions on its own.

1) ***Support Vector Machine****:* The support vector machine works on the foundation of a good theoretical learning algorithm to solve regression analysis as well as classification type problems. The SVM for regression analysis can be called as SVR in short. It was invented by Vladimir Vapnik and his co-workers, and first introduced at the Computational Learning Theory (COLT) 1992 conference with the paper [8]. SVM characterizes the maximal margin algorithm for supervised learning models. In the maximal margin principle, SVR tries to find the optimal hyperplane which maximizes the margin and minimizes the error. Compared to classification problems, regression analysis outputs a continuous variable. The SVR approach defines a margin of tolerance ϵ where no penalty is given to errors. At the same time, it punishes the wrong estimation with a cost-insensitive symmetric hinge loss function.

An alternate approach in SVR applications is the kernel modification. A kernel which possible to transform the given data set to higher dimensional space to derive a liner decision boundary. A properly chosen Radial Basis Functions (RBF) had employed as a kernel function in this work. RBF is also called the Gaussian Kernel which means that each feature vector of the dataset in the transformed dimensional space influenced by the Gaussian observation.

2) ***Deep Neural Network****:* Deep Neural Network is an important step in the machine learning algorithms. Their learning methods are trying to model data with complex architectures and distributions by combining different non-linear transformations. In this work, a general fully connected DNN is implemented. The other main categories of deep learning methods are Convolutional Neural Network (CNN) and Recurrent Neural Networks (RNN). The elementary bricks of deep learning are the artificial neurons (perceptrons) which are inspired by biological neurons. An artificial neuron combines the input signals with adaptive weights and uses an activation function to deliver the output to be estimated. An in-depth discussion about the architecture as well as the adopted parameters has given in section IV.

III. Methodology

A. Overview of the Work

A global overview of the work is portrayed in figure 1. We start by mapping gate-level netlist into the probabilistic graph model. As the structural information of gate-level netlist is transformed into the probabilistic graph, the statistical properties of a graph node conventionally equivalent to that of a sequential (flip-flop) or logic (gate) element of the circuit are exposed. To execute this preliminary part of the work, different user-defined VPI functions had been written in C/C++ and linked to standard EDA logic simulators. The VPI library is able to extract all the relevant details of the gate-level netlist and formats them into a probabilistic graph model through GML graph attributes. In the next stage of the work, an SVM-Regressor (SVR) - a standard machine learning algorithm and fully connected DNN based on the deep learning algorithm,

were adopted as the learning-frameworks of the features from the probabilistic graph.

The feature matrix X for the implemented learning-frameworks is obtained by the random walk method using the node2vec algorithm. This algorithm can provide the feature dataset for the Circuit Under Test in a desired dimensional space within fractions of seconds. The random walk method gives a feature vector corresponding to a node by preserving neighborhood structure. The feature vector is mainly based on transition probabilities from source to target nodes in the neighborhood area and also the degree of nodes.

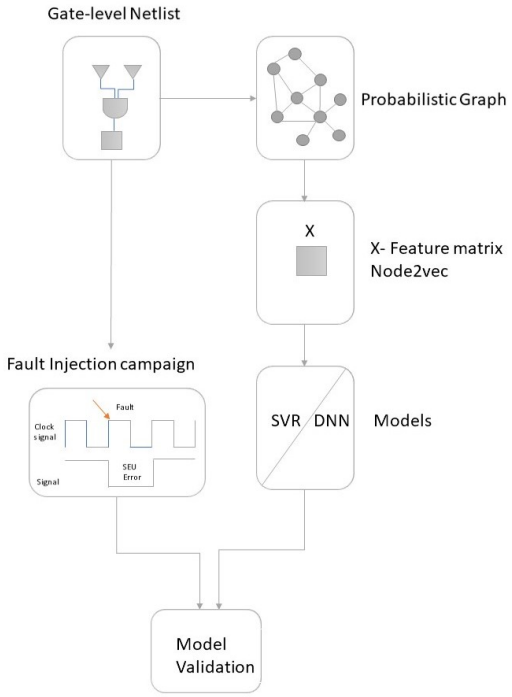


Fig. 1. Systematic block diagram of the scientific work

We choose a first principle approach - a basic, straightforward fault injection and simulation campaign, as a reference model and as a comparison baseline. This way, more stringent validation of the expected goals became possible. As observed from figure 1, the fault injection based ground truth data is shuffled and has split with a test size of 40% and a Training size of 60%. After training the learning models, predicted FDR values of flip-flops has been compared with the test vectors from the fault injection campaign FDR data. The ML/DL algorithms had been implemented in python with the help of Keras and Scikit-learn libraries which are available as open-source machine learning libraries for the Python programming language.

B. Node2vec: Scalable Feature Learning on Graphs

The node2vec algorithm proposed by Aditya Grover in [6] is endowed here in its novelty. The node2vec algorithm is a framework for learning continuous feature representations in the graph network. It maps the nodes in the graph into the desired dimensional feature space which maximizes the likelihood of preserving the network neighbourhood of nodes. Node2vec algorithm can apply to any given directed or undirected, weighted or unweighted edge networks.

Nowadays, representing a dataset in a graphical domain becomes a very useful (and obligatory) tool. We use this approach for predicting and visualizing the probability factors over nodes and edges. The netlist from the gate-level abstraction of the circuits is successfully represented in the graph domain. For performing a prediction analysis, a careful effort is required to develop a feature vector space that suitable for different learning algorithms. This requirement has achieved with the node2vec algorithm.

The feature learning framework of the node2vec algorithm had been formulated as a maximum likelihood optimization problem. The given network can be represented as $G = (\nu, \varepsilon)$, where ν represents vertices or nodes and ε represents the edges between the vertices. $f : V \rightarrow \mathbb{R}^d$ is the mapping function from a node to d dimensional feature space, where V stands for a whole set of vertices. f is a matrix with size of $|V| \times d$. A neighborhood sampling strategy S is used to define a network neighbourhood as $N_s(u)$ of a source node u. The framework optimizes the objective function f by maximizing the log-probability of observing a network neighbourhood $N_s(u)$ for a node u, conditioned on its feature representation. The objective function is given by:

$$\max_{f} \sum_{u \epsilon V} logPr\left(N_s(u)|f(u)\right). \tag{1}$$

The sampling strategy developed for node2vec is a flexible random walk that interpolates two important sampling strategies termed as Breadth-first Sampling (DFS) and Depth-First Sampling (DFS). In BFS, the sampling nodes are the very immediate neighbors of the source node whereas, in DFS the neighbors have been obtained by sampling sequentially at increasing distance from a source node. The two important factors in the node2vec algorithm are flexible biased random walk and search bias α. Let consider a source node u and a random walk length l and c_i denote the i^{th} node in the walk from source node $c_0 = u$. The probability of c_i given c_{i-1} is generated by:

$$P(c_i = x \mid c_{i-1} = v) = \left\{ \begin{array}{cc} \frac{\pi_{vx}}{Z} & \text{if} \quad (v,x) \in E \\ 0 & \text{Otherwise} \end{array} \right\} \tag{2}$$

where, where π_{vx} is the unnormalized transition probability between nodes v and x, and Z is the normalizing constant.

The search bias factor α is a major factor in calculating π_{vx}. Consider a random walk that just traversed the edge (t,v) and resides on node v. As a next step in the random

walk, an unnormalized transition probability π_{vx} on the edge (v,x) leading from v, is estimating. The unnormalized transition probability is set to $\pi_{vx} = \alpha_{pq}(t,x).w_{vx}$, where:

$$\alpha_{pq}(t,x) = \left\{ \begin{array}{ll} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{array} \right\} \quad (3)$$

and w_{vx} is the weight of the edge. In the case of unweighted edge, $w_{vx} = 1$. The d_{tx} is the shortest path between t and x. Parameter p is called the Return Parameter and it controls the likelihood of immediately revisiting node in the walk. q is called an In-Out parameter which allows the search to differentiate between inward and outward nodes. Here, feature space with dimension 8 had extracted. The feature vectors of three arbitrary flip-flops had plotted in figure 2 for giving an illustration of the vector's statistical variance.

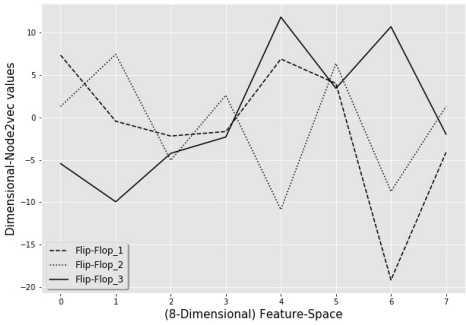


Fig. 2. Feature vector of three arbitrary flip-flops

C. Regression Evaluation Metrics

1) ***Mean Squared Error (MSE):*** If $\hat{y}_i$ is the predicted value and y_i is the true value corresponding to the i^{th} sample, then the mean squared error to be estimated over n samples is defined as,

$$MSE(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2 \quad (4)$$

The regression error will become minimal as MSE approaches to zero.

2) ***R - Squared Score (R^2):*** also known as the coefficient of determination. If $\hat{y}_i$ is the predicted value of the i^{th} sample, and y_i is the corresponding true value, then the coefficient of determination estimated over n samples defined as,

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2}{\sum_{i=0}^{n-1} (y_i - \bar{y}_i)^2} \quad (5)$$

where, $\bar{y}_i = \frac{1}{n} \sum_{i=0}^{n-1} y_i$. Numerical value 1 indicates a good regression fit, while 0 indicates a worse fit.

3) ***Explained variance score (EVS):*** f $\hat{y}$ is the predicted value of the target value y, then Explained variance score estimated over n samples is defined as,

$$EVS(y, \hat{y}) = 1 - \frac{Var\{y - \hat{y}\}}{Var\{y\}} \quad (6)$$

where, Var is the square of the standard deviation. The best possible score is 1.

IV. Result : Modeling and Validations

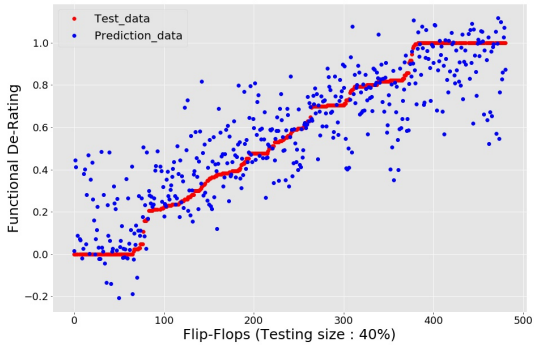
To test the applicability of the node2vec based features for the machine learning frameworks in the system reliability evaluation, a validation effort has been performed on a 10-Gigabit Ethernet MAC IEEE 802.3 standard circuit. Experimenting with fault injection for each flip-flop independently and documenting how probable is the fault to affect the overall function of the circuit as the Functional Derating factor provides the reference dataset for the validation. About one thousand two hundred and two (1202) flip-flops have used for evaluating the prediction models. The circuit is accessible at OpenCores as the 10-Gigabit Ethernet project.

A. Result Analysis : SVR

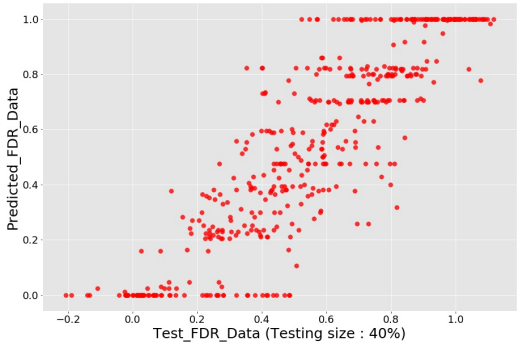
The prediction result of Support Vector Machine Regression are provided in figure 3a and jointly presented a scatter plot in figure 3b respectively. The corresponding evaluation metrics have been given in Table II. In SVR, we use the RBF kernel function which described as,

$$K(X, X') = \exp\left(-\gamma \left\| X - X' \right\|^2\right). \quad (7)$$

The X and X' are the two data points in vector form. The kernel K maps them to higher dimensional vector space. γ is called the spread of the kernel function and, it tuned to γ = 0.01. The other important parameter is epsilon ϵ which, responsible for error tolerance and set to ϵ = 0.0125. The parameter C is the regularization scheme and, proper value is chosen for the penalty factor C. Here C = 10. A grid-search cross-validation method tunes the parameter values. From the prediction diagram 3a, the predicted values approximating the original values which, sorted in ascending order by values. The scatter plot in figure 3b indicating a good correlation between predicted and original test data. But there is still a space for improvement because the scatter plot having a variance between the axial components. The metrics R^2 from Table II is indicating the good regression fit of prediction with original data. It is almost 69%. If the predicted values approximate more likely to the tested data, the R^2 will tend to the numerical value 1. In the same way, the metric MSE form Table II is equal to 0.027 and, it will close to 0 when the approximation becomes better. The metric EVS also mentioned in Table II.

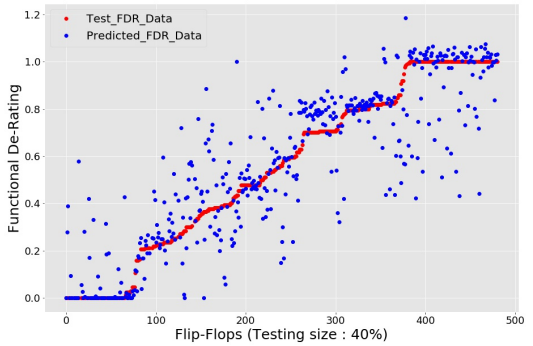


(a) Prediction over 40% Test data

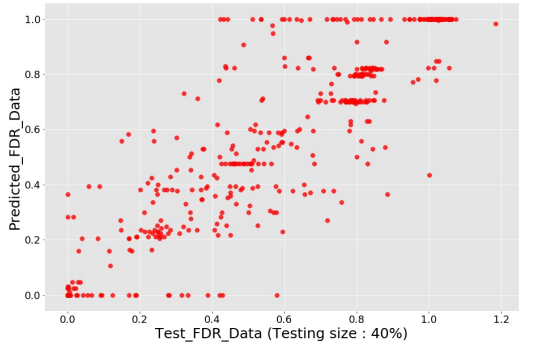


(b) Scatter plot between prediction and true value

Fig. 3. Regression by SVR model



(a) Prediction over 40% Test data



(b) Scatter plot between prediction and true value

Fig. 4. Regression by DNN Model

B. Result Analysis : DNN

The DNN architecture had been chosen according to table I. The input layer is nothing but the feature vectors. The Dense_1, Dense_2, Dense_3, and Dense_4 are the hidden layers. Dense_5 is called the output layer which outputs the estimated regression values. Each hidden layer is a fully-connected dense layer where the number of inputs to each neuron is equal to the output size of the previous layer. The weights of neuron inputs and the bias factor are the parameters that need to be optimized. The hyper-parameters termed as loss = 'Mean Squared Error', optimizer = 'Adam' and batch_size = 10 are chosen according to cross-validation method. The Dense_1 layer has a shape of 126 neurons. With the input feature vector of dimension 8, DNN training for getting a good prediction accuracy becomes difficult. Therefore, the Dense_1 layer will map the low dimensional input vectors to a high dimensional space. DNN will show significant performance with a higher dataset dimension.

Figure 4a shows that the DNN method provides an adequate prediction. A majority of the estimations are close to the true values, which indicates a good R^2 value. From Table II, the R^2 is 0.77. The MSE value is 0.0259, which indicating that the mean error is also low. Figure 4b provides the corresponding scatter plot, showing the correlation between original test values and predictions. Here also, we can see the variance between the two axis components.

TABLE I
DNN ARCHITECTURE

Layer	Output shape	Parameters
Dense_1	126	25326
Dense_2	64	8128
Dense_3	36	2340
Dense_4	12	444
Dense_5	1	13

C. DNN vs SVR

Metrics form Table II indicate a dominant performance of DNN in terms of R^2, EVS and MSE. The score EVS is

TABLE II
METRIC EVALUATION FOR DIFFERENT REGRESSION MODELS
(TRAINING SIZE = 60 %)

Model	MSE	EVS	R^2
DNN	0.025995	0.770322	0.770169
SVR	0.027359	0.690909	0.689758

used to measure the discrepancy between model-driven values and actual data. The high value near to 1 shows the model is providing a valuable prediction. It appears that the DNN model performs better than SVR. But other facts need to be highlighted. In Table III, the time needs to execute different models had compared. The fault injection campaign over 1202 flip-flops of the Ethernet-MAC circuit took nearly five days per Modelsim software. SVR seems to be very fast while the DNN needs to optimize a comparatively large set of parameters, as explained in Table I. But, when compared to traditional fault injection methods, it should be considered that ML/DL models depend 60% true detests that generated by traditional fault injection methods.

TABLE III
TIME PERFORMANCE OF DIFFERENT MODELS

Model	Time
Fault Injection (1 Modelsim)	5 days
Fault Injection (7 Modelsim)	17 hours
SVR	$<$ 1 minute
DNN	6 minutes

Finally, the DNN and SVR have been compared using 95% Confidence Interval (CI) and Mean values between predicted and original values. This comparison showed in figure 5. Here DNN performs comparatively better because the difference between the means of the respective predicted and the target values is small compared to that of SVR.

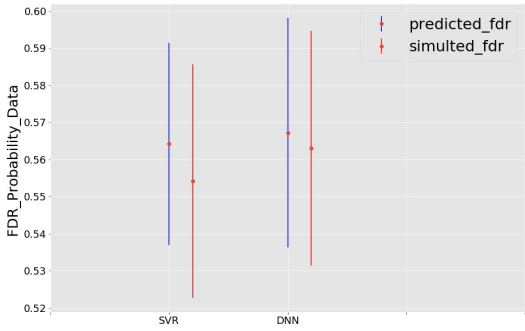


Fig. 5. CI comparison : SVR Vs DNN

V. FUTURE WORK

Even though the implemented ML/DL models are achieving their reasonable accuracy within a very short interval of time, all those algorithms need high quality training data. In our case, we have used a $50\% - 60\%$ of the database obtained through first principles methods (fault simulation) for the training process. The real-time data processing applications will not accept this fact to an extent. Every circuit and its electrical characteristics vary from one to another. It could be useful to solve the issue of the training data set by using Graph Convolutional Neural Network (GCN) [2], that needs only $5\% - 10\%$ of training data. Recent research work about this idea had published in [11]. Another development direction is to develop acceptable prediction over gate-level netlist with the node2vec feature matrix and with more advanced graph-based deep neural architectures. A GCN based probability distribution comparison has shown in 6. According to the comparison, a graph convolutional neural network can reach an acceptable prediction goal.

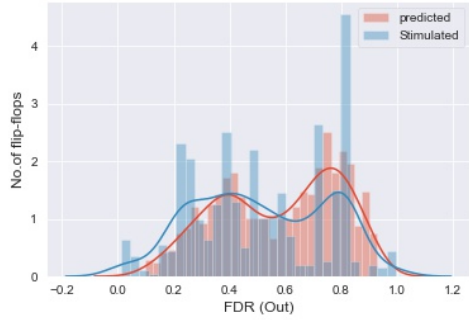


Fig. 6. Histogram comparison of GCN model

VI. CONCLUSION

The works implemented in this paper depict the importance of extracting a low-dimensional feature matrix from a gate-level netlist of logic circuits by the node2vec algorithm. This feature matrix has validated using SVR and DNN machine learning algorithms. These algorithms have compared with different regression metrics and diagrams. The whole experiment is proving that the extracted feature matrix by the node2vec algorithm, can be used to perform ML/DL algorithms successfully. This feature space can also apply to complex neural network architectures to reduce the estimation time of different circuit reliability factors.

REFERENCES

[1] M.Nicolaidis, "Soft Errors in Modern Electronic Systems," Springer Publishing Company Incorporated, 1^{st} ed, 2010.
[2] T.N. Kipf, M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," ICLR 2017.

[3] W.L. Hamilton, R. Ying, J. Leskovec, "Representation Learning on Graphs: Methods and Applications," Published in the IEEE Data Engineering Bulletin, Available: http://arxiv.org/abs/1709.05584, September 2017.
[4] F. Scarselli and M. Gori and A. C. Tsoi and M. Hagenbuchner and G. Monfardini, "The Graph Neural Network Model," IEEE Transactions on Neural Networks, Vol. 20, Jan 2009.
[5] M. Defferrard, X. Bresson and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," Curran Associates, Inc., pp. 3844–3852, NIPS 2016
[6] A. Grover and J. Leskovec, " node2vec: Scalable Feature Learning for Networks," Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.
[7] D.P. Kingma and B. Jimmy, " Adam: A Method for Stochastic Optimization," International Conference on Learning Representations, 2014.
[8] V.Vapnik; The Nature of Statistical Learning Theory, 2nd Edition.Springer, NewYork,(2001).
[9] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In 22nd SIGKDD Conference on Knowledge Discovery and Data Mining, 2016.
[10] L. Breiman. Random forests.Machine Learning, 45:5–32, 2001.
[11] Aneesh Balakrishnan, Thomas Lange, Maximilien Glorieux, Dan Alexandrescu and Maksim Jenihhin . "Modeling Gate-Level Abstraction Hierarchy Using Graph Convolutional Neural Networks to Predict Functional De-Rating Factors". In NASA/ESA Conference on Adaptive Hardware and Systems (AHS), 2019.

Appendix 2

II

A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, “Modeling gate-level abstraction hierarchy using graph convolutional neural networks to predict functional de-rating factors,” in *2019 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, IEEE, jul 2019

2019 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)

Modeling Gate-Level Abstraction Hierarchy Using Graph Convolutional Neural Networks to Predict Functional De-Rating Factors

Aneesh Balakrishnan*†, Thomas Lange*‡, Maximilien Glorieux*, Dan Alexandrescu*, Maksim Jenihhin†
*iRoC Technologies, Grenoble, France
†Department of Computer Systems, Tallinn University of Technology, Tallinn, Estonia
‡Dipartimento di Informatica e Automatica, Politecnico di Torino, Torino, Italy
{aneesh.balakrishnan, thomas.lange, maximilien.glorieux, dan.alexandrescu}@iroctech.com maksim.jenihhin@taltech.ee

***Abstract*—The paper is proposing a methodology for modeling a gate-level netlist using a Graph Convolutional Network (GCN). The model predicts the overall functional de-rating factors of sequential elements of a given circuit. In the preliminary phase of the work, the important goal is making a GCN which able to take a gate-level netlist as input information after transforming it into the Probabilistic Bayesian Graph in the form of Graph Modeling Language (GML). This part enables the GCN to learn the structural information of netlist in graph domains. In the second phase of the work, the modeled GCN trained with a functional de-rating factor of a very low number of individual sequential elements (flip-flops). The third phase includes the understanding of GCN models accuracy to model an arbitrary circuit netlist. The designed model validated for two circuits. One is the IEEE 754 standard double precision floating point adder and the second one is the 10-Gigabit Ethernet MAC IEEE 802.3 standard. The predicted results compared to the standard fault injection campaign results of the error called Single Event Upset (SEU). The validated results are graphically pictured in the form of the histogram and sorted probabilities and evaluated with the Confidence Interval (CI) metric between the predicted and simulated fault injection results.**

***Index Terms*—Probabilistic Graph Model (PGM), Graph Convolutional Neural Network (GCN), Functional De-rating, Single-Event Upset (SEU). Gate-Level Netlist, Graph Modeling Language (GML)**

I. Introduction

System engineering advances and focusing on the integration of small-scale technologies in the system building process. The realization of full potential micro- and nanoscale devices highlights the challenges faced by electronics businesses industries in maintaining or improving their technological competitiveness. System engineering and its challenges keep the design engineers more concentrating on the reliability issues with their designed systems. Focusing on the reliability problems occurring with micro- and nanoscale technology development and its impact on everything from the design phase to actualized products in the health, automotive, aerospace, communication, and many other fields, the system design engineers considering all possible methodological precautions to prevent reliability issues based on their criticality. The industrial customers demanding high-quality reliable devices and in order to meet the requirements, research and design departments proposing different metrics which ensures a default standard quality and reliability. One of the major threats in the system's reliability is the Single Event Effects (SEE) due to the cosmic rays and electromagnetic radiation. Cosmic rays are particles that hit the Earth's atmosphere from space. They include protons, helium nuclei (like α radiation), and electrons (like β radiation). The radiations like gamma and X rays, which are electromagnetic and indirectly ionizing radiations. The two major consequences of the SEE are Single Event Transients (SET) and Single Event Upset (SEU). The effect of SEU and SET at the functional level of the circuit is known as functional de-rating factors. They are more closely examined here with help of exhaustive and accelerated fault injection campaigns.

The important aspect of the fault injection campaign is the more reliable and accurate information over other different mathematical models. In contrary to this point, the effort in terms of time is non-feasible from the perspective of a designer. The effort in the non-feasible dimension of work can be reduced by implementing different statistical and mathematical models. This research goal achieving through the proposed model and automate the assessment of different reliability factors within feasible time constraints and making a trade-off with accuracy.

A. Motivation

Even though above-explained networks are eligible to do deep learning, the implemented model predominantly depending on the neural network as referenced in [4] and [3]. The cited paper [4] extends the current neural networks to a new model called Graph Neural Network (GNN) and process the data in graph domains. There is a lot of scientific areas of engineering which deals the information in the graph domains. This point is considered to be a decisive moment of the thought towards a representation of the gate-level circuit information in graph domains and feeding to a graph neural network

This work was supported by the RESCUE ETN project. The RESCUE ETN project has received funding from the European Union's Horizon 2020 Programme under the Marie Skodowska-Curie actions for research, technological development and demonstration, under grant No. 722325

2471-769X/19/$31.00 ©2019 IEEE
DOI 10.1109/AHS.2019.00007

for processing it. Here, the implemented model adopts a form of Graph Convolutional Network (GCN) proposed by Thomas.N.Kipf in [2], which is another version of generally called graph neural networks. This model is particularly briefed in section IV and V.

B. Organization of the Paper

The introduced paper includes nine sections in total. Section I generalizes the facts and issues in the field of reliability engineering and followed by the main motivation of this work as well as the organizational structure of the paper. Section II gives a background introduction to neural networks and different reliability factors of the micro-electronic systems. This part dedicated to explaining the background of this work. In Section III, the main methodological implementation overview is given, whereas in section IV and V, GCN model and it's neuron implementation explained with mathematical equations. That is, sections IV and V together constitutes the methodology and model architecture and their in-depth view. Section VI illustrates the results and their validations in terms of 95 % Confidence Interval (CI), histograms and sorted order of FDR probabilities. Section VII describes the main model drawbacks. Future works and their importance with their probability to achieve, are discussed in section VIII and In Section IX, the whole work and it's holistic approaches are concluded.

II. Background

A. Interpretation of Standard Electrical Terms

1) ***SEE Analysis Concepts****:* As the term suggests, a single event effect (SEE) results as the penetration consequences of the energetic radiation particle. The main consequence effects are classified as two categories destructive and non-destructive. The Single Event Upset (SEU) and Single Event Transient (SET) are considered to be non-destructive and soft-errors. The radiation hazards like Single Event Latchup (SEL) are categorized under hard-errors (or) destructive-type faults. An elaborate explanation for different radiation hazards can be referenced from [1]. This work is mainly contributing to the derating analysis of Single Event Upset in sequential elements. The quantitative analysis of SEE is based on different derating factors, called Functional derating, Logical derating, Temporal derating, and Electrical derating.

2) ***Electrical Derating****:* The Electrical Derating (EDR) evaluating the effect of modeled logic SET pulse that has the same effect in the circuit as the original analog SET pulse. SET pulse can be modeled logically as an inversion of the output signal amplitude of combinational cells in gate-level abstraction. The effect of such types of anomalies with various electrical factors like electrical pulse width and electrical amplitude range defines how well a transient error obstructs the standard signal propagation in the given circuit.

3) ***Temporal Derating****:* Temporal Derating (or) Time derating associate to the opportunity window ascribed to SEE error (SET (or) SEU) and it's probability to be latched to the downstream sequential elements like Flip-Flop, Latch and Memory.

4) ***Logical Derating****:* The logical vulnerability of the SEU within the combinational cell network based on their logic functions is quantified with masking effect probability, termed as Logical Derating (LDR) factor.

5) ***Functional Derating****:* Functional Derating evaluates how likely the soft error propagate to make an observable impact on the functioning of the circuits or systems.

B. The reasoning of Graph Convolutional Neural Network

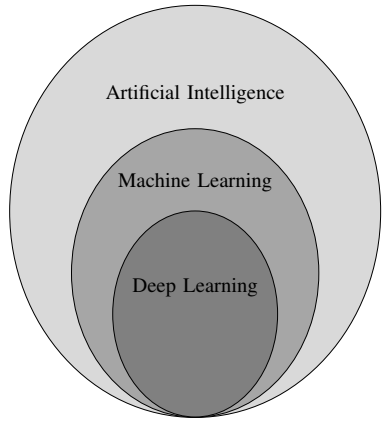


Fig. 1. A relational analysis of Artificial Intelligence (AI), Machine Learning (MI) and Deep Learning (DL)

The part actually gives a clear idea of the relationship between Artificial Intelligence (AI), Machine Learning (ML) and Deep Learning (DL) to the reader. The main point of view is, Deep Learning or Deep Neural Networks (DNN) or Artificial Neural Networks (ANN) are commonly considered as a subset of machine learning which in turns derived from the concept of artificial intelligence. The machine learning in which data parsed for learning phase and then apply the learned dependencies of the data features to arrive at a decision, whereas, in case of deep learning algorithms, it appears in layers to create an Artificial Neural Network (ANN) that can learn and make intelligent decisions on its own. The artificial intelligence considered to be a global idea of ML and DL and it can be defined by a way of enabling the machine (e.g. a computer) to attain a given task based on a stipulated set of rules called an algorithm.

As mentioned in the above paragraph, the deep neural networks are able to make intelligent decisions on its own, the work which experimented here mainly based on a neural network, called Graph Convolutional Neural Network. Intelligent network like GCN is actually different from traditional neural networks algorithm and slightly varied from traditional Convolutional Neural Networks (CNN). A normal neural network consists of staked hidden layers, where each of the

neurons (or) nodes from the current layer receives input from all the nodes from the previous layer, commonly known as dense layers. Then performs a dot product of the data at the input of the neuron and the weights of the neuron and passed through an activation function respectively. These determined values passing to the successive layers by concatenating the input, hidden and output layers together. CNN is different from the traditional way of constructing the dense layered neural network. In CNN, the initial input features are convolved with kernel input filters and then down-sampled through a pooling layer and finally directed to a normal fully connected neural network.

III. Overview of the Work

A better overview of the work portrayed in figure 2. Before stepping into the detailed structure of the whole work, it is very relevant to brief the importance of mapping the gate level netlist into the probabilistic graph model. The more the mapping achieve accuracy, the more the model delivers a valid result because the graph structure maintains the required statical information. In order to execute the preliminary aim of the work, different user-defined Verilog Procedural Interface (VPI) functions had written and it in turn applied to extract all the relevant details of the gate level netlist and formatted into a probabilistic graph model through GML attributes. As stepping forward into the successor stage of the work, GCN adopted as the model in order to learn the whole designed probabilistic graph. The more comprehensively explained hierarchical architecture of GCN updated in the successive sections.

The netlist representation in graph domains subsequently used to extract the adjacency matrix, which represented by A in the figure 2. Correspondingly, a feature matrix X also obtained by the random walk method using the node2vec algorithm. The random walk method gives a feature vector corresponding to a node with respect to its neighboring nodes. The feature vector is mainly based on transition probabilities from source to target nodes and also the degree of nodes.

These are the two main inputs given to the GCN model. GCN then commenced learning the whole netlist as a probabilistic graph. As soon as, it processes the adjacency matrix and feature matrix, a model of the netlist is delivered. After that, this model is used for the training phase and testing phase for accomplishing the FDR prediction goal. Finally, the predicted data is compared with the fault injection campaign FDR data.

The whole deep learning framework was implemented in MXNet.

IV. Graph Convolutional Network

A. Recent Literature History

Different prodigious research work had been introduced in the past decades of years, for generalizing the conventionally established neural network like Recurrent Neural Network (RNN) or Convolutional Neural Network (CNN) for working on arbitrarily structured graphs, even though it is a great challenging problem.

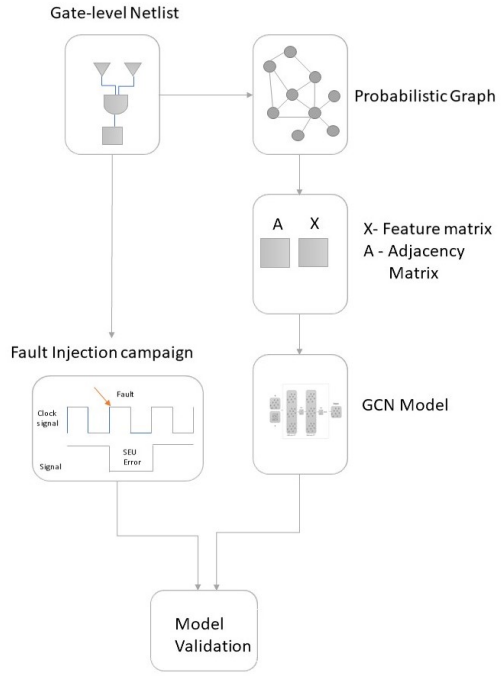


Fig. 2. Systematic block diagram of the scientific work

This work is mainly based on the GCN neural network. A similar spectral approach introduced in [5]. By the GCN model, it is able to exemplify the spectral rule approach in the graphical learning process and it achieves significantly faster training times with higher predictive accuracy and also reaching state-of-the-art classification results on a number of benchmark graph datasets.

B. Architecture

Figure 3 provide a architectural view of GCN. The work made a GCN model of two hidden layers as given in figure 3. The first layer in this work contains 4 hidden nodes and the second layer contains 2 hidden nodes. These two hidden layers stacked between the input layer and the output layer. The input layer contains a number of nodes which equivalent to the gate-level netlist elements of the circuits. It varies from circuit to circuit. The model can able to model even for a large number of elements of the circuit by this time. But it is difficult to say a limit now. Both hidden layer's nodes activated by the non-linear function called a hyperbolic tangential function (Tanh). During the training phase, the model is updating at each step and optimized by an adaptive learning rate optimization algorithm called 'Adam' [7]. The dimension of the hidden

layers can be chosen by arbitrarily and it depends on the parsed adjacency matrix.

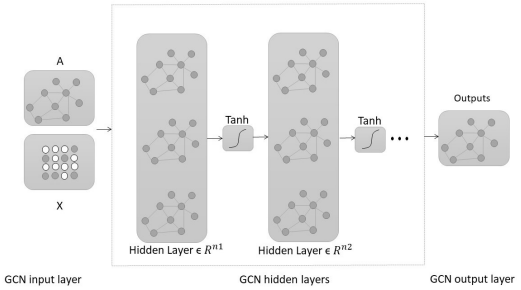


Fig. 3. GCN model [2]

C. Model

The Graph Convolutional Network is a powerful neural network architecture for machine learning on graphs. Following paper [2], revealing the fact that most of the graph neural networks has been addressing a common architecture in general, which lead to the name called Graph Convolutional Neural Networks (GCN). The convolution name comes after using the filter parameters shared across all locations of the graph.

1) Model Definition: The created probabilistic graph model of the gate-level netlist embedded into the GCN network with the intention of learning the function of features in the graph. The graph described as a $G = (\nu, \varepsilon)$, where ν represents vertices or nodes and ε represents the edges between the vertices. The graph characterized as,

$\odot$: Every nodes i is attributed with feature vector x_i of dimension D. So for N nodes, we have feature matrix X: $N \times D$.

$\odot$: Another important parameter is the adjacency matrix A, which indicates the graph structure.

$\odot$: The propagation rule will produce a node-level output of Z: $N \times F$, where the F represents a feature vector of each output node.

$\odot$: Every neural network layer can be represented as in equation 1.

$$H^{(l+1)} = f(H^{(l)}, A) \tag{1}$$

Where $H^{(l+1)}$ represents the any hidden layer node matrix at $(l+1)^{th}$ level and it equivalent to the function of previous hidden layer node matrix H^l at l^{th} level and the adjacency matrix A. H can be taken as the feature matrix X at initial level, ie $H^{(0)} = X$ and Z at final level. Z represents the graph level output.

2) Model Propagation Rule: In this whole paper, an exact propagation model for the Graph Neural Network is adapted to tackle the prediction problem.

A simple form of the layer-wise propagation rule abbreviated as:

$$f(H^l, A) = \sigma(AH^lW^l) \tag{2}$$

Where, W^l is the l^{th} neural network weight matrix and $\sigma()$ is the activation function like Rectified Linear Unit (ReLU), while this work utilizes a hyperbolic tangent activation function (Tanh). Even though the above propagation rule seems to be very simple, it was proved to be very powerful. The major disadvantage of this kind of model is the adjacency matrix A, which not normalized so that multiplication of A with feature matrix will change the scale of feature matrix completely. The second problem as mentioned by the authors of this model is, the model does not consider the self-features by a node itself. And the problem is completely taken away by providing an identity matrix for the nodes.

The major problem overcame by a normalizing matrix A. Normalization achieved by an inverse diagonal node degree matrix D, such as the rows of $D^{-1}A$ sums to 1. So the multiplication becomes more similar to taking the average of neighboring nodes. This lead to symmetric normalization i.e, $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, and it more than just a mere averaging of neighboring features. These combined methods used in this work as a propagation rule which exactly similar to the way implemented in paper [2] and final layer-wise propagation rule provided as:

$$f(H^l, A) = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^lW^l) \tag{3}$$

where, $\hat{A} = A + I$; with I defined as identity matrix and $\hat{D}$ is the diagonal degree node matrix of $\hat{A}$.

3) Input Feature Matrix: In order to generate a feature matrix corresponding to the nodes in the probabilistic graph, we use a node2vec algorithm provided by [6]. *node2vec* is an algorithmic framework for learning continuous feature representations of nodes in networks. According to this algorithm, it maps nodes to the low-dimensional feature space which maximizes the likelihood of preserving network neighborhoods of nodes. This objective is optimizing by the stimulated biased random walks. It preserves a spectrum of equivalences from homophily to overall structural equivalence, by anticipating a balanced exploration-exploitation trade-off.

V. GCN Neuron Model

Figure 4 represents a single slice of neuron pipeline which implemented in the neural network. The GCN model neighborhood aggression is typically different from the basic neighborhood aggregation algorithm as mentioned in 4. It is clear that to mention that, no bias factor is added and trained in the model. A similar weight matrix W_k used for the self-node embedding and neighbor nodes embedding. This improves and achieves more parameter sharing across the network and down-weights the higher degree neighbors. The important thing to notice is the normalization factor which varies across the neighbors instead of a simple average. In equation 4, the node v is abbreviated for the node targeted for embedding

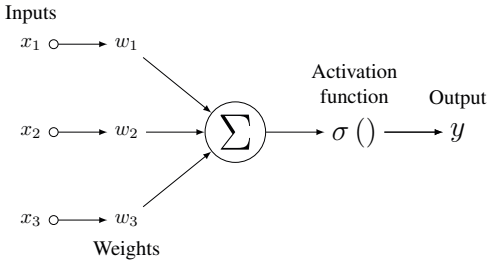


Fig. 4. A neuron model for embedding nodes

process, while $N(u)$ in equation 5 represents the neighbouring nodes of v. h_v^k given in equation 5 is k^{th} layer node v aggregator as indicated in figure 5. The equation 5 pictorially represented in figure 4. $\sigma()$ indicates a non-linear function, simply named as an activation function in figure 4.

$$h_v^0 = X_v \tag{4}$$

$$h_v^k = \sigma\left(W_k \sum_{u \epsilon N(v) \bigcup v} \frac{h_u^{k-1}}{\sqrt{\mid N(u) \mid\mid N(v) \mid}}\right) \tag{5}$$

$$Z_v = h_v^K \tag{6}$$

The variable h_v^0 shows a node v at the input layer and its input equivalent to the node v features vector X_v extracting using a node2vec algorithm. The variable h_v^K denotes node embedding of a node v at the last layer K of neural network and the output node's embedding with its features space abbreviated as Z_v.

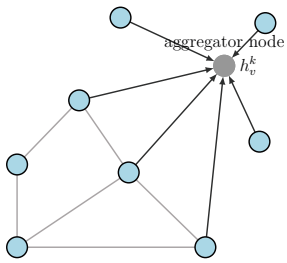


Fig. 5. Aggregation model of a node

Figure 5 shows an aggregator node in the network which collects related information and features of neighboring nodes.

VI. Result : Modeling and Validations

As mentioned earlier in the paper, the model tested with two circuits. The very first one is the double precision floating point adder which extracted from the double precision floating point core as a submodule, which meets the IEEE 754 standard and available in the OpenCores website. The second circuit is also accessible from OpenCores as 10-Gigabit Ethernet project, where Management Data Input/Output (MDIO) function of this module designed to meet 10-Gigabit Ethernet IEEE 802.3 standard. In MAC design based on the Xilinx LogiCORE 10-Gigabit Ethernet MAC, the transmitter and the receiver incorporate the reconciliation layer. Therefore the receive engine, as well as transmit engine, will be specifically designed to interface the client and the physical layer.

A. *Double precision floating point adder*

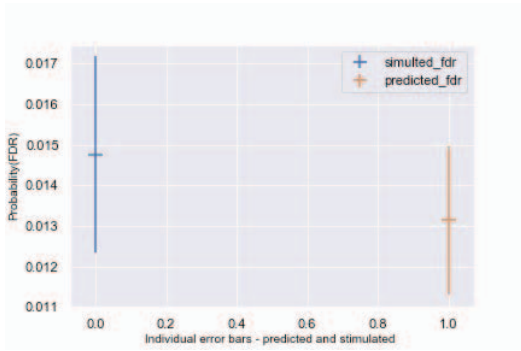


Fig. 6. An overall FDR confidence interval (CI) comparison between predicted and stimulated data

Figure 6 actually represents the confidence interval comparison of the predicted Functional Derating (FDR) data of flip-flops with the FDR data generated from random fault injection campaign. The CI calculated in python, by finding the mean of the flip-flop's FDR distribution and their FDR distribution error for 95% confidence interval. There are no electrical features extracted from the circuit's gate-level netlist to train the upholding neural network model. The training had done with less than 10 flip flops FDR. The overall comparison indicates the prediction almost following the stimulated SEU fault's FDR data.

As observed from the histogram graph depicted in figure 7, the prediction of the FDR probability distribution function (PDF) of the flip-flops comparatively very close to the original PDF of the flip-flops.

Figure 8 compares the sorted FDR value of simulated and predicted data. This sorted FDR plot only shows how an overall functional derating curve behaves with respect to flip flops. In fact, it does not provide any individual flip-flip comparison. The work is currently extending to do that. This comparison plotted to provide an intuition to the reader that

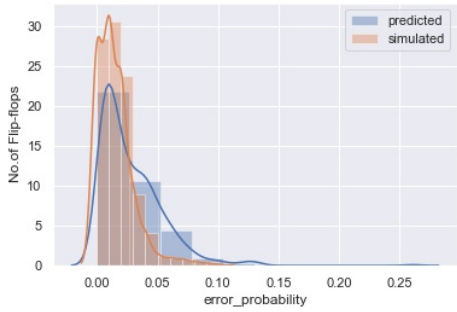


Fig. 7. Histogram Comparison

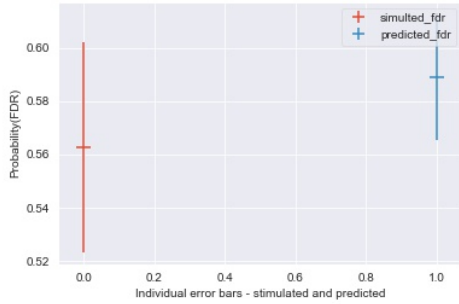


Fig. 9. Representation of CI comparison

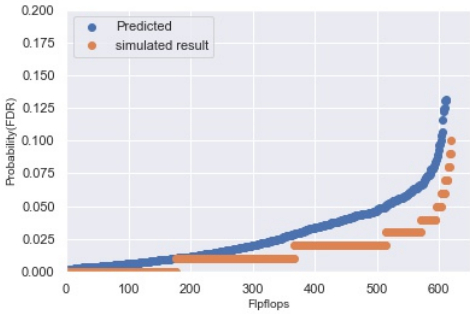


Fig. 8. Sorted FDR probability graph

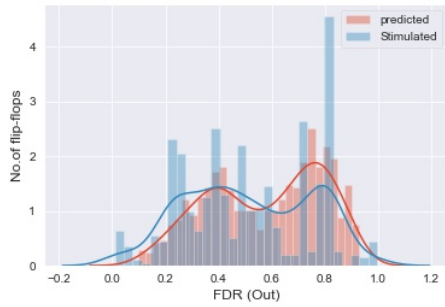


Fig. 10. Representation of Histogram comparison (Filetring out some outliers)

the model is actually able to get a reasonable approximation with respect to their independent structural information. This is specifically understandable when the sorted FDR graph in figure 12 from Ethernet MAC compared here, which entirely different from floating point adder.

B. Ethernet MAC

Here the modeling tries to validate on Ethernet MAC circuit. This reveals how powerful is this algorithm to predict on the completely different histogram with a training sequence of 5 flip-flops (ie, less than 1 % of overall flip-flop number). Figure 9 represents the overall confidence interval comparison of the predicted FDR data with FDR data obtained from fault injection campaign on the sequential elements in each clock-cycle independently. Figure 10 represents the PDF where some of the data points are filtered, which considered being outliers within the data space and plotted the remaining data. Simultaneously figure 11 detailing the histogram comparison for full data obtained through the simulation process but here accuracy of the histogram prediction achieved through a comparatively higher number of epochs.

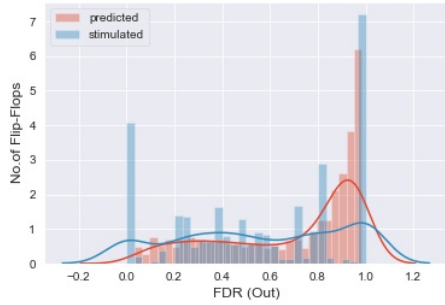


Fig. 11. Representation of Histogram comparison without filtering

VII. Model Drawback

Even though GCN models are achieving their accuracy within a reasonable period of time, the stability for providing good results can be degraded if we increase the number of

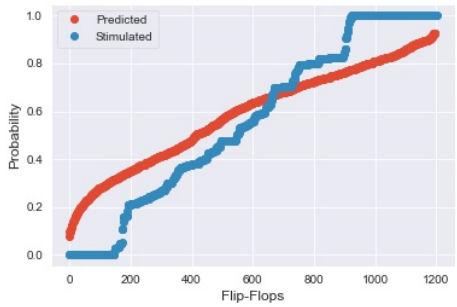


Fig. 12. Sorted FDR probability comparison

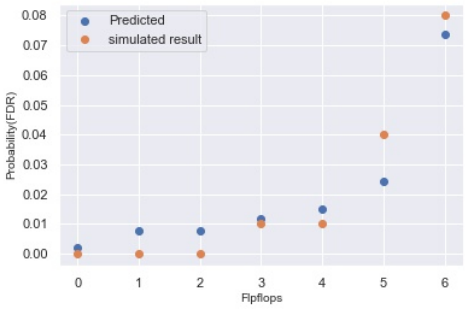


Fig. 13. The trained flip-flops and it's predicted values without sorting

hidden layers of graph convolutional neural networks beyond a certain number. This fact is very important if we consider very large circuits. But some researchers coming with new optimization methods to overcome the challenges faced by GCN.

VIII. Future work

A. Individual flip-flop's FDR prediction

All the above analysis explaining an overall distribution and overall data envelope comparison (like histogram comparison), but the algorithm is not producing a comparison of individual FDR data prediction. This aim could be achievable. This clearly concluded from figure 13 because the individual predicted FDR of trained flip-flop sequence pretty well approximating to it's the original FDR. This example taken from the case of floating point adder circuit.

Now after examining the trained sequence and it's predicted values from figure 13, we can hope to extend this work with the training phase composed of a higher number of flip-flops for achieving more accurate values.

B. Classification or clustering of registers based on FDR

It is already beginning to contemplate a future important application based on this model. It is the ability to do clustering registers based on the trained and predicted FDR. Once the model started to succeed in the prediction of individual FDR, then the classification aim will eventuate in reality.

IX. Conclusion

The works implemented in this paper depict the importance of modeling of FDR due to the soft error called SEU in microelectronic systems using a GCN network. An achieved goal by this model is the modeling of an arbitrary circuit with good accuracy and can predict the distribution of FDR derating factors. The detailed graphical comparison of predicted and stimulated FDR data for two completely different circuits, explicitly shows the prediction capability of the model. Future work for predicting more accurate individual flip-flop's FDR data going on, which may result in another dimension of applications including clustering of registers.

References

[1] M.Nicolaidis, "Soft Errors in Modern Electronic Systems," Springer Publishing Company Incorporated, 1^{st} ed, 2010.
[2] T.N. Kipf, M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," ICLR 2017.
[3] W.L. Hamilton, R. Ying, J. Leskovec, "Representation Learning on Graphs: Methods and Applications," Published in the IEEE Data Engineering Bulletin, Available: http://arxiv.org/abs/1709.05584, September 2017.
[4] F. Scarselli and M. Gori and A. C. Tsoi and M. Hagenbuchner and G. Monfardini, "The Graph Neural Network Model," IEEE Transactions on Neural Networks, Vol. 20, Jan 2009.
[5] M. Defferrard, X. Bresson and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," Curran Associates, Inc., pp. 3844–3852, NIPS 2016
[6] A. Grover and J. Leskovec, " node2vec: Scalable Feature Learning for Networks," Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.
[7] D.P. Kingma and B. Jimmy, " Adam: A Method for Stochastic Optimization," International Conference on Learning Representations, 2014.

Appendix 3

III

A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, “Composing graph theory and deep neural networks to evaluate SEU type soft error effects,” in *2020 9th Mediterranean Conference on Embedded Computing (MECO)*, IEEE, jun 2020

Composing Graph Theory and Deep Neural Networks to Evaluate SEU Type Soft Error Effects

Aneesh Balakrishnan*†, Thomas Lange*‡, Maximilien Glorieux*, Dan Alexandrescu*, Maksim Jenihhin†
*iRoC Technologies, Grenoble, France
†Department of Computer Systems, Tallinn University of Technology, Tallinn, Estonia
‡Dipartimento di Informatica e Automatica, Politecnico di Torino, Torino, Italy
{aneesh.balakrishnan, thomas.lange, maximilien.glorieux, dan.alexandrescu}@iroctech.com maksim.jenihhin@taltech.ee

***Abstract*—Rapidly shrinking technology node and voltage scaling increase the susceptibility of Soft Errors in digital circuits. Soft Errors are radiation-induced effects while the radiation particles such as Alpha, Neutrons or Heavy Ions, interact with sensitive regions of microelectronic devices/circuits. The particle hit could be a glancing blow or a penetrating strike. A well apprehended and characterized way of analyzing soft error effects is the fault-injection campaign, but that typically acknowledged as time and resource-consuming simulation strategy. As an alternative to traditional fault injection-based methodologies and to explore the applicability of modern graph based neural network algorithms in the field of reliability modeling, this paper proposes a systematic framework that explores gate-level abstractions to extract and exploit relevant feature representations at low-dimensional vector space. The framework allows the extensive prediction analysis of SEU type soft error effects in a given circuit. A scalable and inductive type representation learning algorithm on graphs called GraphSAGE has been utilized for efficiently extracting structural features of the gate-level netlist, providing a valuable database to exercise a downstream machine learning or deep learning algorithm aiming at predicting fault propagation metrics. Functional Failure Rate (FFR): the predicted fault propagating metric of SEU type fault within the gate-level circuit abstraction of the 10-Gigabit Ethernet MAC (IEEE 802.3) standard circuit.**

***Index Terms*—GraphSAGE (Graph Based Neural Network), Gate-level Circuit Abstraction, Deep Neural Networks, Functional Failure Rate (FFR), Single Event Upset (SEU), Single Event Transient (SET) and Soft Errors.**

I. INTRODUCTION

System engineering focuses on the integration of new small-scale technologies, which constantly advancing the state of the art. Current quality requirements from industrial standards and end-user requirements for high dependability applications expedite reliability modeling and assessment into an increasingly significant endeavor. The aggressive technology node scaling increased the vulnerability of radiation-induced soft errors. The issues due to radiation-based effects, particularly, Single Event Effects (SEEs) seriously impact the circuit's reliability and, the effects of impacts on the functional behavior of the circuit are challenging to evaluate. A valuable approach to tackle the challenge is the fault injection (or) simulation principle that provides precise and accurate information about circuit behaviour under stress and allowing the calculation of actual circuit-level reliability metrics.

This work was supported by the RESCUE ETN project. The RESCUE ETN project has received funding from the European Union's Horizon 2020 Programme under the Marie Skłodowska-Curie actions for research, technological development and demonstration, under grant No. 722325

A. Motivation

As mentioned above, the exhaustive fault injection method is the ultimate reliability assessment method in terms of accuracy, but it is very inconvenient in terms of time and EDA licenses; which, makes this approach infeasible on medium and large scale circuits. Therefore, a new test methodology has proposed here. The fundamental idea is to provide an alternate solution to avoid unreasonable test costs by maintaining good statistical significance in results of proposed scope. Research proposals based on Graph Theory and Deep Learning (DL) techniques are more advanced and greatly favoured by researchers to learn statistical dependencies of system-function on related parameters. This motivation develops into a method of applying GraphSAGE algorithm and trying to find the best way to develop relevant feature databases from the gate-level netlist and subsequently applying to a downstream deep neural network for the functional failure reliability metric assessment.

B. Related Works

Application of Artificial Intelligence (AI) and Deep Learning approaches to extract feature database of information in the graph network domain, were benefited in different fields. In recent years, different supervised and unsupervised DL approaches have proposed for graphical node embedding. The process of leveraging a node's features into a vector form is called the node embedding. Node2vec [1], Graph Convolutional Networks (GCN) [2] and GraphSAGE [3] have recently gained much attention from researchers for node embedding process. The application of graph-based neural network algorithms (GCN and node2vec) for circuit's reliability modeling, have proposed in papers [4] and [5] respectively. There is sufficient literature for machine learning (ML) applications in system reliability engineering. But, most of the classical machine learning algorithms rely on black-box modeling (not transparent in modeling the metrics). Here, we aiming a framework which

978-1-7281-6949-1/20/$31.00 ©2020 IEEE

could learn the structural information of circuit's gate-level abstraction in an unsupervised way (without the true target-probability information) based on graphSAGE algorithm and applied these node embedding vectors to a downstream deep learning algorithm. A proper mathematical fault propagating metric given in eq.1 has modeled in this scenario. The analyzed results providing the case of much better numerical superiority in fault propagational metric predictions and interestingly reducing the time complexity.

C. Organization of the Paper

The organization of the paper includes a brief introduction followed by sections II, III, IV, and V. Section II covers not only the theoretical background of the physical phenomena and mathematical functions to be modeled but also a brief description of graph theory and graph-based neural networks. The workflow of the framework has provided in section III. Section IV illustrates the results of the fault propagating metric predictions and, finally, a conclusion to the holistic approaches provided in section V.

II. BACKGROUND & METHODOLOGY

A. Reliability Modeling at Gate-Level

Single Event Upset (SEU) and Single Event Transient (SET) are the principal consequences of Single Event Effects (SEEs). Single Event Effects are challenging phenomena to analyse or predict when the silicon material of the circuit interacted with the radiation particles. The Single Event Upset widely used here as a prominent SEE representative, and use-case mainly implies an inversion of the stored value in a flip-flop, latch, or memory cell as the result of the radiation-induced charge. Single Event Transient represents a transient pulse of an arbitrary width due to the radioactive event and probably propagate through the combinatorial network and latched to the downstream sequential element. Among SEU and SET events, more probably SEUs will change the state sequences of the circuits and lead to a classified functional failure of the circuits. The functional failure rate due to SEU ($FFR_{i,seu}$) at the given flip-flop (i), predicting through this framework. The fault propagational probability metric $FFR_{i,seu}$ described as:

$$FFR_{i,seu} = FIT_{i,seu} \cdot \prod_{j \in T,L,F} DR_{ij} \tag{1}$$

$$FFR_{seu} = \sum_{i \in FF} FFR_{i,seu} \tag{2}$$

where, $DR_{iT}, DR_{iL},$ and DR_{iF} represent the fault derating or masking factors such as Temporal Derating (TDR), Logical Derating (LDR) and Functional Derating (FDR) respectively. Similarly, $FIT_{i,seu}$ denotes the rate of soft errors at the flip-flop (i) in Failure-In-Time (FIT) unit. Readers could refer the papers [6]–[10] for the deep insights about the radiation-induced soft errors and their inevitable intrusive nature in the functioning of microelectronic devices in aggressive radiation environments.

*1) **Temporal Derating**:* Temporal (or time) derating represents the opportunity window of an event (SET or SEU) and it's probability to be latched to the downstream sequential elements like flip-flop, latch or memory.

*2) **Logical Derating**:* The porpagational probability of SEU or SET, within the combinational (or) sequential cell networks based on their logical boolean functions is quantified as logical masking probability (or) logical derating factor.

*3) **Functional Derating**:* The probability of the SEU/SET event affects the function of the circuit's actual application. Even though the possibility of changing the circuit's state sequences is significant due to SEU/SET, the effect may be benign or masked because of the application scope.

B. Graph Theory and Deep Learning Algorithms

*1) **Graph Theory**:* The graph theory is renowned for a mathematical representation of the data objects and their pairwise relationships in a graph model. In this context, the gate-level abstraction of the circuit has transformed into a graph network where vertices (ν) analogous to the flip-flops and gates, and the directed edges (ε) represent the connection between them from input ports to output ports direction. The mathematical graph-function G of the transformed network given as:

$$G = (\nu, \varepsilon) \tag{3}$$

*2) **GraphSAGE**:* The GraphSAGE [3], a general inductive framework which leverages node's feature information to efficiently generate node embeddings for previously unseen data. GraphSAGE could be also explained as a graph based neural network with sampler and aggregator functions. Basically the GraphSAGE framework learn a function that generates the node embeddings by sampling and aggregating features from a node's local neighbourhood. Most common approaches like node2vec algorithm [1] require the availability of all the graph-nodes during the training phase of the node embedding process, and those approaches are inherently transductive and generally unable to postulate the learning function to unseen nodes. But an inductive node embedding meant to be an optimized generalization across the graph with same form of features. That is, we can leverage the node features of unseen graph part of a circuit by the embedding generator which trained once with a more generalized graph models of the circuit. This embedding part provides not only the local role of nodes in the graph but also their global positions. A sampler function defines the node's neighborhood definition through a uniform sampling of a fixed number of nodes instead of sampling the entire neighborhood space at each depth-wise iteration. It will result in boosting the optimal usage of memory and reduce runtime complexity. Generally, usage of the word 'Depth' means a measure of a fixed distance from the source node for the neighborhood search. At each iteration of depth, an aggregator function has employed. From the state-of-art of the graphSAGE framework, numerous aggregator functions are available like

Mean aggregator, Long short-term Memory (LSTM) aggregator, Pooling aggregator and Graph Convoultional Network (GCN) based aggregator. Here we implemented a Pooling aggregator with help of a python neural network libraries. The basic idea of the graphSAGE simplified and explained in the figure 2.

*3) **Deep Neural Network**:* Deep Neural Network (DNN) is an important step in machine learning applications. DNNs are trying to model data of complex distributions by combining different non-linear transformations. The elementary bricks of deep learning approaches are artificial neurons (perceptrons), which are inspired by biological neurons. An artificial neuron combines the input signals with adaptive weights and uses an activation function to deliver the output. In this work, a general fully connected DNN has implemented. The other main categories of deep learning methods are Convolutional Neural Network (CNN) and Recurrent Neural Networks (RNN).

C. Fault Injection Simulation Paradigm

As above mentioned, the true database required to train and predict the fault propagating metric ($FFR_{i,seu}$), obtained through an exhaustive Fault Injection (FI) campaign. In an exhaustive FI campaign, an SEU type fault injected independently at each flip-flop in each clock cycle of the time duration between transmission and reception of the input packets, as given in figure 1. If the injected fault (SEU) in a single clock cycle propagates through the circuit and subsequently causes the circuit's function to fail, it will account for the functional failure. Finally, $FFR_{i,seu}$ was obtained by summing the functional failures per flip-flop over the total number of clock cycles required for the operation. In total, 1100 flip-flops from different blocks of the circuit (such as TX, RX, Wishbone Interface, Fault State-machine, and Sync_clk), are tested and recorded functional failure rates ($FFR_{i,seu}$) as true database.

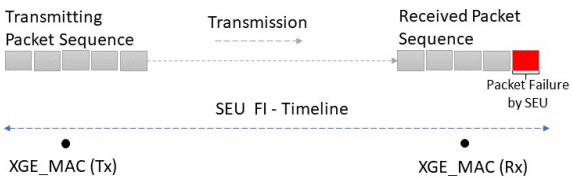


Fig. 1. Transmission between XGE_MAC Transmitter (TX) and it's Receiver (RX)

III. METHODOLOGY ILLUSTRATION

The whole approach has explained through successive workphases as follows:

A. Phase I

The proposed work implements a method to map the gate-level netlist as a Probabilistic Bayesian Graph (PGB) in the Graph Modeling Language (GML) format. To accomplish this goal, a Verilog Procedural Interface (VPI) based library function (a user-defined library) linked to a standard simulation tool (ModelSim/open-source tool). A gate-level netlist mapped to the graph model has represented in figure 2. Parallelly, the FI database simulated in this phase of the workflow.

B. Phase II

In the second phase of the approach, a feature matrix (X) corresponding to graph nodes extracted using the GrpahSAGE algorithm. As mentioned in section II-B2, GraphSAGE includes two principal steps. The premier step was the sampler algorithm. The sampler algorithm defines the neighbourhood space of a source node. In this scenario, we defined the parameter $K = 2$, which means that the sampler will sample up to the depth of 2 neighbourhood space. In the second step of the GraphSAGE algorithm, an aggregator has implemented at each depth ($1 \leq k \leq K$). This could be seen in Phase II of figure 2, where blue and green line indicates the aggregators at depth $k = 1$ and $k = 2$ respectively. Here, a max-pooling aggregator was implemented. The mathematical abstraction of the pooling aggregator [3] formulated as:

$$AGGRE_k^{pool} = max(\{\sigma(W_{pool} h_{u_i}^k + b), \forall u_i \in N_k(v)\}), \quad (4)$$

where equation 4 represents the aggregator function at depth k and it basically a neural network with parameters W_{pool} and b. Parameters optimized through unsupervised learning. $N_k(v)$ represents k-neighbourhood of vertex v and $h_{u_i}^k$ indicates the aggregated neighborhood vector and, σ is the activation function of the neural network. In this way, we could represent the whole GraphSAGE algorithm as a graph-based neural network. At the end of this phase, each node reformed into a corresponding vector and alternatively form a matrix representation (X) of the circuit as given in figure 2.

C. Phase III

Phase III of figure 2 elucidates the DNN algorithm that exercised for prediction purposes. There are two parts included in phase III. The first part is the training part of DNN, and the second one is the testing part of DNN. In the training part, 40 % of the feature matrix and corresponding target probability metric from FI - database, are taken to postulate a hypothesis that best describes the target probability distribution ($FFR_{i,seu}$) by supervised learning method. The optimized parameters (Weights and Bias) of the best fit of the target distribution should provide as model parameters. In the testing part, the proposed model applies to an unknown input vector and predicts the target probability metric. The DNN architecture consists of 5 dense layers, including the input and the output layers.

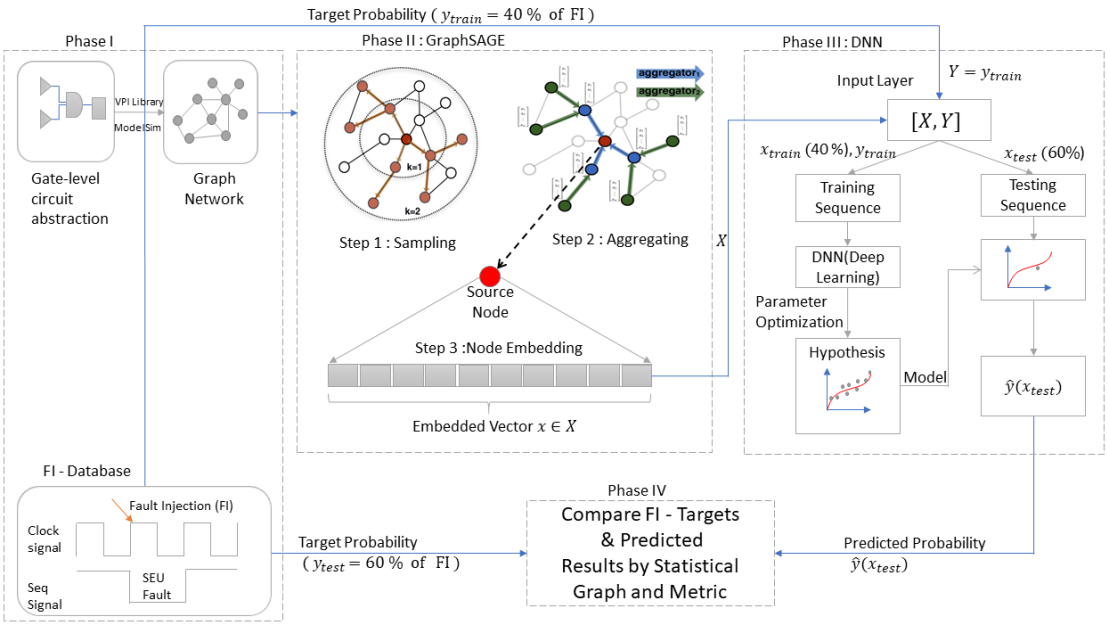


Fig. 2. A Systematic workflow diagram of the implemented scientific work

D. Phase IV

The final phase includes a comparison between the predicted and target probability metrics. The compared results plotted in figure 3, as well as the impacts of results provided in table I.

IV. EXPERIMENTAL RESULTS

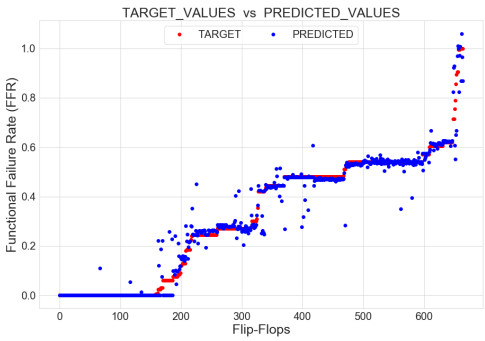


Fig. 3. Graph Comparison : Mean Absolute Error (MAE) = 0.0186 and coefficient of determination (R^2 Metric) = 0.96 with a test size of 60%

In figure 3, the functional failure rates of flip-flops to be predicted were shown in red color, and the corresponding predicted probabilities have shown in blue color. The case study has conducted with the gate-level circuit of the 10-Gigabit Ethernet MAC. The graphical comparison gives a visualization of how well the prediction replicates the observed database. In this case, the DNN prediction achieves the coefficient of determination (R^2) value of approximately 0.96, where the best model fit value of R^2 metric is 1, and the worst value is 0. In statistics, the R-squared (R^2) value is the measure of goodness-of-fit of the proposed regression model and, the projected R-squared value (0.96) able to explain most of the variation in the response data. The entire work repeated and achieves a good prediction accuracy with other standard circuits (e.g., The USB 1.1 Function IP Core).

TABLE I
IMPACT OF PREDICTION IN TIME AND TOOL REQUIREMENTS

Model	Time	Tool	Model Fit (R^2)
Fault Injection	17 hours	7 Modelsim	Target Model
Fault Injection	$\approx$ 5 days	1 Modelsim	Target Model
GraphSAGE + DNN (Test + Training)	$<$ 10 minutes	1 Modelsim	0.96

Table I outlines the impacts of accelerated predictions in terms of time and simulation tool requirements. Even-though GraphSAGE and DNN based ensemble algorithm provide a significant reduction in the required test resources without com-

promising the quality of modeling, the implemented algorithm depends on 40% of FI-database for training the downstream DNN as picturized in Phase III of fig.2. But, it is quite impressive to note that the test and training phase of the whole algorithm takes only less than 10 minutes.

V. CONCLUSION

An accelerated testing methodology; that is scalable and very cost-effective in resource handling, has developed for medium and largescale circuits to predict Functional Failure Rate due to SEU type fault without dropping the significance of the statistical modeling.

REFERENCES

[1] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 07 2016, pp. 855–864.

[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[3] W. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 1024–1034. [Online]. Available: http://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs.pdf

[4] A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, "Modeling gate-level abstraction hierarchy using graph convolutional neural networks to predict functional de-rating factors," in *2019 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2019, pp. 72–78.

[5] A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, "The validation of graph model-based, gate level low-dimensional feature data for machine learning applications," in *2019 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, 2019, pp. 1–7.

[6] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, 2005.

[7] M. Ebrahimi, A. Evans, M. B. Tahoori, E. Costenaro, D. Alexandrescu, V. Chandra, and R. Seyyedi, "Comprehensive analysis of sequential and combinational soft errors in an embedded processor," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1586–1599, 2015.

[8] D. Alexandrescu, E. Costenaro, and M. Nicolaidis, "A practical approach to single event transients analysis for highly complex designs," in *2011 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, 2011, pp. 155–163.

[9] D. Alexandrescu and E. Costenaro, "Towards optimized functional evaluation of see-induced failures in complex designs," in *2012 IEEE 18th International On-Line Testing Symposium (IOLTS)*, 2012, pp. 182–187.

[10] R. Baumann, "The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction," in *Digest. International Electron Devices Meeting,*, 2002, pp. 329–332.

Appendix 4

IV

M. Jenihhin, M. S. Reorda, A. Balakrishnan, and D. Alexandrescu, “Challenges of reliability assessment and enhancement in autonomous systems,” in *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, IEEE, oct 2019

Challenges of Reliability Assessment and Enhancement in Autonomous Systems

Maksim Jenihhin[1], Matteo Sonza Reorda[2], Aneesh Balakrishnan[1,3], Dan Alexandrescu[3]

[1]*Tallinn University of Technology, Estonia, maksim.jenihhin@taltech.ee*
[2]*Politecnico di Torino, Italy, matteo.sonzareorda@polito.it*
[3]*IROC Technologies, France, { aneesh.balakrishnan |dan.alexandrescu }@iroctech.com*

***Abstract*— The gigantic complexity and heterogeneity of today's advanced cyber-physical systems and systems of systems is multiplied by the use of avant-garde computing architectures to employ artificial intelligence based autonomy in the system. Here, the overall system's reliability comes along with requirements for fail-safe, fail-operational modes specific to the target applications of the autonomous system and adopted HW architectures. The paper makes an overview of reliability challenges for intelligence implementation in autonomous systems enabled by HW backbones such as neuromorphic architectures, approximate computing architectures, GPUs, tensor processing units (TPUs) and SoC FPGAs.**

Keywords— reliability, safety, fault tolerance, autonomous systems, standards.

I. Introduction

Recent rapid expansion of autonomous systems has enabled numerous unprecedented novel services and businesses. However, the unleashed benefits come along with computationally extremely challenging mission- and safety-critical application scenarios. The gigantic complexity and heterogeneity of today's advanced cyber-physical systems and systems of systems is multiplied by the use of avant-garde computing architectures to employ artificial intelligence based autonomy in the system. The setups such as swarms of autonomous robotic vehicles are already on the doorstep and call for novel intelligent approaches for reliability that are often the key enabling factor for a new product or technology on the way to market. This success is supported by the connectivity solutions being developed in the IoT research discipline that is also moving towards enhanced autonomy of the connected intelligence-enabled things [1].

Expectations for reliability are very wide as also the variety of autonomous systems. The latter are driven by a number of killer applications listed below:

- *autonomous vehicles* in the automotive domain is the dominant application in terms of funding and recent research efforts invested, includes cars with Autonomous Driving (AD) through levels 3 to 5 of autonomy;
- *aircrafts* with different degree of autonomy, e.g. employing the 'fly-by-wire' reliability-critical systems;
- *unmanned aerial vehicles* (UAVs) commonly known as drones, both fixed-wing and rotary (quadcopters), these days are equipped with very high degree of intelligence and tend to operate autonomously individually or in *UAV swarms* (autonomous System of systems);
- *unmanned ground vehicles* (UGVs) include among others rapidly developing *self-driving delivery robots* (e.g. [2]) and *farming robots*;
- *unmanned underwater vehicles* (UUVs), e.g. robotic fishes, and *unmanned boats* (also *unmanned surface vessel* (USV)) are heading at long-term operation in harsh environments;
- *autonomous spacecrafts* such as satellites and autonomous landers for remote missions often with limited communication capabilities;
- *autonomous military and law enforcement applications*, e.g. that may be dual use of the above mentioned systems but also specific weapons, e.g. autonomous missiles.

Today, autonomous systems are quickly getting on top of the hype cycle [3] and several very recent studies have started to look into the enabling aspects of such systems, e.g. from the standards perspective [4], from the security perspective [5], or for a specific application [6]. In this paper, we make an overview of reliability challenges in autonomous systems. The rest of this paper is organized as follows. Section II outlines the challenging attributes of autonomous systems. Section III and IV target at understanding industrial standards and the key concepts in reliability and safety for autonomous systems Sections V and VI analyze specific requirements introduced by novel applications and architectures and Section VII discusses reliability enhancement. Section VIII wraps up the paper.

II. Autonomous Systems' Attributes from the Reliability Perspective

The attributes of an *autonomous system* (AS) from the reliability requirements perspective may be summarized in the following set of challenges:

External attributes:

a. Specific application domains and operating environments, i.e. dangerous, tedious, remote/hardly- or in-accessible for human involvement;
b. Limited availability and latency of external support for repair or critical decision-making;
c. Real-time constraints (often hard real time);
d. An AS is usually a cyber-physical system immersed into physical world through intensive interaction by sensors (also implying sensor fusion) and actuators [7];

978-1-7281-2260-1/19/$31.00 ©2019 IEEE

e. Often, several ASs are combined into a System of Systems (SoS) [8] with intensive machine-to-machine communication and resource sharing, enabling computing continuum and complex distributed computing architectures, e.g. edge-to-fog computing;
f. An AS imply subjectively higher expectations to reliability level and lower tolerance to unsafe behavior compared to a human-operated system.

Internal attributes:

g. High complexity of the computing architectures capable to run computation-intensive evolvable artificial intelligence software (e.g. the novel GPUs, the TPU for Google's TensorFlow and similar);
h. Specificity of Hardware Neural Networks implementations that are rather a "sea of elements", with reduced structural/functional modularization of hardware;
i. ASs are built utilizing a combination of many very new untested in-field technologies;
j. An AS implementation has strong dependency on the quality of assumptions about the ambient, often dynamic and uncertain environment.

The state-of-the-art academic solutions, e.g. [9],[10],[11], are either incapable or inefficient to tackle this union of challenges. The practical reliability drivers in today's designs are industrial standards in different application domains such as [12] and its application-specific derivatives, e.g. [13], that do not address cross-layer approaches. The standards mostly address functional safety of the complete system rather than just a component and depend on the integrated operation of all sensors, actuators, control devices, and other units. Therefore, the functional safety standards are usually unspecific to the solutions at the integrated circuit (IC) level. The standard for IC stress test [14] in the automotive domain covers requirements for a subset of reliability issues (such as NBTI, HCI, electromigration, etc.) at the chip level. However, many recent application domains unleashed by the unmanned systems, e.g. UAVs, remain uncovered [15],[16].

III. Understanding Reliability Standards in Autonomous Systems

Autonomous systems will enable huge societal changes (and possibly progress). As expected, stringent safety and reliability expectations and requirements are firmly set in international standards, implicit customer expectations and, not unexpectedly, insurance policies. Autonomous systems are also an emerging industrial field and are very likely to stay with us for a very long time. Accordingly, it is very probable that many successive, evolutionary or revolutionary standards will be issued to govern them. International standards are the clearest and most authoritative prescribers regarding reliability and safety. The list of current or under-development standards in this field includes:

- IEC 61508 [12] (Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems) is aimed at all industrial fields and is the template for many application-specific standards;
- One of the most well-known derivatives of the previous is ISO 26262 [13], which addresses the functional safety of automotive systems;
- IEC 62279 is an adaptation of [12] for railway applications;
- ISO 13849 [17] is a safety standard which applies to parts of machinery control systems that are assigned to provide safety functions;
- AC 25.1309-1A [18] (System Design and Analysis) provides background for important concepts and issues within airplane system design and analysis;
- RTCA/DO-254 [19] (Design Assurance Guidance for Airborne Electronic Hardware) provides guidance for the development of airborne electronic hardware.

Since change is a permanent feature of the industrial progress, expectations and requirements constantly evolve. While intended to be robust and durable, standards are not safe from being prone to latest fashions and currents in the industry or from being influenced by companies and organizations looking to promote their own position and offering.

Particularly, the terminology and dictionary of any standard is a faithful snapshot of the particular context at the time of the writing and often suffers from updates, changes of signification, meaning overcharges and obsoleteness during the expected lifetime of a standard and even more so when a new standard is devised. The goal of this Section is to pinpoint some basic topics that are common to the different standards and faced by most of them. They are summarized in Table I.

Many standards include a part related to *Terminology*. In this category, the signification and a clear definition of the key terms shall be presented and elaborated. However, the specific meaning can hide behind an ordinary word, requiring a more in-depth discussion and explanation and investing the simple term with a fundamental weight. The "Terminology" category would thus benefit from a *Concepts* sub-category.

As soon as the key terms and concepts have been introduced, the standards are fast to move to the explanation of their core methodology, framework and principles. The *Methodology* category covers these aspects. In their various proposed methodologies, many of the standards address "risks" to the safety of the intended applications and set a mix of quantitative and qualitative requirements and expectations for these risks. These objectives and goals will be captured in the *Requirements* category. The reliability and safety of any application will have to be checked against the applicable requirements and improved

TABLE I. Main Topics in the Reliability and Functional Safety Standards

Terminology	Methodology	Requirements	Assessment	Management	Environment
Vocabulary	Development:	Hazard & Risks	Models	Online	Electrical
Concepts	System-level	Classification	Probabilistic	Offline	Thermal
	Hardware-level	Event rates	Simulation	Diagnostic	Mechanical
	Software-level	Mitigation		Maintenance	Radiation

until its behavior fulfils the expectations of the intended standard. Accordingly, the taxonomy will have to include the *Assessment* and *Management* categories.

Lastly, any application is designed to work safely and reliably in a given setting. The *Environment* category would capture the entirety of electrical, thermal, mechanical, radiative conditions to which the application will be subjected.

In practice, the concerns about reliability may mix together with those about feasibility (especially when target features are particularly challenging). For this reason, independently on standards and regulations, general safety praxis can be utilized e.g. by using the ALARP method ("as low as reasonably practicable") and providing justification for benefits of the society against the involved risks.

IV. Key Concepts in Reliability and Functional Safety for Autonomous Systems

For autonomous systems, but not only, the notions of "*reliability*" and "*safety*" comprise as many significations as engineers from different industries want to invest in them. Loosely, reliability represents the probability of a system to fail, i.e. higher reliability means less failures, while safety generally means that the system fails in a safe way. A reliable system can be unsafe while a safe system can be unreliable. Furthermore, systems can be made arbitrarily safe and reliable with a corresponding investment of resources and time. Requirements for reliability and safety can be quantitatively and qualitatively very different but standards are often aggressive in setting high requirements for both safety and reliability. The most straightforward approach to address both reliability and safety is to rank risks and hazards according to their impact (*safety*) and to expect that the probability of risks (*reliability*) decreases inversely to their impact. An aggregated event rate (often measured in terms of Failure in Time, or FIT), may be associated to the system and/or component according to their role but with an underlying understanding of the risks that make up the "Failure" key term.

In this way, quantity and quality, safety and reliability are harmoniously integrated. However, reliability engineers will find that this task is relatively difficult as two opposing concepts still need to be conciliated: objective versus subjective. The qualificative of "*Objective*" can be applied to any physical measurements. As an example, technology fault rates can be expressed accurately; a "*Soft Error Rate*" is an objective measurement of the susceptibility of a technological process under radiations. Faults propagate through the circuit and system and can become Failures. Various methods, such as static and dynamic ones, can accurately and undisputedly (thus objectively) predict the fact that a fault occurring in a deeply-embedded logic cell instance can propagate and affect a primary system output. The question that the reliability engineers and their design colleagues must answer now is whether this fault consequence represents a failure or not, what are the actual consequences and, more importantly, where exactly in terms of risk levels the failure needs to be classified. This is the "*Subjective*" part and standards try to address this by a prescriptive, function-based assessment. However, in practice, the whole procedure provides some freedom and margins to reliability engineers that can argue for a less critical classification of possible fault outcomes.

Fig. 1. Faults, errors and failures in a system

Probabilistic risk evaluation and management is a core concept of many reliability assessments. Only a fraction of technological faults will propagate through the circuit and become errors, i.e., erroneous data or values stored instead of correct information. Only a percentage of errors will become failures causing observable deviations of the system behaviour. Furthermore, failures can be classified in criticality classes. If error detection/correction/management features are implemented, they can address faults, errors and failures at any design level and can reduce the percentage of events graduating from one level to the upper one (see Fig. 1).

A first, fundamental contributor to the quality of an autonomous system is the *quality of the underlying implementation technology*. The manufacturing process must present a well-characterized, preferably low intrinsic defect and fault rate, resiliency to environmental challenges and a good, well known aging and degradation performance. Moreover, the technology providers (foundries) must offer their customers a full ecosystem with the tools, IPs and solutions for reliable and safe circuit design.

A second contributor lies in integrating into the system some solutions for *lifetime performance assurance*. The classical bathtub curve is no longer an evidence and the reliability of the system must be managed during the expected lifetime through online and offline monitoring, embedded sensors, test instruments and safety mechanisms (see Fig. 2).

Configurability and Adaptability, to environment and workload challenges, as well as to intrinsic degradation and aging, are

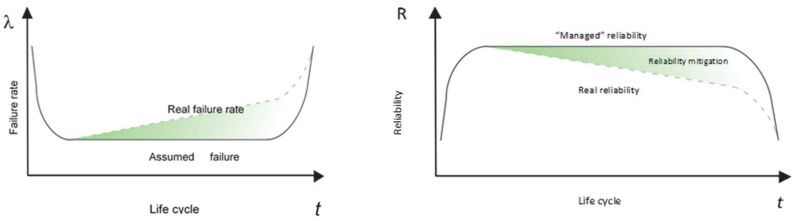


Fig. 2. Managed lifetime reliability (courtesy of the RESIST project).

important for today's autonomous systems running dynamic applications in diverse environments.

Lastly, the evolution to "*Self-*" Everything (self-monitoring, self-calibration, self-adaptation, self-configuration, etc.) is an important industry trend and goal that can provide solutions for more reliable and safer autonomous systems.

V. Novel Autonomous Systems' Application-Specific Requirements for Reliability Validation

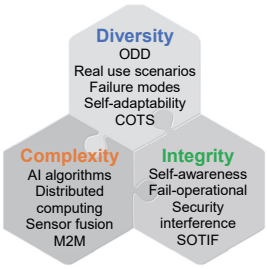


Fig 3. The three clusters of application requirements in autonomous systems

The plethora of new applications unleashed by autonomous systems introduce novel or bring to the front existing requirements for the system reliability validation. These can be represented in the following three clusters as shown in Fig. 3.

A. Diversity

One of the main challenges for the autonomy is the diversity of the environment and operational conditions. A solution to simplify the problem is to split it to a limited (countable) number of *Operational Design Domains* (ODD), e.g. as introduced by the NHTSA agency. Considering a particular autonomous system, these may include such factors as operational terrain, environmental and weather conditions, communication modes, etc. [23].

Proper reliability validation implies capturing *real world scenarios* along with *real failure modes* that may imply a significant amount of statistical data collected. For example, the calculations in [24] demonstrate that this results in billions of miles of in-field test. This approach is also valid for any-scale safety-critical autonomous systems, e.g. UGVs [2].

The implementation of a complex autonomous system or even a system of systems may involve a *diversity of available fault tolerance structures* in the components employed, e.g. involving COTS parts along with hardened ones. Moreover, as mentioned in the above section, the diversity of autonomous systems is amplified by their features, such *self-adaptability* and *self-configuration*.

B. Complexity

Autonomous systems have introduced a conceptually new level of complexity for reliability validation. Here, application of *artificial intelligence algorithms* is the main factor for the rapid increase of complexity in HW architectures (see Section VI). This includes in particular *smart distribution of computation* and related *advanced heterogeneous communication schemes*.

Both system's mission tasks and decision-making autonomy imply complexity for enabling *sensor fusion* from a multitude of integrated diverse sensors.

Vehicle-to-vehicle communication (V2V) has enabled efficient autonomy in complex dynamic networks of machines, e.g. heterogeneous swarms, at the price of very large overall systems' complexity.

Today, the industrial classification of autonomous systems' ICs in million gates (MG) assumes the following ranges: small-scale design <32MG; medium-scale <200MG; large-scale >200MG.

C. Integrity

The main system integrity requirements brought to the front by autonomous systems use the *self-monitoring* and corresponding *self-awareness* that became a must and significantly support reliability enhancement. To cope with the diversity and complexity of environments the concepts such as *responsibility-sensitive safety* and *safety of the intended functionality* (SOTIF) [4] target at the autonomous system misuse cases and establish framework for responsibility sharing.

The challenge of high-severity failures in the system often dictates the requirement of the system to be *fail-safe* or even *fail-operational* thus also establishing diverse reliability requirements for modules to be available in the degraded mode. Here the complete system may benefit in some cases from the swarming of agents when the mission tasks can be re-allocated and accomplished even if some agent fails.

Finally, reliability validation is challenged by interference with requirements by other design aspects [20], [21] and correlation with security requirements in particular (e.g. [22] by ACEA association). For example, a new security standard under development (ISO 21434) is aiming at defining a Cybersecurity Assurance Level (CAL), similar to the ASIL concept [13].

VI. Challenges by Novel HW Architectures

In most cases, the requirements specified for autonomous systems cannot be fulfilled without resorting to advanced computing architectures and semiconductor technologies.

Concerning the architectures, this means that conventional CPU-based ones are substituted by alternative ones, including not only multicore devices, but also special modules, such as *General-Purpose Graphic Processing Units* (GPGPUs). Since Artificial Intelligence is widely used in autonomous systems, accelerators specifically targeting neural networks, such as the *Tensor Processing Units*, or TPUs, by Google, are intensively investigated and increasingly adopted. Complex innovative architectures such as the massively parallel, low-precision floating-point compute *Intelligence Processing Unit* (IPU) [25] require the smallest technology nodes to allow the necessary density of transistors bringing along the atomic scale reliability challenges. Finally, for some applications it may be convenient to include in the architecture some *FPGA modules* able to provide the flexibility to dynamically change the hardware supporting the implemented functions. All the mentioned components may be used as single devices or integrated into even more complex Systems on Chip (SoCs).

Moving from conventional architectures to the mentioned ones rises several important issues, not only in terms of hardware design complexity (e.g. in terms of validation) and software development and qualification, but also in terms of getting a sufficient understanding of their sensitivity to possible faults.

While several theoretical and experimental analysis provided information about the *Architecture Vulnerability Factor* [26] of traditional architectures [27], only preliminary results have been provided concerning the new ones and about the main modules they are composed of [28],[29],[30]. Moreover, these analyses are currently made more complex by the difficulty in getting representative open source models for the new architectures [31]. Similarly, the impact of possible faults affecting the new architectures when executing some common kinds of applications used in autonomous systems (e.g. those related to video processing and to neural networks implementation) is only partly understood [32].

On the other side, the requirements in terms of complexity, speed, power consumption and miniaturization force the adoption of advanced semiconductor technologies. Even if we stick to CMOS technologies, those that are going to be used for autonomous systems are widely unknown in terms of reliability. Hence, their adoption in safety-critical applications (as those of autonomous vehicles often are) asks first for effectively developing *new fault models*, given that the currently adopted ones are largely unsuitable to deal with the new defects characterizing these technologies. Secondly, *solutions* to detect and possibly tolerate these faults should be identified, taking into account that they should be able to trade-off their effectiveness with several other parameters, including cost and time-to-market. Finally, even when solutions will be available, their adoption will be possible only if *EDA tools* supporting them in a fully integrated manner with respect to the design and test flow will exist.

All the above issues are likely to become even more critical when non Von Neumann architectures and post-CMOS technologies will start to be adopted [33].

VII. Reliability Enhancement

This Section aims at briefly summarizing some crucial points related to the current status of the art in the area of solutions able to achieve the required level of reliability when the electronic part of autonomous systems is considered.

As explained in the previous Sections, ASs are likely to expand significantly in the next years, provided that some of the technical and organization issues we are summarizing will be successfully overcome. Since now, we can imagine some trends that may be followed in the next future, based on what is happening in some representative and more advanced domains within the wide area of ASs, such as the automotive one.

Since the adopted technologies are intrinsically less reliable, a first trend goes into the direction of developing *solutions at the architecture level* that may guarantee by construction the target level of reliability. Solutions based on Duplication With Comparison (DWG), such as lockstep, are increasingly adopted when fault detection is the main target [34]. Unfortunately, their extension to the new architectures described in Section VI is not straightforward, although similar solutions implemented at the software level have been successfully explored already, especially with regular structures such as those of GPGPUs. Clearly, the adoption of hardware DWG architectures requires the development of specific products targeting safety-critical applications, only. Some recent products (e.g. Xavier by NVIDIA) go in this direction. Other products (e.g. the solution named Split-Lock introduced by ARM), although based on more conventional architectures, allow the user to dynamically decide whether to use the available redundancy to increase performance or reliability.

Given the high cost of solutions able to tolerate faults resorting to hardware redundancy (e.g., based on Triple Modular Redundancy, or TMR), alternative solutions exploiting *reconfiguration* seem particularly attractive [35],[36]. In particular, they may provide a mechanism to extend the lifetime of adopted circuits, whose span is quickly shrinking in advanced technologies, and tends to be increasingly dependent on the operating environment and workload [37]. Additionally, the new technologies may increase the chance that multiple faults occur in a logic block, as it already happens for memory ones, and this may increase the negative impact of fault accumulation, which can hardly be managed via TMR. Reconfiguration can be applied at different levels and managed either directly in hardware, or resorting to Operating System features. In all cases, suitable techniques able to quickly detect (and possibly locate) faults are crucial. Existing *Design for Testability structures* (e.g. supporting Logic BIST for chunks of logic, or Memory BIST for memory modules) already introduced to support end-of-manufacturing test may be re-used for this purpose. In other cases users resort to functional solutions, e.g. based on the so-called Self-Test Libraries [38], which rely on the Software-based Self-test approach [39],[40]. This solution seems particularly effective for in-field test of complex systems, since it allows to exploit self-test code provided by the developer of the modules composing the system and able to achieve a given fault coverage, which is integrated by the user into the application code and run when required to achieve the target reliability figures (e.g., at the Power-On, or periodically, or when specific error conditions happen). The results produced by the system when executing such pieces of code allow the detection of possible faults [41]. Preliminary results show that the same approach can be extended also to new architectures, such as GPGPUs [42].

Another interesting research direction which is proving to be promising for autonomous systems lies in *trading-off precision with reliability*. Preliminary results about techniques where hardware resources saved by moving to a lower precision are used to increase reliability are shown in [43].

Given the complexity of the systems, we are targeting, in most cases they will integrate components designed and produced by different companies. Hence, cross-layer approaches to reliability are highly promising [44], but require a clear definition of what each level should guarantee in terms of reliability, and how this can be validated.

It is worth mentioning that ASs are characterized by a wide variety of scenarios and constraints. In some of them, different and highly independent systems will *cooperate* to achieve a given target (Systems of Systems, or SoSs). In such cases, new paradigms may emerge even from the point of view of reliability. The complexity and heterogeneity of the resulting SoS may favor the adoption of holistic solutions which will enable it to implement highly innovative features, such as self-reconfiguration, self-test, and implicit robustness.

As a final comment, we would like to emphasize the already mentioned *increasing importance of security* for autonomous

systems [5]. Several works highlighted that facing security often requires adopting solutions based on opposite strategies with respect to those required by reliability and test, e.g. in terms of system status observability. For this reason, integrated solutions identifying suitable trade-off between opposite constraints are required [45].

VIII. Conclusions

The paper has presented an overview of reliability challenges related to the novel and very rapidly developing domain of autonomous systems. The challenges of reliability assessment and enhancement stem from a set of general attributes, novel applications' specific requirements and new hardware architectures of autonomous systems. The way forward for the research community and industry lays in understanding the new needs, collaboration towards comprehensive solutions and adoption of new appropriate standards.

Acknowledgments

This research was supported in part by projects H2020 MSCA ITN RESCUE funded from the EU H2020 programme under the MSC grant agreement No.722325, by the Estonian Ministry of Education and Research institutional research grant IUT19-1 and by European Union through the European Structural and Regional Development Funds.

References

[1] Intel, The Three Phases of the IoT Revolution and the Resources Developers Need to Get Started, May 2, 2017, https://intel.com/
[2] Starship Technologies, The Self-Driving Delivery Robot, 2019, https://www.starship.xyz/
[3] Gartner Top 10 Strategic Technology Trends for 2019, October 15, 2018, https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technologytrends-for-2019/
[4] R. Mariani, "An overview of autonomous vehicles safety," *2018 IEEE International Reliability Physics Symposium (IRPS)*, Burlingame, 1-6.
[5] Stefan Katzenbeisser, Ilia Polian, Francesco Regazzoni, Marc Stottinger, "Security in Autonomous Systems", 2019 24th IEEE European Test Symposium (ETS)
[6] Shakhatreh, H., et al (2018). Unmanned Aerial Vehicles: A Survey on Civil Applications and Key Research Challenges. CoRR, abs/1805.00881
[7] M. Törngren et al. (2018): How to Deal with the Complexity of Future Cyber-Physical Systems? http ://www.mdpi.com/2411-9660/2/4/40/htm
[8] Khabbaz Saberi, et al, (2018). On functional safety methods: A system of systems approach, 12th IEEE Int. Systems Conference, SysCon 2018
[9] M. Ottavi et al., "Dependable Multicore Architectures at Nanoscale: The View From Europe", Design & Test, IEEE, vol.32, no.2, pp.17,28, 2015
[10] Aleksandrowicz G. et al. (2018) Designing Reliable Cyber-Physical Systems. In: Fummi F., Wille R. (eds) Languages, Design Methods, and Tools for Electronic System Design. Lecture Notes in Electrical Engineering, vol 454. Springer, Cham
[11] A. Jutman, C. Lotz, E. Larsson, M. Sonza Reorda, M. Jenihhin, J. Raik, et al., "BASTION: Board and SoC test instrumentation for ageing and no failure found," DATE 2017, pp. 115-120.
[12] IEC 61508-1:2010 Functional safety of electrical/electronic/programmable electronic safety-related systems, 2010, International Electrotechnical Commission
[13] ISO 26262-1:2011/2018 "Road vehicles — Functional safety", 2018, International Standardization Organization
[14] AEC-Q100 Failure Mechanism Based Stress Test Qualification for Packaged Integrated Circuits, Rev-H September 11, 2014, Automotive Electronics Council
[15] TÜV, „The emerging regulatory landscape for unmanned aircraft systems", White Paper, 2018
[16] M. Mlezivova, "Unmanned Aircraft as a Subject of Safety and Security", Magazine of Aviation Development, 6(3):12-16, 2018
[17] Functional Safety of Machinery: EN ISO 13849, http://www.iec.ch/search/?q=13849
[18] http://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_25.1309-1A.pdf, 1988
[19] RTCA/DO-254, Design Assurance Guidance for Airborne Electronic Hardware, http://www.do254.com/
[20] H2020 MSCA ETN RESCUE project, http://rescue-etn.eu/
[21] X. Lai, A. Balakrishnan, T. Lange, M. Jenihhin, T. Ghasempouri, J. Raik, D. Alexandrescu, Understanding Multidimensional Verification: Where Functional Meets Non-Functional, Microprocessors and Microsystems, 102867, ISSN 0141-933, 2019
[22] "Principles of Automobile Cybersecurity", European Automobile Manufacturers' Association ACEA, 2017
[23] P. Koopman, F. Fratrik, How Many Operational Design Domains, Objects, and Events?, Safe AI 2019: AAAI Workshop on Artificial Intelligence Safety, Jan 27, 2019
[24] K. Nidhi and S.M. Paddock, *Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?*, Santa Monica, Calif.: RAND Corporation, RR-1478-RC, 2016. As of June 14, 2019: https://www.rand.org/pubs/research_reports/RR1478.html
[25] Intelligence Processing Unit (IPU) - Graphcore's technology for accelerating machine learning and AI https://www.graphcore.ai/technology
[26] S. Mukherjee et al., "A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor", 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003.
[27] S. Mirkhani, S. Mitra, C.-Y. Cher, J. Abraham, "Efficient soft error vulnerability estimation of complex designs", 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)
[28] F. F. dos Santos, P. Foletto Pimenta, C. Lunardi, L. Draghetti, L. Carro, D. Kaeli, P. Rech, "Analyzing and Increasing the Reliability of Convolutional Neural Networks on GPUs", IEEE T RELIAB, 2019, 68, 2, 663 – 677
[29] B. Fang, K. Pattabiraman, M. Ripeanu, and S. Gurumurthi, "GPU-Qin:A methodology for evaluating the error resilience of GPGPU applications," IEEE Int. Symp. Perform. Anal. Syst. Softw., Mar. 2014,pp. 221–230.
[30] J. Tan, N. Goswami, T. Li, and X. Fu, "Analyzing soft-error vulnerability on GPGPU microarchitecture," IEEE Int. Symp. Workload Characterization, Nov. 2011, pp. 226–235.
[31] B. Du, Josie E. Rodriguez Condia, Matteo Sonza Reorda, "An extended model to support detailed GPGPU reliability analysis", 2019 14th International Conference on Design & Technology of Integrated Systems In Nanoscale Era (DTIS)
[32] Fernando Fernandes dos Santos, Luigi Carro, Paolo Rech, "Kernel and layer vulnerability factor to evaluate object detection reliability in GPUs", IET Computers & Digital Techniques, 2019 13(3), 178 – 186
[33] Y. Wang et al. "Taming Emerging Devices' Variation and Reliability Challenges with Architectural and System Solutions", 2019 IEEE 32nd Int. Conf. on Microelectronic Test Structures (ICMTS), 90 – 95
[34] X. Iturbe, B. Venu, J. Jagst, E. Ozer, P. Harrod, C. Turner, J. Penton, "Addressing Functional Safety Challenges in Autonomous Vehicles with the Arm TCL S Architecture", IEEE Design & Test, 35(3), 7 – 14, 2018,
[35] M.G. Gericota, G.R. Alves, J.M. Ferreira, "A self-healing real-time system based on run-time self-reconfiguration", 2005 IEEE Conference on Emerging Technologies and Factory Automation
[36] Lukas Märtin, Anja Nicolai, "Towards self-reconfiguration of space systems on architectural level based on qualitative ratings", 2014 IEEE Aerospace Conference
[37] A. Sivadasan, S. Mhira, A. Notin, A. Benhassain, V. Huard, E. Maurin ; F. Cacho, L. Anghel, A. Bravaix, "Architecture- and workload-dependent digital failure rate", 2017 IEEE Int.Reliability Physics Symposium (IRPS)
[38] F. Pratas et al. "Measuring the effectiveness of ISO26262 compliant self test library", 2018, Int. Symp. on Quality Electronic Design (ISQED)
[39] L. Chen, S. Dey, "Software-based self-testing methodology for processor cores", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2001, 20(3), 369 – 380
[40] A.Oyeniran, R.Ubar, M.Jenihhin, C.C. Gursoy, J. Raik, High-Level Combined Deterministic and Pseudoexhaustive Test Generation for RISC Processors, 2019 IEEE European Test Symposium (ETS)
[41] P. Bernardi, R. Cantoro, A. Floridia, D. Piumatti, C. Pogonea, A. Ruospo, E. Sanchez, S. De Luca, A. Sansonetti, "Non-Intrusive Self-Test Library for Automotive Critical Applications: Constraints and Solutions", 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)
[42] Stefano di Carlo, Josie E. Rodriguez Condia, Matteo Sonza Reorda, "On the in-field test of the GPGPU scheduler memory", 2019 IEEE Int. Symp. on Design and Diagnostics of Electronic Circuits & Systems (DDECS)
[43] Fernando Fernandes dos Santos et al., "Reliability Evaluation of Mixed-Precision Architectures", 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA), Pages: 238 – 249.
[44] Eric Cheng, et al "Tolerating Soft Errors in Processor Cores Using CLEAR (Cross-Layer Exploration for Architecting Resilience)", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 37(9), 1839 – 1852
[45] Jennavive Benko, et al "Security and Resiliency of Coordinated Autonomous Vehicles", 2019 Systems and Information Engineering Design Symposium (SIEDS).

Appendix 5

V

T. Lange, A. Balakrishnan, M. Glorieux, D. Alexandrescu, and L. Sterpone, “On the estimation of complex circuits functional failure rate by machine learning techniques,” in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks – Supplemental Volume (DSN-S)*, IEEE, June 2019

2019 IEEE/IFIP International Conference on Dependable Systems and Networks Supplemental

On the Estimation of Complex Circuits Functional Failure Rate by Machine Learning Techniques

Thomas Lange*†, Aneesh Balakrishnan*‡, Maximilien Glorieux*, Dan Alexandrescu*, Luca Sterpone†
*iRoC Technologies, Grenoble, France
†Dipartimento di Informatica e Automatica, Politecnico di Torino, Torino, Italy
‡Department of Computer Systems, Tallinn University of Technology, Tallinn, Estonia
{thomas.lange, aneesh.balakrishnan, maximilien.glorieux, dan.alexandrescu}@iroctech.com luca.sterpone@polito.it

***Abstract*—De-Rating or Vulnerability Factors are a major feature of failure analysis efforts mandated by today's Functional Safety requirements. Determining the Functional De-Rating of sequential logic cells typically requires computationally intensive fault-injection simulation campaigns. In this paper a new approach is proposed which uses Machine Learning to estimate the Functional De-Rating of individual flip-flops and thus, optimising and enhancing fault injection efforts. Therefore, first, a set of per-instance features is described and extracted through an analysis approach combining static elements (cell properties, circuit structure, synthesis attributes) and dynamic elements (signal activity). Second, reference data is obtained through first-principles fault simulation approaches. Finally, one part of the reference dataset is used to train the Machine Learning algorithm and the remaining is used to validate and benchmark the accuracy of the trained tool. The intended goal is to obtain a trained model able to provide accurate per-instance Functional De-Rating data for the full list of circuit instances, an objective that is difficult to reach using classical methods. The presented methodology is accompanied by a practical example to determine the performance of various Machine Learning models for different training sizes.**

***Index Terms*—Transient Faults, Single-Event Effects, Fault Injection, Gate-Level Netlist, Machine Learning, Linear Least Squares, k-Nearest Neighbors, Support Vector Regression**

I. Introduction

Today's reliability standards and customers' expectations set tough targets for the quality of electronic devices and systems. Among other reliability threats, transient faults, such as Single-Event Upsets in sequential/state logic and Single-Event Transients in combinatorial logic, are known to contribute significantly to the overall failure rate of the system. Therefore, estimating the Soft-Error Rate of modern complex circuits is a challenging and important task.

Circuits' susceptibility to transient faults/single events is caused by faults occurring in the circuit's cells and their subsequent propagation in the system, possibly causing observable effects (failures) at the system level. The impact of Single-Event Upsets and Single-Event Transients in individual state and combinatorial cells has been extensively studied [1], [2] and for many applications, identified as the leading contributor to the overall failure rate exhibited by the circuit.

This work was supported by the RESCUE project which has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 722325.

A. Objective of Our Methodology

De-Rating or Vulnerability Factors are a major tool used in todays Functional Safety analysis. Since it is difficult and computationally intensive to get accurate per-instance Functional De-Rating data for the full list of circuit instances by using classical methods, we propose an approach using Machine Learning algorithms to assist this procedure. Previous works have shown that the masking effects and thus the vulnerability factors can be related to certain characteristics of the circuit, such as circuit structure and signal probability [3]–[5]. Thus, we assume that machine learning models are able to learn and predict the Functional De-Rating by using such characteristics. Therefore, an analysis flow is presented which uses Machine Learning models to predict the Functional De-Rating factors of individual flip-flops. A set of features is described to characterise each flip-flop instance in the circuit individually. The flip-flop features are used to train the Machine Learning model in a supervised learning approach. The trained model is able to predict the remaining Functional De-Rating values for the flip-flop instances not used for training. The proposed methodology is validated in a practical example and compared against a full flat statistical fault injection campaign.

B. Organisation of the Paper

The rest of this paper is organised as follows: Section II summarises the definition of Single-Event Effects and the different de-rating mechanism. Further, regression in context of supervised Machine learning is explained. The proposed methodology and the used feature set is described in section III. In section IV the proposed method is validate on a practical example by using different Machine Learning models which are compared to each other. Section V summarises this paper and gives concluding remarks as well as prospects for future work.

II. Background

A. De-Rating Mechanism

Erroneous data in one of the memory or logic points of a circuit can be produced by the propagation of a Single-Event Transient (SET) or Single-Event Upset (SEU). SETs are the result of the collection of charge deposited by ionising particles on combinatorial logic cells. SEUs are the change of the logic state of a discrete sequential element, such as a latch,

978-1-7281-3028-6/19/$31.00 ©2019 European Union
DOI 10.1109/DSN-S.2019.00021

a flip flop or a memory cell. In the data path between flip-flops, four de-rating mechanisms [6], [7] significantly reduce the impact of SETs and SEUs on the effective error rate.

Electrical De-Rating (EDR): The transient is filtered due to pulse narrowing and or an increase of the rise and fall time during its propagation. By the time it reaches the end of the path, either it has been completely filtered or the voltage transition is below the switching threshold.

Temporal De-Rating (TDR): The erroneous state reaches the input of a flip-flop but outside the latching window, thus it is not sampled.

Logical De-Rating (LDR): The erroneous state is prevented from propagating due to the state on another controlling input of a gate such as a zero value on an `AND2` gate.

Functional De-Rating (FDR): The erroneous state is considered at an applicative level. This means even when an SEU/SET does propagate (e.g. is not logically or temporally masked), the impact at the function of the circuit can vary, and in many cases is benign. Thus, considering the faults at an applicative level, the de-rating depends on the criteria defining the acceptable behaviour of the circuit during the execution of an application and the fault classifications (correctable, uncorrectable, not detected by the hardware but detected by the software, if a retry is possible, if there is a time limit to receive the correct result, etc.)

These de-rating mechanisms are used to evaluate the probability of the propagation of a fault and are usually estimated by using probabilistic algorithms and simulation based approaches. Thereby, especially the simulation based approaches are very computationally intensive.

B. Supervised Regression with Machine Learning

Machine Learning (ML) is the concept of a machine learning from examples and making predictions based on its experience, without being explicitly programmed [8]. ML algorithms are usually build upon a mathematical model which uses sample data (also called training data) in order to make predictions or decisions. The machine learning process usually consists of two phases, namely the training or learning phase and the prediction phase. The learning phase can be further grouped in 1) supervised learning and 2) unsupervised learning.

Supervised learning algorithms try to model the relationship and dependency between the input features and the target output in such a way that the output values for new data points can be predicted based on the learned relationships. The main tasks of supervised learning models are classification and regression. While classification algorithms are used when the outputs are restricted to a limited set of values, regression algorithms are used when the outputs may have any numerical value within a range.

In contrast to supervised learning, the unsupervised learning models try to find structures in the data set without external labelling or classification. The two main tasks in this type of machine learning methods are clustering and dimensionality reduction. Clustering algorithms are organizing the given data into groups by similarity and dimensionality reduction is compressing the data by reducing redundancy, while maintaining the overall structure.

The objective of this work is to predict continuous Functional De-Rating factors for individual flip-flops with the help of Machine Learning models. Thus, the proposed methodology is based on supervised regression and is presented in the following section.

III. Methodology

This section presents the proposed methodology to estimate Functional De-Rating factors per flip-flop instance by using Machine Learning regression models. Therefore, the implemented approach is described in detail, including the feature set to characterise each flip-flop instance and the evaluation metrics used to measure the performance of the models.

A. Functional De-Rating Estimation Flow

The implemented procedure to estimate the Functional De-Rating factors is shown in Fig. 1. It is based on the gate-level netlist of the circuit and a corresponding testbench, which are used to extract the features for each flip-flop in the circuit (the set of flip-flop features is described in section III-B). Further, they are used to determine the FDR factors for one part of the circuit by using statistical fault injection. The determined FDR factors per flip-flop and the associated flip-flop features form the training data set, used to train the ML model. The size of the training data set is defined by the training size and thus, also defines the number of fault injections to perform.

All ML models have internal model parameters, which are determined during the training process by the ML algorithm. Additionally, most of the ML models also have hyperparameters, which, in contrast to the internal parameters, are manually set before the training process and are not derived by the training algorithm itself. Therefore, several instances of the model need to be trained and evaluated for different hyperparameters (the used evaluation metrics are described in section III-C). A common method to determine the best hyperparameters is to first evaluate the model with randomly selected values for these parameters in a given distribution (random search). Afterwards a more detailed grid search is performed within the region of the values obtained by the random search [9]. The in this way obtained trained model can be used to estimate the FDR values of the remaining flip-flops.

In this paper we further intend to validate and measure the performance of the proposed approach against a full statistical fault injection campaign (see section IV). In order to ensure that the model is not only trained for one particular training and test data set we use the cross-validation technique. Thereby, the model is trained and evaluated against multiple train and test splits of the data. Several subsets, or cross validation folds, of the data set are created and each of the folds is used to train and evaluate a separate model. Thus, we

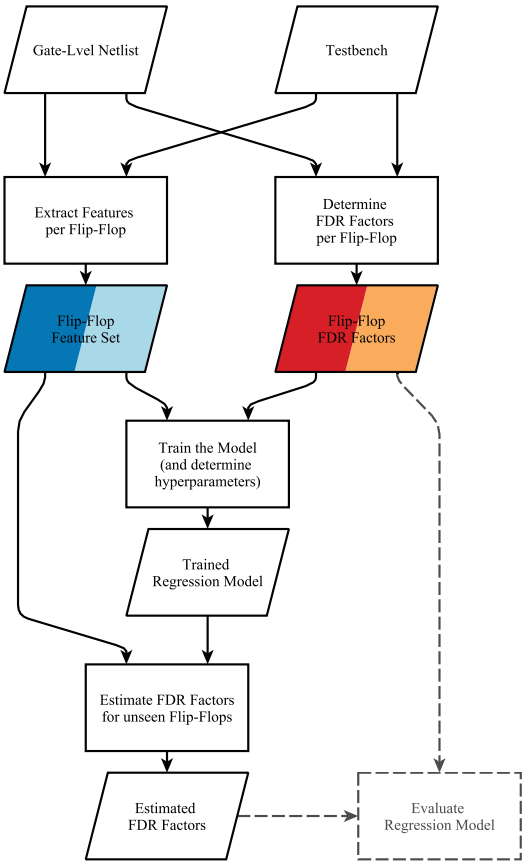


Fig. 1. Procedure to Estimate and Evaluate the Functional De-Rating

are obtaining a more stable measure of how the model is likely to perform on average, instead of relying only on one single training and test data set [10]. In section IV our methodology is evaluated on a practical example and therefore, a ten fold stratified cross validation is used.

B. Flip-Flop Feature Set

The proposed feature set to characterise each flip-flop instance, combines static elements, such as cell properties, circuit structure and synthesis attributes, as well as dynamic elements, such as signal activity. In order to extract the features describing the circuit structure the the gate-level netlist was converted into a graph representation. Thus, graph algorithms, such as Dijkstra's algorithm to find the shortest path, could be used to extract the features. For each flip-flop FF_i the following structural features have been extracted.

Flip-Flop Fan-In This parameter describes how many flip-flops are directly connected (only through combinatorial logic) to the input of the target flip-flop FF_i.

Flip-Flop Fan-Out This parameter describes to how many flip-flops the target flip-flop's FF_i output is directly connected (only through combinatorial logic).

Total Flip-Flops from FF_i This parameter refers to the number of flip-flops which are influencing the input of the target flip-flop FF_i. It represents the total number of flip-flops which are connected to the target flip-flop within the full circuit.

Total Flip-Flops to FF_i This parameter refers to the number of flip-flops which are influenced by the target flip-flop's FF_i output. It represents the total number of flip-flops which are connected to the output of the target flip-flop within the full circuit.

Connections from Primary Input This parameter describes how many primary inputs are connected to the target flip-flop's FF_i input.

Connections to Primary Output This parameter describes to how many primary outputs the target flip-flop FF_i is connected.

Proximity from Primary Input This parameter refers to the proximity of the primary input to the target flip-flop FF_i. It represents the number of stages from the connected primary inputs to the target flip-flop. Thereby, the maximum, average and minimum number of stages are considered.

Proximity to Primary Output This parameter refers to the proximity of the target flip-flop's FF_i output to the primary output. It represents the number of stages from the target flip-flop to the primary outputs. Thereby, the maximum, average and minimum number of stages are considered.

Part of Bus This parameter describes if the target flip-flop FF_i is part of a bus or not.

Bus Position If the target flip-flop FF_i is part of a bus, then this parameter represents the position within the bus. It is set to -1 if the flip-flop is not part of a bus.

Bus Length If the target flip-flop FF_i is part of a bus, this parameter represents the length of the bus. It is set to 0 if the flip-flop is not part of a bus.

Connections to constant drivers This parameter refers to the number of connected constant drivers to the target flip-flop FF_i. It represents how many constant drivers are connected to the flip-flop's input within the circuit.

Has Feedback Loop This parameter refers to the situation in which the output signal of the target flip-flop FF_i is passed to its input, directly or through several flip-flop stages.

Depth of Feedback Loop If the target flip-flop FF_i has a feedback loop, directly or through several flip-flop stages, this parameter describes the minimum number of stages. It is set to -1 if there is no feedback loop.

Further features were extracted which are related to the synthesis of the circuit and were obtained by using Synopsys Design Compiler. The following synthesis related features have been extracted for each flip-flop FF_i.

Flip-Flop Drive Strength This parameter describes the drive strength of the target flip-flop FF_i selected by the synthesis tool.

Combinatorial Fan-In This parameter describes the number of combinatorial elements connected to the target flip-flop's FF_i input up to the previous flip-flop stage.

Combinatorial Fan-Out This parameter describes the number of combinatorial elements which are driven by the target flip-flop's FF_i output up to the next flip-flop stage.

Combinatorial Path Depth This parameter describes the depth of the combinatorial stage at the target flip-flop's FF_i output.

To consider the workload of the circuit, features are required which describe the dynamic behaviour of the flip-flops. Therefore, the signal activity for each flip-flop is extracted. These are obtained by simulating the gate-level netlist with the corresponding testbench and tracing the signal changes at the output of the flip-flops. For each flip-flop FF_i the following dynamic features have been extracted.

@0 This parameter refers to the time the output of the target flip-flop FF_i is at logical `0`. It represents the time ratio the flip-flop's output has been at `0` in relation to the total testbench run time.

@1 This parameter refers to the time the output of the target flip-flop FF_i is at logical `1`. It represents the time ratio the flip-flop's output has been at `1` in relation to the total testbench run time.

State Changes This parameter refers to the number of changes the target flip-flop's FF_i output has performed. It represents the number of changes from `0` to `1` and vice versa.

C. Regression Model Evaluation Metrics

The performance of the Machine Learning model is evaluated by using several metrics. In the following description of these metrics, $\hat{y}_i$ is the value of the i-th sample predicted by the model and y_i is the corresponding true/expected value.

Mean Absolute Error The mean absolute error (MAE) describes the average absolute difference of the expected values to the predicted values. It is calculated over n_{samples} by the following equation (values closer to zero are better)

$$\text{MAE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=1}^{n_{\text{samples}}} |y_i - \hat{y}_i| \tag{1}$$

Maximum Absolute Error The maximum absolute error (MAX) describes the maximum difference of the expected values to the predicted values. The equation

$$\text{MAX}(y, \hat{y}) = \max_{i \in [1, n_{\text{samples}}]} |y_i - \hat{y}_i| \tag{2}$$

calculates the metrics (values closer to zero are better).

Root Mean Squared Error The root-mean-square error (RMSE) describes the square root of the quadratic error of the expected values. In comparison to the mean absolute error the root-mean-square error gives a higher weight to larger errors. It is calculated over n_{samples} by the following equation (values closer to zero are better)

$$\text{RMSE}(y, \hat{y}) = \sqrt{\frac{1}{n_{\text{samples}}} \sum_{i=1}^{n_{\text{samples}}} (y_i - \hat{y}_i)^2} \tag{3}$$

Explained Variance The Explained Variance (EV) measures the proportion to which a model accounts for the variation (dispersion) of a given data set. If $\text{Var}(X)$ is the variance, the square of the standard deviation, of a random variable X then the explained variance is calculated as follows

$$\text{EV}(y, \hat{y}) = 1 - \frac{\text{Var}(y - \hat{y})}{\text{Var}(y)} \tag{4}$$

The best possible values is 1 and lower values are worse.

Coefficient of Determination The coefficient of determination (R^2) provides a measure of how well future samples are likely to be predicted by the model. If $\bar{y}$ is the mean of the expected values, the coefficient of determination can be calculated by

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n_{\text{samples}}} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n_{\text{samples}}} (y_i - \bar{y})^2} \tag{5}$$

and the best possible value is 1 (lower values are worse).

IV. Estimating Functional De-Rating Factors by Using Machine Learning

In this section the presented methodology is evaluated on a practical example. Therefore the Ethernet 10GE MAC Core from OpenCores is used. This circuit implements the Media Access Control (MAC) functions for 10 Gbps operation as defined in the IEEE 802.3ae standard. The 10GE MAC core has a 10 Gbps interface (XGMII TX/RX) to connect it to different types of Ethernet PHYs and one packet interface to transmit and receive packets to/from the user logic [11]. The circuit consists of control logic, state machines, FIFOs and memory interfaces. It is implemented at the Register-Transfer Level (RTL) and is publicly available on OpenCores.

The corresponding testbench writes several packets to the 10GE MAC transmit packet interface. As packet frames become available in the transmit FIFO, the MAC calculates a CRC and sends them out to the XGMII transmitter. The XGMII TX interface is looped-back to the XGMII RX interface in the testbench. The frames are thus processed by the MAC receive engine and stored in the receive FIFO. Eventually, the testbench reads frames from the packet receive interface and prints out the results [11]. During the simulation all sent and received packages to and from the core are monitored and recorded. This record is used as the golden reference for the fault injection campaign.

By synthesising the design using the NanGate FreePDK45 Open Cell Library [12], the gate-level netlist was obtained and 1054 flip-flops have been identified. First, a full flat statistical fault injection campaign was performed to get the Functional De-Rating factors for each flip-flop. Further, the respective features for each flip-flop have been extracted. These values

are used to train and evaluate different regression models as described in the previous section.

A. Flat Statistical Fault Injection Campaign

In order to provide an objective measure of the sensitivity of each flip-flop, a flat statistical fault injection campaign was performed on the gate-level netlist. The fault injection mechanism is implemented by inverting the value stored in a flip-flop using a simulator function. The faults are injected at different times during the active phase of the simulation, when packets are sent and received through the user packet interface.

For each of the 1054 flip-flops 170 fault injection simulations were performed. The simulation run was considered as a functional failure when the final received packages contained payload corruption or the circuit stopped sending or receiving data. Eventually, the Functional De-Rating factor was calculated by the number of simulation runs with a functional failure divided by the number of total simulation runs.

B. FDR Estimation by Using Different Regression Models

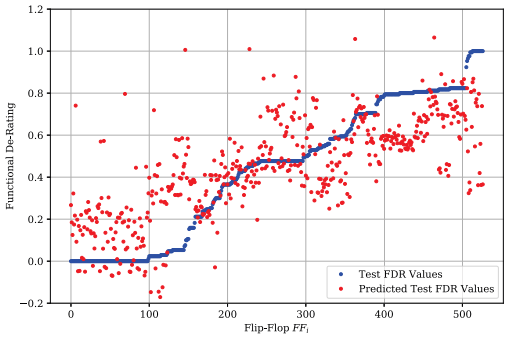
The results of the full statistical fault injection campaign and the extracted flip-flop features are forming the training and test data set, which are used to train and evaluate different Machine Learning models. All models are implemented using Python's scikit-learn Machine Learning framework [13], with a cross validation fold of 10 and a training size of 50 %. Further, the learning curve was determined, which describes the performance of the model for different training sizes.

1) Linear Least Squares Regressor: The Linear Least Squares algorithm fits a linear model which expects the target value to be a linear combination of the input variables. Thereby, the algorithm aims to minimise the residual sum of squares between the observed responses in the training dataset y, and the responses predicted by the linear approximation $\hat{y}$.

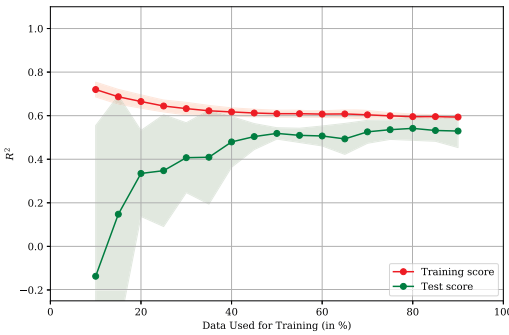
The performance of the Linear Least Squares model is given in Table I. A regression by using the trained model on an example test data fold is shown in Fig. 2a and Fig. 2b shows the learning curve of the model.

2) k-Nearest Neighbors Regressor: The principle behind the k-Nearest Neighbor (k-NN) regressor is to use feature similarity to predict values of new data points. The new point to predict is assigned a value based on how closely it resembles to the points in the training set. A weighted average of the k nearest neighbors is used to predict the value, where the weight is calculated by the inverse of the distances and the distance itself can be any metric measure, such as the Manhattan or Euclidean distance. Hence, the model defines k and the distance metrics as hyperparameters.

The best values for the hyperparameters k and the distance metrics, found during the training phase by using random and grid search, are $k = 3$ and the Manhattan distance. The resulting performance of the k-NN model is listed in Table I. A regression with the trained model on the example test data fold is shown in Fig. 3a and Fig. 3b shows the learning curve for the k-NN model.



(a) Estimation of the example test data fold (training size = 50%)



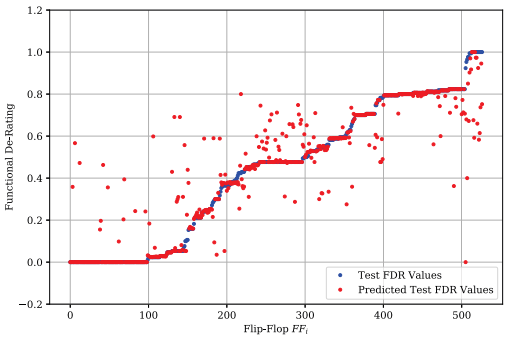
(b) Learning Curve (cross validation fold = 10)

Fig. 2. Regression with the Linear Least Squares model

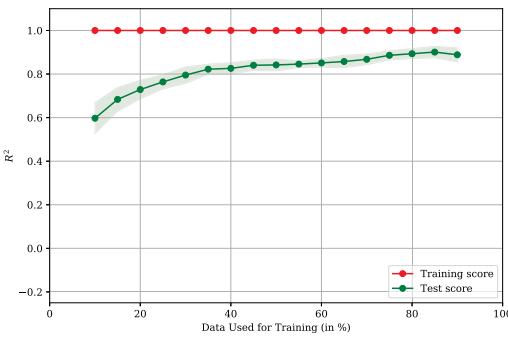
3) Support Vector Regression with RBF Kernel: The goal of the Support Vector Regression (SVR) is to find a function that deviates from the target value by a value not greater than ε for each training point, and at the same time is as flat as possible. The SVR can be extended to use nonlinear kernel functions, which perform a transformation of the input values and map them to a higher dimensional space. This is useful for regression problems which cannot adequately be described by linear models. In this paper the Radial Basis Function (RBF) was used as kernel function. The model defines several hyperparameters, such as the penalty factor C, the size of the ε-tube, and γ to control the RBF kernel function.

The performance of the Support Vector regression is given in Table I. The best hyperparameters of the model found during the training by using random and grid search are $C = 3.5$, $\gamma = 0.055$ and $\varepsilon = 0.0.025$. A regression with the trained model on the example test data fold is shown in Fig. 4a and the learning curve of the model is shown in Fig. 4b.

4) Comparison: By comparing the performance of the different models, shown in Table I, it can be seen that the Linear Least Squares model is rated the worst. The other models are performing much better which suggests that the extracted features are not linear dependent to the Functional

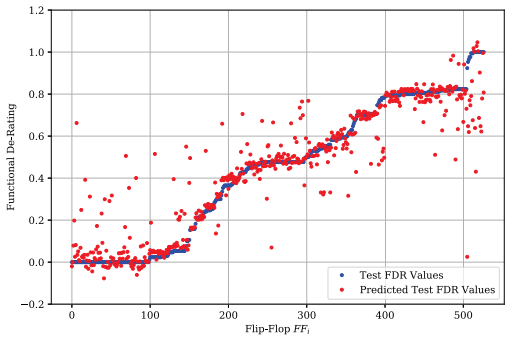


(a) Estimation of the example test data set (training size = 50%)

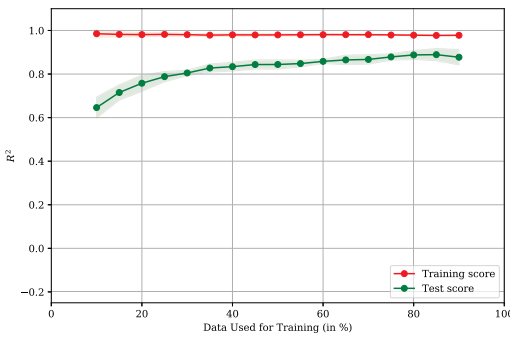


(b) Learning Curve (cross validation fold = 10)

Fig. 3. Regression with the k-Nearest Neighbors model



(a) Estimation of the example test data set (traning size = 50 %)



(b) Learning Curve (cross validation fold = 10)

Fig. 4. Regression with the Support Vector Regressor with RBF Kernel

De-Rating factor and the problem can not be solved with linear models. Further, all models show that the performance does not improve significantly with training sizes higher than 50 %. This means, by using the proposed method, the cost for a fault injection campaign can be reduced by half. Further, the learning curves show a more aggressive optimisation can be achieved (a cost reduction up-to $5\times$) in exchange of a slight reduction in accuracy ($< 10\,\%$).

TABLE I
PERFORMANCE RESULTS FOR DIFFERENT REGRESSION MODELS
(CROSS VALIDATION = 10, TRAINING SIZE = 50 %)

Model	MAE	MAX	RMSE	EV	R^2
Linear Least Sqares	0.165	0.944	0.218	0.520	0.519
k-NN	0.050	0.907	0.124	0.843	0.842
SVR w/ RBF Kernel	0.063	0.849	0.124	0.845	0.844

V. CONCLUSION AND FUTURE WORK

In this paper we proposed a new methodology to assist the Functional Failure analysis by using Machine Learning models. The methodology helps to reduce the cost of computing the Functional De-Rating factors of sequential logic of a circuit. Specifically, the methodology allows the computation of factors per individual instances, which is particularly difficult to obtain using state-of-the-art approaches such as clustering, selective fault simulation or fault universe compaction techniques. Therefore, we propose a feature set to describe the individual flip-flops and an estimation flow to train and evaluate the Machine Learning model.

The methodology was evaluated in a practical example. The comparison of the performance of different models has shown that the linear model is not able to fit the problem. Further, the practical example has shown that training sizes of 20% to 50% provides appropriate performance, which means that the cost for a classical statistical fault injection campaign could be reduced by 2 up to 5 times.

The focus for future work should lie on evaluating further non-linear models, such as Decision Tree Regressor, Multi-Layer Perception Neural Networks, or using boosting algorithms. Additionally, further features should be considered to improve the overall performance of the models. However, also a dimension reduction should be taken into account in order to avoid the curse of dimensionality and the value of each feature needs to be evaluated separately [14].

REFERENCES

[1] R. C. Baumann, "Radiation-Induced Soft Errors in Advanced Semiconductor Technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, Sep. 2005.

[2] N. Seifert, P. Slankard, M. Kirsch, B. Narasimham, V. Zia, C. Brookreson, A. Vo, S. Mitra, B. Gill, and J. Maiz, "Radiation-Induced Soft Error Rates of Advanced CMOS Bulk Devices," in *2006 IEEE International Reliability Physics Symposium Proceedings*, Mar. 2006, pp. 217–225.

[3] I. Wali, B. Deveautour, A. Virazel, A. Bosio, P. Girard, and M. Sonza Reorda, "A Low-Cost Reliability vs. Cost Trade-Off Methodology to Selectively Harden Logic Circuits," *Journal of Electronic Testing*, vol. 33, no. 1, pp. 25–36, Feb. 2017.

[4] O. Ruano, J. A. Maestro, and P. Reviriego, "A Methodology for Automatic Insertion of Selective TMR in Digital Circuits Affected by SEUs," *IEEE Transactions on Nuclear Science*, vol. 56, no. 4, pp. 2091–2102, Aug. 2009.

[5] P. K. Samudrala, J. Ramos, and S. Katkoori, "Selective Triple Modular Redundancy (STMR) Based Single-Event Upset (SEU) Tolerant Synthesis for FPGAs," *IEEE Transactions on Nuclear Science*, vol. 51, no. 5, pp. 2957–2969, Oct. 2004.

[6] H. T. Nguyen and Y. Yagil, "A Systematic Approach to SER Estimation and Solutions," in *2003 IEEE International Reliability Physics Symposium Proceedings, 2003. 41st Annual.*, Mar. 2003, pp. 60–70.

[7] D. Alexandrescu and E. Costenaro, "Towards Optimized Functional Evaluation of SEE-Induced Failures in Complex Designs," in *2012 IEEE 18th International On-Line Testing Symposium (IOLTS)*, Jun. 2012, pp. 182–187.

[8] E. Alpaydin and F. Bach, *Introduction to Machine Learning*, ser. Adaptive Computation and Machine Learning Series. MIT Press, 2014.

[9] J. Bergstra and Y. Bengio, "Random Search for Hyper-parameter Optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.

[10] R. Kohavi, "A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 1137–1143.

[11] Andre Tanguay, "10GE MAC Core Specification," Jan. 2013.

[12] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal, "FreePDK: An Open-Source Variation-Aware Design Kit," in *2007 IEEE International Conference on Microelectronic Systems Education (MSE'07)*, Jun. 2007, pp. 173–174.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[14] G. V. Trunk, "A Problem of Dimensionality: A Simple Example," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 3, pp. 306–307, Jul. 1979.

Appendix 6

VI

X. Lai, A. Balakrishnan, T. Lange, M. Jenihhin, T. Ghasempouri, J. Raik, and D. Alexandrescu, “Understanding multidimensional verification: Where functional meets non-functional,” *Microprocessors and Microsystems*, vol. 71, p. 102867, nov 2019

Microprocessors and Microsystems 71 (2019) 102867

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Understanding multidimensional verification: Where functional meets non-functional

Xinhui Lai[a,*], Aneesh Balakrishnan[a,b], Thomas Lange[b,c], Maksim Jenihhin[a], Tara Ghasempouri[a], Jaan Raik[a], Dan Alexandrescu[b]

[a] Department of Computer Systems, Tallinn University of Technology, Akadeemia 15A, Tallinn 12618, Estonia
[b] IROC Technologies, 2 Square Roger Genin, 5th floor, Grenoble, 38000, France
[c] Dipartimento di Informatica e Automatica, Politecnico di Torino, Turin, Italy

ARTICLE INFO

Article history:
Received 25 February 2019
Revised 28 June 2019
Accepted 5 August 2019
Available online 5 August 2019

Keywords:
Extra-functional verification
Functional verification
Survey
Taxonomy
Security verification
Reliability verification
Power verification
Machine learning

ABSTRACT

Advancements in electronic systems' design have a notable impact on design verification technologies. The recent paradigms of Internet-of-Things (IoT) and Cyber-Physical Systems (CPS) assume devices immersed in physical environments, significantly constrained in resources and expected to provide levels of security, privacy, reliability, performance and low-power features. In recent years, numerous extra-functional aspects of electronic systems were brought to the front and imply verification of hardware design models in multidimensional space along with the functional concerns of the target system. However, different from the software domain such a holistic approach remains underdeveloped. The contributions of this paper are a taxonomy for multidimensional hardware verification aspects, a state-of-the-art survey of related research works and trends enabling the multidimensional verification concept. Further, an initial approach to perform multidimensional verification based on machine learning techniques is evaluated. The importance and challenge of performing multidimensional verification is illustrated by an example case study.

1. Introduction

Recently, several prominent trends in electronic systems design can be observed. Safety-critical applications in the automotive domain set stringent requirements for electronics certification, the Internet-of-Things (IoT) and Cyber-Physical Systems (CPS) devices are immersed in physical environments, significantly constrained in resources and expected to provide levels of security and privacy [1], ultra-low power feature or high performance. Very complex electronic systems, including those built from the non-certified for reliability commercial-off-the-shelf components, are used for safety- and business-critical applications. These trends along with gigascale integration at nanoscale technology nodes and multi-/many-processor based systems-on-chip architectures have ultimately brought to the front various *extra-functional aspects* of the electronic systems' design at the chip design level. The latter include security, reliability, timing, power consumption, etc. There exist numerous threats causing an electronic system to violate its specification. In the hardware part, these are design errors (bugs), manufacturing defects and variations, reliability issues, such as soft errors and aging faults, or malicious faults, such as security attacks. Withal, there can also be bugs in the software part.

Hardware design model verification detects *design errors* affecting *functional* and *extra-functional* (interchangeably referred as *non-functional*) aspects of the target electronic system. Strictly, the sole *task of extra-functional verification* of a design model is limited to detecting deviations that cause violation of extra-functional requirements. In practice, it often intersects with the task of functional verification [2,14], thus establishing *a multidimensional space for verification*. A "grey area" in distinction between functional and extra-functional requirements may appear when an extra-functional requirement is a part of design's main functionality. E.g., security requirements for some HW design can be split into extra-functional and functional sets if the design's purpose and specified functionality is a system's security aspect, e.g. it is a secure cryptoprocessor.

The contributions of this paper are a taxonomy for multidimensional hardware verification aspects, a state-of-the-art survey of related research works towards enabling the multidimensional verification concept. Further, an approach is evaluated which performs multidimensional verification by using machine learn-

* Corresponding author.
E-mail address: xinhui.lai@taltech.ee (X. Lai).

https://doi.org/10.1016/j.micpro.2019.102867

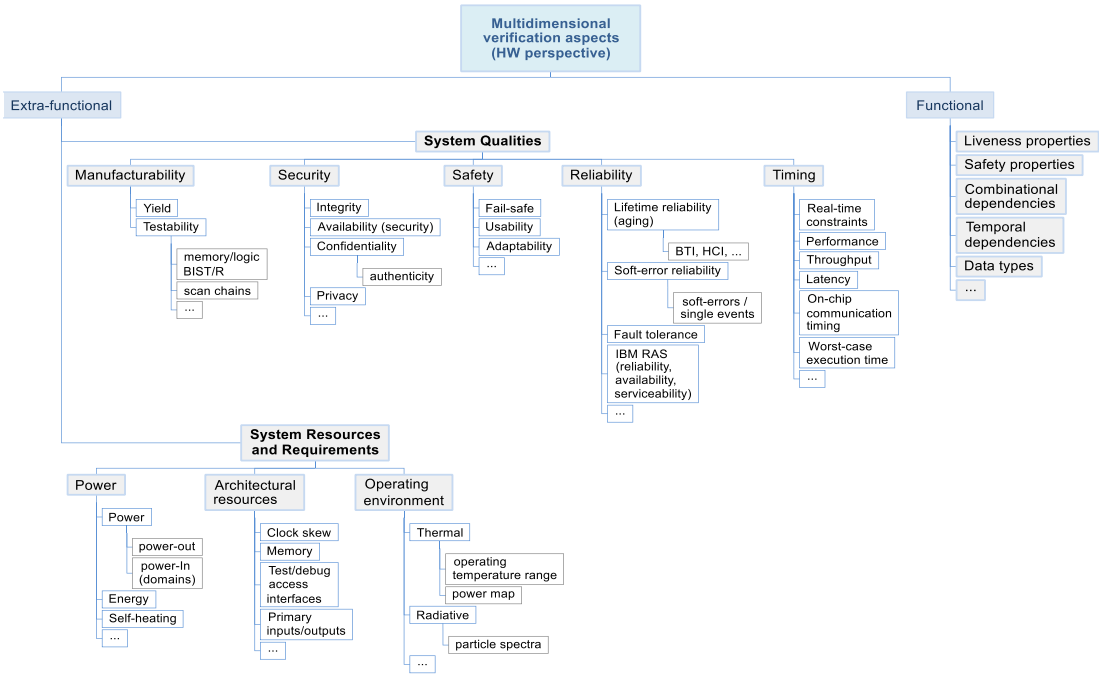


Fig. 1. Taxonomy of multidimensional verification aspects.

ing techniques. The rest of this paper is organized as follows. Section 2 provides a taxonomy of multidimensional verification aspects. Sections 3 proposes a state-of-the-art survey with the key trends in verification for the main extra-functional aspects. Section 4 discusses the multidimensional verification challenges and presents a motivational example for the functional and power verification dimensions. Section 5 proposes adoption of machine learning techniques for support of design's multi-aspect features extraction and verification. Finally, Section 6 draws the conclusions.

2. Taxonomy of multidimensional verification aspects

In practice, relevance of each functional and extra-functional aspect strongly depends on the design type, target system application and specific user requirements. Following the design paradigm shifts, a number of extra-functional aspects have recently received significant academic research attention e.g., security. At the same time, there already exist established industrial practices for measuring and maintaining particular design qualities, e.g. the RAS (Reliability-Availability-Serviceability) aspect introduced by IBM [6]. While in the software engineering discipline, the taxonomy of extra-functional requirements has a comprehensive coverage by the literature [7–12], it cannot be directly re-used for the HW verification discipline because of significant difference in the design models.

Fig 1 introduces a taxonomy of multidimensional verification aspects derived from the performed literature review. The conventional *functional concerns* are *safety and liveness properties, combinational and temporal dependencies* along with *data types*, however this list can be extended for particular designs. The extra-functional aspects can be strictly categorized into two groups: *System Qualities* and *System Resources and Requirements* (in bold). The main system qualities for extra-functional verification are manufacturability of the design, security, in-field safety, reliability during the operational lifespan and a set of timing aspects. The second group embraces the power and architectural resources as well as design constraints set by the operational environment.

Several extra-functional aspects such as *manufacturability*, i.e. primarily *yield* and *testability* against manufacturing defects, *fault-tolerance,* reliability (subject to transient, intermittent and permanent hardware faults) and several aspects from the *System Resources and Requirements* group do not have a direct correspondence in the software engineering discipline because of the distinct nature of faults and specification violations. Other aspects such as *real-time constraints* are very similar between the two domains.

3. Trends in extra-functional verification

Table 1 presents a survey of recent publications targeting extra-functional and multidimensional verification. Here, along with the specific extra-functional aspects details about the design model and verification approach are outlined, i.e., the design under verification type, verification engine, the level of abstraction, design representation language, compute model and the tool operated in the research. We pointed out such key points for all the recent up to 10-year old studies in this area. Further, in the following subsections, we focus on understanding trends for the extra-functional aspects that have the strongest attention in the literature, i.e. security, in-field reliability, timing and power.

3.1. Security aspects

Security is difficult to quantify as today there are no commonly agreed metrics for this purpose [1]. The key targeted security services [16] commonly represented as extra-functional aspects for verification are *confidentiality, integrity* and *availability.* Verifying

Table 1
Survey of the state-of-the-art solutions for extra-functional and multidimensional verification.

Pub.	Year[a]	Extra-functional aspect[b]					Design under verification	Verification engine	Abst. level[e]	Design representation language	Compute model	Tool
		Security	Reliability[c]	Timing[d]	Power	Other aspects						
[19]	2009	confidentiality, integrity	–	–	–	–	HW/SW system	formal, correct-by-construction	SL	AADL	–	OSATE
[20]	2018	confidentiality	–	–	–	–	NoC	unbounded model-checking	RTL	VHDL/Verilog, PSL	LTL	–
[21]	2016	integrity, confidentiality	○	–	–	–	NoC	simulation, HW monitors	RTL	VHDL/Verilog	–	–
[22]	2014	integrity	○	–	–	–	NoC	formal	GL	VHDL/Verilog	–	SurfNoC
[23]	2017	integrity, confidentiality	–	–	–	–	RSN	model check	RTL	ICL	Craig interpolation	CIP solver
[24]	2015	integrity	–	–	–	–	SoC	simulation	RTL	VHDL/Verilog	–	–
[25]	2016	integrity, confidentiality	–	–	–	–	ALU	equivalence check	GL	–	QBF-SAT	–
[26]	2017	integrity	–	–	–	–	SoC	semiformal	GL	–	–	JasperGold SPV
[27]	2016	confidentiality	–	–	–	–	RSN	model check	RTL	ICL	Craig interpolation	CIP Solver
[28]	2017	confidentiality	–	–	–	–	control systems	formal	SL	ASLan++	–	CL-AtSe
[29]	2017	integrity	–	–	–	–	IP cores	semiformal	GL	VHDL	–	mini-SAT
[30]	2015	integrity	–	–	–	–	ISA, pipeline	model check	RTL	–	CTL, LTL	nuXmv SMV
[31]	2018	confidentiality	–	–	–	–	cache	model check	SL	–	CTL	–
[32]	2014	confidentiality	–	–	–	–	cache	model check	RTL/SL	–	FSM	Murphi
[33]	2017	confidentiality	–	–	–	–	cache	model check	RTL/SL	–	FSM	CacheAudit
[34,35]	2013	integrity, confidentiality	–	–	–	–	IPs and SoCs	formal	RTL, GL	Verilog	–	JasperGold SPV
[36]	2018	•	•	–	–	–	MPSoC	model check	SL, RTL	–	Timed Automata	UPPAAL
[41]	2017	–	•	–	–	–	CPS	model check	SL	AADL	Timed Automata	UPPAAL
[42]	2015	–	SER	–	–	–	IP cores	formal	GL/RTL	LDDL	LDDL	Coq
[43]	2010	–	SER	–	–	availability, serviceability	processor	fault inject.	GL	Verilog	–	IBM in-house
[44]	2016	–	•	–	–	availability, serviceability	SoC	fault inject.	RTL	–	–	–
[45,46]	2016	–	○ (LTR)	–	○	thermal	Smart Systems	simulation	SL	IP-XACT, SystemC-AMS	–	–
[47]	2018	–	•	–	–	–	CPS	formal /simulation, HW monitors	RTL	VHDL/Verilog	multiple	multiple

(*continued on next page*)

Table 1 (*continued*)

Pub.	Year[a]	Extra-functional aspect[b]					Design under verification	Verification engine	Abst. level[e]	Design representation language	Compute model	Tool
		Security	Reliability[c]	Timing[d]	Power	Other aspects						
[48]	2014	–	SER	–	–	–	IPs	SAT solver	RTL	VHDL	–	–
[49]	2010	–	SER	–	–	–	IPs, processor	simulation	RTL	VHDL/Verilog	–	–
[50]	2014	–	SER	–	–	–	memory	circuit-level simulation	circuit level	–	–	INFORMER
[51,52]	2018	–	–	comm. constraint	○	–	NoC	fault inject.	RTL	VHDL	–	QoSinNoC
[53]	2011	–	–	RT	–	–	memory	model check	RTL	REAL/AADL	–	Ocarina
[54]	2010	–	–	RT	–	–	Scheduler of RT system	model check	–	Promela	Time Petri-net	SPIN
[55]	2010	–	–	latency	–	–	RT emb. system	model check	SL	AADL	–	YICES
[56]	2017	–	–	performance	○	–	NoC, HW/SW architectures	simulation	SL	Graph Assembly Language	connectivity graphs	ArchOn
[58]	2012				●	–	IPs	simulation	SL	SystemC	–	–
[59]	2016				●	–	DSP cores	simulation	SL,GL, RTL	SystemC	–	Powersim
[62]	2017	–	–	RT	○	–	automotive CPS	model check	SL	C, EAST-ADL	Timed Automata	UPPAALsdv
[63]	2016	–	–	–	●	–	IPs	Semiformal ABV	RTL	VHDL/Verilog; SystemC	Hidden Markov Model	–
[64]	2012	–	–	execution time	○	–	distributed emb. system	simulation	SL	SystemC	–	–
[65]	2016	–	–	performance RT	○	thermal	HW/SW platform	semiformal	RTL,TLM,SL	UML,C++,VHDL SystemC-AMS	HIF	HIFSuite
[66]	2009	–	–	throughput	–	–	SoC/FPGA	simulation	RTL	Verilog/VHDL	–	Modelsim
[67]	2018	–	–	throughput	–	–	NoC	simulation	RTL	System Verilog	–	UVM
[68]	2014	–	–	–	–	connectivity	SoC	symbolic model checking	RTL, TLM	Verilog	–	Incisive Formal Verifier
[70]	2016	–	–	–	–	memory consistency	processor	simulation	ISA	ruby	–	McVerSi
[73]	2011	–	–	–	●	thermal	SoC	simulation	SL,GL, RTL	SystemC	–	Power-Mixer, -Depot, -Brick
[74]	2015	–	–	–	●	–		simulation	SL,TLM	SytemC	–	Power Kernel Tool
[75]	2011	–	–	–	●	–	SoC	simulation	SL	SystemC	–	Powersim

[a] *only conference, journal and industrial white papers published in the last 10 years were selected for this survey.*
[b] ● *– this aspect is the main focus in the paper;* ○ *– this aspect is partially addressed.*
[c] *LTR – lifetime reliability; SER – soft-error reliability;.*
[d] *RT – real-time constraints;.*
[e] *GL – gate level; SL – system level; ISA – instruction set architecture level; TLM – transaction level model.*

security aspects is highly dependent on the type of attack and the attacker model assumed.

Many of the existing works in security verification (e.g. [22,24,26,29,30]) are focusing on the integrity attribute, mostly addressing hardware trojan detection. There also exist works that additionally target [19,21,23,25,34] or are exclusively considering [27,28] the confidentiality aspect. Several solutions in security verification are restricted to specific target architectures or types of modules such as Reconfigurable Scan Networks (RSNs) [23,27] or macro-asynchronous micro-synchronous pipelines [30]. To that end, for complex hardware architectures (e.g. large IEEE1687 Reconfigurable Scan Networks or MPSoCs) the specific on-chip security features to be verified also tend to be very sophisticated. These may include on-chip mechanisms for attack prevention (firewalls, user management, communications' isolation), attack protection (traffic scrambling, encryption) and attack resilience (checkers for side-channel attacks, covert channel detection, attack recovery mechanisms). Several works consider security verification for NoC-based MPSoCs. [20] proposes a method to formally verify the correctness and the security properties of a NoC router. Some solutions in the security verification of NoCs do indirectly address reliability due to the fact that they implement hardware monitors that allow avoiding both, attacks and in-field faults [21,22].

According to recent surveys [37] and [38] cache access driven side-channel attacks have become a major concern in hardware security. In modern processors, deep hierarchy of cache memory is implemented to increase system performance. However, this makes modern computing systems, including IoT devices, vulnerable to cache side-channel attacks. There exist several works addressing verification of the cache security. In [31], the authors propose.

Computation Tree Logic (CTL) based modeling of timing-driven and access-driven cache attacks. This work concentrates on formally describing the attack types. Zhang and Lee [32] models cache as a state machine and proposes a metric based on the non-interference condition to evaluate the access-based cache vulnerability. Canones et al. [33] proposes a model to formally analyze the security of different cache replacement policies. None of the above-mentioned works consider multiple dimensions, or aspects.

An approach that is designed for modeling a multitude of extra-functional aspects is the model-based engineering example of Architecture Analysis and Design Language (AADL) [19]. While, in principle, AADL allows representing several extra-functional aspects (called quality attributes in AADL), Hansson et al. [19] only concentrates on analysis of confidentiality as a part of verifying security in a system with multiple levels of security. The authors in [36] have targeted a general Uppaal Timed Automata based multi-view hardware modeling and verification approach taking into consideration of the security view. The survey of related literature clearly shows that, up to this moment, there is virtually no work considering security verification in combination with other extra-functional aspects.

3.2. Reliability aspects

The key drivers for the reliability aspect in today's designs are the recent industrial standards in different application domains such as IEC61508, ISO26262, IEC61511, IEC62279, IEC62061, RTCA/DO-254, IEC60601, etc. Integrated circuits used in high-reliability applications, e.g. complying with high (Automotive) Safety Integrity Level - (A)SIL, must demonstrate low failure rates (modelled by FIT – Failures in Time) and high fault coverage (e.g. Single-Point Failure Metric SPFM and Latent Fault Metric LFM). These requirements ultimately mandate extra-functional validation efforts for reliability analysis, such as Failure Mode and Effects (Criticality) Analysis - FME(C)A and imply generalized use of methods and features, such as safety mechanisms, for error management. *Functional safety* is a property of the complete system rather than just a component property because it depends on the integrated operation of all sensors, actuators, control devices, and other integrated units. The goal is to reduce the residual risk associated with a functional failure of the target system below a threshold given by the assessment of severity, exposure, and controllability.

The dominant threats for reliability are, first, random hardware faults such as transient faults by radiation-induced single event effects or soft errors [15], i.e. a subject for *Soft-Error Reliability* (SER). Second, these are extreme operating conditions, electronic interference and intermittent to permanent faults by process or time-dependent variations, such as aging induced by Bias Temperature Instability (BTI) [13], where the latter is a subject for *Life-Time Reliability* (LTR). Reliability verification challenge is emphasized by the adoption of advanced nanoscale implementation technology nodes and high complexity of systems, utilizing tens or hundreds of complex microelectronic components and embedding large quantities of standard logic and memory. Moreover, these designs integrate IP cores from multiple design teams making reliability evaluation task to be scattered and complex. Initiatives such as RIIF (Reliability Information Interchange Format) [39], allow the formalization, specification and modeling of extra-functional, reliability properties for technology, circuits and systems.

Similar to other aspects, reliability in large complex electronic systems, e.g. safety-critical CPSs, may be tackled starting at high level of abstraction. System's fault tolerance is formally checked using UPPAAL and timed automata models generated from AADL specifications [41]. HW design models and tools at such a level also enable verification of interference of several extra-functional design aspects [36]. There are research works relying on design soft-error reliability verification by fault-injection campaigns, e.g. [49], or formal analysis, e.g. error-correction code (ECC) based mechanisms against single-bit errors in memory elements [48]. Burlyaev and Fradet [42] proposes a general approach to verify gate-level design transformations for reliability against single-event transients by soft errors that combines formal reasoning on execution traces. Thompto and Hoppe [43] and Kan et al. [44] focus on the RAS (Reliability, Availability and Serviceability) group of extra-functional aspects outlined by IBM for complex processor designs where embedded error protection mechanisms and designs intrinsic immunity (due to various masking) to errors is evaluated by fault injection. Vinco et al. [45,46] propose extensions to system descriptions in the IP-EXACT format to enable multi-layer representation and simulation of several mutually influencing extra-functional aspects of smart system designs such as lifetime reliability, power and temperature. A complex approach to verification of multiple reliability concerns (soft errors, BTI, etc.) across layers in industrial CPS designs is proposed in [47] as a collaborative research result in the IMMORTAL project. Last but not least, addressing the need for reliability verification automation tools, in [50], authors propose a fully automated tool INFORMER to estimate memory reliability metrics by circuit-level simulations of failure mechanisms such as soft-errors and parametric failures.

The survey clearly shows that currently there is a very small number of works considering verification of reliability together with other aspects.

3.3. Timing aspects

Functional temporal properties are essential part of sequential designs' specification that are often modelled for functional verification by Computational Tree Logic (CTL), applied for formal approaches, and Linear Temporal Logic (LTL) temporal assertions expressed arbitrarily, e.g. in Property Specification Language (PSL), System Verilog Assertions (SVA) or systematically, e.g. in Universal

Verification Methodology (UVM). In the extra-functional context, these can be extended to specific requirements and properties such as: *real-time (RT), performance, throughput, latency, on-chip communication time constraint, worst-case execution time* constraints, etc. Several works have been widely studying these timing properties. Some researchers are mainly focused on generating timing properties to reduce the verification effort, for example, state space and cost [54,56,65]. Other works instead use the timing properties to assess whether the system under verification is correctly functioning or not [55,62,64]. In the following, we discuss state of the art for each timing aspect.

A real-time system describes hardware and software systems subject to a *real-time constraint*, that ensures response within a specified time. The correctness of the function depends both on the correctness of the result and also the timeliness of the periods. In [54], an approach to verify the timed Petri-Net model is proposed. A non-instantaneous model is abstracted from the timed Petri-Net model in a hierarchical structure. The non-instantaneous model which is verified with a model-checking tool is used to reduce the state space of the timed Petri-Net model for verification with a satisfiability modulo theories solvers [76,77]. The timed Petri-Net is used to model the interacting relations of the software components and the binding relations between software and hardware in a certain period of time. Görgen et al. [65] introduces a tool called CONTREX to complement current activities in the area of predictable computing platforms and segregation mechanisms with techniques to compute real-time properties. CONTREX enables energy-efficient and cost-aware design through analysis and optimization of real-time constraint. The authors in [62] proposed a method to combine real-rime constraint aspect of a model with energy-aware real-time (ERT) behaviors of the model into UPPAAL for formal verification.

Throughput is a measure of how many units of information a system can process in a given amount of time. In [66], a verification environment has been proposed to estimate the throughput of a SoC. The intention of the paper is to judge whether the verification system can handle SOC verification and provide the necessary performance in terms of speed and throughput. Khamis et al. [67] introduced a Universal Verification Methodology (UVM) environment to measure throughput of a NoC. UVM is a SystemVerilog class library explicitly designed to help and build modular reusable verification components and test-benches. It is an industry standard, so it is possible to acquire UVM IP from other sources and reuse them.

Performance refers to the amount of work which is done during a process, for instance, executing instructions per second. In [56], a framework has been developed to analyze performance of a system design. The framework is based on stochastic modeling and simulation and it is applied on a set of NoC topologies. The methodology uses a selective abstraction concept to reduce complexity.

When referring to hardware, *latency* is the time required for a hardware component to respond to a request made by another component. However, in the cast of hardware, latency is sometimes referred to as the *access time*. In [55], an analysis tool is developed to work with the AADL models [78] to assure the correctness of a scheduling model that binds the relation of different components in a model.

On-chip communication time constraints refer to the requirements on the start and end times of each task in a system critical path, which is the sequence of tasks that cannot be delayed without delaying the entire system. For instance, in [51] and [52] a framework has been proposed, which is based on a set of quality of service aware NoC architectures along with the analysis methodology including selected relevant metrics that enable an efficient trade-off between guarantees and overheads in mixed-criticality application scenarios. These architectures overcome the notion of strictly divided regions by allowing non-critical communication pass through the critical region, providing they do not utilize common router resources. Such problem formulation is relevant to facilitate the usage of NoC technology by safety-critical industries such as avionics.

The *worst-case execution time* of a computational task is the maximum length of time the task could take to execute on a specific hardware platform. The designer of a system can employ techniques such as schedulability analysis to verify that the system responds fast enough [40]. For instance, Zimmermann et al. [64] presents an approach to generate a virtual execution platform in SystemC to advance the development real-time embedded systems including early validation and verification. These virtual execution platforms allow the execution of embedded software with strict consideration of the underlying hardware platform configuration in order to reduce subsequent development costs and to allow a short time-to-market by tailoring and exploring distributed embedded hardware and software architectures.

Last but not least, a few works also take into account dependencies between several extra-functional aspects. For instance, the work in [62,65] and [56] present the effect of optimizing timing properties (performance and latency) on power consumption or the study in [64] performs the effect of decreasing execution time on power consumption. Such analysis is mostly limited to two extra functional aspects or neglected at all [53–55,69], while design timing constraints can strongly influence not only power consumption but reliability, security, availability, etc. as well as functional properties.

3.4. Power aspects

In commercial flows, verification of the power aspect can be addressed relatively independently from the functional verification dimension. The *power intent* and detailed power modelling can be done starting at TLM or RTL with minimal interference with the HDL functional description, e.g. using the Accellera introduced Unified Power Format (UPF) employed for power-aware design verification automation by commercial tools especially with the latest UPF3.0 [60] or Cadence/Si2 Common Power Format CPF [61]. For the advanced device implementation technologies, power specification implies *multi-voltage design* with up to tens of *power domains* and may consider dynamic and adaptive voltage scaling.

In the recent research works, design verification against the power aspect is performed at different abstraction levels with a trade-off between speed and accuracy. Some works such as [58,59,74,75] perform power analysis at system level targeting high simulation speed and low power optimization flexibility similar to the accuracy achievable at lower levels. In [58], the authors applied their approach to SRAM and AES encryption IPs and obtained a significant simulation speed-up in comparison to gate-level simulation with a high fidelity of the system-level power simulation. A promising software tool for power simulation in SystemC designs is the Powersim framework [59,75]. In [59], a methodology to estimate the dissipation of energy in hardware at any level of abstraction is proposed. In [75], the authors propose a SystemC class library aimed at calculation of energy consumption of hardware described at system level. The work in [73] introduces a series of tools (PowerBrick (construct power library for standard cell library), PowerMixer (for RTL/gate-level estimator), PoweMixer^{ip} (IP-based model builder), PowerDepot (estimate system-level power consumption)) which can be tightly linked and enable the power analysis from layout, gate-, RT-, IP- to system level with a good simulation speed while retaining high accuracy. The power aspect verification could benefit from a holistic multi-level modelling, such as e.g. [17] available for functional verification. Rafiev et al.

[56], Vinco et al. [45,46], Kang et al. [62], Zimmermann et al. [64], Görgen et al. [65], are aimed at methodologies suitable for specific applications (such as cyber-physical system [62]) that assume verification of extra-functional aspects such as power, timing, thermal at the system level.

This extra-functional aspect has a tight relation to the implementation technology assumed for the synthesis of the design model under verification. With planar bulk MOSFET technology known for exponential growth of the static leakage power for smaller device geometries and employment of FinFET and Tri-Gate-Transistors in the advanced technology nodes, the CMOS device parameters are essential for this analysis [57].

3.5. Machine learning based techniques

The complex problem of multidimensional verification can be assisted by the recent advances in the machine learning discipline. This type of approaches (along with e.g. evolutionary algorithms) is particularly suitable for multi-aspect optimization problems where formal deterministic approaches may lack scalability.

Machine Learning (ML) is the concept of a machine learning from examples and making predictions based on its experience, without being explicitly programmed [82]. Previous works have shown that ML can be used for verification purposes at different levels. In [83], machine learning was introduced in physical design analysis. The feasibility of ML in physical design verification (e.g., lithography hotspot detection) was investigated, and a reference model for application was presented. Based on this work [84] used ML to increase the speed of the performance evaluation (power and area) of a circuit design after physical design by a factor of 40 as well as performing a Design Rule Check. In [85], ML was used to predict the timing behavior of the final floorplan of a circuit during the Place & Route routine and thus, shifting the analysis to an earlier design stage. In [79], the analysis is moved even to higher abstraction level. The high-level synthesis (HLS) resource usage and timing estimation was improved by train ML models with data from real implementations. Thus, the design flow can be assisted with machine learning and predict accurate values even in very early design stages. Machine learning was further applied for Security Verification in [80,81,86], where it was used to detect Hardware Trojans based on features from the Gate Level Netlist. In Section 5, we propose an approach to assist the multidimensional verification flow by using machine learning techniques to estimate a reliability metric, as well as timing metric.

4. The challenges of multidimensional verification

The performed analysis of the state of the art has outlined a gap in methodologies and tools for holistic multidimensional verification of hardware design models.

Different from functional verification, approaches for extra-functional hardware design aspects' verification remain underdeveloped even when tackled in isolation. Here, one of the key issues is a lack of established metrics for verification confidence. For a particular functional verification plan, the functional dimension usually includes conventional structural (code) coverage metrics, functional coverage [3] in form of asserted and assumed properties and design parameters along with stimuli quality assessment by model mutations [18]. The metrics for confidence in extra-functional dimension verification results may be challenging as in practice the requirements are *subjective* and can be specified as a mixture of *quantitative* and *qualitative constraints*. Accurate hardware verification in a particular dimension requires both sufficient extra-functional design modeling and the extra-functional aspects target modeling [36]. There is a limited number of dedicated commercial tools and common standards for extra-functional verification flows. In particular, for the security dimension the JasperGold SPV [35] is one of the few such commercial tools that stand out from the academic research frameworks. Finally, the issue of eliciting the extra-functional requirements [4,5] is a challenging task as ambiguity and (sometimes conflicting) interdependency of the extra-functional aspects in the specifications increases complexity and may leave gaps in the multidimensional verification plans.

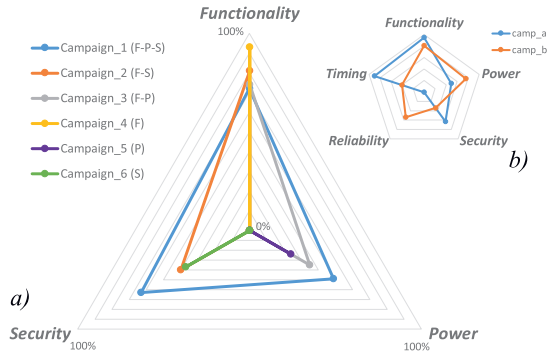


Fig. 2. Multidimensional verification campaigns (Radar-chart n-dimensional visualization).

Unfortunately, there is no established hardware design methodology supporting multidimensional verification plans for mutually influencing functional and extra-functional aspects. There is a very limited number of research works going beyond analysis of one extra-functional verification aspect under constraints of another as the complexity of the problem grows extremely fast with the number of dimensions (interdependent constraints) and the electronic system size. The first works in this direction are, for example, Vinco et al. [46] and Vain et al. [36].

Ultimately, results of multidimensional verification campaigns proposed in this work are subject to be represented in a multi-dimensional space, as illustrated in Fig. 2a. Here is shown an illustration of six hypothetical independent verification campaigns in a three-dimensional verification space. A verification campaign in this example shows the level of confidence in the different dimensions - (F)unctionality, (P)ower and (S)ecurity. In this illustrative example, only three aspects are taken into consideration. Obviously, on the demand the verification engineers can involve different dimensions. Here, the different colors of the lines represent different multi-dimensional spaces e.g. as Campaign_1 in blue lines stand for the verification result considering three extra functional aspects i.e., functional, power and security aspects at the same time. The figure shows the interdependency of these three requirements and thus can help the designers to choose the most suitable design combination. Subsequently, Compaign_2 represents the combination of functional and security aspects, Compaign_3 demonstrates the combination of functional and power aspect, etc. Thus the *Radar-charts, as shown in* Fig. 2b, are an instrument for summarizing multidimensional verification results for a large number of dimensions, (where the dimensions can be ordered to emphasize correlation or interdependencies between adjacent dimensions).

4.1. Motivational example

Single-dimension verification campaigns ignoring interdependencies between the dimensions may lead to gaps in the overall electronic system quality. As an example to show the importance of multidimensional verification, let us consider an actual verification campaign of an open-source NoC framework Bonfire [71,72].

```
process(write_en, write_pointer) begin --write pointer bug
  if write_en = '1' then
  write_pointer_in <= write_pointer(0)&write_pointer(3 downto 1);          -- Bug f1!
  else
    write pointer in <= write pointer;
  end if;
end process;

process(read_en, empty, read_pointer) begin --read pointer bug
  if (read_en = '1' and empty = '0') then
    read_pointer_in <= read_pointer(0)&read_pointer(3 downto 1);           -- Bug f1!
  else
    read pointer in <= read pointer;
  end if;
end process;
```

```
process(write_en, write_pointer )begin --write pointer
  if write_en = '1' then
  write_pointer_in <= write_pointer(2 downto 0)&write_pointer(3);
  else
    write_pointer_in <= write_pointer;
end if;
end process;

process(read_en, empty, read_pointer) begin --read pointer
  if (read_en = '1' and empty = '0') then
    read_pointer_in <= read_pointer(2 downto 0)&read_pointer(3);
  else
    read_pointer_in <= read_pointer;
  end if;
end process;
```

Fig. 3. Bug f1 and its correction.

```
process(Healthy_packet, reset_counters, healthy_counter_out) begin
  if reset_counters  = '1' then
      healthy_counter_in <=  (others => '0');
  elsif Healthy_packet = '1' then                  -- Bug p1!
      healthy_counter_in <= healthy_counter_out + 1;
  else
      healthy_counter_in <= healthy_counter_out;
  end if;
end process;
```

```
process(Healthy_packet, reset_counters, healthy_counter_out,  faulty_counter_out) begin
  if reset_counters  = '1' then
      healthy_counter_in <=  (others => '0');
  elsif Healthy_packet = '1' and faulty_counter_out /= std_logic_vector(to_unsigned(0, faulty_counter_out'length)) then
      healthy counter in <= healthy counter out + 1;
  else
      healthy_counter_in <= healthy_counter_out;
  end if;
end process;
```

Fig. 4. Bug p1 and its correction.

The design under verification is in RTL VHDL and implements a 2×2 NoC infrastructure (processing elements excluded). The verification plan considered 2-dimensional verification campaign targeting *functionality* and *power consumption* requirements. For the former, assertion-based functional verification by simulation was employed targeting statement, branch, condition and toggle coverage metrics and satisfaction of a set of temporal simple-subset PSL assertions. For the latter, a set of power targets were extracted for the targeted silicon implementation assuming a particular switching activity (set to 12 mW in this example).

Among documented design errors in the Bonfire project, the bug *f1,* as shown in Fig. 3, is an example of a functional misbehavior due to improper usage of write and read pointers in the FIFO. The figure represents the code errors in the red line and the corrected versions of the code lines in blue. The bug f1 and the bug p1 demonstrate the error in Figs. 3 and 4, respectively. The bug *p1* causes violations of specified power consumption targets because of unnecessary excessive use of a fault-tolerance structure related counter. The report of such a power consumption is described in Table 2. The power consumption is shown in the cell Total Power which is composed of the dynamic power, i.e. the Switching Power in the interconnects and the Internal Power in the logic cells, and the insignificant (for the target technology) static leakage power Leak Power. As summarized in the first row, for the bug f1 the Total Power is equal to 10.211 (consistence with the power consumption requirement). Similarly, in the third row, which represents the power consumption for the correct version of the code, the total power is equal to 10.184. This report prove that even if there is a bug (bug f1) in the code but still the power consumption requirement is met. In contrary, for the bug p1, even though there is no functional errors, the Total Power consumption is reported which is equal to 22.137. Thus the bug p1 results in a double power consumption compared to the correct implementation and violates the power targets in the specification. This fact prove that it is critical to know how and where the code should be modified in order to reduce the power consumption as well as maintain functional correctness. In general, the above simple motivation example demonstrates the challenge of interdependency of different aspects when requirements in more than one dimension are present.

5. Machine learning to tackle the challenges of multidimensional verification

As it can be seen in the previous sections, multidimensional verification is a complex multi-aspect optimization problem. Machine learning algorithms are known to be able to learn complex relationships and have been used for several optimization problems. Section 3.5 has shown that machine learning techniques were already successfully used for estimating several different single verification metrics. This suggests that machine learning can be also used for solving multidimensional verification problem. There-

Table 2
Power consumption of the Bonfire system implementation: corrected and with bugs f1 and p1.

Bonfire system Implementation	Switching Power (mW)	Internal Power (mW)	Leak Power (pW)	Total Power (mW)
with *f1* bug	0.783	9.427	7.50e+05	10.211
with *p1* bug	0.757	21.379	6.93e+05	22.137
corrected	0.666	9.518	7.43e+05	10.184

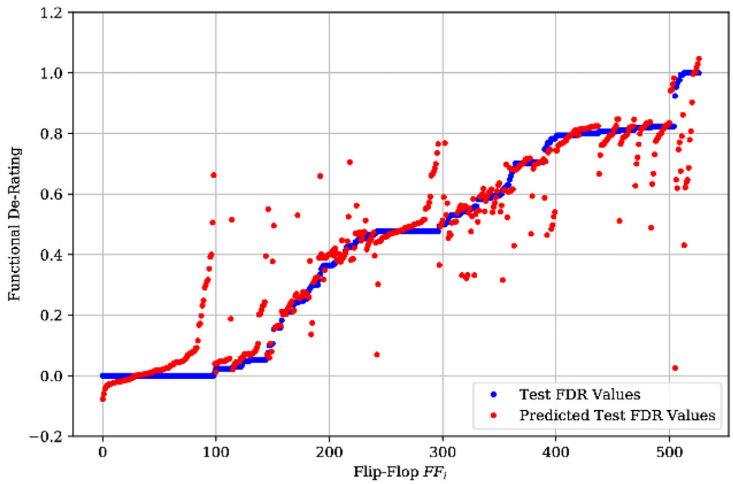


Fig. 5. Prediction of Functional De-Rating factors of the test data set by using a Support Vector Machine regression model (Training Size = 50%, Coefficient of Determination $R^2 = 0.844$).

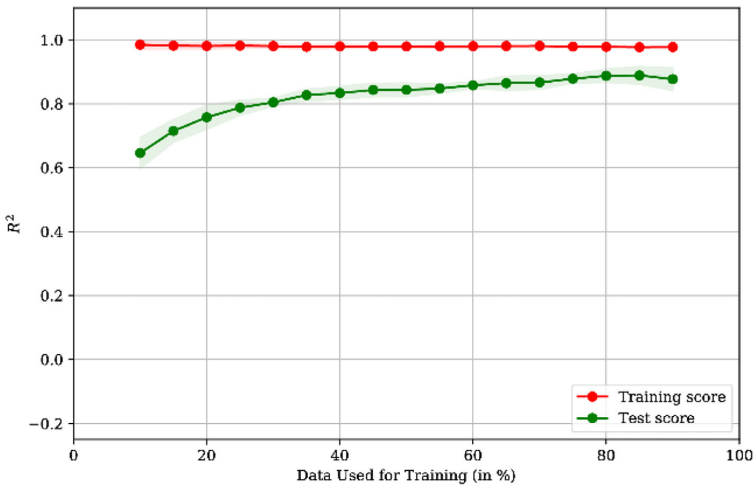


Fig. 6. Learning Curve for the Functional De-Rating prediction by using a Support Vector Machine regression model with different training sizes.

fore, an initial approach is proposed which is based on machine learning techniques in order to tackle this multidimensional verification challenge.

5.1. Proposed methodology

The proposed approach targets to predict two different verification metrics based on the same feature set extracted from the gate-level netlist of a given circuit. These two different metrics are Prediction of De-Rating and Path delay. The first metric to predict is the De-Rating or Vulnerability Factor, which are related to the reliability verification flow and a major metric of the failure analysis. The second metric is the path delay and related to the timing analysis. This metric is usually obtained during the synthesis or place and route stage of the design development. Therefore, machine learning can help to shift the analysis to an earlier design stage.

A possible application scenario consists in extracting a list of circuit feature and training a ML tool with a limited set of reference inputs (the values of the selected circuit features) and

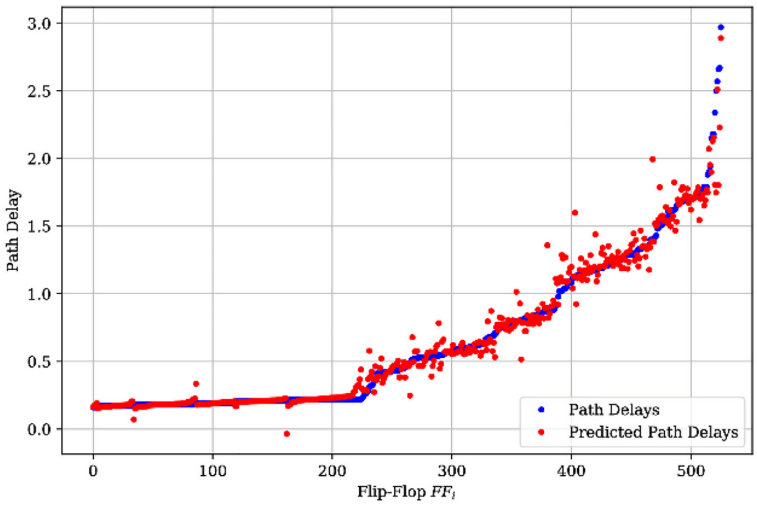


Fig. 7. Prediction of Path Delays of the test data set by using a Support Vector Machine regression model (Training Size = 50%, Coefficient of Determination $R^2 = 0.975$).

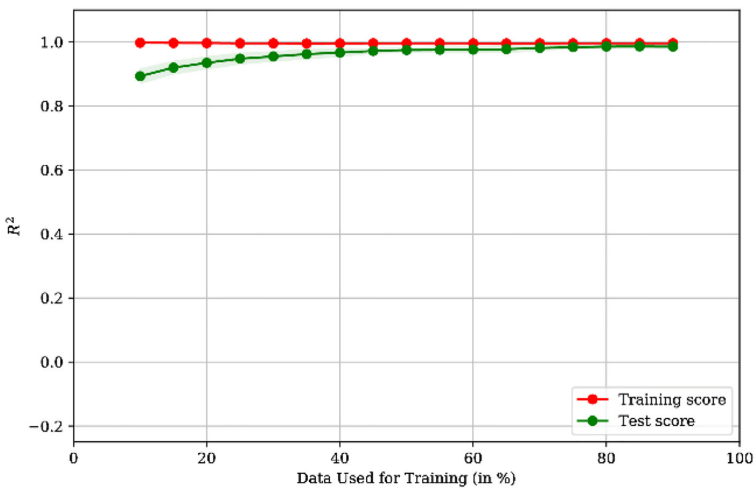


Fig. 8. Learning Curve for the Path Delay prediction by using a Support Vector Machine regression model with different training sizes.

expected outputs (reliability and timing metrics). Depending on the exhaustiveness of the training campaign, the trained ML tool can provide actual reliability metrics from a limited list of circuit features while spending far less resources (CPU time, EDA tools licenses, man-power) than using classical methods.

5.2. Prediction results

The proposed idea was implemented and evaluated on a practical example. Therefore, a set of features is defined which characterizes each flip-flop instance in the circuit. The feature set is composed of static elements (cell properties, circuit structure, synthesis attributes) and dynamic elements (signal activity). After extracting the features for the full list of circuit instances, reference data was obtained. The Functional De-Rating per flip-flop was determined through first-principles fault simulation approaches and the path delay was extracted by a classical static timing analysis. One part of the reference dataset is used to train the machine learning model and the remaining data is used to validate and benchmark the accuracy of the trained tool.

As a circuit under test, the Ethernet 10GE MAC Core was used which is available as RTL description from OpenCores. The circuit consists of control logic, state machines, FIFO controllers and memory interfaces. By synthesizing the design with NanGate FreePDK45 Open Cell Library, 1054 flip-flops have been identified and the corresponding features have been extracted.

Several machine learning models have been evaluated, such as the Linear Least Squares, Ridge (with linear and non-kernels), k-Nearest Neighbors (k-NN), Decision Tree (CART) and Support Vector regression (SVR, with linear and non-linear kernels). It has been noted that especially the linear models are not very suitable to predict the reliability metrics. The non-linear models perform much better and the Support Vector regression with Radial Basis Function (RBF) as kernel functions is among the best. There-

fore, the SVR model with RBF kernel function is used for the following presentation of the prediction results. Figs. 5 and 7 show the prediction of the two metrics. When 50% (527 flip-flops) of the data are used to train the model and the remaining 50% was used to evaluate the model. The performance of regression models is usually evaluated by using the Coefficient of Determination (R^2) score and the model reaches a score of $R^2 = 0.844$ to predict the Functional De-Rating and $R^2 = 0.975$ to predict the path delay. Figs. 6 and 8 show the learning curve of the model. This curve describes the performance of the model for different sizes of the data set used for training and the remaining data set used for the evaluation. The learning curves suggest that by using more than 50% of the available data for training doesn't significantly improve the prediction performance. However, it can also be seen that by using less than 50% still a valuable prediction can be performed. Thus, by allowing a slight reduction of accuracy, the cost of an exhaustive analysis can still be reduced.

The practical example has shown that machine learning can be successfully applied for different verification purposes. In order use ML to support the multidimensional verification problem, features from different design stages need to be extracted and used to train a unified model or several separated models. These can be used to predict the required verification metrics.

6. Conclusion

In the recent years, numerous extra-functional aspects of electronic systems were brought to the front and imply verification of hardware design models in multidimensional space along with the functional concerns of the target system. Targeting at understanding of this new verification paradigm, we have performed a comprehensive analysis of the state of the art and presented a taxonomy for multidimensional hardware verification aspects, an up-to-date survey of related research works and trends towards enabling the multidimensional verification concept and investigated the potential of machine learning based techniques to support the concept. As the result of the performed analysis of the state of the art we have outlined a gap in methodologies and tools for holistic multidimensional verification of hardware design models and the key challenges.

Declaration of competing interest

All authors have participated in (a) conception and design, or analysis and interpretation of the data; (b) drafting the article or revising it critically for important intellectual content; and (c) approval of the final version.

Acknowledgments

This research was supported in part by projects H2020 MSCA ITN RESCUE funded from the EU H2020 programme under the MSC grant agreement no. 722325, by the Estonian Ministry of Education and Research institutional research grant no. IUT19-1 and by European Union through the European Structural and Regional Development Funds.

References

[1] I. Verbauwhede, Security adds an extra dimension to IC design: future IC design must focus on security in addition to low power and energy, in: IEEE Solid-State Circuits Magazine, 9, Fall 2017, pp. 41–45.
[2] W. Chen, S. Ray, J. Bhadra, M. Abadir, L.C. Wang, Challenges and trends in modern SoC design verification, in: IEEE Design & Test, 34, Oct. 2017, pp. 7–22.
[3] A. Piziali, Functional Verification Coverage Measurement and Analysis, Springer, 2008.
[4] S. Ullah, M. Iqbal, A.M. Khan, A survey on issues in non-functional requirements elicitation, in: Int. Conf. on Computer Networks and Information Technology, Abbottabad, 2011, pp. 333–340.
[5] L.M. Cysneiros, E. Yu, Non-Functional requirements elicitation, in: J.C.S. do Prado Leite, J.H. Doorn (Eds.), Perspectives On Software Requirements. The Springer International Series in Engineering and Computer Science, 753, Springer, Boston, MA, 2004.
[6] M.L. Fair, Reliability, availability, and serviceability (RAS) of the IBM eServer z990, IBM J. Res. Dev. 48 (3.4) (May 2004) 519–534.
[7] L. Chung, B. Nixon, E. Yu, J. Mylopoulos, Non-Functional requirements, Software Engineering, Kluwer Academic, 2000.
[8] P. Singh, A.K. Tripathi, Exploring problems and solutions in estimating testing effort for non functional requirement, Int. J. Comput. Technol. 3 (2b) (2012) 284–290.
[9] E.R. Poort, N. Martens, I. Van de Weerd, H. Van Vliet, How architects see non-functional requirements: beware of modifiability, in: Requirements Engineering: Foundation for Software Quality, Springer, Berlin Heidelberg, 2012, pp. 37–51.
[10] D. Ameller, C. Ayala, J. Cabot, X. Franch, How do software architects consider non-functional requirements: an exploratory study, in: Requirements Engineering Conference (RE), 2012, pp. 41–50.
[11] M. Glinz, On non-functional requirements, in: Requirements Engineering Conference, 2007. RE'07, IEEE, 2007, pp. 21–26.
[12] L. Motus, Analytical study of quantitative timing properties of software, 5th EUROMICRO Workshop on Real-Time Systems, 1993.
[13] M. Jenihhin, G. Squillero, T.S. Copetti, V. Tihhomirov, S. Kostin, M. Gaudesi, F. Vargas, J. Raik, M. Sonza Reorda, L. Bolzani Poehls, R. Ubar, G.C. Medeiros, Identification and rejuvenation of NBTI-Critical logic paths in nanoscale circuits, JETTA 32 (3) (June 2016) 273–289.
[14] J. Bhadra, M.S. Abadir, L.C. Wang, S. Ray, A survey of hybrid techniques for functional verification, in: IEEE Design & Test of Computers, 24, 2007, pp. 112–122.
[15] S. Mukherjee, Architecture design for soft errors, Morgan Kauf (2008).
[16] A. Ptzmann, M. Hansen, Anonymity unlinkability undetectability unobservability pseudonymity and identity management, A Consolidated Proposal for Terminology version 0.31, 2008.
[17] R. Ubar, et al., Diagnostic modeling of digital systems with multi-level DDs, in: R. Ubar, J. Raik, Vierhaus H.Th. (Eds.), Design and Test Technology For Dependable, SoC, 2011, pp. 92–118.
[18] V. Guarnieri, Mutation analysis for SystemC designs at TLM, in: 2011 12th Latin American Test Workshop (LATW), Porto de Galinhas, 2011, pp. 1–6.
[19] J. Hansson, B. Lewis, J. Hugues, L. Wrage, P. Feiler, J. Morley, Model-Based verification of security and non-functional behavior using AADL, IEEE Secur. Priv. (2009) 1–1.
[20] J. Sepulveda, D. Aboul-Hassan, G. Sigl, B. Becker, M. Sauer, Towards the formal verification of security properties of a Network-on-Chip router, in: 2018 IEEE 23rd European Test Symposium (ETS), Bremen, 2018, pp. 1–6, doi:10.1109/ETS.2018.8400692.
[21] T. Boraten, D. DiTomaso, A.K. Kodi, Secure model checkers for Network-on-Chip (NoC) architectures, in: 2016 Int. Great Lakes Symposium on VLSI (GLSVLSI), Boston, MA, 2016, pp. 45–50.
[22] H.M.G. Wassel, Networks on chip with provable security properties, IEEE Micro. 34 (3) (May-June 2014) 57–68.
[23] M.A. Kochte, M. Sauer, L.R. Gomez, P. Raiola, B. Becker, H.J. Wunderlich, Specification and verification of security in reconfigurable scan networks, in: 2017 22nd IEEE European Test Symposium (ETS), Limassol, 2017, pp. 1–6.
[24] L.W. Kim, J.D. Villasenor, Dynamic function verification for system on chip security against hardware-based attacks, IEEE Transa. Reliab. 64 (4) (Dec. 2015) 1229–1242.
[25] Hu Wei, et al., Imprecise security: quality and complexity tradeoffs for hardware information flow tracking, in: IEEE/ACM Int. Conference on Computer-Aided Design (ICCAD), Austin, TX, 2016, pp. 1–8.
[26] A. Nahiyan, M. Sadi, R. Vittal, G. Contreras, D. Forte, M. Tehranipoor, Hardware trojan detection through information flow security verification, in: 2017 IEEE International Test Conference (ITC), Fort Worth, TX, 2017, pp. 1–10.
[27] M.A. Kochte, R. Baranowski, M. Sauer, B. Becker, H.J. Wunderlich, Formal verification of secure reconfigurable scan network infrastructure, in: 2016 21th IEEE European Test Symposium (ETS), Amsterdam, 2016, pp. 1–6.
[28] M. Rocchetto, N.O. Tippenhauer, Towards formal security analysis of industrial control systems, in: ACMA sia Conf. Comput. Commun. Secur., 2017, pp. 114–126.
[29] M. Yoshimura, T. Bouyashiki, T. Hosokawa, A hardware trojan circuit detection method using activation sequence generations, in: 2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC), Christchurch, 2017, pp. 221–222.
[30] F.K. Lodhi, S.R. Hasan, O. Hasan, F. Awwad, Formal analysis of macro synchronous micro asychronous pipeline for hardware trojan detection, in: NORCAS, Oslo, 2015, pp. 1–4.
[31] S. Deng, W. Xiong, J. Szefer, Cache timing side-channel vulnerability checking with computation tree logic, in: Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy, ser. HASP '18, New York, NY, USA, ACM, 2018, p. 2. 1–2:8.
[32] T. Zhang, R.B. Lee, New models of cache architectures characterizing information leakage from cache side channels, in: Proceedings of the 30th Annual Computer Security Applications Conference, ser. ACSAC'14, New York, NY, USA, ACM, 2014, pp. 96–105.
[33] P. Canones, B. Köpf, J. Reineke, Security analysis of cache replacement policies, CoRR, 2017 arXiv:1701.06481.

[34] Z. Hanna, Verifying security aspects of SoC designs with Jasper app, (white paper), Jasper Design Automation (Cadence), 2013.
[35] JasperGold Security Path Verification App, Cadence, http://www.cadence.com.
[36] J. Vain, A. Kaur, L. Tsiopoulos, J. Raik, M. Jenihhin, Multi-view modeling for MP-SoC design aspects, in: 2018 16th Biennial Baltic Electronics Conference (BEC), Tallinn, 2018, pp. 1–6.
[37] Q. Ge, Y. Yarom, D. Cock, G. Heiser, A survey of microarchitectural timing attacks and countermeasures on contemporary hardware, J. Cryptogr. Eng. 8 (1) (2018) 1–27.
[38] Y. Lyu, P. Mishra, A survey of side-channel attacks on caches and countermeasures, J. Hardw. Syst. Secur. 2 (1) (Mar 2018) 33–50.
[39] A. Savino, S. Di Carlo, A. Vallero, G. Politano, D. Gizopoulos, A. Evans, RIIF-2: toward the next generation reliability information interchange format, IEEE IOLTS (2016) 173–178.
[40] C. Liu, J. Layland, Scheduling algorithms for multi programming in a hard real time environment, J. ACM 20 (1973) 46–61.
[41] F.S. Gonçalves, D. Pereira, E. Tovar, L.B. Becker, Formal verification of AADL models using UPPAAL, in: 2017 VII Brazilian Symposium on Computing Systems Engineering (SBESC), Curitiba, 2017, pp. 117–124.
[42] D. Burlyaev, P. Fradet, Formal verification of automatic circuit transformations for fault-tolerance, in: 2015 Formal Methods in Computer-Aided Design (FMCAD), Austin, TX, 2015, pp. 41–48.
[43] B.W. Thompto, B. Hoppe, Verification for fault tolerance of the IBM system z microprocessor, in: Design Automation Conference, Anaheim, CA, 2010, pp. 525–530.
[44] S. Kan, M. Lam, T. Porter, J. Dworak, A case Study: pre-Silicon SoC RAS validation for NoC server processor, in: MTV, 2016, pp. 19–24.
[45] S. Vinco, M. Lora, E. Macii, M. Poncino, IP-XACT for smart systems design: extensions for the integration of functional and extra-functional models, in: 2016 Forum on Specification and Design Languages (FDL), Bremen, 2016, pp. 1–8.
[46] S. Vinco, Y. Chen, F. Fummi, E. Macii, M. Poncino, A layered methodology for the simulation of extra-functional properties in smart systems, in: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 36, 2017, pp. 1702–1715.
[47] G. Aleksandrowicz, et al., Designing reliable cyber-physical systems, in: F. Fummi, R. Wille (Eds.), Languages, Design Methods, and Tools for Electronic System Design. Lecture Notes in Electrical Engineering, 454, Springer, Cham, 2018.
[48] Eli Arbel, Shlomit Koyfman, Prabhakar Kudva, Shiri Moran, Automated detection and verification of parity-protected memory elements, in: Proc. IEEE/ACM ICCAD, 2014, pp. 1–8.
[49] M. Maniatakos, Y. Makris, Workload-driven selective hardening of control state elements in modern microprocessors, in: VTS, 2010, pp. 159–164.
[50] S. Ganapathy, R. Canal, D. Alexandrescu, E. Costenaro, A. González, A. Rubio, INFORMER: an integrated framework for early-stage memory robustness analysis, in: 2014 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2014, pp. 1–4.
[51] S. Avramenko, S.P. Azad, S. Esposito, B. Niazmand, M. Violante, J. Raik, M. Jenihhin, QoSinNoC: analysis of qos-Aware NoC architectures for mixed-criticality applications, in: 21st IEEE Int. Symp. DDECS, 2018, pp. 1–6.
[52] S. Avramenko, Upgrading QoSinNoC: efficient routing for mixed-criticality applications and power analysis, in: IEEE VLSI-SoC, Verona, 2018, pp. 1–6.
[53] S. Rubini, F. Singhoff, J. Hugues, Modeling and verification of memory architectures with AADL and REAL, in: 2011 16th IEEE International Conference on Engineering of Complex Computer Systems, Las Vegas, NV, 2011, pp. 338–343.
[54] H. Wang, X. Zhou, Y. Dong, L. Tang, A hierarchical verification procedure of timed petri-net model for real-time embedded systems, in: 2010 2nd International Conference on Information Engineering and Computer Science, Wuhan, 2010, pp. 1–4.
[55] H. Wang, X. Zhou, Y. Dong, L. Tang, Timing properties analysis of real-time embedded systems with AADL model using model check, in: IEEE Int. Conf. on Progress in Informatics and Computing (PIC), 2010, pp. 1019–1023.
[56] A. Rafiev, F. Xia, A. Iliasov, A. Romanovsky, A. Yakovlev, Selective abstraction for estimating extra-functional properties in Networks-on-Chips using archon framework, in: 2017 17th International Conference on Application of Concurrency to System Design (ACSD), Zaragoza, 2017, pp. 80–85.
[57] P. Khondkar, Low-Power Design and Power-Aware Verification, Springer, 2018.
[58] D. Lorenz, Non-invasive power simulation at system-level with systemc, PATMOS 2012. LNCS (7606), Springer, 2012.
[59] S. Orcioni, et al., Energy estimation in SystemC with powersim, Integr. VLSI J. (55) (2016) 118–128.
[60] ANSI/IEEE 1801-2015 - IEEE Standard for design and verification of Low-Power, energy-aware electronic systems, March 2016,
[61] Si2 Common Power Format, v2.1, Silicon Integration Initiative, 2014.
[62] E.Y. Kang, D. Mu, L. Huang, Q. Lan, Verification and validation of a cyber-physical system in the automotive domain, in: 2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Prague, 2017, pp. 326–333.
[63] A. Danese, G. Pravadelli, I. Zandonà, Automatic generation of power state machines through dynamic mining of temporal assertions, in: 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, 2016, pp. 606–611.
[64] J. Zimmermann, S. Stattelmann, A. Viehl, O. Bringmann, W. Rosenstiel, Model-driven virtual prototyping for real-time simulation of distributed embedded systems, in: 7th IEEE Int. Symposium on Industrial Embedded Systems (SIES'12), Karlsruhe, 2012, pp. 201–210.
[65] R. Görgen, et al., CONTREX: design of embedded mixed-criticality CONTRol systems under consideration of EXtra-Functional properties, in: 2016 Euromicro Conference on Digital System Design (DSD), Limassol, 2016, pp. 286–293.
[66] A.W. Ruan, Y.B. Liao, P. Li, W.C. Li, W. Li, Throughput estimation for modelsim simulator tool based HW/SW co-verification system, in: 2009 International Conference on Communications, Circuits and Systems, Milpitas, CA, 2009, pp. 1014–1018.
[67] M. Khamis, S. El-Ashry, A. Shalaby, M. AbdElsalam, M.W. El-Kharashi, A configurable RISC-V for noc-Based MPSoCs: a framework for hardware emulation, in: 2018 11th International Workshop on Network on Chip Architectures (NoCArc), Fukuoka, 2018, pp. 1–6.
[68] JasperGold Connectivity Verification App, Cadence, http://www.cadence.com.
[69] S.K. Roy, Top level SOC interconnectivity verification using formal techniques, in: The 8th Int. Workshop on Microprocessor Test and Verification, Austin, TX, USA, 2008, pp. 63–70.
[70] M. Elver, V. Nagarajan, McVerSi: a test generation framework for fast memory consistency verification in simulation, in: 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA), Barcelona, 2016, pp. 618–630.
[71] S.-P. Azad, B. Niazmand, K. Janson, J. Raik, Github Bonfire Project (2017), https://github.com/Project-Bonfire/ (accessed 1 June 2019).
[72] S.P. Azad, From online fault detection to fault management in Network-on-Chips: a ground-up approach, in: 2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), Dresden, 2017, pp. 48–53.
[73] S.-C. Fang, C.-C. Weng, C.-K. Tseng, C.-W. Hsu, J.-L. Liao, S.-Y. Huang, C.-L. Lung, D.-M. Kwai, SoC power analysis framework and its application to power-thermal co-simulation, in: 2011 Int. Symp. on VLSI Design, Automation and Test, April 2011, pp. 1–4.
[74] G. Vece, M. Conti, S. Orcioni, Transaction-level power analysis of VLSI digital systems, Integr. VLSI J. 50 (2015) 116–126.
[75] M. Giammarini, M. Conti, S. Orcioni, System-level energy estimation with powersim, in: 2011 18th IEEE Int. Conf. on Electronics, Circuits and Systems (ICECS), December 2011, pp. 723–726.
[76] Bell Labs, Verifying Multi-threaded Software with Spin, (1980). http://spinroot.com/ (accessed 1 June 2019).
[77] SMT Steering Committee, The International Satisfiability Modulo Theories (SMT) Competition. http://www.smtcomp.org/ (accessed 1 June 2019).
[78] Carnegie Mellon University, Architecture Analysis and Design Language. http://www.aadl.info/aadllcurrentsite/ (accessed 1 June 2019).
[79] S. Dai, Y. Zhou, H. Zhang, E. Ustun, E.F.Y. Young, Z. Zhang, Fast and accurate estimation of quality of results in high-level synthesis with machine learning, in: 2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2018, pp. 129–132.
[80] K. Hasegawa, M. Oya, M. Yanagisawa, N. Togawa, Hardware Trojans classification for gate-level netlists based on machine learning, in: 2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS), 2016, pp. 203–206.
[81] K. Hasegawa, M. Yanagisawa, N. Togawa, Hardware Trojans classification for gate-level netlists using multi-layer neural networks, in: 2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS), 2017, pp. 227–232.
[82] E. Alpaydin, F. Bach, Introduction to Machine Learning, MIT Press, 2014.
[83] B. Yu, D.Z. Pan, T. Matsunawa, X. Zeng, Machine learning and pattern matching in physical design, in: The 20th Asia and South Pacific Design Automation Conference, 2015, pp. 286–293.
[84] B. Li, P.D. Franzon, Machine learning in physical design, in: 2016 IEEE 25th Conference on Electrical Performance Of Electronic Packaging And Systems (EPEPS), 2016, pp. 147–150.
[85] L. Bai, L. Chen, Machine-Learning-Based early-stage timing prediction in SoC physical design, in: 2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT), 2018, pp. 1–3.
[86] K. Hasegawa, M. Yanagisawa, N. Togawa, Trojan-feature extraction at gate-level netlists and its application to hardware-Trojan detection using random forest classifier, in: 2017 IEEE International Symposium on Circuits and Systems (ISCAS), 2017, pp. 1–4.

Xinhui Lai is one of the early stages researchers in the RESCUE European Training Network. She is doing her research work and Ph.D in Tallinn University of Technology which is one of the institutions evolved in the RESCUE project. She got her bachelor and master diploma in Electronic Engineering from Politecnico di Torino, Italy, in October 2014 and April 2017 respectively. Her research is focused on design error functional verification and automated debug, i.e. localization and correction, as well as verification of extra-functional interdependent aspects in nanoelectronic system design such as security, reliability, power/performance envelope.

Aneesh Balakrishnan is an Early Stage Researcher in the RESCUE European Training Network. Within the project, he is a Research and Development Engineer at iRoC Technologies, France and also a Ph.D. Candidate in the department of computer systems at Tallinn University of Technology, Estonia. Aneesh has a master degree in Communication and Multimedia Engineering from Friedrich-Alexander University, Erlangen-Nurnberg, Germany in July 2016 and holds a Bachelor of Engineering in Electronics and Communication Engineering from Anna University of Technology, India. His current research will address today's high-performance designs requirements in term of validation and reliability. The objective of the research is to significantly enhance and develop new statistical, probabilistic methods and algorithms.

Thomas Lange is an Early Stage Researcher in the RESCUE European Training Network. Within the project he is a Research and Development Engineer at iRoC Technologies, France and also Ph.D. Candidate in Computer and Systems Engineering at Politecnico di Torino, Italy. Thomas holds a Bachelor's and Master's degree in Computer Engineering from Technische Universität Berlin. In his research he is investigating the effects of transient faults for high reliability applications in harsh environments. His main interest includes the development of new models and assessment techniques for transient faults, as well as new mitigation and error management techniques with the focus on hardware capabilities.

Maksim Jenihhin is a professor of Computing Systems Reliability at the Department of Computer Systems of Tallinn UT. He holds M.*Sc.* and Ph.D. degrees in Computer Engineering from Tallinn UT (2004 and 2008 respectively). His research interests include methodologies and EDA tools for hardware design, verification and debug as well as nanoelectronics reliability and manufacturing test topics. Maksim is a project coordinator for H2020 RESCUE - Interdependent Challenges of Reliability, Security and Quality in Nanoelectronic Systems Design.

Tara Ghasempouri is a Postdoctoral Researcher at Tallinn University of Technology in Computer Systems department. Her group is mainly focused on three categories such as fault tolerance, verification and safety/security of systems. She is interested in finding innovative solutions for the verification process at the different level of abstraction. Her research topic is also focused on Hardware Security. She received a Ph.D. degree in Computer Science from University of Verona, Italy. During her Ph.D. program, she has researched on different kind of verifications and specially Assertion-based Verification on the embedded system. She is a member of IEEE Computer Society and she was a reviewer for different conferences such as ETS, VLSI-SOC and etc.

Jaan Raik is a professor of digital systems verification at the Department of Computer Systems of Tallinn University of Technology and the leader of the Center for Dependable Computing Systems Design (DCSD). Prof. Raik received his M.*Sc.* and Ph.D. degrees in Computer Engineering from Tallinn University of Technology in 1997 and in 2001, respectively. He is a member of IEEE Computer Society, HiPEAC and a member of steering/program committees of several conferences. He has co-authored more than 200 scientific publications.

Dan Alexandrescu (IEEE Member'07-Senior Member'13) is the CEO of IROC Technologies. Dan holds a M. Eng. in Electronics from Politehnica Bucharest, Romania, a M.A.S. in Microelectronics from Joseph Fourier University, Grenoble, France and a Ph.D in Microelectronics from INPG, Grenoble Institute of Technology, France. He specializes in the design, optimization and improvement of highly-reliable microelectronic circuits. He contributed to the organization of reliability-centric workshops and symposia (Program Co-Chair for multiple IOLTS editions, Finance and General Co-Chair for several SELSE editions) and he prepared many publications in the field of reliability and radiation-induced effects.

Appendix 7

VII

A. Balakrishnan, D. Alexandrescu, M. Jenihhin, T. Lange, and M. Glorieux, “Gate-level graph representation learning: A step towards the improved stuck-at faults analysis,” in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pp. 24–30, 2021

2021 22nd International Symposium on Quality Electronic Design (ISQED) | 978-1-7281-7641-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ISQED51717.2021.9424256

Gate-Level Graph Representation Learning: A Step Towards the Improved Stuck-at Faults Analysis

Aneesh Balakrishnan[1 2], Dan Alexandrescu[1], Maksim Jenihhin[2], Thomas Lange[1 3], Maximilien Glorieux[1]
[1]iRoC Technologies, Grenoble, France
[2]Department of Computer Systems, Tallinn University of Technology, Tallinn, Estonia
[3]Dipartimento di Informatica e Automatica, Politecnico di Torino, Torino, Italy
[1]E-mail: {aneesh.balakrishnan, dan.alexandrescu, thomas.lange, maximilien.glorieux}@iroctech.com
[2]E-mail: maksim.jenihhin@taltech.ee

Abstract—As the circuit implementation predominantly focuses on the higher density and performance with the technology scaling, more adverse types of faults and effects have been investigated by the system designers. Naive and compatible testing approaches are required to apprehend the emerging technological issues, and that will ensure high reliability and quality to the systems' functional behaviors. The unidentified permanent faults, in particular stuck-at faults, have very adverse impacts on the functional quality of circuits under stressful workloads. This paper focuses on the single stuck-at faults simulation and proposing an Artificial-Intelligence (AI) based algorithm for the prediction of Functional Failure Rate (FFR) of each net (netlist wire) for a given set of input patterns. The statistical prediction also provides an improved way of estimating the Functional Fault Coverage (FFC) and the effectiveness of the input test vector. The introduced algorithm is an accelerated methodology which will significantly reduce the time complexity by 60%, while compared to the traditional exhaustive fault-injection approaches. The case study has been conducted with the gate-level circuit of the 10-Gigabit Ethernet MAC.

Index Terms—GraphSAGE (Graph Based Neural Network), Line Graph, Deep Neural Networks, Functional Failure Rate (FFR), Fault Coverage, Single Stuck-at Faults.

I. INTRODUCTION

Advanced high-quality chips endorse challenging scenarios in developing industrial specifications by including standard quality metrics. A lower DPPM (Defective Parts per Million) level is an example of a highly demanded test metric by the industrial end-users. Such requirements imply elaborated production tests that generally include single stuck-at faults, transitional faults, path delays, bridging effects, and cross-talks. The functional fault coverage is applied as a standard metric for expressing the quality of test patterns. To characterize the stuck-at fault coverage, a valid set of test patterns, and an efficient fault simulation approach are needed [1].

The fault diagnosis approaches locate the faults that cause functional failures in the circuit. The prior works [2], [3], [4], and [5] significantly use the single stuck-at faults models to perform the fault diagnosis. The fault simulation method has been chosen in these papers to infer a statical observation by comparing the performance between the faulty circuit and fault-free circuit. The single stuck-at fault diagnosis efforts also extended scientifically to solve the problems of multiple stuck-at faults with limitations. The works that have been explicated in [6], [7], [8], and [9] tried to reduce the fault-space for multiple suck-at fault diagnosis through different concepts. Effect-cause analysis and guided probing belong to such fault space deduction approaches. Electron-beam probing as a fault diagnosis approach was applied in work [10], but electron-beam probing is a time-consuming and laborious task. Research papers [11] and [12] deduced the suspected faults by algorithmically generated sensitizing input pairs without probing internal nets. The works that have been provided in [13], [14], and [15], tried to utilize the stuck-at fault model for the analysis of bridging faults. Also, in work [16], both single and multiple fault simulations were applied together for the fault space reduction technique to ease the problems of fault diagnosis. In all these proposed approaches, fault simulations and its running time are inevitable factors. The single stuck-at fault model and its simulation within their premises have been explored in fault diagnosis with different perceptions.

The fault-space reduction algorithms in the cited works depend heavily on the stuck-at fault simulations. As a result, the aggressive chip density scaling jeopardized the simulation-based fault diagnosis methodologies. All these facts lead the researchers in the reliability and testing fields to develop mathematical algorithms to reduce the overhead of fault-simulations. Also, in the last decade of years, there is no much sophisticated-literature that combines the simple stuck-at model and advanced mathematical algorithms for fast inference of functional failures due to single stuck-at faults. All these key factors contribute to the proposed Artificial Intelligence (AI) based framework.

Application of Artificial Intelligence (AI) to extract feature information from the probabilistic network domain [17], has emerged as a prominent tool for graphical node embedding. The process of leveraging a node's features into a vector form is called the node embedding. The applications of graph-based neural network algorithms (GCN, DNN, and graphSAGE) for circuit's reliability modeling, have been proposed in papers [18], [19], and [20], respectively. The papers [18] and [19] proposed algorithms that were used for the evaluation of the propagation probability of Single Event Upsets (SEUs) in a transductive way. On the other hand, the paper [20] implemented an inductive type framework. The transductive model is not exactly building a predictive model. A completely

new unseen data point (a net in the circuit netlist here) force the transductive algorithm to re-run the training phase. But inductive learning builds a predictive model that can also apply to the unseen data. The prior work [20] implemented the scientific workflow, which constitutes of graphSAGE and Deep Neural Network (DNN) algorithms for evaluating the soft-error effects [21] on the functional level. Additional to the works in [20], this paper provides:

1) An adapted framework that introduces an edge-to-vertex graph transformation principle to analyze the effects of permanent faults instead of soft-errors.
2) An algorithm that illustrates: how to predict the propagation of induced stuck-at-1 and stuck-at-0 faults at each net (wire in a netlist).
3) Better numerical superiority in the fault propagation probability predictions and interestingly reduces the time complexity by 60%.
4) The characterized failure vulnerabilities result in the reduced set of fault locations for fault diagnosis.

The transformed gate-level graph is the source for extracting the features of nets to model the effects of stuck-at faults at the functional level. The proposed framework goes beyond the classical machine learning algorithms (like support vector machine, logistic regression, and linear regression) by replacing the black-box modeling with transparent modeling of the metrics (i.e., white-box modeling [22]). In white-box modeling, machine learning algorithms have been adapted to interpret or explain the results associated with the model.

The rest of the paper organized as follows: sections II, III, IV, V, VI and VII. Section II covers the backgrounds of Artificial Intelligence (AI) models and frameworks. The application of graph theory and the graphSAGE algorithm as methodologies is briefed in section III. The workflow of the framework is chronicled in section IV. Section V is dedicated to the results and discussions. A conclusion to the holistic approaches is provided in section VI. Finally, the work is acknowledged in section VII.

II. Background: Deep Graph Learning

A. Graph Theory

A gate-level circuit can be transformed into a probabilistic graph network where vertices (ν) analogous to the flip-flops and gates, and the directed edges (ε) represent the connection between them. The mathematical graph-function G of the transformed netlist is given as:

$$G = (\nu, \varepsilon) \tag{1}$$

B. Graph Transformation Principle

The line graph LG is an edge-to-vertex (or) edge-to-node dual representation of graph G. The graph transformation is obtained by associating the vertex of LG with the edge of the graph G. The two vertices of LG are connected with an edge if and only if the corresponding G-edges of that two vertices have a common vertex [23]. A simple and classical example of graph transformation of G into a line graph LG is illustrated in Fig.1. When G represents a gate-level circuit as in (1), then LG epitomizes the dual form of the gate-level, where nets are the nodes.

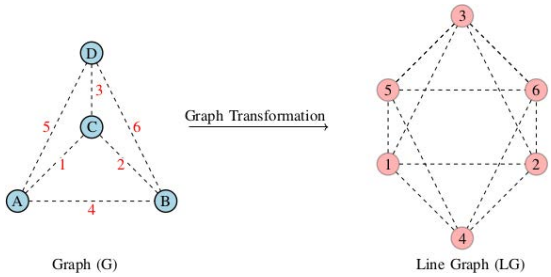


Figure 1: Edge-to-vertex transformation of a graph (G)

C. GraphSAGE

The graphSAGE [24], is a general inductive framework that leverages the node's feature information into efficient node embedding. An inductive node embedding symbolizes an optimized generalization across the graph with the same form of features. Therefore, we can leverage the node features of the unseen graph part of a circuit by the embedding generator that trained once. The graphSAGE is defined as a graph-based neural network with sampler and aggregator functions. Here, we have implemented a pooling aggregator with the help of a Python neural network libraries. This embedding provides the local-role of nodes in the graph as well as their global positions also. The basic idea of the graphSAGE was simplified and explained in the workflow.

D. Deep Neural Network

Deep Neural Networks (DNNs) [25], [26] are trying to model complex distributions of data by combining different non-linear transformations. Here, DNN is the model for the binomial observations P(k|e,N,w_i). After DNN training, the generated hypothesis predicts the failures.

III. Methodology

Fig. 2 provides a visualization of different algorithms that were placed to model the error-observation probability at the functional level. The graphSAGE algorithm is the introduced graph embedding neural network that deduces the hidden information corresponding to a net (w_i) from the netlist, where the netlist is a probabilistic graph of nets and components as vertices and edges. The explicated information about a net (w_i) is the structural peculiarities that decide the chance of an injected fault at w_i to cause functional failures. Simultaneously, the injected fault at w_i for each input vector leads to a binomial type failure probability distribution [27]. P(k|e,N,w_i) explicitly models that failure distribution for N input vectors, where k represents total failure outcomes (e) in number for N injected stuck-at faults at w_i. The whole modelling-approch in Fig. 2 can be explained as the convolution ($\circledast$) between $f(N\text{vectors})$ and $f(\text{faulty netlist}, w_i)$, and that convolution is proportional to P(k|e,n,w_i) as given in (2).

$$f(N\text{vectors}) \circledast f(\text{faulty netlist}, w_i) \propto P(k|e, n, w_1) \tag{2}$$

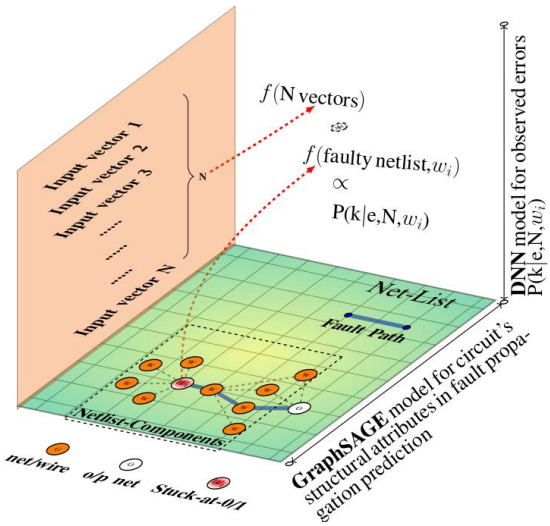


Figure 2: Modeling of functional failure probability

IV. Algorithmic Workflow

Fig. 3 chronicled the whole research approach in block-diagrams, and it includes two parts: the fault-injection campaign and the research approach. In the first part (fault-injection campaign), the raw database (FFRs) for the fault propagation analysis is generated. The Fault-Injection campaign and its details are acknowledged below. In the second subsection, more detailed view of the AI model is provided.

A. *Fault-Injection Simulation*

At the gate-level, the functional failure modeling of stuck-at faults is further advancing to accelerate the failure evaluation effectively. We have statistically simulated the fault injections and, empirically derived the Functional Failure Rate (FFR) factor of each net (gate-level wire) for stuck-at-1 and suck-at-0 faults. Finally, the fault coverage of the given set of 64 input test vectors (64 transmitted packets) is calculated from the observed results of fault injection campaign. The proposed algorithm is used to predict the empirically derived metrics. In this scenario, stuck-at-1 and stuck-at-0 fault injections at gate-level have been achieved through modifying the data of each net to logic-1 and logic-0, respectively. The injected value acts as a permanent fault for the entire operation. In total, 3437 nets from different blocks of the circuit (such as TX, RX, Wishbone Interface, Fault State-machine, and Sync_clk), were tested. A high-level representation of the campaign has been shown in Fig. 3. Equations (3) and (4) express the Functional Failure Rate factor of each net with respect to stuck-at-0 and stuck-at-1 fault, respectively.

$$FFR_{i,SA_0} = \frac{N_{FF_0}}{NF_{SA_0}} \tag{3}$$

$$FFR_{i,SA_1} = \frac{N_{FF_1}}{NF_{SA_1}} \tag{4}$$

In (3) and (4), i indicates each net. N_{FF_0} (or) N_{FF_1} represent the number of functional failures due to stuck-at-0 (or) stuck-at-1 faults. Similarly, NF_{SA_0} and NF_{SA_1} counts the maximum number of injected stuck-at-0 and stuck-at-1 faults per net. The maximum of NF_{SA_0} (or) NF_{SA_1} is simply equivalent to the number of input test patterns (i.e., 64 packets). The significance in evaluating the fault coverage of the test patterns is inevitable [28]. The equation (5) defines the Functional Fault Coverage (FFC) metric.

$$FFC = \frac{\text{Total Faults Detected as Functional Failures}}{\text{Total Injected Faults}} \tag{5}$$

The FFC is a metric that decides the quality of test pattern in producing a prominent subset of all possible functional failures without concerning about 100% possible failures. With the advent of small-scale technology integration in the electronics-chips, a more reliable test pattern according to single stuck-at fault models becomes a desideratum. In this experiment, the quality of the algorithm in estimating the FFC metric is also evaluated.

B. *Research Approach: The AI Model*

In the second part (Research Approach), the graph theory and Deep Learning (DL) are applied to estimate the fault-injection inference. The research approach fundamentally exploits FI database to characterize the fault propagation model. The research approach part can be viewed as four phases.

*1) **Phase I: The Probabilistic Network**:* After the fault-injection campaign, a probabilistic graph G is generated, which represents the gate-level netlist. A Verilog Procedural Interface (VPI) library function was linked to a standard simulation tool (ModelSim) to design the graph G. The graph G includes vertices that imply the gates and flip-flops in the netlist. Then, an edge-to-vertex transformation was applied to graph G. The resulting line graph LG contains the vertices that analogous to the nets/wires in the netlist. Fig. 4 shows those steps in order. The line graph LG was used in the successive algorithm (graphSAGE) to generate the feature matrix of nets for a downstream DNN prediction.

*2) **Phase II: GraphSAGE**:* In the second phase, the grpahSAGE algorithm extracts a feature matrix (X) corresponding to the line graph LG nodes. The graphSAGE includes two principal steps. The first step is the sampler algorithm. The sampler algorithm defines the neighborhood space of a source node. In this scenario, we defined the parameter $K = 2$, which means that the sampler samples up to the depth of the 2 neighbor space. In the second step of the graphSAGE algorithm, an aggregator is implemented at each depth ($1 \leq k \leq K$), that aggregates features from the local neighborhood of the source node. The aggregators were visualized in Fig. 5, where blue and green lines indicate the aggregators at depth $k = 1$ and $k = 2$ respectively. Here, the max-pooling aggregators were implemented. Equation (6) formulates the mathematical abstraction of the max-pooling aggregator [24].

$$AGGRE_k^{pool} = max(\{\sigma(W_{pool} h_{u_i}^k + b), \forall u_i \in N_k(v)\}) \tag{6}$$

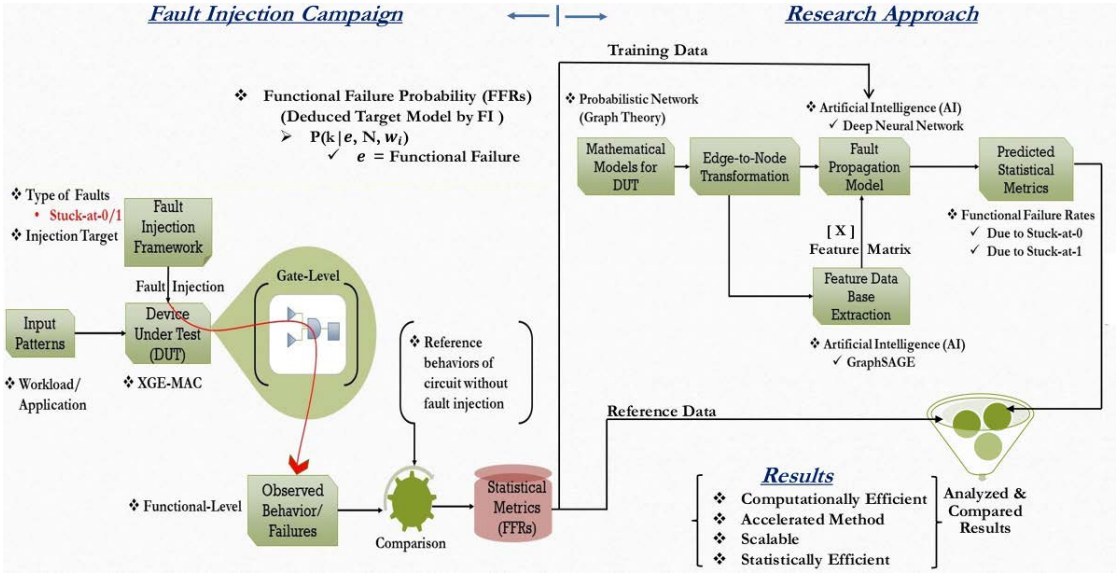


Figure 3: A Systematic workflow diagram

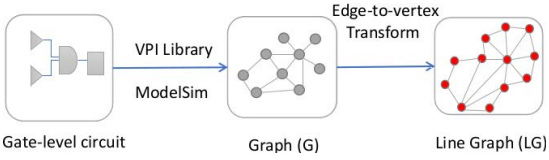


Figure 4: Development of a line graph from a netlist

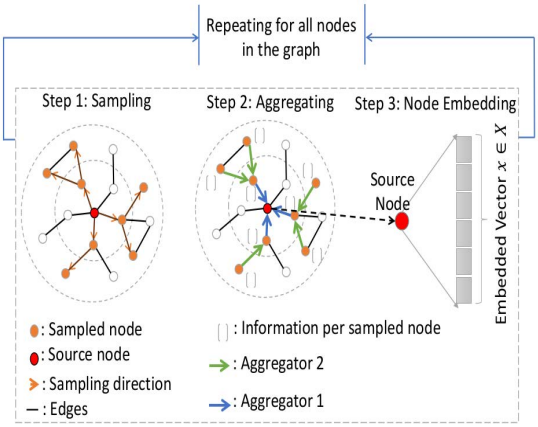


Figure 5: GraphSAGE Algorithm

where (6) represents the aggregator function at depth k, which is a graph neural network with parameters W_{pool} and b. An unsupervised learning method has optimized those parameters. $N_k(v)$ represents k-neighborhood of vertex v and $h^k_{u_i}$ indicates the aggregated neighborhood vector and, σ is the activation function of the neural network.

In the third step, each node is reformed into a vector. All three steps repeat for all the nodes in the line graph and produce a matrix representation (X) of the circuit. Fig. 5 illustrates the node-to-vector mapping (node embedding).

*3) **Phase III: DNN**:* Fig. 6 outlines the DNN algorithm that was exercised for prediction purposes. There are two parts in Fig. 6. The first part is the training part of DNN, and the second one is the testing part of DNN. In the training part, 40 % from the feature database X and corresponding target probability metric from FI-database have been randomly selected as x_{train} and y_{train}. The chosen resources postulate a hypothesis that best describes the target probability distributions (FFR_{i,SA_0} and FFR_{i,SA_1}) through the supervised learning method. The optimized parameters (Weights and Bias) of the best-fit provided the model parameters. In the testing part, the proposed model is applied to an unknown input vector from x_{test} and predicts the target probability metric (FFRs) as provided in Fig. 6. The DNN architecture consists of 5 dense layers, including the input and the output layers. The input layer consists of 50 nodes. The input vector to DNN is a 50-dimensional vector per net, as provided by the graphSAGE algorithm. The hidden layers are fully connected feed-forward neural networks, each having nodes 64 (1^{st}-hidden), 32 (2^{nd}-hidden), and 12 (3^{rd}-hidden) respectively. The output layer has only one node that performs the prediction. A Rectified Linear-Unit (ReLU) is the activation function that is benefitted in the last layer for prediction while the hidden layers take the advantage of the Hyperbolic Tangent (Tanh) as the activation function. Adam [29] is an adaptive learning rate optimization

algorithm that has been used here as an optimizer, and finally, the loss function is updated with the Mean Squared Error (MSE) function.

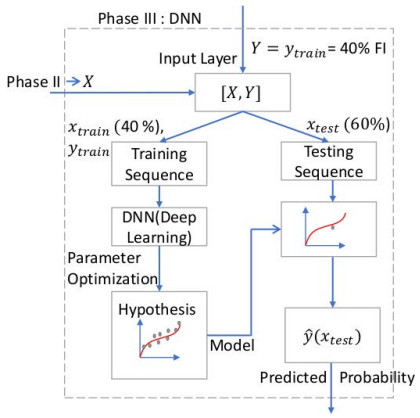


Figure 6: DNN Algorithm

*4) **Phase IV: Result Analysis**:* The final phase includes a comparison between the predicted and reference probability metrics, as given in Fig. 3. The compared results are plotted in Fig. 7, 8, 9, and 10.

V. Case Study and Results

The case study experiment was conducted with the gate-level circuit of the 10-Gigabit Ethernet MAC [30]. Among the synthesized nets, the simulation-campaign injected faults on 3437 nets, where avoided nets are redundant in their functions. The redundant nets includes the parallel connections also. The overall simulation paradigm was summarized in Fig. 3. Fig. 7 and 8 graphically represent the functional failure rates of nets due to stuck-at-1 and stuck-at-0 faults, respectively. The red color legend (in Fig. 7 and 8) represents reference values, and the blue color legend represents corresponding predicted probabilities. Both graphs were obtained for 40% random training size, which explicitly indicates that only 60% (2062 nets) of total nets (3437) are plotted in Fig. 7 and 8. The experiments with different training sizes have led to the conclusion that the training size is better to be 40% for both predictions. The 40% training size provides a significantly higher correlation between predicted and reference values, compared to other training sizes of less than 40%. The correlation property is slightly improving with above 40% training size. Such observations are plotted in Fig. 9 and 10. The graphical visualizations in Fig. 7 and 8 demonstrate how well the prediction replicates the observed reference values. In both cases, the regressions have achieved adequate values for the coefficient-of-determination (R^2) metric (approximately 0.93 for stuck-at-1 and 0.94 for stuck-at-0). Generally, the best model fit value of R^2 metric is 1, and the worst value is 0. In statistics, the R-squared (R^2) [31] value is the measure of goodness-of-fit of a regression model.

From Fig. 7 and 8, we can characterize the nets based on their functional-failure vulnerabilities with an acceptable prediction error. When the failure rates of nets numerically close to 1, then nets are more critically characterized. This information provides a more reliable approach for minimizing the cluster size of critical fault-locations in the fault diagnosis. The implemented algorithm depends on 40% of the fault-injection database. However, it is quite impressive to note that the test and training phase of the whole algorithm takes only less than 10 minutes. The holistic algorithm (graphSAGE + DNN) can process a netlist of 10,000 components without any additional running time on a CPU machine of 16GB RAM. The trade-off between the netlist size and the processing time will be investigated in the future with further experiments.

Table I provides the impacts of prediction in terms of required time and also provides the Functional Fault Coverage (FFC) metric (5) of test patterns and its corresponding predicted value. The FFC value is predominant in the circuit diagnosis to evaluate the effectiveness of the test pattern. The fault coverage close to 90% will adequately explain the observability of single stuck-at faults. Note, here, the test patterns/packets were chosen randomly for this experiment, and were not optimized for the best. The low FFC value (27.5%) of the given test patterns is approximately predicted by the framework (28.2%). If fault-injection simulation chooses the test pattern with a higher FFC value, then the probabilistic failure observation by fault-injection per net will also increase. This increased failure observation provides more information to the algorithm and subsequently helps the algorithm to estimate the stuck-at fault failure probability as discussed in Fig. 7 and Fig. 8 with high accuracy. The random test patterns are chosen to model the real-time scenarios where it is not possible to expect the high possibility of large fault-coverage inputs to the device under test.

TABLE I:
IMPACTS OF PREDICTION IN SIMULATION RESOURCES
(FAULT-INJECTION (FI) CAMPAIGN - 7 MODELSIM)

Parameters	Stuck-at-1	Stuck-at-0
R^2	0.93	0.94
Training & prediction	$<$ 10 mins	$<$ 10 mins
Training Data (40%)	$\approx$ 1.2 hrs	$\approx$ 1.2 hrs
Full Database (FI)	$\approx$ 3 hrs	$\approx$ 3 hrs
	FFC Calculation	
Empirically calculated FFC	= 0.275	
Predicted FFC	= 0.282	

VI. Conclusion

The paper has validated a naive artificial intelligence approach to predict the functional failure vulnerability of each net at the gate-level, as well as the effectiveness of the test patterns. The proposed technique succeeds in reducing the time-complexity of exhaustive fault-injection by 60%. The developed framework depends on the gate-level circuit, and the extracted features have been used for the prediction of realistic fault propagation probabilities. The developed method combines a single stuck-at model and advanced mathematical algorithms, which will able to cluster the nets into critical and non-critical groups depending on the applications. The

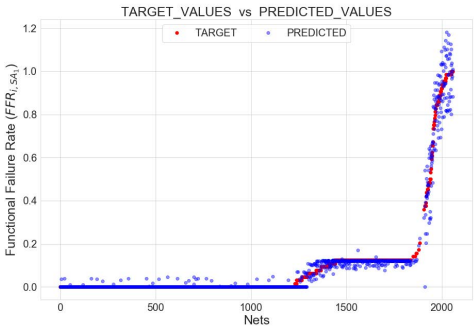


Figure 7: Stuck-at-1: R^2 = 0.93 (40% training size)

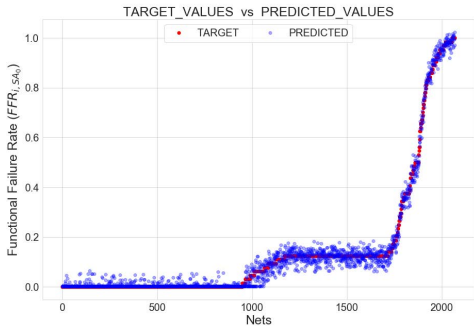


Figure 8: Stuck-at-0: R^2 = 0.94 (40% training size)

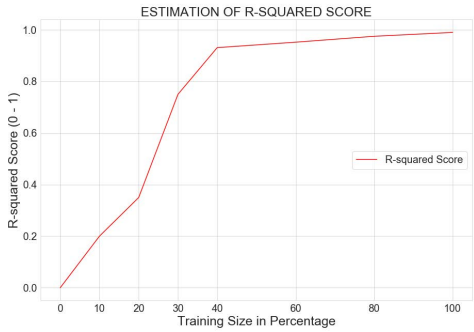


Figure 9: Stuck-at-1: R^2 and training size(%) relation

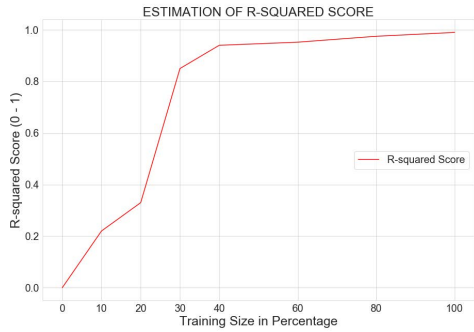


Figure 10: Stuck-at-0: R^2 and training size(%) relation

results open a new possible dimension to a scalable and very cost-effective simulation tool for the production of dependable micro-electronics systems.

VII. Acknowledgements

This work was supported by the RESCUE ETN project. The RESCUE ETN project has received funding from the European Union's Horizon 2020 Programme under the Marie Skłodowska-Curie actions for research, technological development and demonstration, under grant No. 722325.

References

[1] A. Veneris and I. N. Hajj, "Design error diagnosis and correction via test vector simulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 12, pp. 1803–1816, 1999.

[2] S. Venkataraman, W. K. Fuchs, and J. H. Patel, "Diagnostic simulation of sequential circuits using fault sampling," in *Proceedings Eleventh International Conference on VLSI Design*, 1998, pp. 476–481.

[3] P. G. Ryan, S. Rawat, and W. K. Fuchs, "Two-stage fault location," in *1991, Proceedings. International Test Conference*, 1991, pp. 963–.

[4] J. A. Waicukauski and E. Lindbloom, "Failure diagnosis of structured vlsi," *IEEE Design Test of Computers*, vol. 6, no. 4, pp. 49–60, 1989.

[5] J. A. Waicukauski, V. P. Gupta, and S. T. Patel, "Diagnosis of bist failures by ppsfp simulation," in *18th IEEE International Test Conference*, Sep.1987, p. 480–484.

[6] M. Abramovici, "A maximal resolution guided-probe testing algorithm," in *Proceedings of the 18th Design Automation Conference*, ser. DAC '81. IEEE Press, 1981, p. 189–195.

[7] Abramovici and Breuer, "Multiple fault diagnosis in combinational circuits based on an effect-cause analysis," *IEEE Transactions on Computers*, vol. C-29, no. 6, pp. 451–460, 1980.

[8] N. Kuji and T. Tamama, "Integrating an electron-beam system into vlsi fault diagnosis," *IEEE Design & Test of Computers*, vol. 3, no. 04, pp. 23–29, jul 1986.

[9] T. Yamada, S. Hamada, T. Matsumoto, T. Takahashi, and T. Nakayama, "Method of diagnosing multiple stuck-at-faults in combinational circuits," *Systems and Computers in Japan*, vol. 22, no. 11, pp. 21–30, 1991. [Online]. Available: https://doi.org/10.1002/scj.4690221103

[10] N. Itazaki, T. Sumioka, S. Kajihara, and K. Kinoshita, "Automatic fault location using e-beam and lsi testers," in *Proceedings of 1993 IEEE 2nd Asian Test Symposium (ATS)*, 1993, pp. 255–260.

[11] H. Takahashi, N. Yanagida, and Y. Takamatsu, "Multiple stuck-at fault diagnosis in combinational circuits based on restricted single sensitized paths," in *Proceedings of 1993 IEEE 2nd Asian Test Symposium (ATS)*, 1993, pp. 185–190.

[12] N. Yanagida, H. Takahashi, and Y. Takamatsu, "Multiple fault diagnosis by sensitizing input pairs," *IEEE Design Test of*

Computers, vol. 12, no. 3, pp. 44–, 1995.

[13] S. Millman, E. McCluskey, and J. Acken, "Diagnosing cmos bridging faults with stuck-at fault dictionaries," in *1990 International Test Conference*. Los Alamitos, CA, USA: IEEE Computer Society, sep 1990. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TEST.1990.114104

[14] I. Pomeranz and S. M. Reddy, "Location of stuck-at faults and bridging faults based on circuit partitioning," *IEEE Transactions on Computers*, vol. 47, no. 10, pp. 1124–1135, 1998.

[15] Jue Wu and E. M. Rudnick, "Bridge fault diagnosis using stuck-at fault simulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 4, pp. 489–495, 2000.

[16] H. Takahashi, K. O. Boateng, K. K. Saluja, and Y. Takamatsu, "On diagnosing multiple stuck-at faults using multiple and single fault simulation in combinational circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 3, pp. 362–368, 2002.

[17] J. Pearl, "Chapter 3 - markov and bayesian networks: Two graphical representations of probabilistic knowledge," in *Probabilistic Reasoning in Intelligent Systems*, J. Pearl, Ed. San Francisco (CA): Morgan Kaufmann, 1988, pp. 77 – 141.

[18] A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, "Modeling gate-level abstraction hierarchy using graph convolutional neural networks to predict functional derating factors," in *2019 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2019, pp. 72–78.

[19] A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, "The validation of graph model-based, gate level low-dimensional feature data for machine learning applications," in *2019 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, 2019, pp. 1–7.

[20] A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, "Composing graph theory and deep neural networks to evaluate seu type soft error effects," in *2020 9th Mediterranean Conference on Embedded Computing (MECO)*, 2020, pp. 1–5.

[21] M. Nicolaidis, *Soft Errors in Modern Electronic Systems*. Springer, Boston, MA, 2011.

[22] O. Loyola-González, "Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view," *IEEE Access*, vol. 7, pp. 154 096–154 113, 2019.

[23] J. L. Gross and J. Yellen, *Graph Theory and Its Applications, 2nd ed.* Boca Raton, FL: CRC Press, 2006.

[24] W. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 1024–1034. [Online]. Available: http://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs.pdf

[25] R. J. Schalkoff, *Artificial Neural Networks*, 1st ed. McGraw-Hill Higher Education, 1997.

[26] R. Hecht-Nielsen, "Neurocomputing: picking the human brain," *IEEE Spectrum*, vol. 25, no. 3, pp. 36–41, 1988.

[27] Springer, *Binomial Distribution*. New York, NY: Springer, 2008, pp. 44–45.

[28] J. Schat, "On the relationship between stuck-at fault coverage and transition fault coverage," in *2009 Design, Automation Test in Europe Conference Exhibition*, 2009, pp. 1218–1221.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[30] A. Tanguay. (2008) 10ge mac core specification. [Online]. Available: https://opencores.org/websvn/filedetails?repname=xge_mac&path=%2Fxge_mac%2Ftrunk%2Fdoc%2Fxge_mac_spec.pdf

[31] Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Coefficient_of_determination

Appendix 8

VIII

D. Alexandrescu, A. Balakrishnan, T. Lange, and M. Glorieux, “Enabling cross-layer reliability and functional safety assessment through ML-based compact models,” in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, IEEE, jul 2020

26th IEEE International Symposium on On-Line Testing and Robust System Design (IOLTS)

Enabling Cross-Layer Reliability and Functional Safety Assessment Through ML-Based Compact Models

Dan Alexandrescu
IROC Technologies
Grenoble, France
dan@iroctech.com

Aneesh Balakrishnan
IROC Technologies,
Grenoble, France
Tallinn University of Technology, Tallinn 12618, Estonia
aneesh.balakrishnan@iroctech.com

Thomas Lange
IROC Technologies
Grenoble, France
Dipartimento di Informatica e Automatica, Politecnico di Torino, Torino, Italy
thomas.lange@iroctech.com

Maximilien Glorieux
IROC Technologies
Grenoble, France
maximilien.glorieux@iroctech.com

***Abstract*—Typical design flows are hierarchical and rely on assembling many individual technology elements from standard cells to complete boards. Providers use compact models to provide simplified views of their products to their users. Designers group simpler elements in more complex structures and have to manage the corresponding propagation of reliability and functional safety information through the hierarchy of the system, accompanied by the obvious problems of IP confidentiality, possibility of reverse engineering and so on. This paper proposes a machine-learning-based approach to integrate the many individual models of a subsystem's elements in a single compact model that can be re-used and assembled further up in the hierarchy. The compact models provide consistency, accuracy and confidentiality, allowing technology, IP, component, sub-system or system providers to accompany their offering with high-quality reliability and functional safety compact models that can be safely and accurately consumed by their users.**

Keywords—reliability, functional safety, machine learning, fault model, transient faults, soft errors

I. INTRODUCTION

High quality, reliable and safe electronics requires massive cooperation and exchanges across all the partners of the supply, design and manufacturing flow. A huge quantity of information, addressing functional and extra-functional qualities must be produced accurately, exchanged without loss of fidelity and consumed as intended.

One of the most self-evident examples of such flow of information is the typical foundry>designer>integrator process. The technology provider prepares a complex Process Design Kit that includes technological data, simulation models, design rules information, primitives and possible standard cell libraries from the foundry or a library vendor. Designers make use of this information during the preparation of cells, IP blocks and ultimately, components. Components are used by system integrators on boards, sub-systems and systems.

Design paradigms, workflows, practices and expectations are progressing continuously. While in the past a typical ASIC design process was concerned by the paramount trinity of area/frequency/power, todays requirements are formulated as a vast set of functional and extra-functional specifications. Functional Safety (FuSa) and Reliability (Rel) requirements are increasingly present because of current or upcoming formal standards such as IEC 61508 [1], ISO 26262 [2] and others. These standards demand the calculation and presentation of functional safety and reliability metrics at system-level, in quantitative and qualitative terms. However, computing the Failure Rate of a system to make sure that it fulfils the <10 FITs requirement for an ASIL D product involves data that has been produced by at least three individual entities and transformed by tens of engineers (or even companies) during the design flow. There is a huge potential for loss of fidelity, data misuse, omissions and translation errors. In addition, much of this information is highly proprietary and the transmission of the information from one partner of another can be parasitized by restrictions and disclosure limitations. Some of the data can also facilitate possible reverse engineering or at least disclose critical design information that was intended to be kept secret.

In this paper we present a uniform methodology to evaluate reliability and functional safety metrics based on using compact models build with Machine-Learning (ML) methods. The compact models of a design element (standard cell, IP, block, component, sub-system or system) at a given hierarchical level can be combined through ML techniques in a single compact model that can be used for the next design stage or user. This approach ensures that relevant parameters are retained and impactful during the design flow and that the contribution of the various design elements is well represented at any calculation stage. Additionally, it may provide a way to obscure technology and design information, safeguarding critical IP, while still equipping the users at any design stage with information.

The paper is organized as follows: Section II introduces the motivation for this work, Section III presents the proposed approach, Section IV shows a worked example and it's followed by the Conclusion section.

This work was done as part of the RESCUE project that has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 722325

978-1-7281-8187-5/20/$31.00 ©2020 IEEE

II. MOTIVATION

As an example of a metric calculation flow, let's consider the evaluation of Single Events Effects (or Soft Errors or Transient Faults) to the failure rate of a system:

Firstly, the technology provider (foundry), in a possible collaboration with a library vendor must provide technological (raw) event rate for the various technology elements (standard cells, memory blocks, analog IP such as PLLs and so on) that its customer uses. Sequential cells can be affected by Single Event Upsets (SEU) with rates that depend on the cell state, voltage, temperature and so on. Single Event Transients in combinatorial cells can also depend on the cell fanout. Single or Multiple Bit/Cell Upsets (SBU/MCU/MBU) can impact the data stored in memory blocks and their occurrence rate can depend on physical implementation (aspect, column muxing, scrambling) or design choices such as error management schemes. Ideally, all this data deserves to be captured in a detailed, high fidelity format that accurately reproduces the error rate of the researched event according to a reasonable set of parameters. There is massive precedence for this approach: today's PDK contain large databases, multi-index tables and process information in a variety of formats. While this may happen one day, the current State-Of-The-Art in terms of SEE technological information delivery consist in PDF test reports or a summary spreadsheet.

The designer must use this information as an input in the event rate calculation methodology. As an example, the unit SEU rates indicated by the technology provider must be annotated to each design flip-flop and de-rated according to the role of the flip-flop (SEQ – sequential element) in the design:

$$SEQ\ Fail\ Rate = \sum_{Flip-Flop} Raw\ SEU\ Rate \cdot \prod_{DR} De-Rating \quad (1)$$

Memory SBU/MCU data is tailored according to each instance specifics and the impact of possible error management schemes is integrated.

$$MEM\ Fail\ Rate = \sum_{Memories} \begin{cases} Raw\ SBU\ Rate \cdot Size\ if\ no\ ECC \\ Raw\ MBU\ rate \cdot Size\ if\ ECC \end{cases} \quad (2)$$

The end goal is to calculate at the IP or design level the required overall reliability or functional safety metrics. While in some rare circumstances, the final deliverable is a number or a limited set of numbers, any actual high-level metric will be dependent on a variety of parameters. Firstly, some parameters (such as voltage, temperature) inherited from the underlying technological layer will certainly impact the transient fault error rate. Design settings (speed, number of active cores, buffer or cache sizes, memory modes) can be set according to each application requirements, affecting the actual error rate. Internal features (error detection and/or correction, safety mechanisms) can be activated or not with a direct impact on the design reliability.

The same design can have multiple physical implementation that can also impact reliability. As an example, the usage of packaging materials with different alpha emissivity rates will strongly impact (from x1 to x1000) the final SEU rate.

In conclusion, the set of parameters affecting the FuSa/Rel metrics of a design can be large and impactful. The manufacturer will usually have difficulties in packaging and transmitting this information to the user. There are several options:

- A maximum, worst-case metric. This way, the actual failure rate is guaranteed to be lower than the indicated value. This approach can penalize some applications or systems and can lead to overengineering in trying to manage the failure rate.
- A recommended utilization scenario that leads to a nominal, favorable error rate. This scenario is usually implemented through a "Safety Manual" where the manufacturer describes his vision on how the component should be used in order to fulfil safety goals and to lead to the desired failure rate
- An analytical method where the provider can explicitly describe a function to calculate the actual failure rate according to the implementation. As an example, assuming that the system includes memory instances, then their contribution to the overall failure rate can be described similarly to the equation (2). However, this approach presents the drawbacks of disclosing to the user in a clear format, internal circuit information or raw technology data which can be critical pieces of IP.

A system integrator will have to integrate many individual components from various providers. Translating, adapting and integrating various reliability data presented in dissimilar formats is very challenging. Regardless of these hurdles, the system integrator will have to deliver a final product with a clear set of metrics that can be compared to the requirements. However, even the system reliability metrics can depend on various usage scenarios. The same exact product can show different failure rates when used in a datacenter, a car, an airplane or a satellite. The provider needs to express the metrics of its products according to a set of parameters and has to provide the user with a metric model with a set of parameters that are relevant for the given abstraction level.

The various collaborations between the providers and users of technology, components and systems need to be supported by tools specifically developed for reliability and functional safety efforts. While more specific tools exist, standard spreadsheet tools are widely deployed and used. Data import and export from these tools is inadequate and prone to errors.

Recent standardization efforts from IEEE [3] and Accellera [4] aims at defining an exchangeable interoperability format for functional safety analysis and functional safety verification activities. A format for Reliability exchanges has been proposed previously [5], providing a solution to suppliers and consumers to exchange reliability information in a consistent fashion and to use this information to construct accurate reliability models. All these efforts stress the importance of building models hierarchically and according to the design abstraction level. Accordingly, an uniform, universal method to model functional safety or reliability metrics at any design stage or abstraction level and more importantly, combining the information available at the current level to benefit the next efforts in the pipeline would be opportune and useful.

Using Machine Learning techniques to build such a model has several advantages. Machine Learning algorithms are

known to efficiently learn even complex relationships. Models can be built from various types of input data, such as tables or functions. Therefore, it is suitable at any abstraction and hierarchical level. Several aspects can be combined in a single model, which makes it compact and easy to use by the user in the next design stage. Additionally, the representation of the Machine Learning model can be fundamentally different to the original representation and thus, obscure critical technology and design information.

III. Proposed Methodology

The proposed approach is conceptually straightforward and easy to implement. The methodology relies on compact models build using Machine-Learning techniques and trained using reference data provided by State-of-the-Art (SOTA) methods. The overall FuSa/Rel metrics for the current level can be then calculated and used as reference data to train a compact ML model. The elaborated ML model can then fuel the analysis efforts required for the next design stage or upper hierarchical level or abstraction and can be shared with the corresponding engineer or user.

The phases of this methodology are described below:

Phase I – Data collection

Let's assume that the current hierarchical level or design abstraction is a collection of elements (or components). Each element has one or multiple attached FuSa/Rel metrics and models. The models can be analytical, data or ML. Each individual metric can have a collection of input parameters:

$$Metric_i = f\left(par_{i,0}, par_{i,1}, par_{i,2}, \quad \ldots par_{i,n(i)}\right) \tag{3}$$

Phase II - Integration

At the current level, SOTA approaches will allow us to combine the existing data of the various elements in overall, top-level FuSa/Rel metrics. This metric will depend on all of the parameters of the element metrics, including options, parameters and choices that are applicable at this design level. Obviously, the parameter list can be simplified by optimizing the parameters that are the same or equivalent. As an example, a supply voltage parameter can be applicable to several elements.

$$Metric_{overall} = \sum_i Metric_i = f(par_0, par_1, par_2, \ldots par_m) \tag{4}$$

Phase III – Compact ML Model Elaboration

The equations that composes the overall metric can be then exercised over the validity range of the parameters. The overall metric can also be a collection of data valid over a specific range. The collected data will be then used to train a ML model that will accept as inputs the aggregated set of parameters and will extend to the area covered by the available dataset. Once trained, the ML model is expected to have good if not perfect accuracy over the valid range and is an ideal surrogate or replacement for the discrete overall metric.

The main goal of the training is to create a model which accurately represents the given metrics function or collection of data. The model should be able to accurately recall the trained values and depending on the learned metric, the model should also be able interpolate and extrapolate the data.

In comparison to classical Machine Learning applications this approach is a bit different. Usually the input parameter range is limited and known and for most cases more training data can be generated or gathered. In this way the data to train the model can be increased to improve the accuracy until a specified target is met.

Phase IV – Packaging and Reuse

The elaborated ML model can be then provided to the next engineer or user that can start applying this methodology on the next hierarchical level, flow stage or design abstraction level.

The presented methodology shows distinct advantages. The approach is uniform regardless the location in the design flow or design hierarchy. The ML models are compact, the training is not computationally intensive and implementations are available in a variety of language and programming frameworks. In many cases the trained ML model will hide or obscure sensitive design or technology information.

IV. Worked Example

In this example, we are addressing the calculation of a system Transient Fault / Soft Error Rate. For the purpose of the demonstration, we will use simple, naive equations for modeling the error rates and aggregating the contribution of the various elements. Firstly, let's introduce the following functions for the calculation of the various Soft Error Rates (indicated in FITs/MegaBit or MegaCell):

$$RawSER_{Seq}(V_{dd}) = 100 \cdot \left(1 + (1.2 - V_{dd})\right) \tag{5}$$

(A Flip-Flop has a 100 FITs/Mb at the nominal 1.2V supply voltage; SER decreases at higher voltages and increases at lower). Vdd means supply voltage

$$RawSER_{Comb}(V_{dd}, PW) = 50 \cdot (1 + (1.2 - V_{dd})) \cdot \frac{50}{PW} \tag{6}$$

(A combinatorial cell can exhibit a spectrum of Single Event Transients with decreasing event rate for larger, longer events. We will limit the minimal Pulse Width - PW at 10ps which is the shortest transient that the selected technological process can propagate). PW means Pulse Width (in ps).

The Single Event Effect rate for natural working environments is dominated by the contribution of SEEs caused by **alpha** particles emitted by impurities in the packaging materials and **neutrons** caused by the interaction of high-energy particles with the atmosphere. Both contributions depend on a large number of parameters and it can be difficult to provide a model that integrates the effect of all the parameters.

As an example, the **neutron**-induced SEE rate depends on many factors, including physical location, altitude, solar activity, shielding and so on. Following the approach from [7], let's focus on altitude and cutoff (dependent on location), ignoring solar modulation. In this case, the actual neutron flux (NF) at a given location can be expressed as follows:

$$NF = NF_{ref} \cdot GRF \cdot e^{-\frac{A - A_{ref}}{L}} \tag{7}$$

, where NF_{ref} is the the neutron flux at the reference location (New York = 14 n/sq. cm/h). L is the flux attenuation length for neutrons in the atmosphere (~148 g/cm2). Finally, A is the areal density of the location of interest, Aref is the areal density of the reference location.

$$A = 1033 \cdot e^{-0.03813 \cdot \left(\frac{a}{1000}\right) - 0.00014 \cdot \left(\frac{a}{1000}\right)^2 + 6.4 \cdot 10^{-7} \cdot \left(\frac{a}{1000}\right)^3} \quad (8)$$

While these factors can be described analytically and integrated in the various models, the GRF represents the Geomagnetic Rigidity Factor and varies according to the geographical position. As an example, values of geomagnetic vertical cutoff rigidity used to calculate the relative neutron flux were provided by the Aerospace Medical Research Division of the Federal Aviation Administration's Civil Aerospace Medical Institute. The cutoff data were generated by M.A. Shea and D.F. Smart using the International Geomagnetic Reference Field for 1995 [8][9]. Therefore, the actual GRF values can only be provided as a table of data indexed according the longitude and latitude. This cause a number of issues, including the need to interpolate between the available sparse, low granularity data and the difficulty to integrate tabular data in a compact model.

Machine-Learning Models can cope very efficiently with these difficulties. Firstly, the training can be done on any type of data, analytical, tabular with any type of data representation for the input parameters: linear (for the altitude) or specific (geographical coordinates). Moreover, it will also be able, provided that an adequate model is used, to interpolate GRF data between the locations provided in the tables, allowing the approximate calculation of the GRF for any location.

The neutron-induced error rate is provided as base value for the reference setting (New-York, sea-level) that needs to be multiplied by an acceleration factor calculated according to the actual location and altitude. As an example, the following tables show the acceleration factors for a selection of altitudes and location. The final acceleration factor can be calculated by multiplying the appropriate location factor with the desired altitude factor.

TABLE I. ALTITUDE-DEPENDED NEUTRON DATA

Altitude *(feet)*	***Altitude*** *(m)*	**Neutrons flux** *(n/sq.cm hour)*	**Neutrons flux** *(n/sq.cm second)*	**Neutrons flux** *(relative to sea level)*
0	0	14.0	0.003889	1.0
1000	304.8	18.2	0.005056	1.3
2000	609.6	23.4	0.0065	1.7
5000	1524	47.6	0.013222	3.4
10000	3048	134.6	0.037389	9.6
20000	6096	668.5	0.185694	47.8
30000	9144	2001.1	0.555861	142.9
35000	10668	2993.2	0.831444	213.8
40000	12192	4147.0	1.151944	296.2

TABLE II. LOCATION-DEPENDED NEUTRON DATA

Location	**Neutrons flux** *(relative to sea level)*
Colorado Springs	4.42
Bangalore	1.02
Beijing	0.72
Grenoble, France	1.24

This part fulfils the Phase I of the proposed methodology.

In the following (Phase II), a simple de-rating approach is assumed to calculate the overall Error Rate of an ASIC with 1 Mbit of Flip-Flops and 10 Mbits of Combinatorial cells. The overall Error Rate can be computed as the de-rated contribution of each individual element:

$$SER_{ASIC} = \sum_{element} RawSER_{s|c} \cdot \prod_{LDR,TDR,FDR} DR \quad (9)$$

The de-ratings applicable here are: LDR – Logic De-Rating (the probability of the fault to propagate from a logic/masking perspective), the TDR – Temporal De-Rating (the probability of the fault to arrive during a sensitive opportunity window) and FDR – Functional De-Rating (the probability for an error to become an observable failure). The Fault/Error/Failure dichotomy and the various De-Rating factors are used as presented in [6].

Simple equations or values for the applicable De-Rating factors are presented in the equations below

$$TDR_{Seq}(Freq) = \frac{Slack}{ClockPeriod} = 1 - \frac{Freq[Hz]}{2e6} \quad (10)$$

$$TDR_{Comb}(PW, Freq) = \frac{PW}{ClockPeriod} = PW * Freq \quad (11)$$

$$LDR = 0.25 \quad (12)$$

$$FDR = 0.25 \quad (13)$$

Finally, the overall SER of the ASIC can be described as follows:

$$SER(PW, Freq, V_{dd}) = \left(1Mbit \cdot RawSER_{Seq} \cdot TDR_{Seq} + 10Mbit \cdot RawSER_{Comb} \cdot TDR_{comb}\right) \cdot LDR \cdot FDR \quad (14)$$

and can be unrolled in the following form:

$$SER(PW, Freq, V_{dd}) = \left(1Mb \cdot 100\frac{FIT}{Mb} \cdot \left(1 + (1.2 - V_{dd})\right) \cdot 1 - \frac{Freq[Hz]}{2e6} + 10Mb \cdot 50\frac{FIT}{Mb} \cdot (1 + (1.2 - V_{dd})) \cdot PW * Freq\right) \cdot 0.25 \cdot 0.25 \quad (15)$$

A final customization can be made to clarify the value or the value range for the Pulse Width parameter which is a technology attribute and may not make sense to or be fillable by the final

user. Therefore, the ASIC provider, in agreement with the technology provider, may specify values for the PW according to the working environment. In this example we consider a neutron environment (ground applications) with a typical PW of 50ps, Alpha particles – 10ps and Heavy Ions – 100ps.

It is obvious that the overall SER equation can disclose unit technology data (FIT rates per Mb, typical pulse widths), design structure (1Mb of FF and 10Mb of combinatorial cells) or knowledge (such as calculation of de-rating factors).

The next step (*Phase III – Compact ML Model Elaboration*) consists in exercising the overall SER equation over the range of valid values for the environment, frequency and voltage. The results of this exploration are presented in the following table:

TABLE III. SER VALUES TABLE

Sample	Parameters				SER [FIT]
	Voltage [V]	*Environment*	*Frequency [MHz]*	*Location, Altitude*	
0	0.8	n	0.1	NYC, sea-level	238.551
1	0.8	n	0.2	NYC, sea-level	467.927
2	0.8	n	0.3	NYC, sea-level	697.303
...	...	...	...		...
897	1.5	HI	1.8	N/A	2068.97
898	1.5	HI	1.9	N/A	2183.66
899	1.5	HI	2	N/A	2298.35

Phase III continues with the training of the ML model. In this work we are interested by the applicability of different ML models for the intended usage. Accordingly, several ML regression methods have been evaluated: Linear Models (Linear and Ridge), Kernel Ridge Regressor (with Linear, Polynomial, RBF, Sigmoid Kernels), Decision Tree Models, Neighbors-based Models (K-Nearest and Radius), Support Vector Machine Models (Linear SVR, SVR with various kernels, NuSVR), Multilayer Perceptron Neural Networks.

All models have been trained with 60% of the data set, the train data set. After the training, the models are exercised over the full permitted parameters range. This allows, on the one hand, to measure the accuracy of the model to recall values which were already in the training data set. On the other hand, by testing the model with data not used for training it is evaluated if the model is able to interpolate and extrapolate the given data. The performance of the models is measured by comparing the predicted values against the reference values and calculate the following metrics: "MAE - Mean Absolute Error", "MAX - Max Absolute Error", "RMSE – Root Mean Squared Error", "EV - Explained Variance", "R^2 - Coefficient of Determination".

The following table presents the results for the most accurate and promising models:

1. Ridge Regression with Polynomial Kernel
2. Ridge Regression with RBF Kernel
3. k-Nearest Neighbors Regression
4. Support Vector Regression with Polynomial Kernel

TABLE IV. TOP ML MODELS PERFORMANCES

ML Model	Data Set	ML Model Error/Correlation Measurements				
		MAE	*MAX*	*RMSE*	*EV*	R^2
1	Train	1.688e-06	1.199e-05	2.119e-06	1	1
	Test	1.695e-06	1.331e-05	2.106e-06	1	1
2	Train	2.716e-05	2.545e-04	3.530e-05	1	1
	Test	2.907e-05	6.609e-04	4.421e-05	1	1
3	Train	0	0	0	1	1
	Test	21.91	195.2	30.98	0.99	0.99
4	Train	1.787e-02	6.830e-02	2.233e-02	1	1
	Test	1.756e-02	6.243e-02	2.195e-02	1	1

The results show that the presented Machine Learning models are able to accurately recall the values from the train data set. The error metrics MAE, MAX, and RMSE are very low and the correlation like metrics EV and R^2 are 1 for most of the models. The Predicting the test data set show similar good results, which means that the models are also able to interpolate and extrapolate the given data. The k-Nearest Neighbors regression is able to perfectly recall the data the data used for training the model, due to the nature of its algorithm. However, predicting new values from the test data set, which were not used for training, shows the models weakness. The inter- and extrapolation capability is less good in comparison to other models.

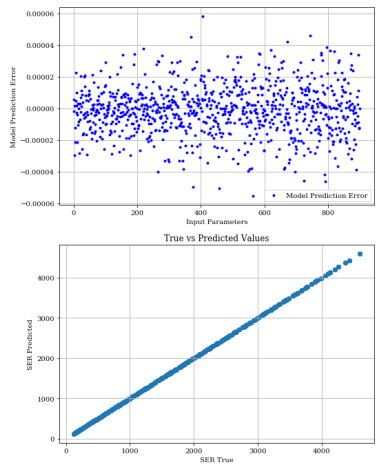


Fig. 1. Results of predicting the train and test data set by using Ridge regression with polynomial kernel.

Fig. 1 shows the graphs representing the model prediction error for the train and test data set, as well as the correlation between the actual and predicted values when the Ridge Regression with polynomial Kernel is used. In comparison Fig. 2 the results for the k-Nearest Neighbors regression is shown. The graphs show as well very clearly, that the k-Nearest Neighbors regression is perfectly able to recall the training data but less efficient when predicting the test data.

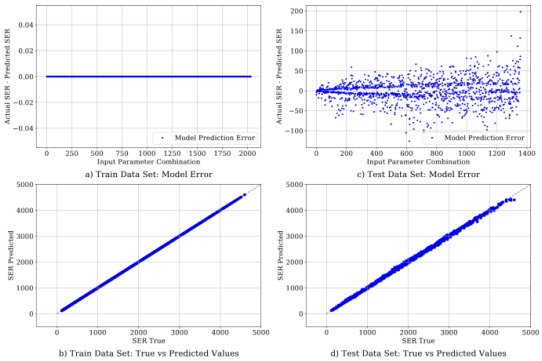


Fig. 2. Results of predicting the train and test data set by using k-Nearset Neighbors regression.

The results show that certain ML models are able to learn the given metric function and it is a good approach for the matching of extensive reference data sets. Depending on the data or function to learn other models can be less effective. As seen in the example they might only be appropriate to recall the data but not to inter- and extrapolate it. Other models might be less effective in general. This means for the given data several models need to be considered and evaluated.

TABLE V. LINEAR REGRESSION MODEL

ML	ML Model Error/Correlation Measurements					Time (s)
	MAE	*MAX*	*RMSE*	*EV*	R^2	
1	152.917	544.768	204.089	0.96	0.96	0.0025

The graphical representation of this model is also telling (Fig. 3)

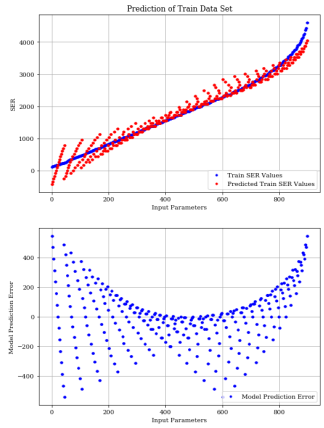


Fig. 3. Example of results of the Linear Regressor Model

Because of the limited and simple training data set, the training of the ML models is fast and doesn't require computationally intensive resources. A single-shot execution of a trained models is very fast for most models, with execution time inferior to the nanosecond.

V. CONCLUSIONS AND PERSPECTIVES

In this paper, a methodology has been proposed to help experts to approach the complexity of hierarchical modeling of reliability and functional safety metrics. Furthermore, the presented approach allows the use, elaboration and distribution of compact ML models in an uniform and systematic manner, minimizing both human and CPU efforts while maintaining high accuracy and fidelity.

While the approach has been validated and seems to work effectively and efficiently on a modest example, further works will have to consider more complex systems with expanded sets of parameters and with more, non-linear fault models.

Finally, this approach can be used to integrate the effect of multiple failure mechanisms (such as Total Ionizing Dose – TID, circuit aging) that can contribute to an overall applicative failure rate. The effects of such mechanisms depend on generic (such as circuit structure, test-vector/workloads, PVT …) and specific (total and rate of received dose, age of the circuit …), The authors intend to to train ML models for a combination of Transient Faults/Soft Errors, TID and Aging Effects using the same fundamental platform of parameters plus a limited set of effect-specific parameters. The trained ML models will be able to quickly evaluate a large variety of effects, encapsulating very useful reliability and functional safety data in a compact and efficient solution that can be used and reused further down the design and manufacturing flow.

REFERENCES

[1] IEC, "Functional Safety and IEC 61508", https://www.iec.ch/functionalsafety/

[2] ISO, "SO 26262-1:2018 Road vehicles — Functional safety" https://www.iso.org/obp/ui/#iso:std:iso:26262:-1:ed-2:v1:en

[3] IEEE, "C/DA P2851 Standard for Exchange/Interoperability Format for Functional Safety Analysis and Functional Safety Verification of IP, SoC and Mixed Signal ICs", https://development.standards.ieee.org/get-file/P2851.pdf?t=102689600003

[4] Accellera, "Functional Safety Proposed Working Group", https://www.accellera.org/activities/proposed-working-groups/functional-safety

[5] A. Evans, M. Nicolaidis, S. Wen, D. Alexandrescu and E. Costenaro, "RIIF - Reliability information interchange format," 2012 IEEE 18th International On-Line Testing Symposium (IOLTS), Sitges, 2012, pp. 103-108.

[6] M. Vilchis, R. Venkatraman, E. Costenaro and D. Alexandrescu, "A real-case application of a synergetic design-flow-oriented SER analysis," 2012 IEEE 18th International On-Line Testing Symposium (IOLTS), Sitges, 2012, pp. 43-48.

[7] JEDEC, "JESD89-3A - TEST METHOD FOR BEAM ACCELERATED SOFT ERROR RATE", https://www.jedec.org/standards-documents/docs/jesd-89-3a

[8] http://www.seutest.com/cgi-bin/FluxCalculator.cgi, retrieved on March 10, 2020

[9] C. E. Barton et al., "International Geomagnetic Reference Field, 1995 Revision: International Association of Geomagnetism and Aeronomy (IAGA) Division V, Working Group 8," in Geophysical Journal International, vol. 125, no. 1, pp. 318-321, Apr. 1996

Appendix 9

IX

A. Balakrishnan, G. C. Medeiros, C. C. Gürsoy, S. Hamdioui, M. Jenihhin, and D. Alexandrescu, “Modeling soft-error reliability under variability,” in *2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 1–6, ***Award: Outstanding Student Research Paper***, 2021

Modeling Soft-Error Reliability Under Variability

Aneesh Balakrishnan*†, Guilherme Cardoso Medeiros‡, Cemil Cem Gürsoy†,
Said Hamdioui‡, Maksim Jenihhin†, Dan Alexandrescu*
*iRoC Technologies, Grenoble, The France
†Department of Computer Systems, Tallinn University of Technology, Tallinn, The Estonia
‡Computer Engineering Laboratory, Delft University of Technology, Delft, The Netherlands
{aneesh.balakrishnan, dan.alexandrescu}@iroctech.com {cemil.gursoy, maksim.jenihhin}@taltech.ee
{G.CardosoMedeiros, S.Hamdioui}@tudelft.nl

2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT) | 978-1-6654-1609-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/DFT52944.2021.9568295

***Abstract*—The Soft-Error (SE) reliability and the effects of Negative Bias Temperature Instability (NBTI) in deep submicron technologies are characterized as the major critical issues of high-performance integrated circuits. The previous scientific research studies provide a comprehensive description that the soft-error vulnerability becomes more severe as the circuit performance degrades with aging. The main reason is the reduction of cell-level critical charge in an aging environment. However, such increased soft-error generation does not necessarily contribute towards circuits' critical functional failures. The proposal of this paper is the experimental investigation of soft error propagation at the aged gate-level by considering the different derating factors like Electrical Derating (EDR), Temporal Derating (TDR), Logical Derating (LDR), and Functional Derating (FDR). As contrary to the previous studies, the results of this work prove that SEU fault propagation probability is reducing in critical paths as time advances while the propagation probability of SET faults is neither reducing nor increasing, but the spot of generation of failure enhancing SETs is shifting within the clock period.**

I. Introduction

System engineering continuously focuses on the aggressive technology scaling which increases the vulnerability of radiation-induced soft-errors [1] [2]. Simultaneously, the time-dependent variabilities like the effects of Bias Temperature Instability (BTI) expedite the reliability assessment of high-performance Cyber-Physical Systems (CPS) into an increasingly challenging job.

Scientific efforts significantly highlighted the relevance of experiments that account for the impact of aging on soft-error reliability. The works [3], [4] and [5] have proposed their methods and chronicled their observations on soft-error susceptibility under circuit aging. A. Gebregiorgis in [6] has presented a cross-layer reliability analysis in the presence of soft-errors, aging, and process variation effects. Similarly, the work [7] provides ramification of aging and temperature on Propagation Induced Pulse Broadening (PIPB) effect of Single Event Transient (SET) pulse for nano-scale CMOS. In [8], F. L. Kastensmidt shows that aging and voltage scaling enhance the Soft Error Rate (SER) susceptibility of SRAM-based FPGAs by two times.

However, the above-cited articles mainly focussing on the increased occurrence of radiation-induced errors on behalf of the declined critical charge Q_{crit} at the technology cells. The SER is not observed in diverse propagational forms of SET and Single Event Upset (SEU) separately. Contrary to previous findings, our results concentrate on the soft-error generation and propagation by considering all the derating factors (EDR, LDR, TDR and FDR) associated with an aged Standard Delay Format (SDF) file. The soft-error reality analysis of aged circuits at higher abstraction (e.g., RTL) is the future vision of this work. The relevant contributions of the paper to the soft-error reliability analysis under aging include the following,

- **Characterization of threshold voltage degradation (ΔV_{th}) for industrial 15 nm technology**
- **Cross-level Modeling of NBTI-induced delay degradation using Artificial Intelligence (AI)**
- **Analysis of derating factors' influence in soft-error propagation and proposing a signal-processing model to locate propagating SETs**

The main motivation of the work is to investigate the soft-error reliability challenges at low-scaled (e.g., 15-nm) technologies with time-dependent voltage variabilities. The downscaled technology results in clock-frequency maximization and provides soft-error propagation a high sensitivity to the path delay variations in the design due to aging. In this work, aging effects are limited to the NBTI. The Positive Bias Temperature Instability (PBTI) remains unconsidered throughout the research based on the facts: the PBTI effect for the NFET transistors at small scale technologies for the High Performance (HP) applications is comparatively low compared to the NBTI effect for PFET transistors [9]. Also, the PBTI effect's dependence on the quality of gate-oxide materials [10] and the complexities in the efficacy to model PBTI's relative voltage degradation, are more than the intended research focus. However, the deduced conclusions for NBTI can extend to the combined effect of NBTI and PBTI.

The paper organized the rest as follows: sections II, III, IV, and V. Section II covers the mathematical formulation of NBTI-induced voltage degradation and cross-level modeling of NBTI-induced delay degradation. The overall experimental setup is briefed in Section III. Section IV explains the results and discussions. Finally, a conclusion to the holistic approaches is provided in section V.

This work was supported by RESCUE ETN project under grant No. 722325.

II. Background

A. Modeling NBTI induced voltage degradation

The primary work of this research activity is to find a mathematical abstraction to estimate the changes in threshold voltage ΔV_{th} due to the NBTI process in the aged circuits, and completely revised from the original papers [11] and [12]. The paper [11] has presented a predictive model for the Negative Bias Temperature Instability (NBTI) of PMOS under both short-term and long-term operations. This model has

TABLE I
Parameters and Units for NBTI Predictive Modeling

Symbol	Quantity	Numerical value in SI unit
K_v	Technology Constant	6.1773e-11
C	Temperature Dependence	2.2987e-12
T_0	Constant	10^{-8} (s/nm^2) $\rightarrow$ 10^{10} (s/m^2)
E_0	Technology-Independent	0.08 V/nm $\rightarrow$ 0.08e9 V/m
E_a	Technology Independent	0.13 eV $\rightarrow$ 2.08260e-20 J
k	Boltzmann Constant	1.38e-23 m^2 kg s^{-2} K^{-1}
$1/n$	H_2 Diffusion Model	1/6
t_{ox}	Oxide Thickness	0.9 * 10e-9 m
ξ_1	Back Diffusion Const.	0.9
ϵ_{ox}	Oxide Permittivity	$3.9 * 8.854 * 10^{-12}$ F/m
K_1	Model Constant	7.5e22.5 $C^{-0.5}\, m^{-2.5}$
E_{ox}	Electric Field (gate oxide)	27.7e-7 V/m

Unit Abbreviations: K = Kelvin, F = Farad; V = Volt, s = Second, m = Meter, nm = Nano-meter, J = Joule, kg = Kilogram.

comprehensively apprehended NBTI dependence on the key transistor-design parameters, based on the reaction-diffusion (R-D) mechanisms. The work in [11] presented a quality model accuracy verification with 90-nm technology while [12] reassured that with industrial 65-nm technology. A characterization of threshold voltage (ΔV_{th}) degradation due to NBTI for 15-nm technology cells, has been presented through this paper, where the models from [11] and [12] are integrated with 15-nm technology parameters as in Table I. A model for the $\Delta V_{th,t}$ considering the long-term effect of NBTI is provided in [11]. At high frequencies, the threshold voltage degradation of long-term evaluation is independent of the frequency [11] [13]. So the ΔV_{th} at time 't' can be written as [11]:

$$\Delta V_{th,t} \approx \left(\frac{n^2 K_v \alpha C t_1 t}{\xi_1^2 t_{ox}^2 (1-\alpha)} \right)^n \tag{1}$$

The parameter α in (1) is the duty-cycle that defines the signal probability in a clock period. Equation (1) refers to the dynamic NBTI, $\forall\, \alpha \in [0,1]$. As $\alpha \rightarrow 1$, (1) is attributed to the static NBTI that corresponds to the case of PMOS under constant stress. A graph between ΔV_{th} and α, at $t = 1$ year is given in Fig. 1. The upper-limit of the degradation model (1) is estimated by a power law model [14] as privided in (2). The static ΔV_{th} calculation at t = 1 year gives 7mV that fits the trends of static ΔV_{th} of various technology-nodes as given in PTM models [15] [16].

$$|\Delta V_{th,t}| = ({K_v}^2 t)^n \tag{2}$$

The formula in (1) is deduced into a simpler form by substituting the parameters of 15nm technology and follows the form of (3), where relation of technology specific factors is represented as separate form.

$$|\Delta V_{th,t}| \simeq \underbrace{\left(\frac{n^2 K_v C t_1}{\xi_1^2 t_{ox}^2} \right)^n}_{\text{Technology dependent}} t^n \left(\frac{\alpha}{1-\alpha} \right)^n \tag{3}$$

B. An AI revolution for NBTI predictive model

The gate-delay degradations of 15-nm technology library cells are characterized through the SPICE simulation at the transistor level. A neural network called Long Short-Term Memory (LSTM) is a perfect method to predict the future delays from past delay (Δt) samples even without an input like ΔV_{th}. Fig. 2 depicts the complete results of LSTM modeling, where the predicted values from previous samples are shown in blue. However, the LSTM model is not a completely dependable one here for generating a delay Δt for the current input ΔV_{th} in a timely important simulation environment. To simplify the exhaustive fault-injection experiment, a Machine Learning (ML) model called Support Vector Machine (SVM) is used to automate the simulation data. Previous scientific literatures [17] and [18] presented a simple polynomial function for modeling variation between Δt and ΔV_{th}. In this work, SVM can perform as a comprehensive model generator for a triangular relation between α, time 't', and Δt, where ΔV_{th} is hidden in the SVM model. The AI revolution is implicitly referring to such complex and compact modeling of indirectly related multi variables. Fig. 3 delegates a SVM model for an inverter. Once SVM is trained offline, it will become a fast online computing model generator of Δt for the inputs α and time 't'. This compact ML model is very much straightforward in exporting the variability from the transistor level to the gate level. Another important factor to choose SVM over LSTM is the low static ΔV_{th} (7mV at t=1 year) by (3), because a large simulation data needs to train LSTM as provided in Fig. 2, where ΔV_{th} ranges from 0 to 0.4V, while ΔV_{th} in Fig. 3 perfectly follows the ΔV_{th} values as provided in Fig. 1.

III. Methodology

A. Phase I: Fault-Injection Campaign

Single Event Effects (SEEs) are the consequences of interactions between the circuit and the radiation particles. SEUs and SETs are widely used here as the prominent representatives of SEEs. The use-case for SEU fault mainly implies an inversion of the stored value in a flip-flop until the clock period changes. The SET represents a transient pulse of arbitrary width at the gate and having the probability to propagates and latches to the downstream sequential element. Fig. 4 shows a block diagram of the fault-injection campaign that infers the statistical functional failure metrics of SEU/SET events. The mathematical model for Functional Failure Rate ($FFR_{i,seu}$) of an SEU event is described in (4) and the Functional Failure Rate due to SET ($FFR_{i,set}$) is formulated in (5).

$$FFR_{i,seu} = FIT \cdot TDR \cdot LDR \cdot FDR \tag{4}$$

$$FFR_{i,set} = FIT \cdot EDR \cdot TDR \cdot LDR \cdot FDR \tag{5}$$

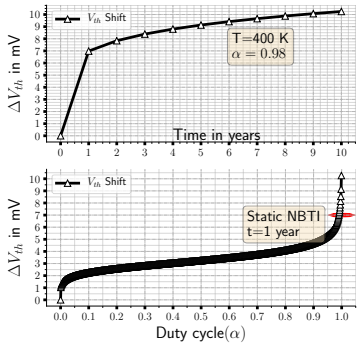


Fig. 1. Mathematical Analysis

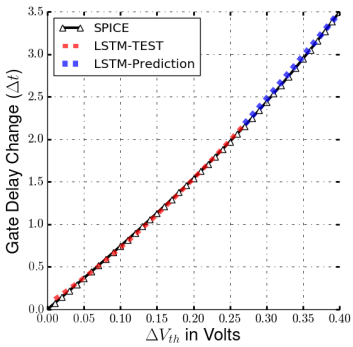


Fig. 2. LSTM Model (Inverter)

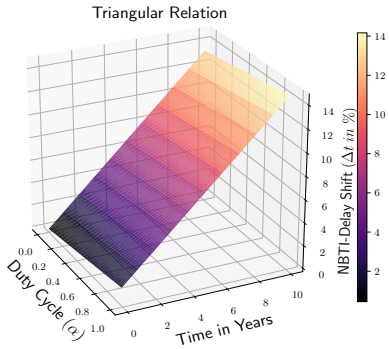


Fig. 3. SVM-Regression Model (Inverter)

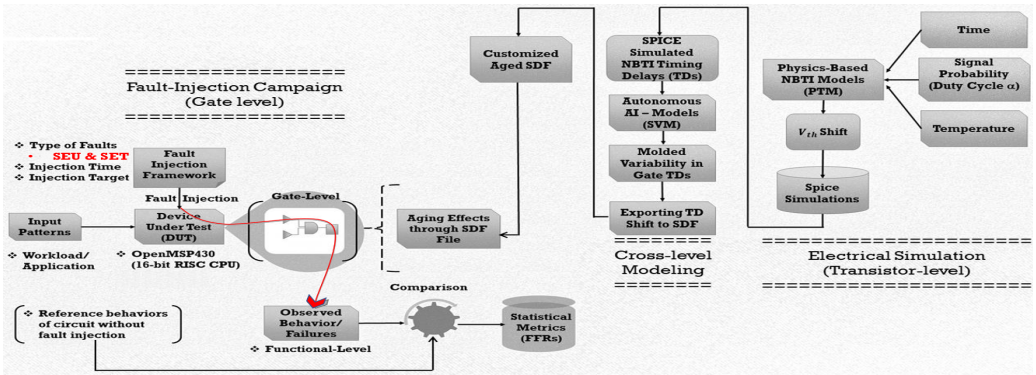


Fig. 4. Delineation of Aging Aware Fault-Injection Framework

where FIT denotes the rate of soft errors at the i^{th} flip-flop/gate in the Failure-In-Time (FIT) unit. Electrical Derating (EDR), Temporal Derating (TDR), Logical Derating (LDR), and Functional Derating (FDR) are more sophistically portrayed in Fig. 5. The classical explanations for each derating factors are available in [1].

B. Phase II: Aging aware gate-level circuit

Fig. 4 represents how an injected fault propagates through the aged circuits. Circuit aging is modeled through degraded propagation delay as the result of the NBTI process at PMOS transistors. Sophisticated cross-layer modeling is also demonstrated in Fig. 4. A way more experimentally proved and scientifically adapted physics-based NBTI models (3) are employed to generate the V_{th} shifts.

Electrical simulations have been carried out using SPICE. Voltage sources have been injected in the gate of PMOS transistors to model the V_{th} shift caused by aging. The voltage introduced by the source was then swept from 0V to 0.4V to represent the effects of different levels of aging. The impact of aging on each logic gate is estimated by measuring the input-to-output time delay, i.e., the time delay from when the input is in $V_{DD}/2$ to when the output rises from '0' to '1' at the $V_{DD}/2$ mark. The spice simulation is conducted for 23 gates from a 15-nm Nangate library which including inverter (NOT), AND, OR, NAND, NOR, XOR, XNOR with input combinations ranging from 2 to 4. The transistor model (SPICE Model Card) for 15-nm Nangate library is customized from HP Predictive Technology Models (PTM) [16].

After the simulation process, the change in propagation delays of combinational cells is submitting to AI models. Fig. 4 depicts the significance of AI models in cross-layer modeling. A versatile model by SVM is used to generate the propagation delay change Δt as a function of duty cycle (α) and time in years. This model is very easy to port to gate-level or even to a higher hierarchical level, and adequately producing Δt values by concurrently changing time and signal probability. Those model-driven values are numerically very close to the original simulation Δt and the test phase of SVM confirms that the Mean Squared Error (MSE) is less than 2% for the given test data.

After testing the compatibility and quality of the model, the model outputs (Δt) are exported to the SDF file. A customized SDF file introduces time-dependent variability at the gate level.

The customization of the SDF file means altering the value under the keyword entry called 'IOPATH' that indicates the gates' Input-Output path delay. Before changing the gates' 'IOPATH' values, corresponding signal probabilities and signal transitions at each terminal are pre-estimated through signal-activity analysis through Verilog Procedural Interface (VPI) functions and Value Change Dump (VCD) files by Modelsim. So that, the changed values should coincide with the real-time input and output transitions between '0' and '1'. To process the above-explained cumbersome work for large IEEE standard SDF files, then a dedicated algorithm is written in python, and the corresponding pseudo-code is presented in Algorithm 1. The algorithm generates an IEEE standard customized aged SDF file that provides an aged behavior while injecting SEUs/SETs in a simulation environment.

Result: Customized Aged SDF File
Estimate ΔV_{th} for 10 years $\forall$ used gates $\in$ Tech.lib;
for *Each used Tech.lib gate* **do**
  for *Each 'i'* $\in$ *SVM model Training* **do**
    SVM : input vector $[\alpha_i, \text{time}_i, \Delta V_{th_i}] \mapsto \Delta t_i$;
  end
  Test SVM model and Save in Python script;
end
Read the SDF file in IEEE standard format;
for *Each cell instance in SDF File* **do**
  Identify the Tech. library cell name ;
  Calculate signal probability (α) from VCD file ;
  if *[α_i, $time_i$] given* **then**
    Calculate ΔV_{th_i} by (3) ;
    SVM.gate = Load trained SVM.gate kernel;
    Δt_i = SVM.gate.predict($[\alpha_i, \text{time}_i, \Delta V_{th_i}]$) ;
    Update the instance's IOPATH in SDF file;
  end
end

Algorithm 1: IEEE Standard Aged SDF File

IV. Results and Discussions

A. Device Under Test

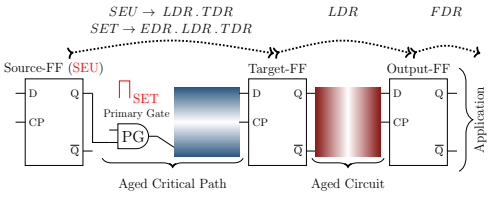


Fig. 5. From SEU/SET Fault to Failure

The case studies are conducted by gate-level circuits of the openMSP430 cores, a 16-bit microcontroller (μC) which is synthesized by the 15-nm NanGate Open Cell Library. As per the static timing analysis of the test case circuit, the critical paths contain a maximum path delay of 250ps. The CPU unit in the MSP430 μC is executing the application 'sandbox' by sourcing a clock period of 2ns. The Fig. 5 demystifies the propagation of SEU/SET faults to the circuit function.

B. Aging impact on SEU fault propagation

The NBTI effect is modeled in terms of propagation delay Δt of logic gates. The setup and hold time constraints of flip-flops dictate the maximum and minimum delays of the logic gates between them. The maximum delay constraint limits the number of consecutive gates on the critical path of a high-speed circuit. In this section, the results are proving that the propagation probabilities of SEU faults are dropping according to the changed delay of the signal path. These results are explaining in Fig. 6 and Fig. 7. In Fig. 6 a term called Polarization Window (PW) is introduced in red color. This is because the time width of the PW completely masks the SEU faults that have been generated inside the window. From 5, it is clear that induced-SEU fault at source flip-flop (Source-FF) reaches the target flip-flop (Target-FF) after a path delay $t_{D_{set}}$ between them. So that, a fault SEU_1 at source-FF between T_0 and T_1 as provided in Fig. 6 not overlaps the latching window (SETUP+HOLD time) when it reaches the target-FF, and masks the SEU_1 fault propagation. But the fault SEU_2 is propagated and latched to the target flip-flop because it is stable during the SETUP and HOLD time of target-FF. After 1 year, the PW is prolonged to $T_0 - T_2$ due to NBTI effect and clearly masks the fault SEU_2 as demonstrated in Fig. 6. Similarly, after 10 years, the faults SEU_1 and SEU_2 are masked and the fault SEU_3 is still propagating to target-FF. These conclusions are statistically derived from the fault-injection campaign of 1582 SEU faults at the path of longest delay as given in Fig. 7, where the upper-row justifies that SEU_1 is masked without aging, and the lower-row justifies that SEU_1 and SEU_2 are masked after 10 years.

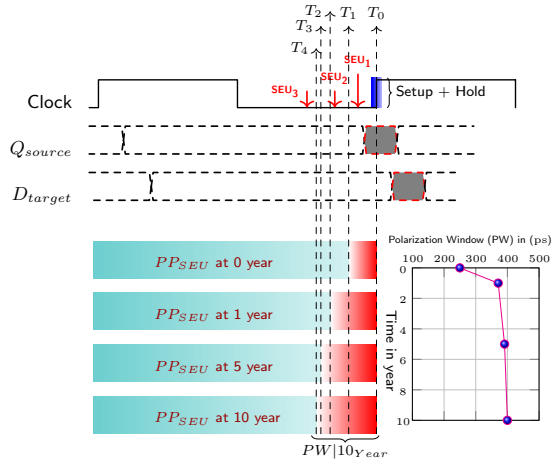


Fig. 6. SEU-Faults' Polarization Window (PW) Elongation

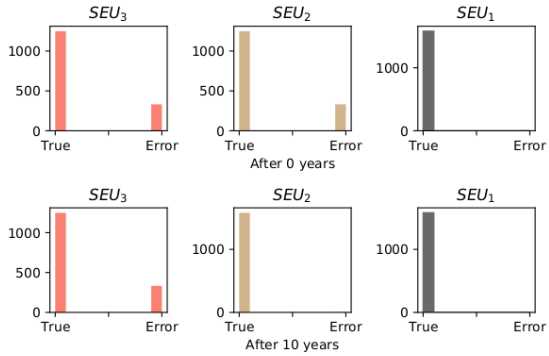


Fig. 7. SEU-Fault Injection Result (After 0-10 years)

C. Impact of aging in SEU caused circuit-functional failures

In Fig. 6, the Propagation Probability (PP_{SEU}) indicates the chances of generated SEUs to propagate through the downstream circuit within an arbitrary clock period T_{clk} and is modeled as an uniform distribution as specified in (6).

$$PP_{SEU} = \begin{cases} \frac{1}{T_{clk}}(T_{clk} - PW|_{\tau_{d_0}}) & \forall \text{ Non-aged } \tau_{d_0} \\ \frac{1}{T_{clk}}(T_{clk} - PW|_{\tau_{d_i}}) & \forall \text{ Aged } \tau_{d_i} \end{cases} \quad (6)$$

The effect of elongated polarization window PW due to aged delays τ_{d_i} in (6) models the increased masking of SEU faults and confirms it through an exhaustive Fault-Injection (FI) campaign as shown in Fig. 8. Color C-2 represents a FI campaign of 74000 injected faults at 188 flip-flops (FFs) of the non-aged circuit, which produce a total of 1181 functional failures as in (4). The 188 FFs are explained as high failure vulnerability FF-instances out of 720 FFs. A 10 year aged model of the same circuit is simulated with an increased FI rate by 50% (color C-1 in Fig. 8). This increment is performed based on the previous studies that the rate of SEU generation increases due to decreased critical charges Q_{crit} while aging.

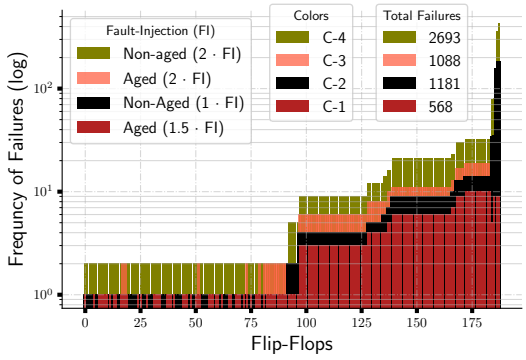


Fig. 8. Functional Failures (4) of SEU-Fault Injection

Even though the FI rate is boosted by 50% for the aged circuit, the number of detected functional failures per flip-flop and the cumulative failures (568), are comparatively less contrary to the case in color C-2 in Fig. 8, which shows high-level masking of SEU faults. As an extension of the analysis, a FI campaign consists of 148000 faults (almost double in number) is performed at the 10-year aged circuit. So that a total of 1088 failures are observed (color C-3 in Fig. 8) and that plot contrasts with a reference bar-graph of FI of 148000 faults at the non-aged circuit (color C-4). FI in C-3 leads to more failures per FF compared to the case in C-2, but still, the overall failures in color C-3 are fewer. This proved that aging causes a masking effect in SEU fault propagation.

D. Aging impact on SET fault propagation

As the circuit ages, considerable propagation-probability deteriorations for SEU faults have been notified. Nevertheless, SETs are not attenuated when traversing the critical paths in response to elongation of the polarization window in Fig. 6. The time-slot at which the generated SETs are latching to target-FF (Fig. 5) is shifting advance in time due to the delay changes in the traversal path of SET. These observations are delineated in Fig. 9, where the propagation delay $t_{D_{set}}$ of SET_1 at gate PG in Fig. 5 varies from 248 ps to 398 ps, which causes a shift in the time slot of propagating SET from T_1 to T_4. The observations are simulated by a SET fault of 20ps in width. Below the timing diagram in Fig. 9, the results of 100 SET fault injections over a second-half cycle of an arbitrary fault-latching clock are provided.

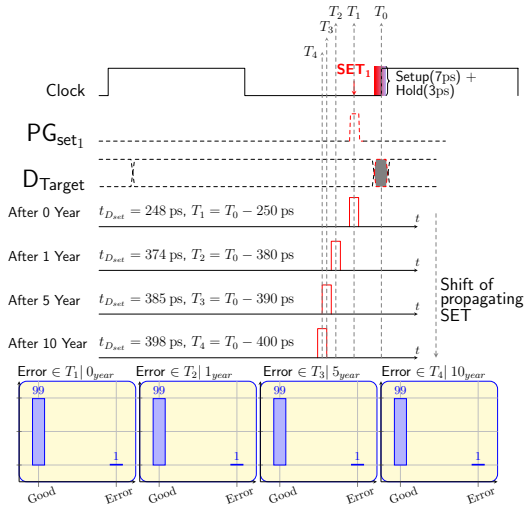


Fig. 9. Shift in Propagating Spot of 20ps-SET

E. Modeling of aging impact on SET fault propagation

The shift in time-spots of propagating SET as provided in Fig. 9, is modeled by the signal processing method. The SET

pulse is assigned by a digital pulse model of width 'w' in gate-level, which possess same electrical boundary properties of a transient pulse at transistor-level. Similarly, the SETUP and HOLD time-periods of a target-FF in Fig. 5, are modeled as a single digital pulse $S_{SH}(t-T_0)$ of width equal to the sum of SETUP and HOLD, and the amplitude is normalized by $1/w$. The delayed impulse function $\delta(t-t_{D_{set}})$ in step-1 of Fig. 10 convolves with induced SETs (G_{set_1} and G_{set_2}), and produces delayed SETs $G_{set_1}(t-(T_1+t_{D_{set}}))andG_{set_2}(t-(T_2+t_{D_{set}}))$ that are analogous to propagated SET pulses through a path delay $t_{D_{set}}$ and reach target-FF as in a real-time scenario of Fig. 5. The convolution and generation of delayed SETs is represented by step-1 and step-2 in Fig. 10. Similarly, step-4 indicates a valid-convolution ($\circledast_v$) [19] between $S_{SH}(t-T_0)$ in step-3 and propagated SETs in step-2, where convolution product is only given for points at which the signals overlap completely. The valid-convolution results in the generation of two impulse functions $\delta(t-(T_2+t_{D_{set}}))$ in blue and $\delta(t-T_0)$in black with amplitudes $A(\delta)$=0 and $A(\delta)$=1 respectively, where $A(\delta)$=1 represents the non-masked fault as given in (7). The case of $A(\delta)<1$ represents the masked fault and various cases of step-4 are presented in Fig. 10.

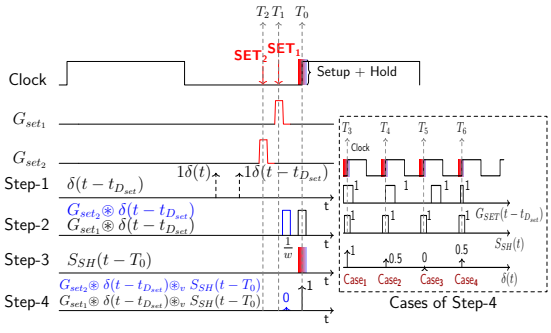


Fig. 10. SET Propagating Model

$$\underbrace{G_{set}(t)\circledast\delta(t-t_{D_{set}})\circledast_v\frac{1}{w}S_{SH}(t)}_{FPP_{i,set}}=\delta(t)*\begin{cases}A(\delta)=1\\0<A(\delta)<1\end{cases}\tag{7}$$

where, $FPP_{i,set}$ is the Fault Propagating Probability of SET pulse at the i^{th} gate. So that, the complex equation (5) can be simplified as:

$$FFR_{i,set}=FIT\cdot FPP_i\cdot LDR_i\cdot FDR_i\tag{8}$$

$FPP_{i,set}$ generate a model which characterize TDR_i and EDR_i in (5). For the case of aging, the path delay $t_{D_{set}}$ in (7) models NBTI caused time-slot shift of propagating SETs and $\frac{1}{w}$ factor in (7) models the width of propagating SETs.

V. Conclusion

Based on the observed results of this research, it has been concluded that voltage variability due to aging shows significant masking property on SEU fault propagation and reduces the effect of increased soft-error susceptibility while aging. Furthermore, this work explicitly reveals that aging shifts the SETs' propagating spots within the clock period. Besides, the latching probability of a SET fault to a downstream flip-flop is successfully modeled by a signal-processing method. The revised AI models and SDF-based aged technology data from this work are considering in the future to emulate aging-aware timing models at the system's RTL abstraction, which enables an early aging analysis in the design process.

References

[1] M. Nicolaidis, *Soft Errors in Modern Electronic Systems*. Springer, Boston, MA, 2011.

[2] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, 2005.

[3] A. P. Shah and P. Girard, "Impact of aging on soft error susceptibility in cmos circuits," in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2020, pp. 1–4.

[4] M. Omaña, D. Rossi *et al.*, "Impact of aging phenomena on latches' robustness," *IEEE Transactions on Nanotechnology*, 2016.

[5] D. Rossi, M. Omaña *et al.*, "Impact of aging phenomena on soft error susceptibility," in *IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, 2011, pp. 18–24.

[6] A. Gebregiorgis, S. Kiamehr *et al.*, "A cross-layer analysis of soft error, aging and process variation in near threshold computing," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2016.

[7] A. Yan, Y. Ling *et al.*, "Aging-temperature-and-propagation induced pulse-broadening aware soft error rate estimation for nano-scale cmos," in *2018 IEEE 27th Asian Test Symposium (ATS)*, 2018, pp. 86–91.

[8] F. L. Kastensmidt, J. Tonfat *et al.*, "Aging and voltage scaling impacts under neutron-induced soft error rate in sram-based fpgas," in *2014 19th IEEE European Test Symposium (ETS)*, 2014, pp. 1–2.

[9] C. Auth, C. Allen *et al.*, "A 22nm high performance and low-power cmos technology featuring fully-depleted tri-gate transistors, self-aligned contacts and high density mim capacitors," in *Symposium on VLSI Technology (VLSIT)*, 2012, pp. 131–132.

[10] S. Sinha, B. Cline *et al.*, "Design benchmarking to 7nm with finfet predictive technology models," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, 2012.

[11] S. Bhardwaj, W. Wang *et al.*, "Predictive modeling of the nbti effect for reliable design," in *IEEE Custom Integrated Circuits Conference*, 2006.

[12] W. Wang, V. Reddy *et al.*, "Compact modeling and simulation of circuit reliability for 65-nm cmos technology," *IEEE Transactions on Device and Materials Reliability*, vol. 7, no. 4, pp. 509–517, 2007.

[13] G. Chen, K. Y. Chuah *et al.*, "Dynamic nbti of pmos transistors and its impact on device lifetime," in *IEEE International Reliability Physics Symposium Proceedings*, 2003.

[14] J. B. Bernstein, "Chapter 3 - failure mechanisms," in *Reliability Prediction from Burn-In Data Fit to Reliability Models*, J. B. Bernstein, Ed. Oxford: Academic Press, 2014, pp. ix–x.

[15] R. Vattikonda, W. Wang, and Y. Cao, "Modeling and minimization of pmos nbti effect for robust nanometer design," in *2006 43rd ACM/IEEE Design Automation Conference*, 2006, pp. 1047–1052.

[16] PTM, "Spice models of predictive technology model (ptm)," in *Website: ptm.asu.edu*, 2012.

[17] S. Kostin, J. Raik *et al.*, "Spice-inspired fast gate-level computation of nbti-induced delays in nanoscale logic," in *IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits Systems*, 2015.

[18] M. Jenihhin, G. Squillero *et al.*, "Identification and rejuvenation of nbti-critical logic paths in nanoscale circuits," *Journal of Electronic Testing*, vol. 32, p. 273–289, 2016.

[19] Numpy, "Valid-convolution," in *Website: numpy.org/doc/stable/reference /generated/numpy.convolve.html*, 2021.

Curriculum Vitae

1. Personal data

Name & Surname	Aneesh Balakrishnan

Birth Date and Place	6 January 1990, Kerala, The India
Nationality	Indian

2. Contact information

Address	iROC Technologies 2 Square Roger Genin 5th floor, Grenoble, 38000, France
Fax	+33 438 129 615
Phone	+33 438 120 763 (office)
E-mail	aneesh.balakrishnan@iroctech.com, anbala@ttu.ee, aneesh.krishh@gmail.com

3. Education

Jan. 2018 – Jan. 2022	Tallinn University of Technology, School of Information Technologies, Computer and Systems Engineering, PhD studies
0ct. 2013 – Apr. 2016	Friedrich-Alexander University, Erlangen-Nurnberg, Germany Communication and Multimedia Engineering, MSc
Oct. 2007– May. 2011	Anna University of Technology Electronics and Communication Engineering, Bachelor in Engineering

4. Scholarships and Project Funding

Jan. 2018 – Jan.2021	RESCUE ETN Project Funded by European Union's Horizon 2020 Marie Curie-Skłodowska actions - grant no. 722325 Recruitment Host Institution (iROC Technologies) Cross-sectoral supervision (Tallinn University of Technology)

5. Language competence

Malayalam	Mother Tongue
Hindi & Tamil	Other Native Languages
English	Professionally Fluent

German	B1-level
French	A2-level

6. Professional employment

Jan. 2018 – Jan. 2022	iROC Technologies, 2 Square Roger Genin, 5th floor, Grenoble, 38000, The France
Position	R&D Engineer in Microelectronics
Feb. 2014 – Jul. 2015	International Audio Laboratories, Fraunhofer IIS, Erlangen, Germany
Position	Student Research Assistant

7. Computer skills

- Operating systems: Linix, Mac, and Windows
- Document preparation: Latex, LibreOffice, Window's Word Documentation
- Programming languages: C/C++, Java
- High-Level Language (software): Tcl/Tk, Matlab, Python, SQL and R
- High-Level Language (Hardware): Verilog,VHDL Matlab, Python, and R
- Shell Scripting: Bash, C Shell and Tcl Shell
- EDA Automation Tools: ModelSim, Synopsis, ICARUS Verilog, VCS
- Scientific Packages: Machine Learning Libraries,Verilog Procedural Interface (VPI) Functions, Datamining Libraries

8. Honours and awards

- Oct. 2021: **Outstanding Student Paper Award**, "Modeling Soft-Error Reliability Under Variability", 34th IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems

9. Defended theses

- 2016, Master Thesis
  International Audio Laboratories, Fraunhofer IIS, Erlangen, Germany
  Topic: Perceptual Evaluation of Magnitude and Phase Quantisation in Audio Coding
- 2011, Bachelor Thesis
  Anna University of Technology, India
  Topic: Home Automation through 230v Power Lines Using X-10 Protocol

10. Field of research

- Quality and Reliability of Complex systems
- Analysis of Technological Transient Faults and their Ramifications
- Computer Science and Artificial Intelligence
- Microelectronics

11. Scientific Works
Scientific Articles

1. A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, "The validation of graph model-based, gate level low-dimensional feature data for machine learning applications," in *2019 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, IEEE, oct 2019

2. A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, "Modeling gate-level abstraction hierarchy using graph convolutional neural networks to predict functional de-rating factors," in *2019 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, IEEE, jul 2019

3. A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, "Composing graph theory and deep neural networks to evaluate SEU type soft error effects," in *2020 9th Mediterranean Conference on Embedded Computing (MECO)*, IEEE, jun 2020

4. M. Jenihhin, M. S. Reorda, A. Balakrishnan, and D. Alexandrescu, "Challenges of reliability assessment and enhancement in autonomous systems," in *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, IEEE, oct 2019

5. T. Lange, A. Balakrishnan, M. Glorieux, D. Alexandrescu, and L. Sterpone, "On the estimation of complex circuits functional failure rate by machine learning techniques," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks – Supplemental Volume (DSN-S)*, IEEE, June 2019

6. X. Lai, A. Balakrishnan, T. Lange, M. Jenihhin, T. Ghasempouri, J. Raik, and D. Alexandrescu, "Understanding multidimensional verification: Where functional meets non-functional," *Microprocessors and Microsystems*, vol. 71, p. 102867, nov 2019

7. A. Balakrishnan, D. Alexandrescu, M. Jenihhin, T. Lange, and M. Glorieux, "Gate-level graph representation learning: A step towards the improved stuck-at faults analysis," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pp. 24–30, 2021

8. D. Alexandrescu, A. Balakrishnan, T. Lange, and M. Glorieux, "Enabling cross-layer reliability and functional safety assessment through ML-based compact models," in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, IEEE, jul 2020

9. A. Balakrishnan, G. C. Medeiros, C. C. Gürsoy, S. Hamdioui, M. Jenihhin, and D. Alexandrescu, "Modeling soft-error reliability under variability," in *2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 1–6, ***Award: Outstanding Student Research Paper***, 2021

10. T. Lange, A. Balakrishnan, M. Glorieux, D. Alexandrescu, and L. Sterpone, "Machine learning to tackle the challenges of transient and soft errors in complex circuits," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, IEEE, jul 2019

11. T. Lange, A. Balakrishnan, M. Glorieux, D. Alexandrescu, and L. Sterpone, "Machine learning clustering techniques for selective mitigation of critical design features," in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 1–7, 2020

12. X. Lai, T. Lange, A. Balakrishnan, D. Alexandrescu, and M. Jenihhin, "On antagonism between side-channel security and soft-error reliability in bnn inference engines," in *2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 1–6, 2021

Conference Presentations

1. NASA/ESA Conference on Adaptive Hard-ware and Systems (AHS), IEEE, jul 2019, Colchester, United Kingdom

2. IEEE Nordic Circuits and Systems Conference (NOR-CAS): NORCHIP and International Symposium of System-on-Chip(SoC),IEEE,oct2019, Helsinki, Finland

3. 9th Mediterranean Conference on Embedded Computing (MECO), IEEE, jun2020, Budva, Montenegro

4. 22nd International Symposium on QualityElectronic Design (ISQED), 2021, California, USA

5. 34th IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, Greece

Elulookirjeldus

1. Isikuandmed

Eesnimi & Perenimi	Aneesh Balakrishnan

Sünniaeg & Sünnikoht	6. jaanuar 1990, Kerala, India
Rahvus	Indialane

2. Kontaktandmed

Aadress	iROC Technologies 2 Square Roger Genin 5th floor, Grenoble, 38000, France
Faks	+33 438 129 615
Telefon	+33 438 120 763 (office)
E-post	aneesh.balakrishnan@iroctech.com, anbala@ttu.ee, aneesh.krishh@gmail.com

3. Haridus

Jaan. 2018 – Jaan. 2022	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Computer and Systems Engineering, PhD
0kt. 2013 – Apr. 2016	Friedrich-Alexander University, Erlangen-Nurnberg, Germany Communication and Multimedia Engineering, MSc
Okt. 2007 – Mai 2011	Anna University of Technology Electronics and Communication Engineering, Bachelor in Engineering

4. Stipendiumid ja Projektide Rahastamine

Jaan. 2018 – Jaan. 2021	RESCUE ETN Project Funded by European Union’s Horizon 2020 Marie Curie-Skłodowska actions - grant no. 722325 Recruitment Host Institution (iROC Technologies) Cross-sectoral co-supervision (Tallinna Tehnikaülikool)

5. Keelteoskus

Malajalami keel	Emakeel
Hindi & Tamili keel	Teised emakeeled
Inglise keel	Professionaalselt ladusalt

Saksa keel	B1-tase
Prantsuse keel	A2-tase

6. Teenistuskäik

Jaan. 2018 – Jaan. 2022	iROC Technologies, 2 Square Roger Genin, 5th floor, Grenoble, 38000, The France
amet	Mikroelektroonika uurimis- ja arendusinsener
Veebr. 2014 – Juuli 2015	International Audio Laboratories, Fraunhofer IIS, Erlangen, Germany
amet	Üliõpilaste uurimisassistent

7. Arvutioskused

- Operatsioonisüsteemid: Linix, Mac, and Windows
- Kontoritarkvara: Latex, LibreOffice, Window's Word Documentation
- Programmeerimiskeeled: C/C++, Java
- Kõrgetasemelised keeled (Tarkvara): Tcl/Tk, Matlab, Python, SQL and R
- Kõrgetasemelised keeled (Riistvara): Verilog,VHDL Matlab, Python, and R
- Shell Scripting: Bash, C Shell and Tcl Shell
- EDA Automatiseeritud tööriist: ModelSim, Synopsis, ICARUS Verilog, VCS
- Teadustarkvara paketid: Machine Learning Libraries, Verilog Procedural Interface (VPI) Functions, Datamining Libraries

8. Autasud

- Okt. 2021: **Outstanding Student Paper Award**, "Modeling Soft-Error Reliability Under Variability", 34th IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems

9. Kaitstud lõputööd

- 2016, Magistritöö
  International Audio Laboratories, Fraunhofer IIS, Erlangen, Germany
  Teema: Perceptual Evaluation of Magnitude and Phase Quantisation in Audio Coding
- 2011, Bakalaureusetöö
  Anna University of Technology, India
  Teema: Home Automation through 230v Power Lines Using X-10 Protocol

10. Teadustöö põhisuunad

- Komplekssete süsteemide kvaliteet ja töökindlus
- Tehnoloogiliste mööduvate rikete ja nende tagajärgede analüüs
- Süsteemitehnoloogia ja Arvutiteadused
- Mikroelektroonika

11. Teadustegevus
Teadusartiklid

1. A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, "The validation of graph model-based, gate level low-dimensional feature data for machine learning applications," in *2019 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, IEEE, oct 2019

2. A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, "Modeling gate-level abstraction hierarchy using graph convolutional neural networks to predict functional de-rating factors," in *2019 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, IEEE, jul 2019

3. A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu, and M. Jenihhin, "Composing graph theory and deep neural networks to evaluate SEU type soft error effects," in *2020 9th Mediterranean Conference on Embedded Computing (MECO)*, IEEE, jun 2020

4. M. Jenihhin, M. S. Reorda, A. Balakrishnan, and D. Alexandrescu, "Challenges of reliability assessment and enhancement in autonomous systems," in *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, IEEE, oct 2019

5. T. Lange, A. Balakrishnan, M. Glorieux, D. Alexandrescu, and L. Sterpone, "On the estimation of complex circuits functional failure rate by machine learning techniques," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks – Supplemental Volume (DSN-S)*, IEEE, June 2019

6. X. Lai, A. Balakrishnan, T. Lange, M. Jenihhin, T. Ghasempouri, J. Raik, and D. Alexandrescu, "Understanding multidimensional verification: Where functional meets non-functional," *Microprocessors and Microsystems*, vol. 71, p. 102867, nov 2019

7. A. Balakrishnan, D. Alexandrescu, M. Jenihhin, T. Lange, and M. Glorieux, "Gate-level graph representation learning: A step towards the improved stuck-at faults analysis," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pp. 24–30, 2021

8. D. Alexandrescu, A. Balakrishnan, T. Lange, and M. Glorieux, "Enabling cross-layer reliability and functional safety assessment through ML-based compact models," in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, IEEE, jul 2020

9. A. Balakrishnan, G. C. Medeiros, C. C. Gürsoy, S. Hamdioui, M. Jenihhin, and D. Alexandrescu, "Modeling soft-error reliability under variability," in *2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 1–6, ***Award: Outstanding Student Research Paper***, 2021

10. T. Lange, A. Balakrishnan, M. Glorieux, D. Alexandrescu, and L. Sterpone, "Machine learning to tackle the challenges of transient and soft errors in complex circuits," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, IEEE, jul 2019

11. T. Lange, A. Balakrishnan, M. Glorieux, D. Alexandrescu, and L. Sterpone, “Machine learning clustering techniques for selective mitigation of critical design features,” in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 1–7, 2020

12. X. Lai, T. Lange, A. Balakrishnan, D. Alexandrescu, and M. Jenihhin, “On antagonism between side-channel security and soft-error reliability in bnn inference engines,” in *2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 1–6, 2021

Konverentsi ettekanded

1. NASA/ESA Conference on Adaptive Hard-ware and Systems (AHS), IEEE, jul 2019, Colchester, United Kingdom

2. IEEE Nordic Circuits and Systems Conference (NOR-CAS): NORCHIP and International Symposium of System-on-Chip(SoC),IEEE,oct2019, Helsinki, Finland

3. 9th Mediterranean Conference on Embedded Computing (MECO), IEEE, jun2020, Budva, Montenegro

4. 22nd International Symposium on QualityElectronic Design (ISQED), 2021, California, USA

5. 34th IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, Greece

ISSN 2585-6901 (PDF)
ISBN 978-9949-83-805-9 (PDF)