



University of Thessaly  
Department of Electrical, Electronic and Computer Engineering  
Volos, Spring 2024

# Static Timing Analysis, Delay Calculation and Timing Sign-Off Algorithms for small scale (sub-40nm) Digital Electronic Circuits

Stavros Simoglou

Supervisors: Christos Sotiriou, Fotios Plessas, George Stamoulis

A thesis submitted in fulfillment of the requirements for the degree of Doctor  
of Philosophy

Αλγόριθμοι Στατικής Χρονικής Ανάλυσης,  
Υπολογισμού Καθυστέρησης και Χρονικής  
Εγγύησης Ψηφιακών Ηλεκτρονικών  
Κυκλωμάτων Μικρής Κλίμακας(κάτω των  
40nm)

## DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this Ph.D. dissertation, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

The declarant

A handwritten signature in black ink, appearing to be 'Stavros Simoglou', written over a horizontal line.

Stavros Simoglou

*Dedicated to my father and grandfather, the two sincerest people I have  
ever met.*

# Acknowledgements

Parts of this dissertation were supported by the Hellenic Foundation for Research and Innovation (HFRI) under the 3rd Call for HFRI PhD Fellowships (Fellowship Number: 6168).

Special thanks to my supervisor Prof. Christos Sotiriou for all 9 years of our collaboration and all Dissertation Committee Professors for their support and useful discussions.

This dissertation's findings are integrated within the University of Thessaly (UTH), CASlab in-grown EDA tool which is a result of collaborative effort so I would like to express my warmest appreciation and gratitude to all CASlab members for contributing to this in every possible way. Especially the former VLSI group members Nikolaos Blias, Giorgos Stanimeropoulos Iordanis Lilitsis and Ilias Gkolfos. It was an honor and privilege to lead this team for all these years. Also, I would like to acknowledge the most senior members of the team Dr. Nikolaos Sketopoulos, (soon to be) Dr. Christos Georgakidis and Mr. Dimitris Valiantzas for our collaboration towards achieving top-notch research results and creating, managing and maintaining top standards.

My sincerest gratitude to Dr. Ahmed Mahdi for contributing the best intuition, professionalism and inspiration to pursuit a career in microelectronics. Thanks to Paul Pereira and Tom Katsioulas for contributing high complexity ideas, intuition and perception of nowadays industry's needs. Thanks to our colleagues Prof. Milos Kristic and Dr. Marko Andjelkovic for our collaboration on the radiation induced effects analysis front, for providing us with the motivation and funding to explore this area and their significant contributions in the projects and the field, in general. Thanks to my current manager in Synopsys Inc. Mr. Adrian Wrixon for providing me with flexible working hours to write up my dissertation and my colleagues Dr. Anton Belov, Dr. Alan Benson

and Dr. Richard Moloney for their support, patience and faith in me.

Finally, my highest appreciation and gratitude to all friends, colleagues, family and my partner in life Sotiria for being there even during the toughest of times.

To any of you who contributed in any way to all of the above but are not mentioned, I would like to express my eternal gratitude and indebtedness.

# Abstract

Modern ASIC design flows are heavily dependent on the Quality of Results (QoR) of Static Timing Analysis (STA), as it has been the timing validation golden standard for decades. The ever pronounced parasitic phenomena, due to the excessive shrinking of process geometries, in conjunction with the ever increasing circuit complexity and scale, have been challenging the accuracy and capacity of the standard STA algorithms. One of the highest influence algorithms for STA QoR is Delay Calculation which also affects performance. This dissertation presents an industrial timing model compatible Delay Calculation methodology that achieves SPICE accurate results while being orders of magnitude faster than SPICE and thus is suitable for use with STA. Furthermore, as circuit reliability has become a major concern for modern mission critical application systems, this thesis presents two novel methodologies of radiation induced Single Event Transients analysis and estimation of circuit sensitivity. The presented methodologies are based on STA principles. One of them is based on industrial EDA tools and analyses the Propagation Induced Pulse Broadening (PIPB) effect. Contrary to that, the other is implemented within Circuits and Systems Laboratory (CASlab) in-house EDA tool and employs STA algorithms and the aforementioned Delay Calculation method to provide a global picture of circuit sensitivity. This can be considered the first step towards creating a generalized Radiation Hardened (RADHARD) ASIC design flow, as its capacity and performance allows its application within closed loop optimization flows. Finally, two of the resultant publications regarding SET analysis were honoured with the Best Paper Award in international IEEE conferences.

# Περίληψη

Οι σύγχρονες ροές σχεδιασμού ASIC εξαρτώνται σε μεγάλο βαθμό από την Ποιότητα των Αποτελεσμάτων (Quality of Results - QoR) της Στατικής Χρονικής Ανάλυσης (Static Timing Analysis - STA), καθώς αποτελεί την πλέον εδραιωμένη μέθοδο επαλήθευσης και επικύρωσης χρονισμού για δεκαετίες. Τα έντονα παρασιτικά φαινόμενα, λόγω της ολοένα και περισσότερης συρρίκνωσης των γεωμετριών των νανοηλεκτρονικών συσκευών, σε συνδυασμό με την αυξανόμενη πολυπλοκότητα των κυκλωμάτων, τόσο σε αριθμό συσκευών όσο και διασυνδέσεων, έχουν αποτελέσει πρόκληση για την επίτευξη της μέγιστης δυνατής ακρίβειας και ταχύτητας των τυπικών αλγορίθμων STA. Ένας από τους αλγορίθμους με την υψηλότερη επιρροή για την QoR της Στατικής Χρονικής Ανάλυσης είναι ο υπολογισμός καθυστερήσεων, ο οποίος επηρεάζει επίσης την απόδοση. Η παρούσα διατριβή παρουσιάζει μια μεθοδολογία υπολογισμού καθυστέρησης, η οποία είναι συμβατή με βιομηχανικά μοντέλα χρονισμού και επιτυγχάνει αποτελέσματα με μέγιστη ακρίβεια σε σύγκριση με την ηλεκτρική εξομοίωση SPICE, ενώ είναι τάξεις μεγέθους ταχύτερη και έτσι είναι κατάλληλη για τους σκοπούς της Στατικής Χρονικής Ανάλυσης. Επιπλέον, καθώς η αξιοπιστία των κυκλωμάτων έχει γίνει μια καίρια ανησυχία για τα σύγχρονα συστήματα mission-critical εφαρμογών, αυτή η διατριβή παρουσιάζει δύο νέες μεθοδολογίες ανάλυσης και εκτίμησης της ευαισθησίας των κυκλωμάτων όταν αυτά λειτουργούν σε περιβάλλοντα με ιοντίζουσα ακτινοβολία. Οι προτεινόμενες μεθοδολογίες βασίζονται σε αρχές STA. Η μία βασίζεται σε βιομηχανικά εργαλεία Αυτοματοποιημένης Ηλεκτρονικής Σχεδίασης (Electronic Design Automation - EDA) και αναλύει το φαινόμενο Propagation Induced Pulse Broadening (PIPB). Αντίθετα, η δεύτερη έχει υλοποιηθεί εντός του εργαλείου EDA του εργαστηρίου Κυκλωμάτων και Συστημάτων (CASlab) και χρησιμοποιεί αλγορίθμους STA και την προαναφερθείσα μέθοδο υπολογισμού καθυστερήσεων για να παρέχει μια ολοκληρωμένη εικόνα της ευαισθησίας του κυκλώματος. Αυτό μπορεί να θεωρηθεί το πρώτο βήμα προς τη δημιουργία μιας

γενικευμένης ροής σχεδιασμού ASIC τα οποία είναι ανθεκτικά στην ακτινοβολία, καθώς η αποτελεσματικότητα και η απόδοσή της επιτρέπουν την εφαρμογή της σε κλειστές ροές βελτιστοποίησης. Τέλος, δύο από τις δημοσιεύσεις που προέκυψαν σχετικά με την ανάλυση ευαισθησίας τιμήθηκαν με το Βραβείο Καλύτερου Άρθρου σε διεθνή συνέδρια του IEEE.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Background</b>	<b>6</b>
2.1	Industrial Timing Models . . . . .	6
2.1.1	Non-Linear Delay Model (NLDM) . . . . .	6
2.1.2	Composite Current Source Timing Model (CCS) . . . . .	7
2.2	Static Timing Analysis . . . . .	10
2.2.1	STA Basics . . . . .	11
2.2.2	STA Algorithms . . . . .	14
2.2.2.1	Delay Propagation . . . . .	14
2.2.2.2	Required Arrival Time (RAT) Propagation . . . . .	15
2.3	Differential Equations and Control Systems Theory . . . . .	18
2.3.1	Passive Circuits and Scientific Representation . . . . .	18
2.3.2	Interconnect Voltage Propagation Fundamentals . . . . .	22
2.3.2.1	Time Domain . . . . .	22
2.3.2.2	Frequency Domain . . . . .	23
2.4	Radiation Induced Glitches - Single Event Transients Generation Mechanisms	25
2.5	Single Event Transients Propagation and Masking Mechanisms . . . . .	27
2.5.1	Electrical Masking . . . . .	27
2.5.2	Timing Window Masking . . . . .	28
2.5.3	Logical Masking . . . . .	29
<b>3</b>	<b>Related Work</b>	<b>31</b>
3.1	Delay Calculation . . . . .	31
3.1.1	Interconnect Delay Calculation . . . . .	34
3.2	Single Event Transients . . . . .	38
<b>4</b>	<b>Proposed Delay Calculation Methodology for STA</b>	<b>40</b>
4.1	Driver Side Delay and Slew Calculation . . . . .	40
4.1.1	Dynamic Capacitance . . . . .	43
4.2	Interconnect Signal Propagation . . . . .	43
4.2.1	Input Voltage Waveform Analysis . . . . .	45
4.2.2	Pi-model Stage Full Waveform Propagation . . . . .	47
4.2.3	RC Tree Interconnect Stage Full Waveform Propagation . . . . .	48
4.3	Generalized Dynamic Capacitance and Driver Voltage Waveform Calculation	50
4.4	Receiver Delay And Slew Approximation . . . . .	53
<b>5</b>	<b>Novel SET Analysis Approaches using STA Principles</b>	<b>57</b>

---

5.1	The SET Analysis Topic Trade-Offs . . . . .	57
5.2	Particle Strike Scenario . . . . .	58
5.2.1	SET Pulse Conversion to STA Compatible Format . . . . .	59
5.3	Glitch Generation and Path Based Analysis Propagation using Industrial Tools . . . . .	60
5.3.1	Particle Strike Scenario Generation . . . . .	62
5.3.1.1	SET Generation . . . . .	62
5.3.1.2	SET Propagation . . . . .	64
5.4	Integrated Glitch Generation and Graph Based Propagation using In-House EDA Tool . . . . .	65
5.4.1	SET Generation at a Target Node . . . . .	65
5.5	GBA-based SET Propagation . . . . .	66
<b>6</b>	<b>Experimental Results</b>	<b>71</b>
6.1	Delay Calculation . . . . .	71
6.1.1	Pi-models, TAU2020 Benchmarks . . . . .	72
6.1.2	Arbitrary RC Interconnects, Approximate Method . . . . .	74
6.2	SET Generation And Propagation . . . . .	76
6.2.1	Industrial Tools PBA-based Flow . . . . .	76
6.2.2	Integrated GBA-based Flow . . . . .	79
<b>7</b>	<b>Conclusion</b>	<b>82</b>
7.1	Publications . . . . .	83
7.1.1	Delay Calculation . . . . .	83
7.1.2	Single Event Transients Analysis . . . . .	83
7.1.3	Miscellaneous . . . . .	84
7.2	Awards . . . . .	84
7.3	Future Work . . . . .	85
	References . . . . .	86

# List of Figures

2.1.1 Example Cell Rise Delay Timing Arc . . . . .	8
2.1.2 Example Current Vector . . . . .	9
2.1.3 Example CCS Single Stage and its Corresponding Electrical Equivalent Model . . . . .	10
2.2.1 Example Synchronous STA Benchmark . . . . .	11
2.2.2 Example Synchronous STA Benchmark Timing Graph . . . . .	12
2.2.3 Master-Slave Flip-Flop . . . . .	12
2.2.4 Flip-Flop Setup ( $t_s$ ) and Hold ( $t_h$ ) Time . . . . .	13
2.2.5 Flip-Flop Timing Arcs . . . . .	13
2.3.1 Simple RLC circuit with attached Voltage Source Excitation . . . . .	21
2.3.2 LTI System's Influence to Excitation Input Signal . . . . .	22
2.4.1 Charged Particle Strike Effect on Simple CMOS Inverter Cross Section . . . . .	25
2.5.1 SET Pulse Electrical Masking . . . . .	27
2.5.2 SET Pulse Timing Window Masking . . . . .	29
2.5.3 SET Pulse Logical Masking . . . . .	30
3.1.1 Non-Continuous Effective Capacitance Regions . . . . .	33
3.1.2 Arbitrary RC Interconnect . . . . .	35
3.1.3 Arbitrary RC Interconnect Tree Representation . . . . .	35
4.1.1 Circuit setup for NLDM and CCS characterization . . . . .	40
4.2.1 PWL Signal to Sum of Ray Equations Analysis . . . . .	47
5.2.1 SET Generation Current Source . . . . .	59
5.2.2 Double-Exponential (DEXP) Typical Current Waveform . . . . .	59
5.2.3 SET pulse decomposition to rise and fall edges for STA engine compatibility . . . . .	60
5.3.1 Industrial tools SET Generation And PBA-based Propagation Flow . . . . .	61
5.4.1 Equivalent Analog Model used during the custom local simulation of SET generation . . . . .	67
5.5.1 Electrical Masking . . . . .	68
6.1.1 Proposed Methodology Driver Waveform VS SPICE for Stage with Arbitrary RC Interconnect. . . . .	76
6.2.1 SET Monte-Carlo Charge Distribution . . . . .	77
6.2.2 90 <sup>th</sup> Worst Delay Timing Path . . . . .	77
6.2.3 90 <sup>th</sup> Worst Delay Timing Path SPICE Waveform Plots . . . . .	78
6.2.4 SET Analysis Runtime Distribution . . . . .	80

# List of Tables

6.1.1 Traditional Effective Capacitance [1] VS Proposed Methodology for Stages with Balanced Pi-Model Interconnect. . . . .	72
6.1.2 Traditional Effective Capacitance [1] VS Proposed Methodology for Stages with Non Balanced Pi-Model Interconnect, $C_1 = 0.1C_2$ . . . . .	74
6.1.3 Traditional Effective Capacitance [1] VS Proposed Methodology for Stages with Non Balanced Pi-Model Interconnect, $C_1 = 0.01C_2$ . . . . .	74
6.1.4 Proposed Methodology VS SPICE for Stages with Arbitrary RC Interconnect	75
6.2.1 Examined Particle Profiles . . . . .	79
6.2.2 Benchmarks Characteristics . . . . .	80
6.2.3 Exhaustive SET Analysis Results . . . . .	81

# List of Algorithms

1	<i>propagate_delay</i> . . . . .	16
2	<i>update_delay</i> . . . . .	17
3	<i>propagate_RAT</i> . . . . .	19
4	<i>update_RAT</i> . . . . .	20
5	<i>voltage_vector</i> . . . . .	41
6	<i>calculate_driver_delay_slew</i> . . . . .	44
7	<i>calculate_net_dynamic_capacitance</i> . . . . .	51
9	<i>Newton's_Method</i> . . . . .	54
8	<i>calculate_driver_waveform</i> . . . . .	56

# Chapter 1

## Introduction

Sign-off Static Timing Analysis (STA) has emerged as the cornerstone standard for timing validation prior to the fabrication stage, a trend that has been firmly established since the advent of Electronic Design Automation (EDA) in the field of Application-Specific Integrated Circuit (ASIC) design [2]. The dynamics of this domain are multifaceted. On one front, the digital ASICs market, characterized by its ever-growing demand, compels ASIC designers to integrate an increasingly large number of instances within a single System on Chip (SoC). This upsurge in complexity and scale makes the capacity of STA engines a critical factor, as they must handle designs of immense magnitude and intricacy.

Concurrently, on another front, there is an unyielding drive to enhance the computational capabilities of these SoCs. This drive often manifests in a push towards end products that are optimized for lower power consumption or reduced area. Such an evolution has necessitated the adoption of progressively smaller process nodes. However, this miniaturization brings its own set of challenges, particularly in the realm of parasitic phenomena, which become more prominent at these scales. A critical aspect of this challenge is that while silicon-based devices generally scale down effectively, and process geometries continue to shrink, the same level of scalability is not observed in metal interconnections. Reducing the size of these interconnections without compromising their conductive and structural integrity is a significant challenge. Consequently, parasitic elements such as interconnect resistance or Miller capacitance [3] assume greater significance. These elements substantially affect the Quality of Results (QoR) in STA.

To address these escalating complexities, the semiconductor industry has invested heavily in developing robust methodologies and advanced tools for accurate STA. This investment has culminated in the development of sophisticated industrial EDA tools capable of estimating timing information with a high degree of accuracy. However, a notable caveat is that the algorithms and methodologies underpinning these tools are typically proprietary, shrouded in confidentiality, and not disclosed to the public domain. Parallel to these developments, the academic and technical literature has seen the proposal of various driver cell delay models. Some of these models have gained traction and been adopted as industry standards. Yet, to date, there appears to be a lack of a comprehensive and conclusive publication that addresses SPICE accurate STA, particularly in the context of arbitrary RC interconnect shapes, without resorting to additional characterization or benchmark classification. Here, 'conclusive' is defined as a publication that integrally combines elements such as driver cell delay calculation, the impact of arbitrary RC interconnect, the influence of the receiver model, and the full waveform propagation through the arbitrary interconnect to the receiver.

Except from circuit validity, another domain which has started gaining traction in recent years is circuit reliability. On the one hand, the recent space race between worldwide enterprises after the commercialization of cargo and manned space missions has been driving a high demand of mission critical systems, which in turn creates demand for more and more complex and capable electronics. On the other hand, the recent worldwide geopolitical tensions between opposing and competing superpowers is driving ever increasing needs for high altitude or space systems, purposed for surveillance, secure communication, and strategic military objectives. This again, causes great demand for more modern capable and reliable electronics. Parallel to these, the extremely significant extent of climate change effects on humanity have created a surge of investment towards environment friendly large scale power production, in an attempt to reduce greenhouse-effect gas emissions as much as possible. Part of these funds is either contributing in the creation and operation of nuclear power plants or research on harnessing nuclear fusion energy and creating financially viable nuclear fusion power production. For such power plants, digitally powered monitoring and control is the only way of operation, both for safety and efficiency reasons.

Reliability is a highly, if not the most, significant factor in all of the systems mentioned above, as these systems are required to operate correctly in harsh environment conditions, like orbiting outside of earth's magnetic field within Van Allen radiation belts, within nuclear power plants, or even in high radiation research projects like particle accelerators. In such environments, radiation, in the form of electromagnetic interference or high energy charged particles, can affect the normal functionality of digital electronics by penetrating the package and creating charge excitation which in turn usually creates bit flips. These may cause all kinds of errors in digital ASICs, ranging from simple and relatively insignificant miscalculation of arithmetic quantities up to complete loss of synchronization between pipeline stages, blocks, communication, etc.

The most common practice of dealing with the overall reliability problem is using multiple identical ASICs per calculation and taking a majority vote, as the probability of simultaneous failure is close to zero, if the chips are placed far enough from each other. This has served well multiple safety critical systems in the past, as reliability was the top priority, while throughput and power efficiency was not of interest at the time. However, this practice can be considered impractical in the modern electronics era we live in, as using three or more identical chips and taking their majority for the purpose of redundancy, requires significant amount of time being spent for inter-chip communication as well as significant amounts of power, which is not always available. For example, a block level Triple Modular redundancy (TMR) system requires more than 3x the power consumed by a single block. Thus, industry has shifted towards experimenting with the introduction of fine grain modular redundancy practices, which target specific circuit components within blocks instead of multiplying the blocks themselves. Such methods multiply Flip-Flops, Latches and Memories, and also introduce majority gates and filters in order to ensure that the aforementioned transient phenomena are handled properly. It is worth noting that, while permanent faults can only be handled by designing more robust devices and interconnections at the standard cell design phase, temporary fault effects, which are usually called soft errors, can be addressed using the above practices within the ASIC backend flow.

Soft errors are a major factor leading to malfunctions in electronic systems deployed in radiation-prone environments. As reported by [4], radiation-induced effects are responsible

---

for over 40% of the anomalies observed in space electronics, with soft errors being the predominant cause, accounting for more than 80% of these issues. In older technologies, soft errors were primarily due to direct particle hits to memory or sequential elements, leading to unintended bit-flips, a phenomenon referred to as Single Event Upsets (SEUs). However, with the progression of technology, another notable source of soft errors has emerged: voltage disturbances in combinational logic triggered by radiation, known as Single Event Transients (SETs). These SETs have increasingly contributed to the total soft error rate in Integrated Circuits (ICs). If the amplitude and duration of these SETs are substantial, they can propagate through the circuit, eventually being registered by memory or sequential logic, thereby causing soft errors. Thus, addressing the impacts of SETs is of paramount importance in the design process of ASICs intended for use with mission critical systems for radiation-prone environments.

Characterizing Single Event Transients (SETs) for digital ASICs is a challenging procedure, due to their analog characteristics. For precise outcomes, it is essential to take into account both the generation and propagation mechanisms of SETs. Presently, the most sophisticated method for examining soft errors is through comprehensive fault injection simulations, utilizing SPICE electrical simulation or Hardware Description Language (HDL) based tools. Nonetheless, these methods present two primary drawbacks. Firstly, conducting fault injection simulations using SPICE is impractical for large-scale designs, due to electrical simulation computational complexity and the need for input vectors. Secondly, even when using HDL simulations, the process can become excessively time-consuming, especially for circuits consisting of millions of gates.

Significant efforts have been made by academia to result in a time-efficient and accurate method for evaluating SETs impact on digital electronics. In general, two approaches have been presented, namely fully a combination of simulations and analytical methods and fully analytical analysis. On the one hand, the former rely either on mathematical or algorithmic models, including Binary Decision Diagrams (BDD)/Algebraic Decision Diagrams (ADD), [5, 6], Boolean satisfiability theory[7, 8, 9] and Error Propagation Probability (EPP) [10, 11, 12, 13]. On the other hand, the purely analytical approaches are better than the former in evaluating logical masking and are typically faster. Unfortunately though, they totally disregard physical and electrical properties of the underlying circuit under

investigation, which in turn leads to lower overall accuracy compared both to simulation-based methods and reality. To cope with these, attempts have been made to create a combination of simulation-based analysis and analytical methods [14, 15, 16, 17]. The proposed methods, in general incorporate SET generation analysis employing SPICE simulations for standard cells only, and then using these results for analytical evaluation of SET propagation. However, it is evident that these approaches suffer from various inaccuracies as simplified SET injection models are typically utilized.

This thesis is separated in two strongly related parts. The first part is about Static Timing Analysis and Delay Calculation using Current Source Models (CSMs) and arbitrary RC interconnect. Also, basic STA algorithms are presented in this part for shake of completeness. The second part is about SETs generation and propagation using both commercial EDA tools and a methodology which is implemented within an academic EDA tool as part of a bigger project, which is currently funded and used by the European industry. It is worth noting that both parts' scientific contribution has been internationally recognized as they resulted in two best paper awards in IEEE conferences. On top of that, the underlying research performed before the PhD endeavour detailed in this dissertation, contributed in winning the TAU 2020 timing contest award.

The next chapters are organized as follows. Chapter 2 provides an overview of all the required fundamental terms and concepts regarding STA, Delay Calculation and its underlying mathematical formulation as well as SET generation and propagation. Chapter 3 provides a brief overview of all relevant publications to this thesis, *i. e.* all works that are related to Delay Calculation and SETs analysis. Chapter 4 contains all proposed methods and concepts for SPICE accurate stage Delay Calculation for STA. Chapter 5 presents the proposed methodologies for SETs generation and propagation using commercial EDA tools as well as the integrated STA engine based approach mentioned above. Finally, all experimental methodologies and results are presented in Chapter 6 and conclusions are drawn in Chapter 7.

# Chapter 2

## Background

This chapter is an overview of all fundamentals regarding Static Timing Analysis and Delay Calculation as well as radiation effects on modern VLSI digital circuits. All described examples are presented, without loss of generality, for longest path (max) analysis and for register-based synchronous designs, meaning that all sequential elements used are Flip-Flops.

### 2.1 Industrial Timing Models

For the past few recent decades, various timing models have been developed to model the delay and transition\_time/slew in digital circuits, as well as to perform timing checks for sequential elements and enforce Design Rule Constraints (DRC). These models generally fall into two primary categories: Voltage Response Models (VRM) and Current Source Models (CSMs). The industry has widely adopted the Non-Linear Delay Model (NLDM) as its VRM of choice. Meanwhile, there are two main types of industrial CSMs: the Composite Current Source Timing model (CCS) and the Effective Current Source Model (ECSM). This study specifically concentrates on the CCS model for its analysis, without loss of generality.

#### 2.1.1 Non-Linear Delay Model (NLDM)

Starting with the NLDM, which is an older and simpler yet well adopted model, it conceptualizes the driver cell's behavior as a voltage source. This response is determined

by the slew at the driver timing arcs's input pin and the capacitive load at the output pin, with the receiver cell being represented as a lumped capacitor. The delay and slew of the driver cell's output pin are stored in two-dimensional (2D) Look-Up Tables (LUTs). These LUTs are indexed by the transition time at the timing arc input pin and the total capacitance of the output net. When the input slew and output load values of the said stage under investigation do not align perfectly with the LUT index grid points of precharacterization data bilinear interpolation is employed to calculate the value. The resultant value, is either the delay or slew, depending on which LUT is accessed.

Additionally, the most detailed form of receiver capacitance in NLDM is provided using attributes `rise_capacitance_range` and `fall_capacitance_range`, which store the capacitance values for rise and fall signal transitions, applicable to either minimum or maximum delay analysis. While this model has been a standard for decades, the evolution of fabrication technologies and the shrinking of electronic circuit device geometries have rendered parasitic electrical phenomena more pronounced, which introduce all kinds of complications in delay calculation, and thus NLDM's accuracy is significantly hindered. Such phenomena are, among others, interconnect coupling, intra-cell nodes coupling and Miller capacitance as well as increased interconnect resistance .

In the context of delay calculation using NLDM, the `.lib` file stores delay and slew data as two-dimensional Look-Up Tables (LUTs). These tables are organized by `input_net_transition` and `total_output_net_capacitance` parameters, and the measurements are calculated and stored during precharacterization. In case the actual stage `input_net_transition` and `total_output_net_capacitance` do not exactly match the index grid points, interpolation is applied to deduce delay and slew for the given loading conditions. An illustration of how a `cell_rise` arc appears in the `.lib` file and its graphical representation is shown in Figure 2.1.1. Consequently, the calculation of driver delay and slew is performed solely by LUT indexing and interpolation, with the receiver modeled as a single lumped capacitance value.

### 2.1.2 Composite Current Source Timing Model (CCS)

In contrast, CCS conceptualizes the driver's response as a current source characterized by infinite resistance, and dependency on time and voltage. The current

```

cell_rise(template) {
  index_1 ("1, 2");
  index_2 ("0.1, 0.2");
  values ("1.1, 1.2", \
          "2.1, 2.2");
}

```

	1	2
0.1	1.1	1.2
0.2	2.1	2.2

**Figure 2.1.1:** Example Cell Rise Delay Timing Arc

information is encapsulated in `current_vector` Look-Up Tables (LUTs), which detail the current versus time waveforms. Each `current_vector` is associated with a specific `input_net_transition` and `total_output_net_capacitance` from the precharacterization phase. Furthermore, every current vector includes a `reference_time` parameter, which represents the delay voltage threshold crossing time of the pre-driver input normalized driver waveform used when characterization is performed.

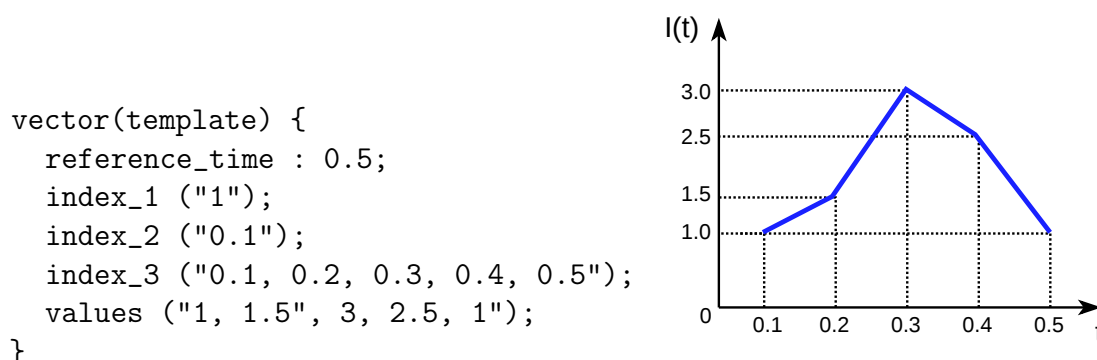
To accurately model Miller capacitance and other receiver input electrical phenomena, receiver capacitance is characterized by two or more distinct 2D LUTs. These tables are indexed by the `input_net_transition` and `total_output_net_capacitance` of the receiver cell. Specifically, there are multiple `receiver_capacitance_rise` and `receiver_capacitance_fall` values, each corresponding to different segments of the waveform. This is more of a legacy model nowadays, as more modern CCS Noise (CCSN) libraries typically contain more than two `receiver_capacitance` regions. However, the only libraries that were available while undertaking this research dissertation contained only two regions for receiver capacitance.

Direct calculation of delay or slew from the current waveform is obviously impractical due to its non-monotonic nature. However, by integrating this waveform and dividing by the capacitance, one can reconstruct the voltage waveform using formula:

$$V(t) = \frac{Q(t)}{C} = \frac{\int i(t)dt}{C}, \quad (2.1.1)$$

where  $V(t)$  is the voltage waveform,  $Q(t)$  is charge which is equal to the integral of current waveform  $i(t)$ , and  $C$  is current vector's `total_output_net_capacitance` value. Subsequent steps involve determining the time instants for lower slew, delay, and upper slew, based on voltage thresholds. Similar to the approach with NLDLM, interpolation

between closest current vector time points is utilized to accurately determine delay and slew. On the receiver side, the driver's waveform slew must be propagated to the receiver's input in order to calculate the receiver capacitance values., This clearly presents an interdependence between driver waveform points calculation and receiver capacitance, as the former depends on the capacitance as 'seen' by the driver, while the latter depends on slew at receiver input, which in turn relies on driver output waveform slew. This process necessitates the use of an iterative procedure until slew converges to a specific value. The aforementioned are thoroughly documented in the related standards and industrial tool manuals [18, 19, 20].

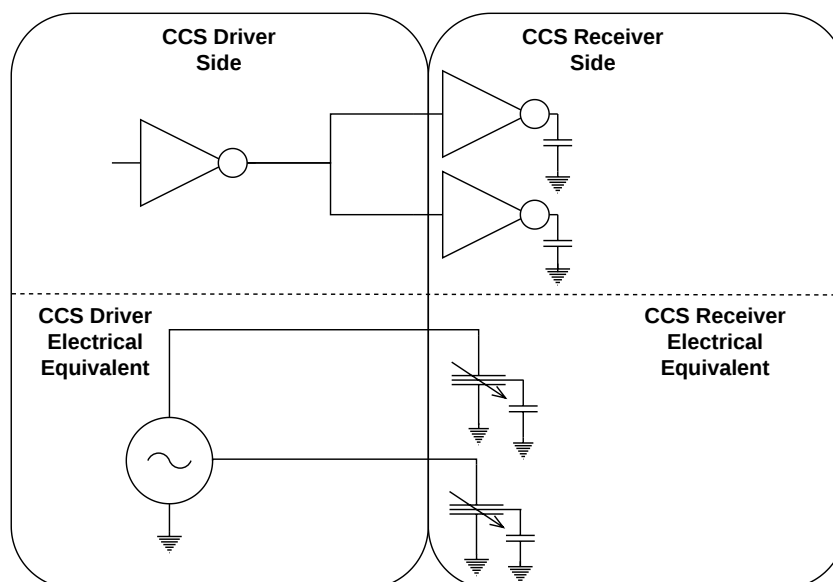


**Figure 2.1.2:** Example Current Vector

The CCS model encapsulates information about the current waveform at the driver output gate pins, storing these waveforms as current vector LUTs. Each LUT entry comprises a pair consisting of `input_net_transition` and `total_output_net_capacitance`, representing the precharacterization loading conditions. Additionally, a `reference_time` is specified to denote the time instant when the input pre-driver waveform crosses the delay threshold voltage, usually 50% of the supply voltage. The current vector of course contains as well the current waveform itself in form of the time index and current values. As illustrated in Figure 2.1.2, `index_1` is associated with `input_net_transition`, `index_2` with `total_output_net_capacitance`, and `index_3` is always storing time. The integration of  $I(t)$  over time, followed by division by the `total_output_net_capacitance`, yields the voltage waveform. This process enables determining the time instants corresponding to the lower and upper slew voltage thresholds, as well as the delay threshold.

Figure 2.1.3 illustrates a configuration consisting of a single-stage driver inverter connected

to a pair of receiver inverters. To accurately model the Miller effect, the CCS timing model represents the receiver's behavior as a variable capacitor, which depends on the receiver's `input_net_transition` and `total_output_net_capacitance`. Consequently, it is imperative to propagate the driver's slew information to the receiver's pin to compute the receiver's capacitance. This establishes a clear mutual dependency between the driver's slew and the receiver's capacitance. To address this complexity, an iterative approach is employed, continuing until driver slew converges. Additionally, the `.lib` standard mandates a minimum of two segments for receiver capacitance to adequately detail both the driver's response and the receiver's behavior. Accordingly, the computation of time instants at the slew and delay threshold voltages is conducted separately for the lower and upper portions of the waveform. Specifically, the lower slew and delay thresholds utilize `receiver_capacitance1`, while the upper slew threshold uses `receiver_capacitance2`.



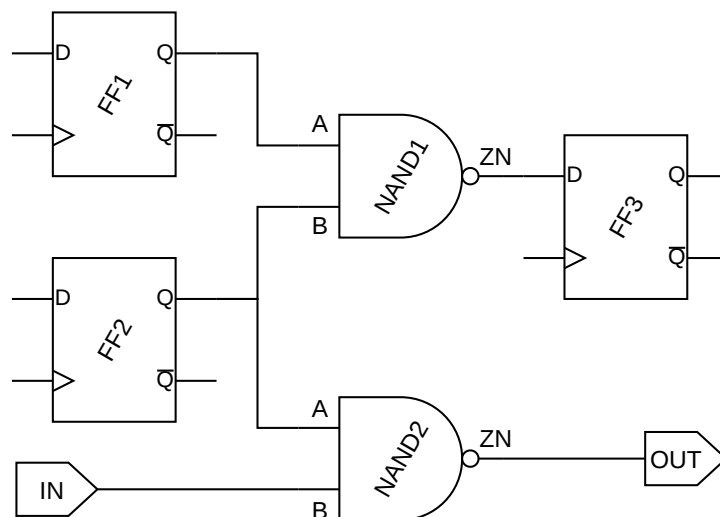
**Figure 2.1.3:** Example CCS Single Stage and its Corresponding Electrical Equivalent Model

## 2.2 Static Timing Analysis

This section briefly presents the basic principles, procedures and fundamentals of Static Timing Analysis (STA) algorithms and timing checks.

### 2.2.1 STA Basics

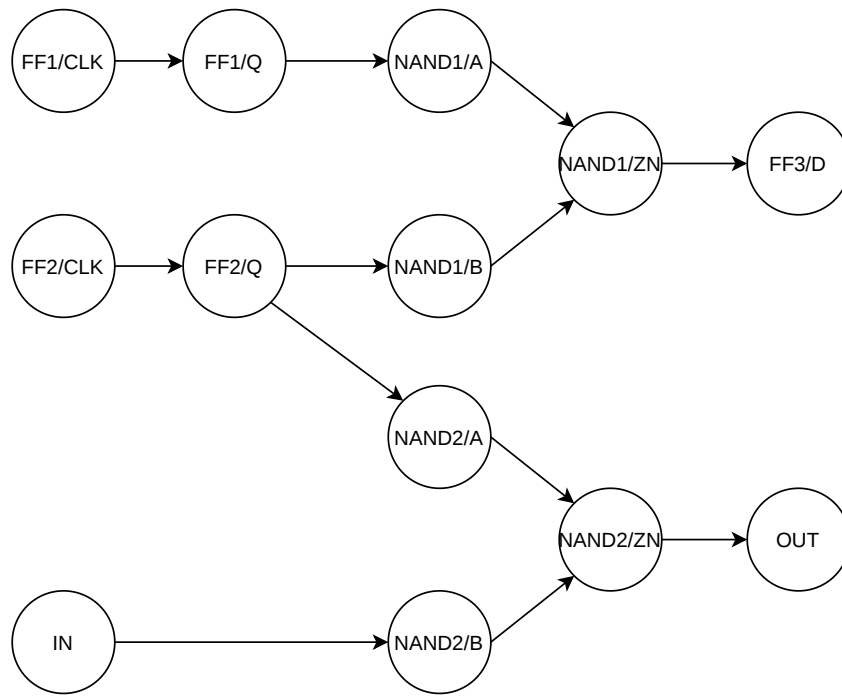
Modern Static Timing Analysis (STA) operates on the principle that the timing graph is a Directed Acyclic Graph (DAG). In this context, the nodes of the timing graph represent the circuit pins, and the edges are the timing arcs, which are input to output timing relations, and the connections between them. The start points of timing paths are marked by data launching pins which are Flip-Flop outputs/clock inputs and top-level Primary Inputs, whereas the timing path end points are identified by Flip-Flop inputs and top-level Primary Outputs. An example circuit and its corresponding timing graph are illustrated in Figures 2.2.1 and 2.2.1, respectively. For max analysis, longest path analysis is conducted on the timing graph, whereas a shortest path analysis is utilized for min analysis [2].



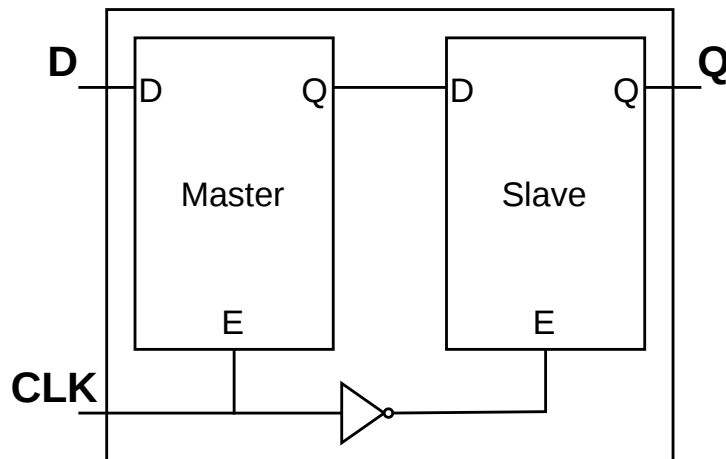
**Figure 2.2.1:** Example Synchronous STA Benchmark

Additionally, the timing checks for sequential elements are performed using the information annotated on the timing graph. The necessary data for these timing checks are included by .lib files. The purpose of conducting these checks is to verify the correct operation of the sequential elements. The following paragraphs, provide a brief description of these timing checks purpose and procedure.

Flip-Flops are typically implemented using a dual latch Master-Slave configuration, as depicted in Figure 2.2.3. This architectural configuration introduces specific timing requirements between the clock and data signals. There is a specific timing window prior to the arrival of the clock signal at the clock pin, during which the data must remain stable. This requirement is critical to guarantee that data is accurately transferred from



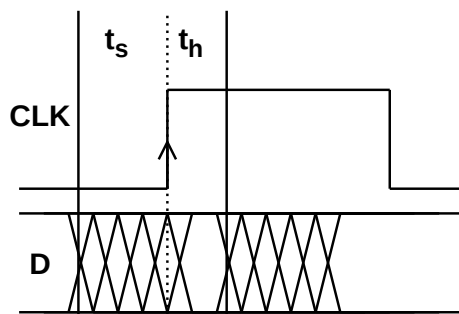
**Figure 2.2.2:** Example Synchronous STA Benchmark Timing Graph



**Figure 2.2.3:** Master-Slave Flip-Flop

the Master latch to the Slave latch, enabling the latter to correctly capture and store the data. This period is defined as the setup time of the Flip-Flop, and it always exists before the triggering edge of the Flip-Flop. In a similar manner, the data must also remain stable for a brief period following the clock edge to ensure the proper data latching. This timing margin is defined as the hold time of the Flip-Flop. Figure 2.2.4 graphically illustrates the setup and hold times.

In addition, recovery and removal timing checks are necessitated by the use of asynchronous reset in Flip-Flops, which are enabled by asynchronous preset and clear pins. To guarantee

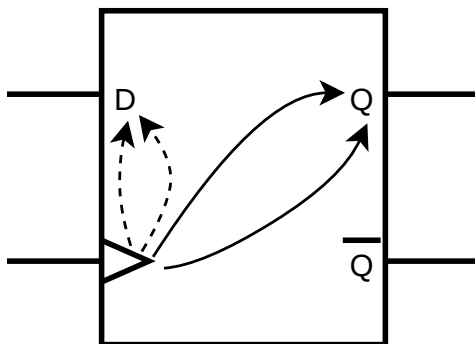


**Figure 2.2.4:** Flip-Flop Setup ( $t_s$ ) and Hold ( $t_h$ ) Time

the proper functionality of these asynchronous operations, recovery and removal checks are essential [2].

Timing checks are also essential in the context of Latch based designs. Latches are also susceptible to setup and hold time violations. In fact they are the reason why setup and hold timing checks are imposed on Flip-Flops as Latches are the building blocks of them. These time margins are defined relative to the latching edge—namely, the clock edge at which the Latch captures the data. This edge corresponds to the falling edge for active-high Latches or the rising edge for active-low Latches [2].

As previously discussed, the timing relationships within components are modelled using timing arcs. Consequently, the cell rise and fall delays between an input and an output pin, as well as slew and setup and hold times, are characterized using timing arcs. Figure 2.2.5 illustrates the timing arcs of a D Flip-Flop sequential element, where dashed arrows represent setup check arcs and solid arrows denote delay arcs.



**Figure 2.2.5:** Flip-Flop Timing Arcs

## 2.2.2 STA Algorithms

As detailed in Section 2.2.1, the circuit is modeled as a timing graph for the purposes of STA. The timing graphs that model synchronous digital circuits are structured as Directed Acyclic Graphs (DAGs), which justifies employing graph theory for their analysis. The subsequent subsections detail the methodologies for the propagation of delay, slew, and Required Arrival Time (RAT) to all timing graph nodes.

### 2.2.2.1 Delay Propagation

The delay and slew at the driver output depend on the `input_net_transition` and the `total_output_net_capacitance`. Thus, in order to compute the delay and slew at the driver output, the slew information of preceding input standard cell pins must be available. This can be interpreted in two different ways, one being that either multiple passes on the output pin are required - one per timing arc -, which is inefficient, the other being that the output pin must be visited once, after all inputs have been visited. Therefore, a strategy that employs levelization is optimal, ensuring that standard cell output pins are processed only after all their predecessor input pins have been addressed. This procedure is naturally facilitated by employing a Topological Sort traversal [21].

Algorithm 1 outlines the methodology for delay propagation utilizing Topological Sort. Specifically, the fastest Topological Sort traversal is performed by employing two FIFO queues: one for the nodes currently being traversed and another for their traced successor nodes. Successor nodes are enqueued only after all their preceding nodes have been visited, a principle which is the practical application of the Topological Sort Criterion. This procedure is presented in detail within Algorithm 1. The algorithm takes a timing graph  $G$  as input and annotates all nodes with delay and slew, resulting in a timing graph  $T$  which is annotated with delay and slew.

The initial steps (Lines 5-10) initialize the delay and slew of the timing graph to zero. Sequential component outputs and Primary Inputs are then enqueued into the successors queue  $S$ . Subsequently, as long as the successors queue is not empty, a levelization step is undertaken: the current nodes are replaced with their successors. This step involves visiting each current standard cell pin to identify their successors and predecessors, which are essential for calculating the `total_output_net_capacitance`

and the `input_net_transition` needed by the delay and slew annotation function `timing_annotate_output_gatepin()`. Following this, each successor input is traversed, with delay and slew propagated to it via the wire delay and slew annotation function `timing_annotate_successor()`. If all predecessors of a successor output have been visited, fulfilling the Topological Sort Criterion, this successor output is enqueued into the Topological Sort Successors queue  $S$ . This procedure is repeated until the timing graph  $T$  is fully annotated with delay and slew.

In instances requiring delay updates, Breadth First Search (BFS) proves to be a more appropriate traversal method compared to Topological Sort. This preference arises because, in scenarios where delay or slew needs to be recalculated starting from a specific standard cell pin onwards, adhering to the Topological Sort Criterion blocks the traversal. This is due to the fact that Topological Sort Criterion cannot be satisfied as the timing graph is being traversed in a similar manner with tracing logic cones, rather than progressing level by level. Algorithm 2 demonstrates the process of updating delay. Notably, this algorithm mirrors the delay and slew propagation mechanism, with the primary difference with Algorithm 1 being the absence of the Topological Sort Criterion check. Additionally, the traversal is terminated once no further delay and slew updates are required.

It is important to highlight that during the course of this research project, an alternative incremental update traversal method was discovered. This method employs a BFS traversal, as previously mentioned, but uniquely leverages the Topological Sort levels as they were calculated by the full timing annotation algorithm beforehand, to sequentially visit all nodes. This approach allows the BFS traversal to function analogously to a Topological Sort, which, under certain conditions, can achieve up to x10 speedup.

### 2.2.2.2 Required Arrival Time (RAT) Propagation

The propagation of Required Arrival Time (RAT) is crucial for calculating the global slack, i.e., the slack at all circuit standard cell pins and IOs. To this end, a backward Topological Sort traversal is utilized. Algorithm 3 describes this procedure. Initially, at lines 5-9, input pins of sequential elements and Primary Outputs (PO) are added to the RAT predecessors queue  $P$ , and the RAT for all pins is set to positive infinity. This initialization step is of great importance since, for longest path analysis, RAT annotation

**Algorithm 1:** *propagate\_delay*


---

```

1 Input: Timing Graph  $G = (V, E)$ 
2 Output: Delay, Slew Annotated Timing Graph  $T = (V, E, delay, slew)$ 
3  $C = \emptyset$ ; // Current Gate Pins FIFO //
4  $S = \emptyset$ ; // Successor Gate Pins FIFO //
5 for (each Node  $v \in V$ ) do
6    $T.delay(v) = 0$ ;
7    $T.slew(v) = 0$ ;
8   if ( $v.is\_PI()$  ||  $v.is\_sequential\_output()$ ) then
9      $S.enqueue(v)$ ; // insert node to successors //
10  end
11 end
12 while ( $S \neq \emptyset$ ) do
13    $C = S$ ; // replace current with successor nodes //
14   while ( $C \neq \emptyset$ ) do
15      $v = C.dequeue()$ ; // get output pin from current //
16      $N = get\_successor\_input\_pins(v, E)$ ; // get successor input pins //
17      $P = get\_predecessor\_input\_pins(v, E)$ ; // get predecessor input pins //
18     if ( $N \neq \emptyset$ ) then
19        $timing\_annotate\_output\_gatepin(v, P, N, T.delay(v), T.slew(v))$ ;
20       for (each Successor Input  $n \in N$ ) do
21         // annotate successor, account for wires too //
22          $timing\_annotate\_successor(v, n, T.delay(n), T.slew(n))$ ;
23         if ( $n.is\_combinational()$ ) then
24            $O = get\_successor\_outputs(n)$ ; // trace successor outputs //
25           for (each Successor Output  $o \in O$ ) do
26             // TopoSort Criterion //
27             if ( $all\_predecessors\_visited(o)$ ) then
28               // insert traced output to successors //
29                $S.enqueue(o)$ ;
30             end
31           end
32         end
33       end
34     end
35   end
36 end

```

---

is performed to derive the worst-case RAT, which is the minimum value. Subsequently, the RAT at Flip-Flop inputs needs to be determined. The RAT at these inputs is calculated as the capturing clock edge arrival minus the setup time of the Flip-Flop, with both setup and hold times depending on the slew at the data pin and the clock pin. Therefore, at lines 10-16, the Flip-Flop's clock pin is identified, setup time is determined based on the slews at the clock and data pins, next capturing edge, (for shake of simplicity equal to

**Algorithm 2:** *update\_delay*


---

```

1 Input: Delay, Slew Annotated Timing Graph  $T = (V, E, delay, slew)$ , Update Start
  Point Pins  $U$ 
2 Output: Updated Delay, Slew Annotated Timing Graph  $T = (V, E, delay, slew)$ 
3  $C = \emptyset$ ; // Current Gate Pins FIFO //
4  $S = \emptyset$ ; // Successor Gate Pins FIFO //
5 for (each Node  $u \in U$ ) do
6   |  $S.enqueue(u)$ ; // insert node to successors //
7 end
8 while ( $S \neq \emptyset$ ) do
9   |  $C = S$ ; // replace current with successor nodes //
10  while ( $C \neq \emptyset$ ) do
11    |  $v = C.dequeue()$ ; // get output pin from current //
12    |  $N = get\_successor\_input\_pins(v, E)$ ; // get successor input pins //
13    |  $P = get\_predecessor\_input\_pins(v, E)$ ; // get predecessor input pins //
14    | if ( $N \neq \emptyset$ ) then
15      |  $diff\_exists = timing\_update\_output\_gatepin(v, P, N, T.delay(v),$ 
16        |  $T.slew(v))$ ;
17      | if ( $diff\_exists == false$ ) then
18        |   continue;
19      | end
20      | for (each Successor Input  $n \in N$ ) do
21        | // annotate successor, account for wires too //
22        |  $diff\_exists = timing\_update\_successor(v, n, T.delay(n), T.slew(n))$ ;
23        | if ( $diff\_exists == false$ ) then
24          |   continue;
25        | end
26        | if ( $n.is\_combinational()$ ) then
27          |    $O = get\_successor\_outputs(n)$ ; // trace successor outputs //
28          |   for (each Successor Output  $o \in O$ ) do
29            |     // insert traced output to successors //
30            |      $S.enqueue(o)$ ;
31          |   end
32        | end
33    | end
34  end
35 end

```

---

the clock period), is retrieved, and finally, the RAT at the Flip-Flop input is calculated. Identically to delay propagation, as long as the Predecessors queue  $P$  is not empty, the current pins queue  $C$  is replaced with Predecessors  $P$ , which is equivalent to a backward levelization step.

Then, as long as the current pins queue  $C$  remains non-empty, each pin is individually

traversed to identify their successor output pins and predecessor outputs. RAT annotation is then achievable based on the RAT of successor outputs and the delay between these successor outputs and the current input. Following this, the predecessor output is explored, and its predecessor inputs are traced. At this juncture, a predecessor input is enqueued into the Topological Sort Predecessors queue  $P$  if, and only if, all its successors have been visited, thereby fulfilling the Topological Sort Criterion. This procedure is repeated until the entire timing graph has been traversed, and thus the entire timing graph is annotated with RAT.

Support for Incremental RAT updates is imperative to facilitate global slack calculation when employing the STA engine as an Application Programming Interface (API). This API must provide timing awareness to other physical design tools, such as timing-driven placers or timing-aware routers. Algorithm 4 outlines the process for Incremental RAT updates. It is crucial to emphasize that the procedure for updating RAT closely mirrors that of full RAT propagation, with the key difference being the use of Breadth First Search (BFS) instead of Topological Sort for traversal. Moreover, RAT updates necessitate traversing the entire backward logic cone, acknowledging that the correspondence between delay and RAT logic cones is not entirely one-to-one. Then again, the full traversal Topological Sort levels can be exploited to accelerate the BFS traversal in the same manner with incremental delay update.

## 2.3 Differential Equations and Control Systems Theory

This section provides the required theory and fundamentals, which are used in circuit analysis and modelling of passive devices impact in signal propagation.

### 2.3.1 Passive Circuits and Scientific Representation

Modern VLSI circuits are described using active devices for standard cells, while interconnection parasitics are modelled as passive RC/RLC networks. For the scope of STA, the former are abstracted using pre-characterized libraries as described in section 2.1, while for the latter the only readily available information for the STA engine is the RC/RLC values and interconnect shape, in the form of a parasitics format file like SPEF or DSPF [22, 23]. Contrary to that, the most widespread scientific representation of

**Algorithm 3:** *propagate\_RAT*


---

```

1 Input: Delay, Slew Annotated Timing Graph  $T = (V, E, delay, slew)$ 
2 Output: RAT Annotated Timing Graph  $R = (V, E, RAT)$ 
3  $C = \emptyset$ ; // Current Gate Pins FIFO //
4  $P = \emptyset$ ; // Predecessor Gate Pins FIFO //
5 for (each Node  $v \in V$ ) do
6   R.RAT( $v$ ) =  $\infty$ ;
7   if ( $v.is\_PO()$  ||  $v.is\_sequential\_input()$ ) then
8      $P.enqueue(v)$ ; // insert node to predecessors //
9   end
10  if ( $v.is\_sequential\_input()$ ) then
11     $FF = get\_FF\_from\_pin(v)$ ;
12     $c = get\_FF\_clock\_pin(FF)$ ;
13     $t_{period} = get\_clock\_period(c)$ ;
14     $t_s = get\_FF\_setup\_time(T.slew(v), T.slew(c), FF)$ ;
15    R.RAT( $v$ ) =  $t_{period} - t_s$ ;
16  end
17 end
18 while ( $P \neq \emptyset$ ) do
19    $C = P$ ; // replace current with predecessor nodes //
20   while ( $C \neq \emptyset$ ) do
21      $v = C.dequeue()$ ; // get input pin from current //
22      $N = get\_successor\_output\_pins(v, E)$ ; // get successor output pins //
23      $p = get\_predecessor\_output\_pin(v, E)$ ; // get predecessor output pin //
24     if ( $N \neq \emptyset$ ) then
25       RAT_annotate_input_gatepin( $v, N, R.RAT(v)$ );
26       // annotate predecessor output, account for wires too //
27       RAT_annotate_predecessor( $v, p, R.RAT(v), R.RAT(p)$ );
28       if ( $n.is\_combinational()$ ) then
29          $I = get\_predecessor\_inputs(p)$ ; // trace predecessor inputs //
30         for (each Predecessor Input  $i \in I$ ) do
31           // TopoSort Criterion //
32           if ( $all\_successors\_visited(i)$ ) then
33             // insert traced input to predecessors //
34              $P.enqueue(i)$ ;
35           end
36         end
37       end
38     end
39   end
40 end

```

---

describing passive RC/RLC circuits is the derivation of the voltage and current differential equation [24, 25]. For demonstration purposes Figure 2.3.1 depicts a simple RLC circuit.

**Algorithm 4:** *update\_RAT*


---

```

1 Input: RAT Annotated Timing Graph  $R = (V, E, RAT)$ , Update Start Points  $U$ 
2 Output: Updated RAT Annotated Timing Graph  $R = (V, E, RAT)$ 
3  $C = \emptyset$ ; // Current Gate Pins FIFO //
4  $P = \emptyset$ ; // Predecessor Gate Pins FIFO //
5 for (each Node  $u \in U$ ) do
6    $P.enqueue(u)$ ; // insert node to predecessors //
7   if ( $v.is\_sequential\_input()$ ) then
8      $FF = get\_FF\_from\_pin(u)$ ;
9      $c = get\_FF\_clock\_pin(FF)$ ;
10     $t_{period} = get\_clock\_period(c)$ ;
11     $t_s = get\_FF\_setup\_time(T.slew(u), T.slew(c), FF)$ ;
12     $R.RAT(u) = t_{period} - t_s$ ;
13  end
14 end
15 while ( $P \neq \emptyset$ ) do
16    $C = P$ ; // replace current with predecessor nodes //
17   while ( $C \neq \emptyset$ ) do
18      $u = C.dequeue()$ ; // get input pin from current //
19      $N = get\_successor\_output\_pins(u, E)$ ; // get successor output pins //
20      $p = get\_predecessor\_output\_pin(u, E)$ ; // get predecessor output pin //
21     if ( $N \neq \emptyset$ ) then
22        $RAT\_annotate\_input\_gatepin(u, N, R.RAT(v))$ ;
23       // annotate predecessor output, account for wires too //
24        $RAT\_annotate\_predecessor(u, p, R.RAT(u), R.RAT(p))$ ;
25       if ( $n.is\_combinational()$ ) then
26          $I = get\_predecessor\_inputs(p)$ ; // trace predecessor inputs //
27         for (each Predecessor Input  $i \in I$ ) do
28           // insert traced input to predecessors //
29            $P.enqueue(i)$ ;
30         end
31       end
32     end
33   end
34 end

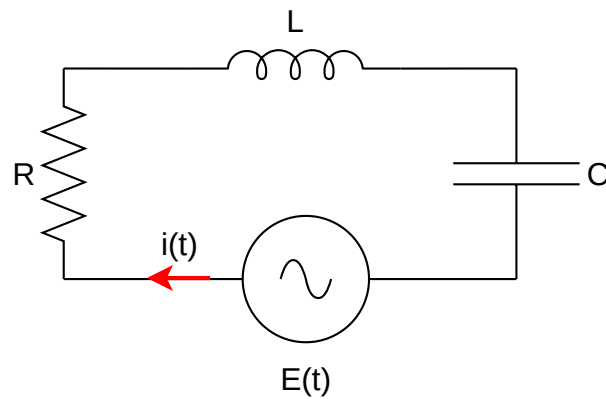
```

---

The following Ordinary Differential Equation (ODE) describes the RLC circuit behavior:

$$L \frac{d^2 i(t)}{dt^2} + R \frac{di(t)}{dt} + \frac{i(t)}{C} = E(t), \quad (2.3.1)$$

where  $L$  is inductance,  $i(t)$  is branch current,  $R$  is resistance,  $C$  is capacitance, and finally  $E(t)$  is the excitation input waveform. While this representation is suitable for scientific study of such simple circuits, it is impractical for real word applications, where millions



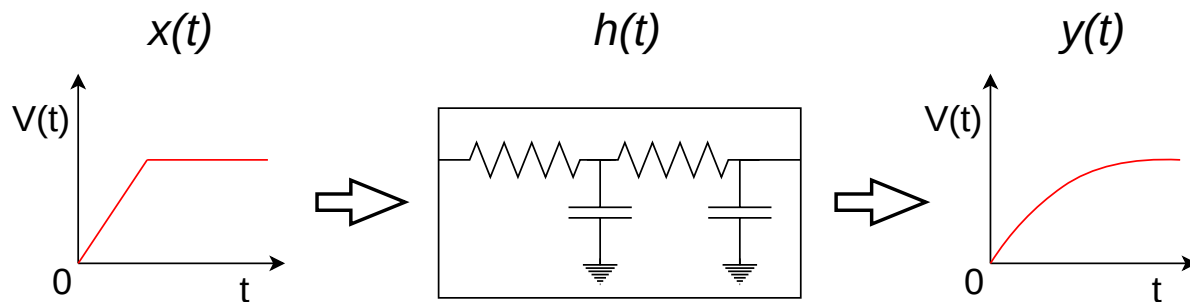
**Figure 2.3.1:** Simple RLC circuit with attached Voltage Source Excitation

of such circuits representing wires are handled, for a wide variety of reasons. The most important ones are the fact that the circuits are never as simple, thus an analytical representation of them is hard to derive, as well as the issues of solving differential equations that may not exactly match any well known ODE form. Thus, a different approach is required to tackle these.

Another important property of passive RC circuits, is their suitability to be expressed as Linear Time-Invariant (LTI) systems [26]. In other words, the resistive, capacitive and inductive parasitic components of the interconnection, as well as network topology is predetermined and does not change as time passes. This property is particularly useful to analyze circuit response independently of the excitation at its input. In order to achieve this, an infinitely small input excitation can be applied to determine the response of the LTI system. This infinitely small excitation is called impulse and is defined by Dirac's delta function

$$\delta(t) = \begin{cases} \infty & \text{if } t = 0 \\ 0 & \text{if } t \neq 0 \end{cases}$$

and the system's response to this impulse is called impulse response. Furthermore, the LTI system representation allows the use of dedicated operations like convolution, in order to mathematically determine output signals of the circuit, given the input excitation mathematical formula and circuit impulse response. Figure 2.3.2 provides an illustrative example of a  $h(t)$  LTI system influencing a ramp input signal  $x(t)$  and the resultant output signal  $y(t)$ .



**Figure 2.3.2:** LTI System's Influence to Excitation Input Signal

## 2.3.2 Interconnect Voltage Propagation Fundamentals

This subsection provides a comprehensive description of signals, LTI systems, and all related related operations and properties, which are perfectly suitable for use with Delay Calculation and STA.

### 2.3.2.1 Time Domain

As discussed above, numerous issues and inconveniences manifest when dealing with such problems purely in the time domain. In more detail, operating solely on time domain cannot be considered convenient as it is particularly difficult to translate the problem to an equivalent ODEs problem that can be easily and efficiently solved. This can be justified by using as an example a modern complex interconnect which is part of the clock network. Such wires are typically driven by high drive clock buffers, thus their overall shape and complexity is significantly high. Furthermore, the existence of coupling capacitors or even resistive loops renders the derivation of an analytical impulse response formula extremely difficult and computationally expensive. Furthermore, the use of LTI system rules per RC segment of the interconnect is not applicable, as mutual interdependence between all node voltages exists, due to the existence of capacitors in the circuit.

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau \quad (2.3.2)$$

Another significant consideration stems from the actual time domain convolution operation itself. Equation 2.3.2 is the mathematical definition of time domain convolution [26]. It is evident that this is an infinite integral operation, which renders it unsuitable for use with STA for the purpose of waveform construction and/or delay and slew calculation. The

reason behind this statement is the actual computational intensity of such an operation in time domain, which ultimately defies the purpose of using STA instead of SPICE simulation for the interconnect.

Finally, a subtle yet significant detail, is the fact that all of the signals used with STA are discrete Piece-Wise Linear (PWL) signals, which is incompatible with the above definition as the generalized convolution operation is a continuous mathematical operator. The version of convolution which is compatible with STA is the discrete form as provided in Equation 2.3.3.

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n - m] \quad (2.3.3)$$

This discrete operation is dependent on the sampling step size used, thus there is great possibility of accuracy considerations to apply, depending on how this sampling step is selected.

All of the aforementioned justify the reasoning behind searching for more suitable alternatives to time domain convolution.

### 2.3.2.2 Frequency Domain

A tool of utmost importance to avoid direct time domain analysis, is the transformation to frequency domain. A straightforward and intuitive depiction of frequency domain functions and transform is the switching from a sinusoidal  $\sin(t)$  waveform to a spinning phasor  $z = A(x + yi)$ , where  $A$  is the waveform amplitude,  $x = \cos(\omega t + \theta)$ ,  $y = \sin(\omega t + \theta)$ . This phasor produces the aforementioned time domain waveform, as it spins while time progresses.

Frequency domain analysis, enables also the use of a variety of mathematical tools, as for example the convenient LTI systems representation and the exploitation of their properties [26]. Identically with all other LTI systems, the interconnect is characterized by its transfer function. As previously mentioned, the output voltage of the interconnect, in response to any input excitation, is described by differential equations, making frequency domain analysis significantly more convenient [26, 27] than attempting direct analytical solutions. An alternative approach involves convolving the input voltage waveform

function with the time domain response of the interconnect. However, this method becomes computationally intensive when conducted exclusively in the time domain. A mathematical tool of the most significant importance in this context is the Laplace transform [26],

$$\mathcal{L}\{f(t)\} = \int_{t=0}^{\infty} f(t)e^{-st} dt$$

which facilitates the transition of the problem from the time domain to the frequency domain. The interconnect transfer function  $H(s)$  and the input voltage waveform function  $X(s)$  can subsequently be multiplied in the frequency domain to derive the output voltage function  $Y(s)$  in the frequency domain. This operation is exactly equivalent to performing convolution in the time domain [26]. It is noteworthy that the input function can assume any form, while it is advantageous to express the interconnect transfer function in the following form:

$$H(s) = \sum_{q=1}^N \frac{k_q}{s + p_q}, \quad (2.3.4)$$

where  $k_q$  are defined as residues and  $p_q$  are called poles. These values hold considerable importance, as they are utilized for the derivation of the time domain output voltage waveform function through the application of the inverse Laplace transform,  $y(t) = \mathcal{L}^{-1}\{Y(s)\}$ . This function is subsequently utilized to compute the delay and slew at the output of the interconnect. For instance, in the scenario where the input voltage waveform is a unit step function, denoted as  $u(t)$ , the resulting output voltage waveform will be of the form:

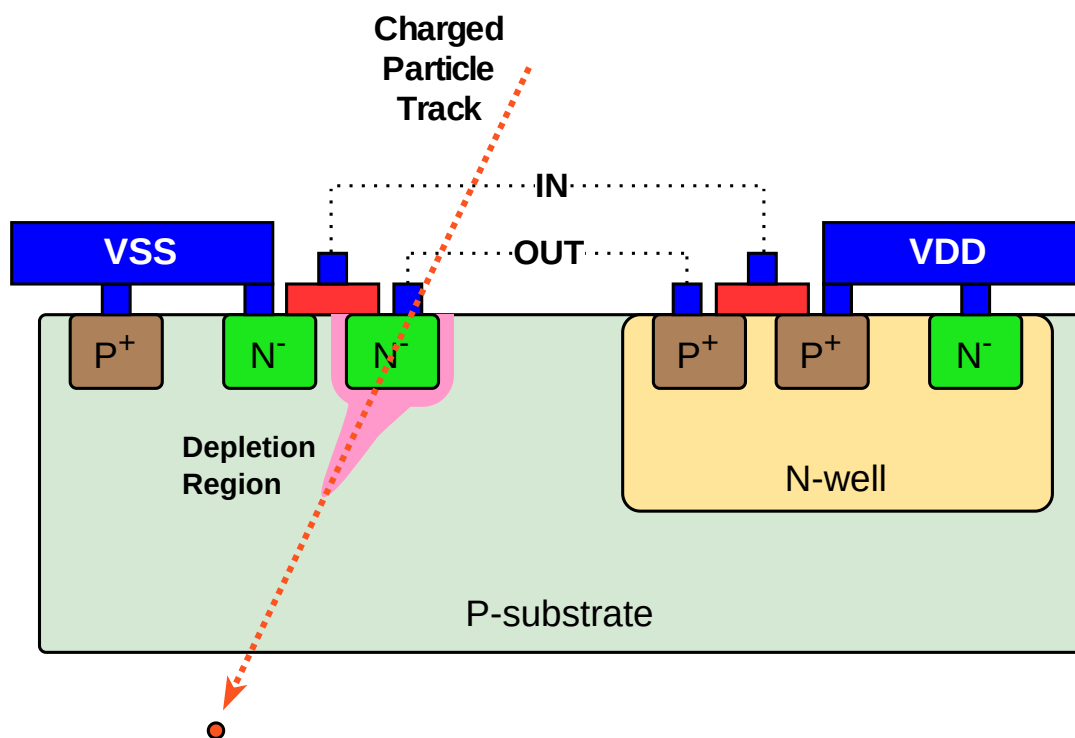
$$y(t) = c\delta(t) + \sum_{q=1}^N k_q e^{-p_q t}, \quad (2.3.5)$$

where  $c$  stems from initial conditions.

Nonetheless, a challenge encountered in this context is that, for interconnects with arbitrary shapes, deriving a transfer function of this nature is neither trivial nor straightforward, due to the possibility for the numerator and denominator to include polynomials of higher order. As a result, approximate methods for determining poles and residues are preferred.

## 2.4 Radiation Induced Glitches - Single Event Transients Generation Mechanisms

The generation of Single Event Transients (SETs) in semiconductor devices is a complex phenomenon primarily driven by the interaction between ionizing radiation and the semiconductor material. This interaction leads to the generation of charge carriers, which can transiently affect the electrical behavior of the device. The primary mechanisms involved in the creation of SETs are detailed in next paragraphs.



**Figure 2.4.1:** Charged Particle Strike Effect on Simple CMOS Inverter Cross Section

Direct ionization is the most straightforward mechanism through which SETs are generated. When a charged particle, such as a proton, neutron, or heavier ion, passes through any semiconductor material, it may ionize the atoms in its path directly. This ionization process creates electron-hole pairs in the substrate. The number of pairs generated is proportional to the energy deposited by the ionizing particle and the material's ionization energy [28].

The transient current pulse induced by the movement of these carriers can temporarily alter the voltage levels within the device, leading to a SET. The characteristics of this

pulse, including its amplitude and duration, are influenced by factors such as the Linear Energy Transfer (LET) of the particle, the doping concentration, and the electric field distribution within the device [29].

In addition to direct ionization, SETs can also result from indirect ionization processes. Secondary particles or photons generated by the initial ionizing event can further interact with the semiconductor, extending the region affected by the ionizing event beyond the immediate path of the primary particle. This can lead to a more dispersed generation of electron-hole pairs and, consequently, a more widespread transient effect.

Furthermore, the manner in which the generated charge carriers are collected and contribute to the transient current is a critical aspect of SET generation. The process of charge collection is influenced by the electric field within the device, which drives the movement of carriers. For instance, in a MOSFET, the carriers generated in the substrate can be collected by the drain, source, or gate terminals, depending on the device's biasing conditions and the location of the ionizing event relative to these regions [28].

Another important fact, is that the impact of technology scaling on SET sensitivity is significant. As devices are made smaller, the charge required to significantly alter the state of a device is reduced, making advanced technology nodes more susceptible to SETs. Additionally, the device architecture, including the layout and doping profiles, can affect the electric field distribution and, thus, the charge collection efficiency and the characteristics of the SET.

Finally, to model and measure SET related phenomena, Technology Computer Aided Design (TCAD) [30, 31] 3D material layer simulations are typically performed. Another alternative is simulation at SPICE level while adhering to the use of models that closely match the TCAD results [28, 29, 32]. However, it is evident that such simulations can not be performed for a complete digital design which may contain millions of standard cells. Thus, this provided the motivation to create a more abstract but also globally applicable methodology for SET analysis.

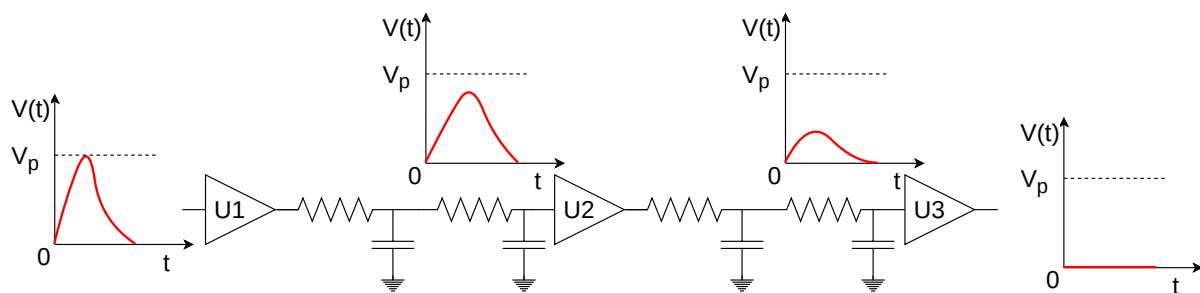
## 2.5 Single Event Transients Propagation and Masking Mechanisms

Except from SET Generation, SET Propagation is as well a crucial issue for analyzing the circuit sensitivity to radiation. Voltage glitch propagation is a complicated issue that requires not only the accuracy and bounding that specific algorithms can provide, similarly with STA for example. Even if delay calculation for the whole voltage pulse can be performed with maximum accuracy, there are still several issues that must be addressed. These issues include circuit coverage, supporting various masking mechanisms and generating data based on probabilistic formulation in case there is not enough information for voltage pulses at all nodes.

In the scope of this work, only the masking mechanisms where investigated as all other issues are part of a different collaborative PhD dissertation. Thus, the following subsections provide brief descriptions about the SET masking mechanisms.

### 2.5.1 Electrical Masking

The first masking mechanism to present is Electrical Masking. Given a generated SET pulse at a circuit stage, the pulse is propagated through forward logic. However, in reality the digital circuit consists not only from standard cells but also interconnections, which typically present a significant amount of parasitic resistance and capacitance. As discussed in section 2.3 the wire parasitics behave as low pass filters, and thus the SET pulse peak voltage  $V_p$  may drop as the pulse propagates through the interconnect, up to a point where it may be completely attenuated.



**Figure 2.5.1:** SET Pulse Electrical Masking

A simplistic example of this is depicted in Figure 2.5.1. In the figure, the SET voltage

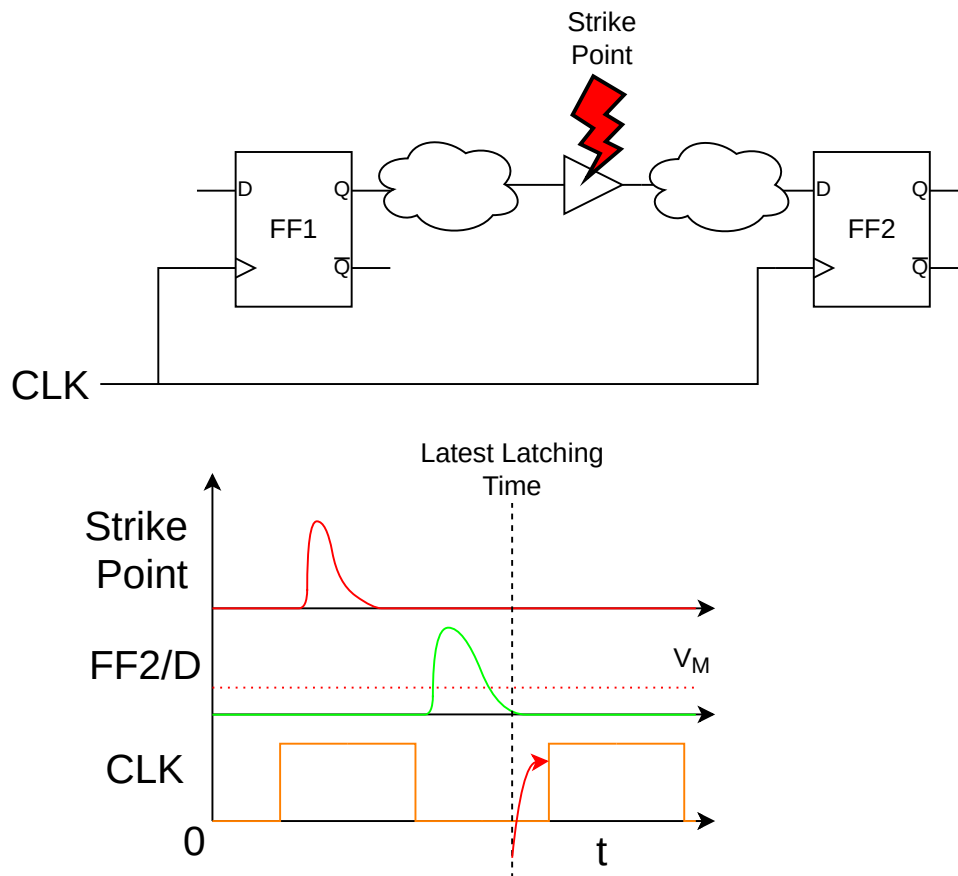
pulse applied at input of U1, propagates through first stage parasitics and arrives at input of U2 where it is now broader and reaches lower peak voltage than the initial  $V_p$ . Then, the exact same happens until arriving at input of U3, where the pulse's peak voltage is below threshold voltage  $V_M$  of U3. Finally there is no pulse at the output of U3 as its threshold voltage  $V_M$  was not crossed, and thus no switching happened internally. This mechanism of subsequent broadening and attenuation up to the point where propagation stops is called Electrical Masking.

### 2.5.2 Timing Window Masking

The next mechanism to present is Timing Window Masking. Again, an arbitrary particle strike point is selected to present the example. The generated SET voltage pulse is propagated through logic and interconnect until reaching the timing path endpoints, which are sequential elements and Primary Output ports. Both endpoint types are considered data capture and storage elements. It is noteworthy that Primary Outputs are considered data capture elements, and it can be explained by assuming that any Primary Output is driving a sequential element in the circuit's environment. All data capture elements are constrained by a clock. Thus, it is obvious that any voltage glitch that arrives before the said clock's capturing edge window, is not captured by the data capture elements.

A simple example of the above is depicted in Figure 2.5.2. For shake of clarity, two Flip-Flops are used for this example, one launching and one capturing, with combinational logic between them. The particle strike happens somewhere within the combinational logic and the SET pulse propagates through it to FF2 input pin D. In the figure, the CLK waveform is the constraining clock waveform. If the pulse arrives before the last time instant that FF2 can capture it (dotted "Latest Latching Time" vertical line) then the pulse is not stored by FF2 and it does not affect circuit functionality.

It is important to note that the term "arrive" does not refer to arrival time as defined in STA nomenclature, but it is a more general term which can be translated as "the pulse voltage is higher than capturing element's  $V_M$ ". Finally, it is interesting to mention that, masking of voltage glitches due to capturing timing window is always happening in digital circuits' normal function and, in a sense, STA accounts for the latest edge of these glitches.



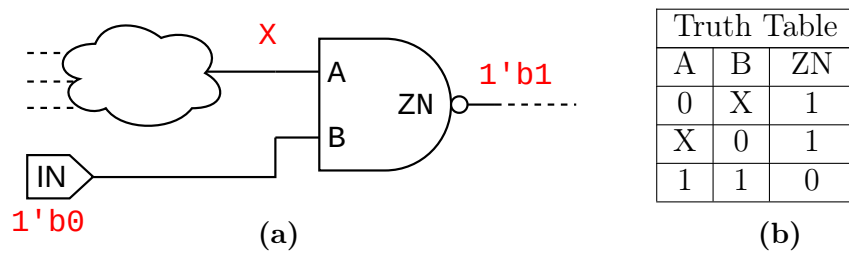
**Figure 2.5.2:** SET Pulse Timing Window Masking

Thus, it is again possible to employ a STA-based approach to account for SETs Timing Window Masking.

### 2.5.3 Logical Masking

Last but not least, Logical Masking is detailed. This mechanism is based on circuit nodes state and refers to SET voltage pulses propagating through the circuit in the presence of steady state values at standard cells side inputs. Arguably, circuit state affects not only SET voltage pulses propagation, but also generation. An intuitive example of this is assuming a standard cell output pin being at  $V_{dd}$  for a specific circuit state and a particle strike occurring at its Pull-Up network. The only pulse that will be generated will be above  $V_{dd}$  and thus it will not affect the next stage.

However, without loss of generality, in this context the focus is on SET propagation, thus the intuition behind Logical Masking is about standard cells inputs at any state that blocks the pulse from propagating forward. This may happen due to a variety of reasons.



**Figure 2.5.3:** SET Pulse Logical Masking

First of all, circuit mode is one of the most common uses of case analysis, *i.e.* setting Primary Input ports at specific boolean values. In general, case analysis is one dominant cause of Logical Masking. Another important situation, is re-convergence of paths. In such situations, the generated SET voltage pulse propagates through a number of paths that diverge from each other and then re-converge at some specific standard cell output pin through different inputs. If the pulses arriving at input pins are overlapping, there is a great chance that one may block the other based on the standard cells output pin boolean function.

The last dominant cause of Logical Masking is the case of boolean implications. To render generation of any SET pulse possible, the particle strike point must be at a specific state as previously described. This however implies that the state of all preceding logic, including all preceding timing paths start points, must allow for this particle strike to happen. This in turn means that the forward logic cone of the particle strike point may be in a restricted set of possible states, which then creates implications for forward logic cone pins. Thus, due to these, Logical Masking may occur for pulses that do not adhere to these possible states.

Finally, a simple example of logical masking due to case analysis is presented in Figure 2.5.3.

# Chapter 3

## Related Work

### 3.1 Delay Calculation

Numerous methodologies for delay calculation, both for standard cell delay and RC interconnect, have been proposed in the literature. This section offers a concise review of these studies, including a fair assessment of their strengths and limitations.

The most critical challenge in computing driver delay, especially within the context of industrial standard timing models, is the modeling of effective capacitance. A work of great importance to addressing this issue is the publication by Puri *et al.* [33], where the authors introduce a closed-form expression for the iterative computation of Effective Capacitance. This is achieved by recursively traversing the interconnect nodes and effectively reducing each RC segment of the interconnect into a single Effective Capacitance value. The derived formula for calculating Effective Capacitance is:

$$C_{eff} = C_1 + C_2 \left(1 - \frac{2RC_2}{tr} (1 - e^{-\frac{tr}{2RC_2}})\right), \quad (3.1.1)$$

where  $tr$  is the RC segment input slew.

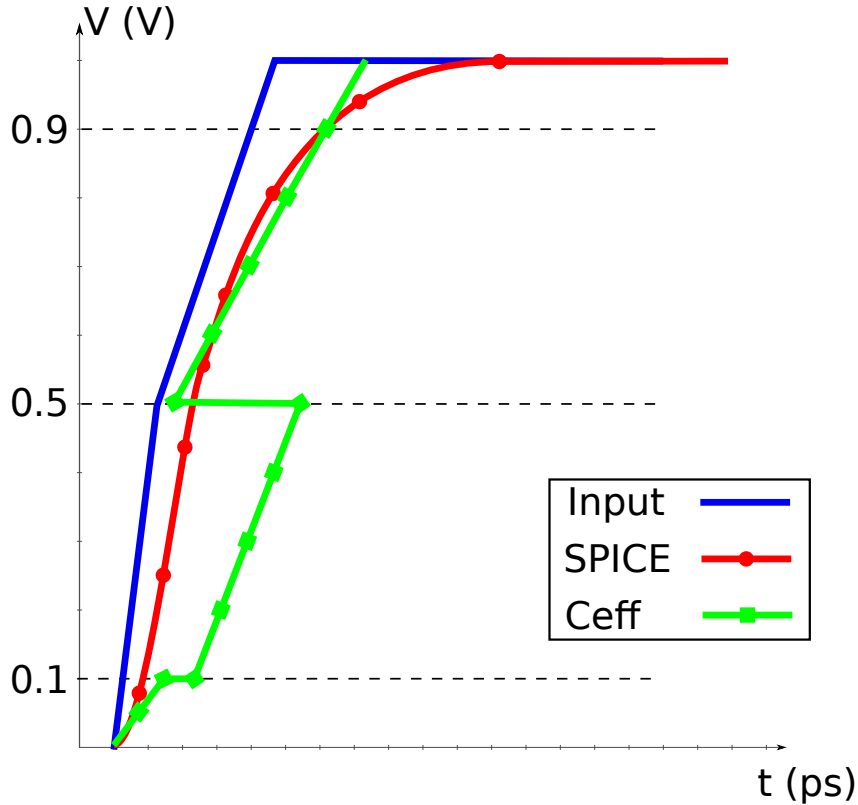
However, the formula presented is limited to only calculating Effective Capacitance up to the delay voltage level, typically 50% of  $V_{dd}$ , without addressing the calculation of Effective Capacitance for slew voltage levels. While this was sufficient at the time of these publications, accurate calculation of slew has become crucial, given its impact on the delay calculation of next stage. The authors also suggest a geometrical method for slew

propagation, as the proposed Effective Capacitance calculations depend on slew at every node. This method falls short in accuracy because it simplifies slews to linear segments while totally disregarding the voltage across  $C_2$  at all times. Another limitation arises with its application to arbitrary interconnects. While the above slew propagation method may yield reasonably accurate results for pi-models, applying it to arbitrary interconnect nodes with fanout greater than two is notably inaccurate. This inaccuracy stems from treating currents in fanout branches as completely independent, which is a direct violation of Kirchhoff's branch currents law.

Another important paper is our publication which presents the methodology we employed in the Tau Workshop 2020 Contest and achieved to get the first place [1]. In our approach, we aimed to build upon the work of Puri *et al.* by developing an Effective Capacitance formula that applies to multiple voltage ranges, specifically 0 - 10%, 10% - 50%, and 50% - 90% of  $V_{dd}$ . While this adaptation achieved satisfactory precision for contest benchmarks, it still inherits the exact same issues related to slew propagation as the original Puri *et al.* methodology. Moreover, modifying Puri *et al.*'s method for slew propagation across these multiple voltage regions proved to be non-trivial. Implementing this adaptation as suggested in [1] led to the derivation of output voltage waveforms that exhibit non-continuous regions, as illustrated in Figure 3.1.1.

An intriguing question emerges here: why does the inaccuracy in slew propagation not compromise the overall accuracy, given that the CCS receiver capacitance is influenced by slew? This question can be addressed by examining the generation of contest benchmarks. The benchmark generator is capable of performing design space exploration based on statistics from real designs, but it operates under the assumption that  $C_1$  equals  $C_2$  for all cases. In section 6.1, this methodology is rigorously tested with imbalanced pi-models, validating the aforementioned arguments. The rationale behind this is quite simple. In the extreme scenario where  $C_1 = 0$ , the driver node essentially functions as the middle point of a voltage divider, that is, between the resistance of the driver cell and that of the interconnect. This completely exposes the effect of inaccuracy of slew propagation on the calculation of Effective Capacitance at the receiver node, as there is no capacitance directly connected to the driver's output. Thus, receiver capacitance directly affects driver waveform as for this example there is no  $C_1$  to dominate calculations. It is worth noting

that in [1], we have provided extensive details and comprehensive background context regarding works even prior to Puri *et al.* [33] publication.



**Figure 3.1.1:** Non-Continuous Effective Capacitance Regions

Finally, the most significant publication in terms of providing intuition regarding Effective Capacitance in this research thesis is found in the publication of Feldmann *et al.* [34]. The authors of this paper discuss the critical need for introducing a newer concept named *Dynamic Capacitance*. They impose a charge equilibrium condition into the computation of Effective Capacitance. This condition stipulates that the amount of electrical charge that can flow from the driver cell through the interconnect can only be as much as the storage capacity of the interconnect capacitors. This stems from the charge conservation law and the rationale behind it is absolutely fundamental. The authors provide Equation 3.1.2 per voltage region  $\{\{T_l, V_l\}, \{T_{l+1}, V_{l+1}\}\}$  for adhering to this, and then provide closed form charge equations to replace the integral of Equation 3.1.2.

$$\int_{T_l}^{T_{l+1}} i_d(t) dt = (V_{l+1} - V_l) C_{d,l} \quad (3.1.2)$$

The analysis and experiments undertaken throughout this research project proved that the proposed methodology for pi-models achieved the exact same results as with this

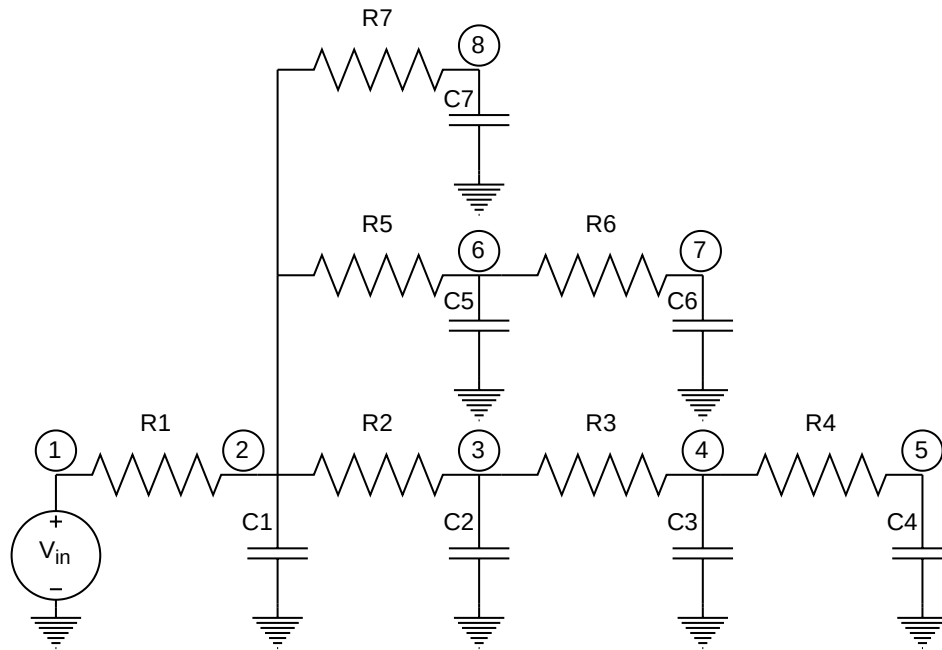
Feldmann *et al.* publication, given that slew is calculated for pi-model outputs using any golden method like SPICE. Thus, even if the proposed methodology is completely different on a mathematical basis compared to this publication, it yields the exact same results at the electrical level for pi-models. On the contrary, with regard to arbitrary interconnect shapes, it was impossible to either derive the provided charge formulas or directly apply them effectively. Furthermore the mathematical derivation of these provided formulas is lacking appropriate explanation or proof in this publication, while also being non-trivial or straightforward. Furthermore, another drawback of this publication is the need to use a Brent's method root-finder in order to solve for Dynamic Capacitance value per voltage region. This may be a way to effectively find a solution for the desired value, although it is costly in terms of execution time, as Brent's method inherits Secant method's convergence or worse [35]. Finally, another important drawback is the lack of any information or method about voltage waveform or slew propagation, which is an absolute necessity for calculation of CCS receiver capacitance values. Despite all of these, the presented intuition behind the charge equilibrium condition functioned as the inspiration for the main idea of the proposed Dynamic Capacitance calculation method of section 4.3.

### 3.1.1 Interconnect Delay Calculation

Apart from driver delay calculation, the computation of interconnect delay and slew significantly affects circuit timing. The Elmore delay [36] is the most well known technique for estimating wire delay. Elmore proposed a closed-form formula to pessimistically estimate an upper bound for wire delay by applying a unit step waveform at interconnect input. Nonetheless, this approach does not offer precise calculation or a tight bound compared to SPICE simulations and does not accommodate arbitrary input waveform shapes. Moreover, all literature methodologies based on Elmore delay are not suitable for full waveform propagation because they only offer conservative estimates for certain voltage thresholds of the actual waveforms. Therefore, a more sophisticated and elaborate mathematical approach is necessary to address all of the aforementioned issues.

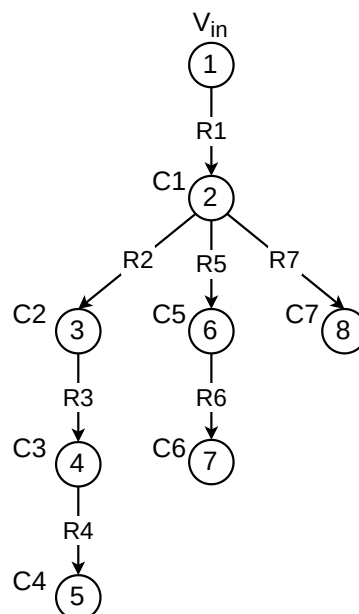
$$t_{Di} = \sum_{i=1}^N C_k R_{ik}, \quad (3.1.3)$$

The literature methodology that was utilized and extended regarding interconnect analysis



**Figure 3.1.2:** Arbitrary RC Interconnect

and delay calculation is the publication of Ratzlaff *et al.* [37], which employs the powerful Asymptotic Waveform Evaluation method [38]. Its main target is to derive the time domain output voltage formula for any arbitrary RLC interconnect shape, by transferring the differential equation problem in frequency domain in the form of Equation 2.3.4, approximating poles and residues using a fast moment-matching approach, and finally



**Figure 3.1.3:** Arbitrary RC Interconnect Tree Representation

solving for the time domain function.

To elaborate further, the initial step involves acquiring the interconnect moments, as referenced by [38]. These moments are fundamental convergence points of the transfer function, which can be obtained by subsequent traversals of the interconnect. A tree-link analysis technique is suggested for this task. An illustration of a typical interconnect is shown in Figure 3.1.2, and Figure 3.1.3 illustrates the same interconnect represented as a tree. It is important to note that STA engines cannot handle resistive loops, which means resistors that connect to either the ground or supply node are not supported [19]. Consequently, it can be safely assumed that the initial moment for every node will be equivalent to the supply voltage, which is the DC analysis voltage. The subsequent steps involve a process which is equivalent with subsequent DC analyses to compute each higher-order moment. This process starts with the calculation of branch currents by traversing the tree in a post-order fashion. The branch current calculation involves multiplying the node capacitance with previous iteration moment and summing up the currents from the children nodes to it. Equation 3.1.4 is used for incoming node current calculation.

$$i_{n,j} = -m_{n,j-1}C_n + \sum_{k=0}^{children} i_{k,j}, \quad (3.1.4)$$

where  $i$  is the branch current component from all leaf nodes to node  $n$ ,  $m_{n,j-1}$  is the previous moment, and  $i_{k,j}$  is the current value for  $j^{th}$  moment of  $k^{th}$  child of node  $n$ . Then, by traversing in pre-order, moment values are calculated in the form of DC voltages, by multiplying the current values of Equation 3.1.4 per node with parent resistance and subtracting from the parent node moment (voltage).

$$m_{n,j} = m_{n-1,j} - i_{n,j}R_{n-i \rightarrow n} \quad (3.1.5)$$

In this way, any required number of moments can be calculated for any node, up to the highest precision allowed by floating point arithmetic precision or processor limits.

The next step is to utilize these calculated moments to obtain the differential equation's characteristic polynomial coefficients  $a_j$ . The roots of this polynomial are the transfer

function pole reciprocals  $\tau_j$  [38].

$$\begin{bmatrix} m_0 & m_1 & \dots & m_{j-1} \\ m_1 & m_2 & \dots & m_j \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ m_j & m_{j+1} & \dots & m_{2j-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \cdot \\ \cdot \\ \cdot \\ a_{j-1} \end{bmatrix} = \begin{bmatrix} -m_j \\ -m_{j+1} \\ \cdot \\ \cdot \\ \cdot \\ -m_{2j-1} \end{bmatrix} \quad (3.1.6)$$

$$\tau^j + a_{j-1}\tau^{j-1} + a_{j-2}\tau^{j-2} + \dots + a_0 = 0, \quad (3.1.7)$$

where  $\tau$  is the pole reciprocal, *i.e.*  $\tau = \frac{1}{p}$ . However, this method may result to non-negative or complex poles due to the mathematical formulation of Equation 3.1.6. In case the roots are complex or non-negative, publications [37, 39] propose a moment shifting/frequency hopping technique. As proposed, the lower order moments can be iteratively dropped, until no complex or non-negative roots are obtained. The authors claim that while there is no way to prove why this is always the case, they never encountered cases where this technique is not providing the desired result. Indeed, throughout the research work conducted for the completion of this thesis, this technique was indeed found as powerful as claimed in the publication.

The last step for frequency domain analysis, is calculating the transfer function residues  $k_q$ . These are calculated by:

$$\frac{k_1}{p_1^{i+1}} + \frac{k_2}{p_2^{i+1}} + \dots + \frac{k_q}{p_q^{i+1}} = m_i \quad (3.1.8)$$

for  $q$  number of pole and residue pairs. At this point, a subtle yet significantly useful observation is that the expanded equations presented in [37], Equation 6, are not consistent with the aforementioned generalized Equation 3.1.8, which is again proposed in [37], Equation 5. Based on this, it is important to provide the correctly expanded formulas, for shake of clarity and completeness. Furthermore, these expanded equations were the ones

that provided the highest possible accuracy achieved in all experiments of section 6.1.

$$\begin{aligned}
\frac{k_1}{p_1} + \frac{k_2}{p_2} + \dots + \frac{k_{q-1}}{p_{q-1}} &= m_0 \\
\frac{k_1}{p_1^2} + \frac{k_2}{p_2^2} + \dots + \frac{k_{q-1}}{p_{q-1}^2} &= m_1 \\
\cdot &\cdot \\
\cdot &\cdot \\
\cdot &\cdot \\
\frac{k_1}{p_1^{2q-1}} + \frac{k_2}{p_2^{2q-1}} + \dots + \frac{k_{q-1}}{p_{q-1}^{2q-1}} &= m_{2q-1}
\end{aligned} \tag{3.1.9}$$

Finally, using Equation 2.3.5, the time domain voltage waveform can be derived. Even if the proposed methodology of publication [37] proved to be significantly useful for calculating transfer function poles and residues, the presented math regarding time domain voltage waveforms are only suitable for step inputs. Then, publication [38], claims that the presented methodology and mathematical formulation can be extended to support ramp input waveforms as well, however the provided example lacks the appropriate level of detail and contains equations that can not be directly derived from the provided explanation. Thus, throughout this research thesis, only frequency domain analysis to obtain poles and residues from publications [37, 38] was employed. The overall procedure of transferring the problem to time domain is presented in section 4.2.

## 3.2 Single Event Transients

Characterizing the effects of SETs is difficult because SET pulses are analog and combinational circuits vary widely. This makes the calculation of electrical and timing masking factors significantly less precise, potentially by several orders of magnitude, compared to calculations of logical masking [40]. To compute electrical and timing masking factors with accuracy, it's crucial to take into account the mechanisms of SET generation and propagation.

A prevalent method for characterizing SET effects involves using electrical simulations, such as SPICE or SPECTRE. This approach entails injecting a current pulse that simulates the effects of SET generation into each stage of the circuit and monitoring the SET response at the primary outputs. The benefit of using electrical simulations lies in their ability to account for the analog characteristics of SETs. Nonetheless, the applicability

of SPICE-like simulations is limited to smaller circuits due to the dramatic increase in runtime with the addition of circuit standard cells and input vectors.

Another method for understanding SET effects focuses on characterizing these effects within individual standard cells, as referenced in various studies [41, 42, 43, 44], and then assessing how these effects propagate through a specific circuit by using the resultant characterization data. This process involves conducting SPICE-like simulations for standard cells within a given library just once, with the outcomes saved in look-up tables (LUTs) in the same manner as timing and power library characterization. While this characterization-based strategy can achieve high levels of accuracy, it also comes with downsides, namely the storage space needed for the LUTs might be excessively large, and analyzing SET effects in a particular circuit could become highly time-intensive due to the need to read-in and access a huge number of characterisation LUTs.

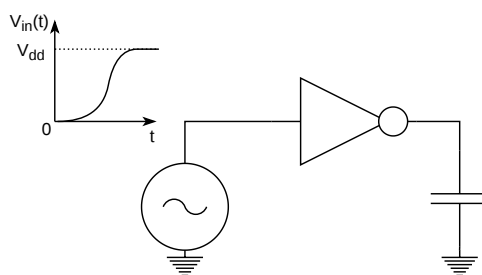
Performing irradiation experiments is another method to study the propagation effects of SETs, as highlighted in publication [45, 46]. This technique allows for the evaluation of SET effects across the entire circuit, including the distribution of SET pulse widths. However, precisely capturing the SET pulse width is impossible due to the inherent randomness in strike locations when actual radiation is used. Irradiation experiments also come with high costs and the prerequisite of a manufactured test chip, rendering them rare and primarily used for validating design solutions or the accuracy of simulation results. A more cost-effective alternative is laser testing [47], which can be performed in-house and offers the advantage of targeting specific circuit nodes for localized SET injection. Nonetheless, this method still requires a manufactured test chip for execution, which is costly both in terms of resources and time.

Thus, the need of a methodology which adheres to electrical and circuit design and analysis rules, while also being able to cover the whole circuit is justified.

# Chapter 4

## Proposed Delay Calculation Methodology for STA

### 4.1 Driver Side Delay and Slew Calculation



**Figure 4.1.1:** Circuit setup for NLDM and CCS characterization

Driver delay and slew calculation is the first step that must be performed in order to complete the stage delay and slew calculation. First of all, the CCS Timing library characterization vectors contain current waveforms. For the scope of this work, it is impractical to directly use current in calculations, as all capacitance formulas are derived using charge  $Q$  which is dependent on voltage. Thus, current vectors must be converted to voltage. All current vectors are characterized using the circuit setup depicted in Figure 4.1.1. The capacitor shown corresponds to `total_output_net_capacitance`, while the slew of the pre-driver waveform of the depicted voltage source corresponds to `input_net_transition`. These values are stored in the vectors and `total_output_net_capacitance` can be used to convert current values

to voltage using formula:

$$V(t) = \frac{Q(t)}{C} = \frac{\int i(t) dt}{C}$$

where  $V(t)$  is voltage,  $Q(t)$  is charge,  $i(t)$  is current and  $C$  is `total_output_net_capacitance`. Charge is the integral of current, and can be calculated as a Riemann sum using the trapezoidal rule. This procedure is depicted in algorithm 5.

---

**Algorithm 5:** *voltage\_vector*

---

```

1 Input: Library Current Vector Waveform  $I(t) = \{t_j, i_j\}$ ,
   total_output_net_capacitance  $C$ , measurement voltage thresholds
    $VM = \{vs_l, vd, vs_h\}$ 
2 Output: Voltage Waveform  $V(t) = \{t_j, v_j\}$ , measurement threshold time points
    $M = \{s_l, d, s_h\}$ 
3 begin
4    $V[0] = 0$ ; // initialize first waveform point voltage
5    $T[0] = I.t_0$ ; // initialize first waveform point time
6    $v = 0$ ;
7   for  $j \leftarrow 1$  to  $V.size()$  do
8      $i_l = I.i[j - 1]$ ; // lower current value
9      $i_h = I.i[j]$ ; // higher current value
10     $t_l = I.t[j - 1]$ ; // lower time point
11     $t_h = I.t[j]$ ; // higher time point
12     $v = \frac{(i_l + i_h) * (t_h - t_l)}{2C}$ ; // segment  $j$  trapezoidal rule voltage //
13     $V[j] = v + V[j - 1]$ 
14     $T[j] = t_h$ 
15    // lower slew threshold //
16    if  $V[j - 1] < vs_l < V[j]$  then
17      |  $s_l = \text{linear\_interpolation}(vs_l, V[j - 1], V[j], t_l, t_h)$ ;
18    end
19    // delay threshold //
20    if  $V[j - 1] < vd < V[j]$  then
21      |  $d = \text{linear\_interpolation}(vd, V[j - 1], V[j], t_l, t_h)$ ;
22    end
23    // upper slew threshold //
24    if  $V[j - 1] < vs_h < V[j]$  then
25      |  $s_h = \text{linear\_interpolation}(vs_h, V[j - 1], V[j], t_l, t_h)$ ;
26    end
27  end
28   $V(t) = \{T, V\}$ 
29 end

```

---

Input is the library current vector waveform  $I(t)$  and it's corresponding capacitance  $C$ .

Also, slew and delay measurement threshold voltages  $VM$  are needed to cache the related measurement time points  $M$  for fast lookup. Output is the calculated voltage waveform  $V(t)$  for this vector. Lines 3 and 4 are used to initialize the first waveform point. Then, in lines 6 - 13, for each current waveform segment, trapezoidal area is calculated and divided with capacitance  $C$ . Each segment value is summed to previous voltage to create the new voltage value. Finally, the calculated time and voltage points are stored in the vector.

The next step is to use the calculated voltage waveforms to calculate delay and slew for any output capacitance and input slew. Algorithm 6 depicts this procedure.

In more detail, algorithm 6 accepts as input the driver timing arc under investigation  $arc_{drv}$ , all receiver capacitance arcs  $C_{rcv,i}$ , slew at driver timing arc input  $s_{in}$  and receiver output loads  $C_{out_{rcv,i}}$ . It is important to remember that in order to capture parasitic capacitive phenomena such as Miller coupling capacitance within each receiver cell, receiver capacitance arcs are modelled as a function of receiver input slew and output load. Thus, there is interdependence between driver output slew and receiver capacitance. The algorithm calculates driver output delay and slew as output. The first step is to access the current vectors from the timing arc and calculate total output net capacitance. In this initial phase slew at receiver inputs is not known. Thus, to begin with, fixed NLDM receiver capacitance values are used. Then, the following procedure is performed iteratively until slew at driver output converges. Four current vectors are used for delay and slew calculation. Every vector contains a corresponding input slew and output load that was used in characterization to produce this vector. Thus, the algorithm retrieves four vectors  $vec_1, vec_2, vec_3, vec_4$  and their corresponding input slew  $tr_l$  and output load  $C_l$  pairs, which bound  $s_{in}$  and  $C_1$ . Note that, here,  $C_l$  is used to calculate lowest slew time point and delay time point. Four corresponding slew low time points are obtained from the vectors. Also, the delay time points are retrieved. By applying bilinear interpolation with these four slew lower points and  $tr_l$  and  $C_l$  pairs, the driver output slew lower point  $s_l$  is obtained. Same applies with delay point  $d_t$ . Then the same steps are applied with  $C_2$  to obtain the upper slew time point  $s_h$ . Driver output slew  $s$  is then obtained at line 28. The whole procedure iterates until  $s$  has converged. Then to calculate timing arc delay, input reference time must be subtracted from delay point. Input reference time depends only on driver input predriver characterization waveform, only driver input slew

is required to calculate it, in the same manner as above points. Finally, to obtain the timing arc delay  $d$ , input reference time  $r_t$  is subtracted from delay time point  $d_t$  in line 34.

### 4.1.1 Dynamic Capacitance

At this point, it is important to discuss the effect of interconnect loading to the driver cell. It is obvious that algorithm 6 is incapable of handling interconnect loading when resistive components exist in the interconnect. As stressed in Section 3.1 the discontinuity of traditional effective capacitance methodologies at voltage region switch points can negatively impact accuracy in waveform creation. Thus, while embracing the same intuition behind the effective charge transfer concept, it is crucial to generalize the methodology to account for all waveform points of driver waveform. In other words, this work proposes a generalized approach to determine dynamic capacitance for all waveform voltage regions, based on Equation 3.1.2. This equation can provide each interconnect node's the dynamic capacitance contribution to the overall dynamic capacitance seen by the driver, if it is rewritten as:

$$C_{d,l} = \frac{\int_{T_i}^{T_{i+1}} i_d(t) dt}{V_{l+1} - V_l} = \frac{Q_{node}}{V_{l+1} - V_l} = \frac{\Delta V_{node} C_{node}}{V_{l+1} - V_l} \quad (4.1.1)$$

Thus, the unknown here is the voltage difference at every interconnect node, for any driver waveform voltage region  $V_{l+1} - V_l$ . The following section provides a complete methodology of determining these voltage values at every interconnect node for all parts of the driver waveform.

## 4.2 Interconnect Signal Propagation

For the interconnect signal propagation problem, there are two separate issues that must be addressed in order to achieve accuracy and reasonable performance. On the one hand, the interconnect input excitation signal must be expressed in a mathematical form that is convenient to calculate the output. Thus, the PWL .lib representation must be converted to a different, strictly mathematical representation. On the other hand, the interconnect transfer function must be calculated to enable convolution with input signal to determine the output. This section analyzes all parts of the mathematical methodology required to

**Algorithm 6:** *calculate\_driver\_delay\_slew*


---

```

1 Input: Driver Library Timing Arc  $arc_{drv}$ , Receiver Capacitance Arcs  $C_{rcv,i}$ , Driver
   Input Slew  $s_{in}$ , Receivers Output Loads  $Cout_{rcv,i}$ 
2 Output: Driver Delay  $d$  and Slew  $s$ 
3 begin
4    $V = \{0\}$ ; // initialize waveform voltages
5    $T = \{0\}$ ; // initialize waveform time points
6    $Vecs = \text{get\_current\_vectors}(arc_{drv})$ ; // get CCS vectors from timing arc
7    $C = \text{sum\_all\_receivers\_NLDM\_capacitance}(C_{rcv,i})$ ; // start with NLDM value
8    $C_1 = C_2 = C$ ; // start with receiver capacitance NLDM value //
9   // initialize delay and slew //
10   $d = 0, s = 0$ ;
11  do
12    // get 4 closer CCS vectors for  $C_1$  //
13     $\{vec_1, vec_2, vec_3, vec_4, tr_l, C_l\} = \text{find\_4\_closer\_vectors}(Vecs, s_{in}, C_1)$ ;
14    // get lower slew and delay time point from 4 closer vectors //
15     $S_l = \{s_{1l}, s_{2l}, s_{3l}, s_{4l}\} = \{vec_1.s_l, vec_2.s_l, vec_3.s_l, vec_4.s_l\}$ ;
16     $D = \{d_1, d_2, d_3, d_4\} = \{vec_1.s_l, vec_2.s_l, vec_3.s_l, vec_4.s_l\}$ ;
17    // get 4 closer CCS vectors for  $C_2$  //
18     $\{vec_1, vec_2, vec_3, vec_4, tr_h, C_h\} = \text{find\_4\_closer\_vectors}(Vecs, s_{in}, C_2)$ ;
19    // get upper slew time point from 4 closer vectors //
20     $S_h = \{s_{1h}, s_{2h}, s_{3h}, s_{4h}\} = \{vec_1.s_h, vec_2.s_h, vec_3.s_h, vec_4.s_h\}$ ;
21    // calculate all 3 points //
22    // slew lower time point //
23     $s_l = \text{bilinear\_interpolation}(S_l, tr_l, C_l, s_{in}, C_1)$ ; // interpolate
24    // delay time point //
25     $d_t = \text{bilinear\_interpolation}(D, tr_l, C_l, s_{in}, C_1)$ ; // interpolate
26    // slew upper time point //
27     $s_h = \text{bilinear\_interpolation}(S_h, tr_h, C_h, s_{in}, C_2)$ ; // interpolate
28    // calculate driver output slew //
29     $s = s_h - s_l$ ;
30    // update receiver capacitance  $C_1, C_2$ 
31     $\{C_1, C_2\} = \text{update\_receiver\_capacitance}(C_{rcv,i}, s, Cout_{rcv,i})$ ;
32  while ( $s$  not converged);
33  // get input reference time  $r_t$ ; depends only on input slew //
34   $\{r_1, r_2\} = \{vec_1.r, vec_2.r\}$ 
35   $r_t = \text{linear\_interpolation}(r_1, r_2, tr_l, s_{in})$ ; // interpolate
36  // calculate driver output delay //
37   $d = d_t - r_t$ ;
38 end

```

---

derive the output waveform, and the resultant output waveform closed form formulas.

### 4.2.1 Input Voltage Waveform Analysis

First and foremost, the existing methodologies detailed in literature typically utilize either step or ramp inputs. While this approach was appropriate for older technologies that relied on NLDM, it falls short in terms of accuracy for modern process technologies, as typically the use of entire waveforms in delay calculation is required. Consequently, this subsection introduces a method for converting input PWL waveforms into a set of equations. These equations maintain the principle of superposition, enabling their use with interconnect transfer functions to accurately derive the output voltage waveform. This conversion is a crucial component of the proposed methodology, especially as the mathematical tool to derive the output waveform is the Laplace transform. The Laplace transform is a linear transformation, as illustrated in Equation 4.2.1, which is the main reason why the input signal must be expressed as a superposition of functions with trivial Laplace transform formula.

$$\mathcal{L}\{\kappa f(t) + \lambda g(t)\} = \kappa \mathcal{L}\{f(t)\} + \lambda \mathcal{L}\{g(t)\}, \quad \lambda, \kappa \in \mathbb{R} \quad (4.2.1)$$

The preservation of the superposition property for the output voltage waveform of an interconnect is contingent upon the ability to express the input waveform through superposition itself, as detailed in any signals and systems book as [26]. Therefore, for any PWL signal expressed as  $t_0, v_0, \dots, t_n, v_n, n > 1$ , where  $t_n$  and  $v_n$  denote the respective time and voltage pairs, it is essential that each segment  $S_{i+1}t_i, v_i, t_{i+1}, v_{i+1}$  is formulated as a properties summation of all preceding PWL segments. In this context, a novel representation for each PWL segment is proposed. This representation takes the form of a ray equation, originating from the point  $t_i, v_i$  and characterized by a slope  $A_i$ . It is important to note that the Heaviside function  $\theta(t)$  is utilized in defining this ray equation.

$$S_{i+1}(t) = (A_i t + b_i) \theta(t - t_i) \quad (4.2.2)$$

$$A_i = \frac{v_{i+1} - v_i}{t_{i+1} - t_i} \quad (4.2.3)$$

$$b_i = v_{i+1} - A_i t_{i+1} \quad (4.2.4)$$

The ultimate goal here is to formulate  $n$  such ray equations, whose collective sum recreates the original PWL signal. This objective can be accomplished by generating a unique ray equation  $r_{i+1}(t)$  for each segment. These individual equations, when added to the ray equations of all preceding segments  $S_i(t)$ , result in the creation of the segment ray equation  $S_{i+1}(t)$ . Essentially, this process involves determining the slope  $A'_i$  and the y-axis intercept  $b'_i$  of  $r_{i+1}(t)$ . This is done by calculating the difference between the previous segment ray equation  $S_i(t)$  and the current segment ray equation  $S_{i+1}(t)$ . To elaborate, the ray equation  $r_{i+1}(t)$  is defined as:

$$\begin{aligned} r_{i+1}(t) &= \\ &= S_{i+1}(t) - S_i(t) \\ &= (A'_i t + b'_i) \theta(t - t_i) \end{aligned} \quad (4.2.5)$$

The subtraction of the slopes of two subsequent PWL segments is used to determine Slope  $A'_i$ :

$$A'_i = A_i - A_{i-1} \quad (4.2.6)$$

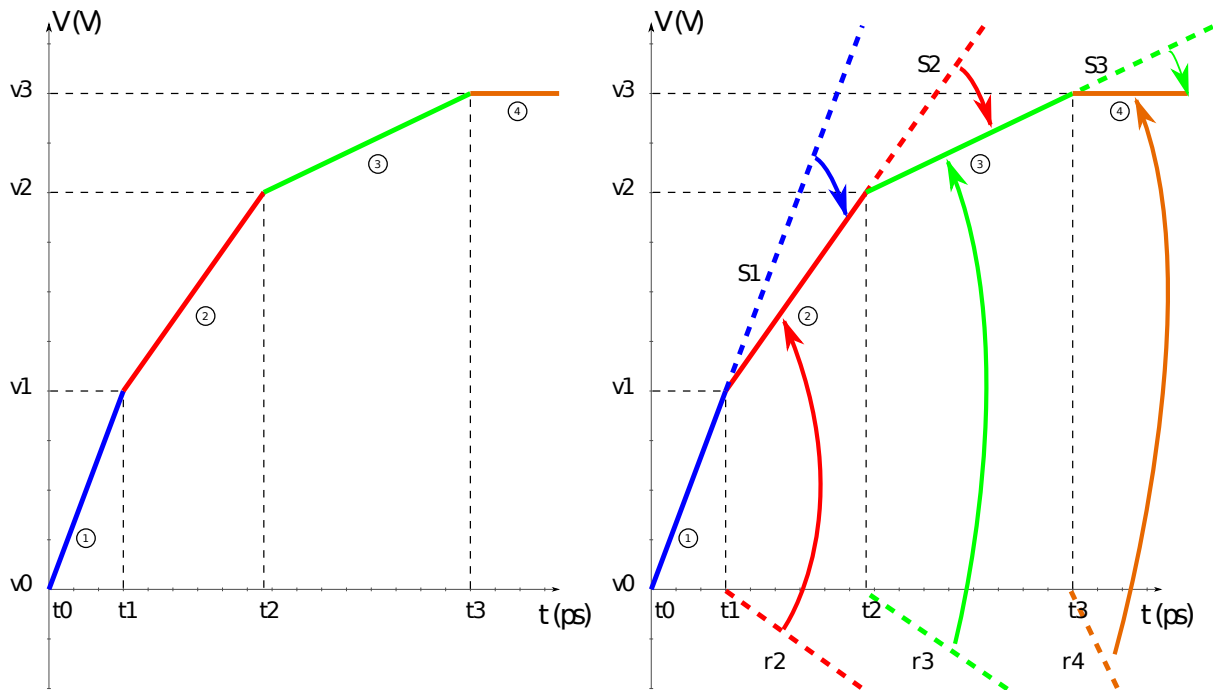
Also, y-axis intercept values  $b'_i$  are determined in the same manner as:

$$b'_i = b_i - b_{i-1} \quad (4.2.7)$$

To capture the saturation portion of the PWL waveform, *i. e.* either the supply voltage or 0, it is necessary for the slope of the final ray equation to be the inverse of the slope from its preceding segment. In other words, this approach will yield a slope of 0, while the value of  $b'_i$  can be determined using Equation 4.2.4. It is important to note that this final segment corresponds to the supply voltage in digital circuits. The sum of these ray equations will precisely match the initial PWL signal. Finally, the Laplace transform for each ray equation is provided in Equation 4.2.8. Here, more emphasis is particularly given

to including Heaviside function in the transformation, in order to define a ray equation instead of an infinite line equation:

$$\begin{aligned}
 \mathcal{L}\{r_{i+1}(t)\} &= \\
 &= R_{i+1}(s) \\
 &= e^{-t_i s} \left( \frac{b'_i}{s} + \frac{A'_i(t_i s + 1)}{s^2} \right), \quad s \in \mathbb{C}
 \end{aligned} \tag{4.2.8}$$



**Figure 4.2.1:** PWL Signal to Sum of Ray Equations Analysis

Figure 4.2.1 depicts an illustrative example of the PWL waveform analysis proposed above. Each PWL segment is derived by summing subsequent ray equations as proposed. So, for example, ray equation of segment No. 3 is produced by adding all subsequent ray equations:

$$S3 = S1 + r2 + r3$$

## 4.2.2 Pi-model Stage Full Waveform Propagation

At this point, the input PWL voltage waveform has been transformed into a format that is convenient for analysis in the frequency domain. The subsequent step involves constructing the pi-model transfer function. When the pi-model interconnect is connected

directly to the driver cell, the capacitor  $C_1$  is in parallel with the driver. According to fundamental principles of circuit analysis, this configuration implies that  $C_1$  does not influence the propagation of the waveform. In contrast,  $C_2$  is connected to the resistor  $R$ , making the pi-model function as a low pass filter. This observation leads to the reasonable assumption that the transfer function of the pi-model will be identical to that of a single-stage low pass filter. Thus, there is no need for any approximation in the calculation of poles and residues. The transfer function for a single-stage low pass filter is illustrated in Equation 4.2.9.

$$H(s) = \frac{p}{s + p}, \quad (4.2.9)$$

where  $p = \frac{1}{RC_2}$ . In general, the output voltage waveform in frequency domain is determined by multiplication of input function with transfer function, thus for each ray equation:

$$\begin{aligned} V_{i+1}(s) &= \\ &= R_{i+1}(s)H(s) \\ &= e^{-t_i s} \left( \frac{b'_i}{s} + \frac{A'_i(t_i s + 1)}{s^2} \right) \frac{p}{s + p}, \end{aligned} \quad (4.2.10)$$

Then, by utilizing the Inverse Laplace transform, the output waveform in time domain is determined as:

$$\begin{aligned} \mathcal{L}^{-1}\{V_{i+1}(s)\} &= \\ &= v_{i+1}(t) \\ &= \theta(t - t_i) * \left( \left( \frac{A'_i}{p} - b'_i \right) (e^{-p(t-t_i)} - 1) + A'_i(t - t_i e^{-p(t-t_i)}) \right) \end{aligned} \quad (4.2.11)$$

Finally, in order to obtain voltage at any given time point, the superposition of all Equations 4.2.11 provides the output voltage waveform  $v_\pi(t)$ .

$$v_\pi(t) = \sum_{i=0}^n v_i(t) \quad (4.2.12)$$

### 4.2.3 RC Tree Interconnect Stage Full Waveform Propagation

Evidently, the majority of the equations presented in section 4.2.2 are not suitable for arbitrary interconnects, such as the one shown in Figure 3.1.2. Therefore, the approach

outlined in section 3.1.1 can be utilized to ascertain the poles and residues of a lower order transfer function, denoted as  $H'(s)$ . This lower order function closely mirrors the actual transfer function for each interconnect node. This approximation is feasible, because the terms of the actual transfer function are decaying with order. In other words, the influence of each higher-order pole-residue pair on the overall value of the transfer function decreases as the order increases.

Following this, the methodology detailed in section 4.2.2 should be applied to determine the output voltage waveform. In frequency domain, the output  $V_{i+1}(s)$  for each segment is calculated by multiplying the input  $R_{i+1}(s)$  from Equation 4.2.8 with the approximate transfer function  $H'(s)$ :

$$\begin{aligned} V_{i+1}(s) &= \\ &= R_{i+1}(s)H'(s) \\ &= e^{-t_i s} \left( \frac{b'_i}{s} + \frac{A'_i(t_i s + 1)}{s^2} \right) \sum_{q=1}^N \frac{k_q}{s + p_q} \end{aligned} \quad (4.2.13)$$

Then using the inverse Laplace transform the output voltage waveform equation  $v_{i+1}(t)$  for each segment can be obtained as follows:

$$\begin{aligned} \mathcal{L}^{-1}\{V_{i+1}(s)\} &= \\ &= v_{i+1}(t) \\ &= \theta(t - t_i) \sum_{q=1}^N \frac{k_q}{p_q} \left( A'_i t - \frac{A'_i}{p_q} + b'_i + \left( \frac{A'_i}{p_q} - b'_i - A'_i t_i \right) e^{-p_q(t-t_i)} \right) \end{aligned} \quad (4.2.14)$$

Finally, voltage waveform  $v_{dnet}(t)$  at any arbitrary RC interconnect node is obtained by applying superposition over all segment output voltage Equations 4.2.14:

$$v_{dnet}(t) = \sum_{i=0}^n v_i(t) \quad (4.2.15)$$

It should be emphasized that the equations proposed are designed to account for any arbitrary number  $N$  of poles and residues. However, extensive experimental investigations, which involved analyzing thousands of distinct delay calculation stages, have demonstrated that the output voltage waveform generated is virtually indistinguishable from the golden

waveform obtained from SPICE electrical simulations for any  $N \geq 2$ . Thus, throughout this thesis, two poles and residues are used in scenarios where there are no coupling capacitors present. In cases involving nets with coupling, the use of three or four poles has been found to yield the desired level of accuracy.

### 4.3 Generalized Dynamic Capacitance and Driver Voltage Waveform Calculation

All presented full waveform voltage propagation methods of previous sections can enable the calculation of Dynamic Capacitance, which is the last missing piece to complete the full waveform calculation at driver side. Again, it is important to remember that CCS current vectors are indexed by driver input slew and output load. Slew at driver input pin is always a given; there are two objectives at this point. The first one is to derive a dynamic capacitance value per waveform voltage region, first for picking the correct current vectors out of characterization data, and second for giving accurate input to the bilinear interpolation function. The second one is to use this dynamic capacitance per region in order to accurately create the driver voltage waveform.

As discussed in Section 3.1, previous academic publications have already proved that deriving a single Dynamic or Effective Capacitance value always achieves more than 2% relative error vs SPICE for modern process nodes where resistance is not negligible. As a result, a multiple voltage region approach is proposed, which is also convenient for the proposed full waveform propagation methods of Section 4.2. Also, in order to maintain clarity and simplicity, the formulation of section 4.2.1 is also used in this section, for defining the driver waveform voltage regions.

Based on this, each voltage region with voltage thresholds  $v_i, v_{i+1}$  and time points  $t_i, t_{i+1}$  is defined by a PWL segment  $S_{i+1}\{\{t_i, v_i\}, \{t_{i+1}, v_{i+1}\}\}$ . A missing part to this is how to select which driver waveform voltage thresholds are better to use in order to maintain a fairly small amount of calculation, while achieving maximum accuracy. After exhaustive experimentation, it is safe to assume that the most balanced choice regarding this trade-off is to always pick voltage thresholds based on characterization data. More specifically, the voltage thresholds that are picked are the voltages in CCS LUTs indexes, while the

**Algorithm 7:** *calculate\_net\_dynamic\_capacitance*


---

```

1 Input: Interconnect SPEF Info  $RC_{net}$ , Receiver Capacitance Arcs  $C_{rcv,j}$ , Voltages  $V$ ,
   Time Points  $T$ , Region index  $i$ 
2 Output: Dynamic Capacitance  $C_{dyn}$ 
3 begin
4    $C_{dyn} = 0$ 
5    $v_{drv,l} = V[i - 1]$ ; // low driver voltage
6    $v_{drv,h} = V[i]$ ; // high driver voltage
7    $t_l = T[i - 1]$ ; // low waveform time point
8    $t_h = T[i]$ ; // high waveform time point
9   // calculate poles and residues with receiver  $C_1$  and  $C_2$  //
10  if  $i == 0$  then
11    | AWE_poles_residues( $RC_{net}, C_{rcv}.C_1$ );
12  end
13  else if  $i == V.size()/2$  then
14    | AWE_poles_residues( $RC_{net}, C_{rcv}.C_2$ );
15  end
16  // tally up all net nodes contribution to  $C_{dyn}$ 
17  for each node  $n$  in  $RC_{net}$  do
18    | // sum up all capacitances attached to  $n$  //
19    |  $C_{node} = \text{sum\_node\_capacitances}(n, RC_{net}, C_{rcv})$ ;
20    | // calculate node voltage for  $t_l$  and  $t_h$  //
21    |  $\{v_{n,l}, v_{n,h}\} = \{v_{dnet}(t_l), v_{dnet}(t_h)\}$ ; // Equation 4.2.15
22    |  $C_{dyn} = C_{dyn} + (v_{n,h} - v_{n,l})C_{node}/(v_{drv,h} - v_{drv,l})$ 
23  end
24 end

```

---

unknown is always time point  $t_{i+1}$ .

Algorithm 8 presents the voltage waveform calculation procedure. The algorithm accepts driver timing arc  $arc_{drv}$ , the attached interconnect information  $RC_{net}$ , all receiver capacitance arcs  $C_{rcv,i}$ , slew at driver timing arc input  $s_{in}$  and receiver output loads  $C_{out_{rcv,i}}$ . To begin, waveform time points  $T$  and voltage points  $V$  are initialized. As with algorithm 6, library current vectors  $Vecs$  are retrieved from the timing arc  $arc_{arc}$ , and total output capacitance  $C_{tot}$  is initialized to the sum of all interconnect capacitors  $C_{net}$  plus the sum of NLDM fixed capacitance values  $C_{rcvs}$  for receiver cells.

First, four closer vectors are obtained and waveform time point  $T[0]$  and voltage are calculated, via bilinear interpolation, using  $C_{tot}$ . Then, for each waveform point, the following procedure is performed. First, dynamic capacitance  $C_{dyn}$  is initially set to  $C_{tot}$ . Then, iterations are performed as dynamic capacitance value  $C_{dyn}$  depends on newly

calculated point  $T[i]$  and vice versa. Again, four closer vectors are obtained from CCS data. Then, timepoints  $t_l$  and  $t_h$  are calculated for voltage points  $v_l$  and  $v_h$  respectively. Here, it is important to emphasize a subtle but significant detail. Previous time point  $T[i - 1]$  was potentially obtained by a different set of four closer vectors from CCS library data than the set which is currently picked to calculate  $t_h$ . This means that the lower point of current voltage waveform segment may differ from  $T[i - 1]$  which in turn may cause discontinuity in the driver voltage waveform. This justifies the need of calculating both  $t_h$  for voltage  $v_h$  and  $t_l$  for voltage  $v_l$  using the four closer CCS vectors set selected for  $t_h$ . Then in order to prevent discontinuity, current waveform time point  $T[i]$  is calculated in line 34 as the sum of previous time point  $T[i - 1]$  with the newly calculated voltage region slew  $t_h - t_l$ .

At this point, knowing the new timepoint  $T[i]$ , dynamic capacitance  $C_{dyn}$  can be calculated using routine `calculate_net_dynamic_capacitance`. Routine `calculate_net_dynamic_capacitance` description will be presented right after the description of algorithm 8. The iterations for each voltage region finish, when  $T[i]$  converges. Upon completion, the driver voltage waveform is known.

Algorithm 7 describes the procedure of calculating dynamic capacitance value  $C_{dyn}$  for a voltage region. The algorithm accepts as inputs the interconnect parasitics information  $RC_{net}$ , all receiver arcs  $C_{rcv,j}$ , waveform voltage and time points  $V, T$  and the voltage region index  $i$ . Of course, the output is dynamic capacitance  $C_{dyn}$  as seen by driver cell for voltage region  $i$ . The algorithm begins with initializing  $C_{dyn}$ , getting the driver waveform low and high voltage points  $v_{drv,l}, v_{drv,h}$  and related time points  $t_l, t_h$  for region  $i$ . Then, based on the waveform part, interconnect transfer function, poles and residues are calculated by employing the powerful methodology of section 3.1.1. Note that this step cannot be omitted as receiver capacitance is different below and above 50% of  $V_{dd}$ . To generalize, any number of receiver capacitance regions may be included in the CCS library data. To address this, the proposed methodology re-iterates at receiver capacitance threshold crossings, in case the receiver capacitance value differs significantly from the previously calculated one, so as to recalculate transfer function poles and residues. This is reasonable as modifying the receiver capacitance directly translates to altering the interconnect transfer function. However, for the purpose of maintaining simplicity in

explanation, the algorithm is presented with only two receiver capacitance values,  $C_1$  and  $C_2$ , without loss of generality.

Then, each node's contribution to dynamic capacitance must be calculated. By expanding Equation 4.1.1, the dynamic capacitance  $C_{dyn}$  for node  $n$  and region  $i$  defined by threshold indexes  $\{j, j + 1\}$ , can be derived as:

$$C_{dyn} = C_{n,j \rightarrow j+1} = \frac{Q_n}{\Delta V_{drv}} = \frac{\Delta v_n C_n}{\Delta V_{drv}} = \frac{v_{n,j+1} - v_{n,j}}{v_{j+1} - v_j} C_n \quad (4.3.1)$$

The terms of this formula are as follows:  $C_{n,j \rightarrow j+1}$  is the Dynamic Capacitance of node  $n$ ,  $C_n$  is actual parasitic capacitance the node attached to the node,  $Q_n$  is the equivalent charge being transferred from driver to the interconnect within the voltage region of focus,  $\Delta V_{drv}$  expresses the voltage difference at driver side and  $v_{n,j+1}$  and  $v_{n,j}$  are the node voltages at timepoints  $t_{j+1}, t_j$ .

Thus it is evident, that by analyzing charge  $Q$  using the capacitor charge formula  $Q = CV$ , dynamic capacitance for each node is the ratio between the voltage difference at node  $n$  and voltage difference at driver side multiplied with capacitance at the node. Thus the unknowns are voltage values  $\{v_{n,l}, v_{n,h}\}$ . Algorithm 7 loops through all interconnect nodes and first of all calculates the sum of all capacitors that may be attached on the node. Then, node voltage values  $\{v_{n,l}, v_{n,h}\}$  are derived, using Equation 4.2.12 for pi models or Equation 4.2.15 for the general case of arbitrary RC. Finally, the algorithm calculates dynamic capacitance for node  $n$ , using Equation 4.3.1 and accumulates it to  $C_{dyn}$ . Upon completion  $C_{dyn}$  is known.

## 4.4 Receiver Delay And Slew Approximation

Finally, the only missing functionality for a complete delay calculation methodology is delay and slew calculation for receiver cells inputs. In general, this issue can be trivially resolved when calculating dynamic capacitance, in case the interconnect is lightly loading the driver, and thus receiver input pins voltage exceeds the upper slew voltage threshold when driver reaches 100%. In contrast, if the interconnect loads the driver cell significantly and resistance is high, it is impossible for receiver cell voltage to exceed upper slew voltage threshold when driver reaches 100%. Thus, Equation 4.2.15 must be used with driver

waveform  $T, V$  calculated by algorithm 8 to propagate the waveform to receiver inputs and calculate slew and delay time points. This may sound trivial given Equation 4.2.15, however by a more detailed inspection both of Equation 4.2.12 and Equation 4.2.15 one can obviously derive that no closed form solution exists for  $t$  and as a consequence it is impossible to directly calculate delay and slew time points.

---

**Algorithm 9:** *Newton's Method*

---

```

1 Input: function  $f(x)$ , first derivative  $f'(x)$ , point  $f(x_r)$ , max iterations number  $i_{max}$ , initial
   guess  $x_{init}$ , tolerance  $\epsilon$ 
2 Output: root  $x_r$ 
3 begin
4    $i = 1$ ;
5    $x_0 = x_{init}$ 
6   while  $i \leq i_{max}$  do
7      $x_r = x_0 - \frac{f(x_0)}{f'(x_0)}$ 
8     if  $|1 - x_0/x_r| < \epsilon$  then
9        $\text{return } x_r$ ;
10    end
11     $i = i + 1$ ;
12     $x_0 = x_r$ ;
13  end
14 end

```

---

The first and foremost idea for solving such an arithmetic issue is to employ Newton's method, which is a derivative-based bracketing root-finder. This is always applicable particularly for Equation 4.2.12 and Equation 4.2.15, as derivatives always exist for these. Considering performance, Newton's method is of superlinear convergence thus it was considered a good candidate solution for this problem. However, the method performed more than 60 iterations for delay and slew time points on average for all benchmarks presented in Section 6.1.

Thus, to cope with the aforementioned issue, a higher order Householder's method root-finder, which employs second and third derivatives, is used to achieve even greater convergence rate [48]. It was experimentally confirmed that this method converges in less than 8 iterations and this is considered adequate for the purpose of calculating delay and slew points. It is worth noting that Householder's method underlying algorithm is the exact same with Newton's method algorithm, however Equation 4.4.1 is used to derive the next candidate root, instead of purely using the first derivative for this purpose, like

in line 6 of algorithm 9.

$$t_{n+1} = t_n - \frac{6y(t_n)y'(t_n)^2 - 3y(t_n)^2y''(t_n)}{6y'(t_n)^3 - 6y(t_n)y'(t_n)y''(t_n) + y(t_n)^2y'''(t_n)} \quad (4.4.1)$$

It is also worth mentioning that, as with Newton's method, Householder's method requires all employed higher order derivatives to exist in order to be applicable. Regarding Equations 4.2.12 and 4.2.15, their second and third derivatives always exist.

For the extreme cases, where Householder's method produces extreme next candidate root values, a classic bisection bracketing rootfinder is adequate to contain the candidate solutions within corresponding time points of 0 and 100% of  $V_{dd}$ . This is rarely happening and can be explained by the fact that Equations 4.2.12 and 4.2.15 are monotonic and continuous functions for all  $t \geq 0$ , but 0 for  $t < 0$  due to the use of Heaviside function.

**Algorithm 8:** *calculate\_driver\_waveform*


---

```

1 Input: Driver Library Timing Arc  $arc_{drv}$ , Interconnect SPEF Info  $RC_{net}$ , Receiver
   Capacitance Arcs  $C_{rcv,j}$ , Driver Input Slew  $s_{in}$ , Receivers Output Loads  $C_{out_{rcv,i}}$ 
2 Output: Driver Voltage Waveform  $V_{drv}(t) = \{t_i, v_i\}$ 
3 begin
4    $V = \{0\}$ ; // initialize waveform voltages
5    $T = \{0\}$ ; // initialize waveform time points
6    $Vecs = \text{get\_current\_vectors}(arc_{drv})$ ; // get CCS current vectors from timing
   arc
7    $C_{net} = \text{get\_net\_total\_capacitance}(RC_{net})$ ; // sum all net capacitors
8    $C_{rcvs} = \text{sum\_all\_receivers\_NLDM\_capacitance}(C_{rcv,j})$ ; // start with NLDM
   value
9    $C_{tot} = C_{net} + C_{rcvs}$ ; // total capacitance
10  // get 4 closer CCS vectors //
11   $\{vec_1, vec_2, vec_3, vec_4\} = \text{find\_4\_closer\_vectors}(Vecs, s_{in}, C_{tot})$ ;
12  // calculate voltage thresholds from 4 closer vectors //
13   $V = \text{voltage\_points\_from\_vectors}(vec_1, vec_2, vec_3, vec_4)$ ;
14  // get first time point of all 4 closer vectors //
15   $\{t_1, t_2, t_3, t_4\} = \{vec_1.T[0], vec_2.T[0], vec_3.T[0], vec_4.T[0]\}$ ;
16  // calculate first waveform time point //
17   $T[0] = \text{bilinear\_interpolation}(t_1, t_2, t_3, t_4, s_{in}, C_{tot})$ ; // interpolate
18  for  $i \leftarrow 1$  to  $V.size()$  do
19     $v_l = V[i - 1]$ ; // lower voltage
20     $v_h = V[i]$ ; // higher voltage
21     $C_{dyn} = C_{tot}$ ; // start with  $C_{tot}$ 
22    do
23      // get 4 closer CCS vectors //
24       $\{vec_1, vec_2, vec_3, vec_4\} = \text{find\_4\_closer\_vectors}(Vecs, s_{in}, C_{dyn})$ ;
25      // calculate voltage thresholds from 4 closer vectors //
26       $V = \text{voltage\_points\_from\_vectors}(vec_1, vec_2, vec_3, vec_4)$ ;
27      // get previous time point of all 4 closer vectors //
28       $\{t_{1l}, t_{2l}, t_{3l}, t_{4l}\} = \{vec_1.T[i - 1], vec_2.T[i - 1], vec_3.T[i - 1], vec_4.T[i - 1]\}$ ;
29      // get current time point of all 4 closer vectors //
30       $\{t_{1h}, t_{2h}, t_{3h}, t_{4h}\} = \{vec_1.T[i], vec_2.T[i], vec_3.T[i], vec_4.T[i]\}$ ;
31      // calculate new low and high time points //
32       $t_l = \text{bilinear\_interpolation}(t_{1l}, t_{2l}, t_{3l}, t_{4l}, s_{in}, C_{dyn})$ ; // interpolate
33       $t_h = \text{bilinear\_interpolation}(t_{1h}, t_{2h}, t_{3h}, t_{4h}, s_{in}, C_{dyn})$ ; // interpolate
34      // new waveform time point is previous + segment slew //
35       $T[i] = T[i - 1] + t_h - t_l$ ;
36      // calculate  $C_{dyn}$  using new time point //
37       $C_{dyn} = \text{calculate\_net\_dynamic\_capacitance}(RC_{net}, C_{rcv}, V, T, i)$ ;
38      while ( $T[i]$  not converged);
39    end
40 end

```

---

# Chapter 5

## Novel SET Analysis Approaches using STA Principles

### 5.1 The SET Analysis Topic Trade-Offs

The following sections describe the proposed methodologies of SET glitch analysis. It is noteworthy here, to emphasize the ultimate goal of the proposed flows. It is reasonable to think that for all cases, there is no way to totally prevent influence of SETs in the normal function of the circuit. However, at this design level, the SET influence can be reasonably mitigated. Overall, there are two competing metrics here, one being accuracy, and the other being performance. Accuracy is directly dependent on two factors, generation/propagation accuracy and design coverage. Then performance is dependent on generation/propagation performance and coverage. Out of all these, coverage is considered the most important of all factors, thus the following compromises are accepted to resolve the trade-off. First of all, propagation performance and accuracy is guaranteed, as the employed STA methodologies guarantee achieving highest possible accuracy for the best possible performance. Then the remaining competing factors are generation accuracy versus generation performance and coverage. However, following the same rationale, generation performance is again prioritized over accuracy. This enables higher coverage, which indirectly translates to higher accuracy in the global scheme of merit.

## 5.2 Particle Strike Scenario

Two main approaches have been proposed in literature for the purpose of modelling and simulating SET-induced current pulses, as outlined in publication [29]. The first approach can be thought of and defined as macro-modeling. For this, the injected charge can be modelled as an individual current source which is then attached to the target gate's output and global ground node. The second approach is micro-modeling, where the induced current is again modelled as set of current sources, but these are directly attached within the internals of the target gate, and more specifically within Pull-Up and Pull-Down network transistor nodes. In the scope of this research work, all investigations are performed for the purpose of analyzing digital ASICs, which means that macro-modelling is selected due to the requirement of dealing with massive circuits with hundreds of thousands gates. However, two different flavours of macro-modelling are used; the first one is used in an industrial tool based flow, where the target is maximum accuracy while performance is second order, thus transistor level simulation is performed for SET generation, with fully detailed transistor level and parasitics description of library standard cells and transistor models. The second one is an integrated flow within UTH CASlab's academic EDA tool, which combines an STA engine with a custom SPICE engine and an analog model that attempts to bound the industrial flow SET Generation as fast as possible. It is worth noting here, that the custom SPICE engine was developed specifically for the purposes of this project.

The double-exponential (DEXP) current source model is the most commonly employed macro-model for simulating SET-induced current, and it is represented by the equation below:

$$I_{SET}(t) = \frac{Q_{total}}{\tau_f - \tau_r} \times (e^{-\frac{t}{\tau_f}} - e^{-\frac{t}{\tau_r}}), \quad (5.2.1)$$

where  $Q_{total}$  is the total collected charge,  $\tau_f$  represents the collection time constant of the NP/PN junction within the affected transistors, which is used to represent the SET glitch decay, and  $\tau_r$  is called the ion-tract establishment time constant. In simpler wording,  $\tau_f$  is the current pulse fall time, while  $\tau_r$  is the current pulse rise time.

Figure 5.2.1 depicts a simple example of attaching the  $I_{SET}$  current source on the output of driver cell.

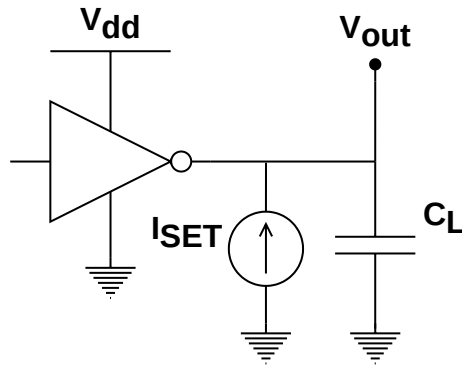


Figure 5.2.1: SET Generation Current Source

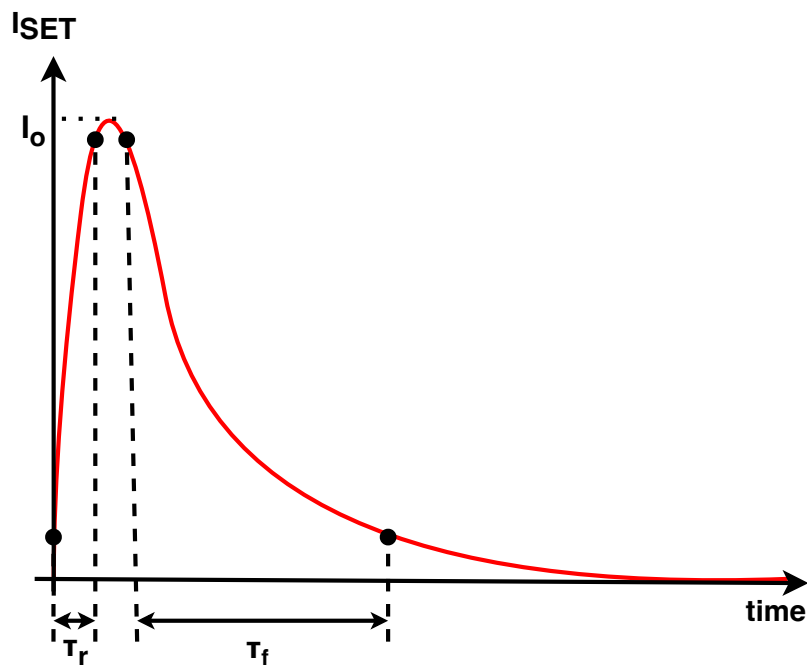


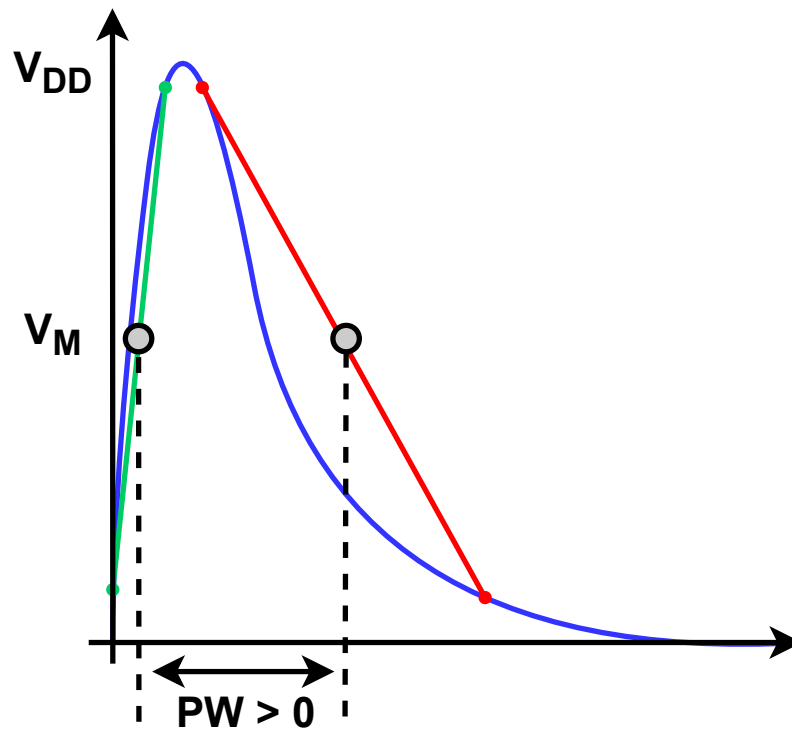
Figure 5.2.2: Double-Exponential (DEXP) Typical Current Waveform

### 5.2.1 SET Pulse Conversion to STA Compatible Format

As discussed in Chapter 2, STA is capable of propagating entire voltage waveforms through paths starting from all timing path startpoints, passing through combinational logic, and reaching the timing path endpoints, namely primary outputs (POs) and the input pins of sequential elements. However, this method is time-intensive and requires significant resources, making it impractical for iterative application across all scenarios, particularly when the analysis data is intended for optimization purposes. When the optimization procedure is targeted toward SET tolerance and radiation hardening, the optimizer will attempt to exploit various strategies and recipes to improve these metrics, highlighting

the importance of the efficiency of the analysis process.

Therefore, it is crucial to encapsulate the critical information of the SET generation pulse into two timing signals, the rise edge and the fall edge, as depicted in Figure 5.2.3. Consequently, every signal edge that is propagated is characterized by its transition time ( $t_{r/f}$ ) and delay ( $t_{d(LH/HL)}$ ). For the phase of SET generation, these measurements are directly extracted from the voltage waveform created in the simulation.



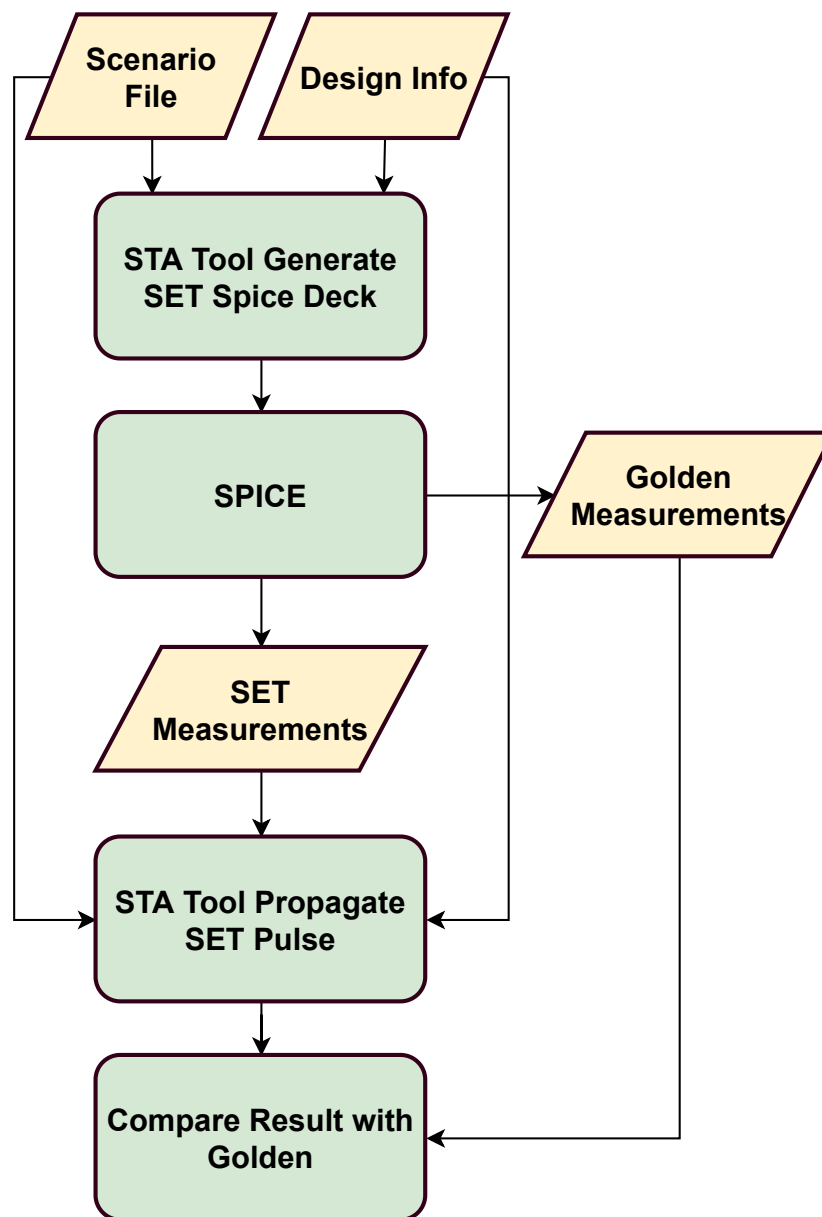
**Figure 5.2.3:** SET pulse decomposition to rise and fall edges for STA engine compatibility

## 5.3 Glitch Generation and Path Based Analysis Propagation using Industrial Tools

The first step towards fast SET analysis was to try and leverage the robustness, accuracy and performance of industrial STA and electrical simulation tools to prove that the intuition of using STA for such a purpose is a viable solution. This resulted in creating a scriptware flow which integrates a commercial SPICE engine for SET generation with an industrial STA tool for glitch propagation. It is worth mentioning that it is evident that the internals of an industrial STA engine can not be affected, thus there is no way of using GBA for glitch propagation as the user has no control of the merging operations performed

for every multi-input cell output pin. Thus, for the above purposes, the industrial STA engine was used only with PBA, as no merging operations are performed for the paths.

In this section, the general scriptware automated flow is presented. This flow utilizes commercial tools for transistor level simulation and Static Timing Analysis (STA) to generate and propagate Single Event Transient (SET) pulses, respectively. Additionally, the process of creating particle strike scenarios is outlined, along with methods for quantifying the PIPB effect and comparing measurements between the golden simulation and the proposed methodology.



**Figure 5.3.1:** Industrial tools SET Generation And PBA-based Propagation Flow

### 5.3.1 Particle Strike Scenario Generation

As a first step, the SET generation scenario must be defined. Each affected path is defined as a pair of 2 points; one point for the SET-affected pin and another one for the timing path endpoint, where the pulse width (PW) measurement must be performed. Also the  $I_{SET}$  current source of Figure 5.2.1 must be defined. It is noteworthy, that the current source can be user-defined and of arbitrary waveform shape, however it is reasonable to select the well adopted double exponential current source model [49] as the generation model of choice, for the scope of this work. Finally, the current source parameters must be configured. Without loss of generality, this can be achieved by providing probability distributions for such parameters. The SET scenario generator was implemented using the most well adopted methodology for random simulation which is the Monte-Carlo method. The implemented simulation is using user defined distributions, to variate the current source parameters. Also, without loss of generality, this work is using normal distributions for generating the particle strike scenarios.

#### 5.3.1.1 SET Generation

In the context of this industrial flow, the next crucial step involves selecting which paths should be thoroughly examined and subsequently generating the necessary SPICE deck for their analysis. To accomplish this, the Path-Based Analysis (PBA) [2] capabilities of the commercial STA engine are employed.

The primary objective of this phase is to identify the most critical path for the specified startpoint and endpoint pins, which are contained in the scenario file. The focus of the entire analysis is exclusively on this specific path, while accounting both for the rising and falling edges of data signals.

While it is certainly possible to extend this analysis to cover complete logical cones, it is important to note that the initial implementation of the scriptware flow was solely designed to assess the methodology's performance and accuracy in isolation. The rationale behind this decision was to ensure that the evaluation process remained free from any external influences, such as side inputs or the complex effects caused by Multiple Input Switching (MIS). Only after thoroughly evaluating the methodology's effectiveness under these controlled conditions it was reasonable to address the proposition of a more comprehensive

analysis involving full logical cones, which will be presented in upcoming sections. This careful and incremental approach is aimed at optimizing both performance and accuracy as we proceed with our industrial flow as well as boosting the overall understanding for the interactions between the standard physics theory behind these problems and its application and analysis on EDA.

Moreover, the STA engine's SPICE deck generation capability is fully utilized, enabling the creation of a comprehensive SPICE deck that fully encloses the entire path. This deck serves a dual purpose, as it is utilized both for generating the Single-Event Transient (SET) voltage pulse during the SET generation phase and for obtaining the golden simulation measurements at path output.

It is important to highlight that all timing path points are preserved for later use in the SET propagation phase. This careful preservation ensures that the timing path of interest remains consistent throughout both the SET generation and propagation steps, thereby maintaining accuracy and reliability in the analysis process.

Regarding the generation of Single-Event Transient (SET) pulses, it is imperative to utilize a transistor-level simulation engine, such as SPICE, specifically for the impacted <driver, interconnect, receiver> stage. This is mandatory as the injected charge leads to a flow of current, which is accurately represented using the  $I_{SET}$  current source depicted in Figure 5.2.1.

It is evident that obtaining an accurate SET pulse necessitates transistor-level simulation due to the dependency on both the interconnect and receiver components, as well as the Pull-Up and Pull-Down networks of the driver. To achieve this accuracy, the affected stage is extracted from the SPICE deck and isolated in a reduced SPICE deck (RDECK). This approach avoids the need to simulate the entire path during the SET generation and propagation processes, streamlining the analysis and accurately measuring execution time.

Both the complete SPICE deck and the Reduced SPICE deck (RDECK) are augmented with a behavioral current source serving as the Single-Event Transient (SET) stimulus, implemented using a SPICE Bsource model.

To enhance the SPICE deck's comprehensiveness, the STA engine contributes non-controlling values for all gate side inputs along the entire path. Additionally, it provides

initial state stimulus for sequential elements within the design, in the form of initially setting their inputs and applying a latching clock edge to them. This particular step is crucial, as without this initial stimulus the sequential elements might be in an undefined initial state thus accuracy is hindered.

In the final steps, measurement commands related to SET rise and fall slew, as well as arrival times, are incorporated into the RDECK. Furthermore, the full SPICE deck includes golden measurements for rise and fall slew and arrival times, which are specifically targeted at the timing path's endpoint. It's worth noting that arrival time measurements are consistently conducted with respect to the launching clock edge as a matter of common truth ground for all paths regardless of clock network and common point accounting of the pipeline.

As a result of all the above steps, two distinct measurement files are generated: one containing the golden results from the complete SPICE deck and another tailored for use within our SET Propagation flow steps.

#### **5.3.1.2 SET Propagation**

Regarding the SET propagation phase, it is important to note that in the proposed scriptware flow, this step is exclusively carried out in accordance with Static Timing Analysis (STA) principles, leveraging a commercial STA engine. Prior to the related publication, this approach was regarded as a pioneering innovation by the publication reviewers and had never been explored in previous publications, to the best of my knowledge.

To begin with, as highlighted in section 5.3.1.1, the stored timing information for the worst-case scenario's critical path is employed to establish the worst-case path for PBA. In addition, the measurement file containing SET rise and fall slews, as well as arrival times, must be accessed. At this point, all the stored information regarding SET pulses becomes accessible for utilization within the STA engine. The propagation of SETs initiates with the rising edge of the pulse, facilitated by the SDC commands `set_annotated_transition` and `set_annotated_delay` as described in [50]. Subsequently, STA is executed, and the resulting timing data at the endpoint of the timing path is recorded.

Similarly, this procedure is carried out for the falling edge. It is important to emphasize

that determining the direction of the pulse when measuring Pulse Width (PW) at the endpoint directly depends on the timing path's unateness. For instance, if a timing path comprises an odd number of inverting arcs from the point affected by the SET to the endpoint, it is considered inverting. Consequently, when the SET pulse's rising edge occurs, it results in a falling edge at the endpoint. Therefore, the number of inverting arcs is counted to determine whether PW should be calculated by subtracting the rise arrival time from the fall arrival time at the endpoint or vice versa.

To finalize, the obtained arrival times and PW at the endpoint are compared to the reference golden results generated by SPICE simulation. The Percentage Error (PE) is computed as follows:

$$PE = \frac{|t_m - t_g|}{t_g} * 100\%$$

where  $t_m$  is the STA measurement value and  $t_g$  is the SPICE golden measurement.

This concludes the industrial scriptware flow and its results will be analyzed in Experimental Results section.

## 5.4 Integrated Glitch Generation and Graph Based Propagation using In-House EDA Tool

After proving the intuition that STA is indeed a suitable tool to address static SET analysis using the above industrial flow, the next step is to assess its use not only on separate timing paths but in the global scheme of things, using GBA for the entire circuit. The following subsections present the proposed methodology for this purpose.

### 5.4.1 SET Generation at a Target Node

In order to simulate a particle strike, creating the waveform of the Single Event Transient (SET) voltage pulse at receivers can be achieved through simulation, as outlined in the previously presented industrial tools flow. However, performing simulations at the transistor-level for the entire driver interconnect receivers stage is a time-consuming process. Therefore, a less elaborate model is required. To facilitate pulse generation within the integrated SET analysis flow, the entire driver, interconnect, receivers stage is

represented using simple components like resistors, capacitors, ideal diodes, and current sources.

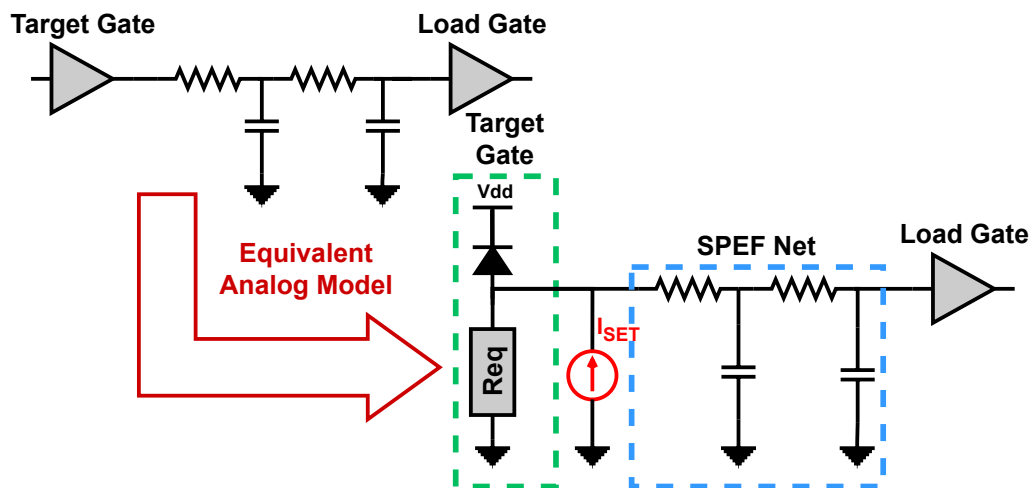
To elaborate further, the driver gate is assumed to be in steady state. For simplicity, only pulses in the positive direction are considered, meaning the driver gate remains at a steady state logical 0 for a positive pulse to occur. The Pull-Down network of the driver is represented by an equivalent resistance  $R_{eq}$ , which can be derived through various heuristic methods. These methods include applying Ohm's law to characterization current vectors in the library, using RC charge or discharge equations, dynamic driver response formulas, among others. For the Pull-Up network of the driver, the PMOS transistors are not conducting, thus the only component having an impact there is the reverse biased diodes between source/drain and bulk. These diodes are forward-biasing when the driver output surpasses  $V_{dd} + 0.7V$ . Note that this 0.7V is the forward bias voltage of the silicon np junction.

It's important to note that, for negative pulses,  $R_{eq}$  would be connected to  $V_{dd}$ , and the diode would be connected to the ground. Subsequently, the SET current source is attached to the output of the driver gate, followed by the interconnect. The information about interconnect parasitics is provided using the IEEE Standard Parasitics Exchange Format (SPEF). Finally, receivers are modeled as capacitors. Figure 5.4.1 illustrates this conversion of the full stage into the less elaborate analog model.

Moreover, to maximize efficiency, a custom SPICE simulator which is natively integrated within the STA engine and is capable of performing fast transient analysis was implemented. It also focuses on storing only essential information for the SET propagation phase, maintaining a minimal memory footprint during simulation. This macro-modeling strategy greatly speeds up the simulation process and consequently enhances capacity, enabling more extensive coverage.

## 5.5 GBA-based SET Propagation

After the generation of the SET pulse is finished and the voltage pulse waveform is separated into rising and falling edges, the process of SET propagation begins. This involves propagating each edge from the point of generation to all endpoints through



**Figure 5.4.1:** Equivalent Analog Model used during the custom local simulation of SET generation

standard cells of the forward logic cone. Similar to the approach typically followed in GBA STA, SET propagation employs a topological sort algorithm. Specifically, this involves traversing the pins of all standard cells within the forward logic cone, ordered by their logical depth or topological level.

In cases where multiple edges arrive at the input pins of standard cells, a merging process similar to that in STA is necessary at the output pin. This process must guarantee the conservative estimation of all possible cases. Conservative here means to pessimistically bound the dynamic behaviour of the circuit as performed in STA. In more detail, STA achieves the aforementioned bounding by selecting the worst-case slew and arrival times at all output pins. This approach might not exactly represent the actual signals that propagate through the logic, but it effectively bounds the worst case using the most pessimistic reasonable information. The term most pessimistic in this context, refers to the largest value when max analysis is performed (setup checks) or smallest value for min analysis (hold checks). In the case of SET propagation, the merging must also consider the absolute worst-case scenario. Here, 'worst case' refers to the scenario with the largest pulse width, as a larger pulse width increases the likelihood of a bit flip being latched by the endpoint sequential elements. Therefore, the merging process should take into account the earliest possible arrival of the pulse's first edge and the latest possible arrival of its second edge.

It is crucial to highlight that the shape of the SET pulse as it moves through a combinational cell is influenced by several factors: the direction of the incoming SET pulse, the timing

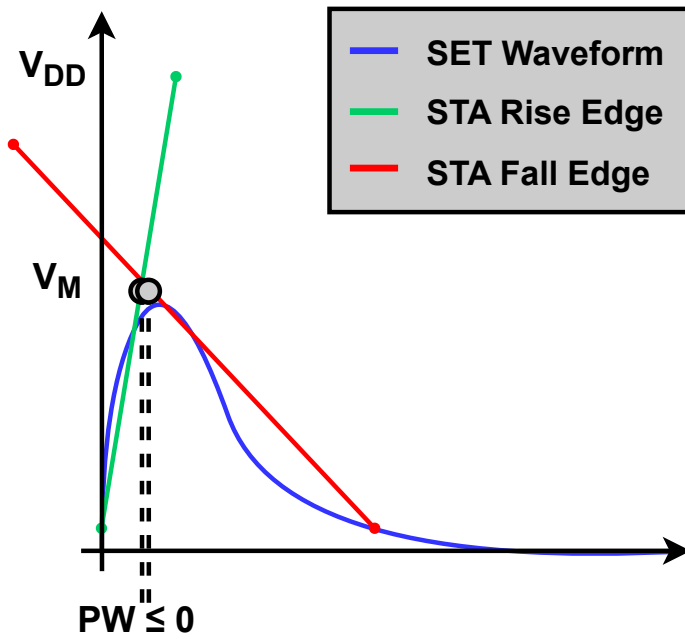


Figure 5.5.1: Electrical Masking

sense of the related timing arcs (positive unate, negative unate or binate), and the voltage levels of the side input pins of the cell. For instance, if the voltage at the side inputs of a standard cell prevents the incoming SET pulse from being transmitted to the output, then the pulse is effectively blocked, which is referred to as logical masking. It is important to note that specifically for binate gates, the direction of the output SET pulse is determined by the voltage levels at the side inputs. If these voltage levels are not predefined, adhering to STA practices, both scenarios of positive and negative unate may lead to the worst-case at the endpoint sequential elements. Consequently, to account for this two separate output SET pulses are created and propagated for such cases.

Furthermore, SET propagation through the forward logic cone can be halted by successive attenuation caused by circuit parasitics. This attenuation, is happening primarily due to wire parasitics which act as high-order low-pass filters. This in turn tends to broaden the pulse width and reduce the peak voltage. As a result, the amplitude of the SET pulse might fall below the threshold voltage  $V_M$  of the standard cell that follows, preventing the pulse from moving to the output pin of the successor. While STA is capable of handling full swing signals with an acceptable degree of accuracy loss when compared to detailed transistor-level simulations like those performed with SPICE, it struggles with non-full swing, partially attenuated signals. Therefore, to determine whether the SET pulse is

electrically masked or not, a heuristic that can be applied is observing the pulse width between the rise and fall edges. The pulse width is obtained by edge arrivals as:

$$PW = \begin{cases} t_d^{fall} - t_d^{rise} & , \text{ positive SET pulse} \\ t_d^{rise} - t_d^{fall} & , \text{ negative SET pulse} \end{cases} \quad (5.5.1)$$

In case the pulse width of the reconstructed SET pulse, after its propagation through a standard cell or interconnect, becomes equal to or less than zero, then it is reasonable to assume that the SET pulse is electrically masked. As shown in Figure 5.5.1, if the pulse width drops below zero, rise and fall edges of the SET pulse become overlapping and either the pulse has become an instant glitch, which is unable to propagate through next gate or its resultant amplitude is lower than  $V_M$ , and thus the SET pulse is masked. Also, in case guarded sequential elements are used to design the circuit, these discard high frequency glitches, thus any pulse with pulse width below this discarding threshold can be considered electrically masked. This threshold can be pre-specified by the user.

Finally, an interesting question arises regarding the plausibility of all scenarios SET propagation. As described above, the pulse edges are propagated or masked based on local criteria such as side input pins voltage levels or attenuation due to filtering from wire parasitics. These metrics may be important at the stage level, however they totally disregard the dynamic behaviour of the circuit as well as the implications from preceding logic. One simple example to elaborate more on this is the following. Assuming a particle strike happens at a standard cell of the 10<sup>th</sup> logic level of a logic cone. For this strike to create a positive pulse, there are two conditions. One is enough energy, while the other is that the output pin must be at the correct state, *i.e.* 0V. Considering that the said output pin must be at 0V, implies that all preceding logic must be at a specific state, and then on top of that, this state of preceding logic creates more implications for all other gates in the forward logic cone. Accounting for all states is of exponential or factorial asymptotic complexity which is totally prohibitive for use with any practical application. Thus a probabilistic model is more suitable to handle the nature of this problem. Also, due to the probabilistic approach, there is no specific way of providing a single quantitative metric to address this problem, like the way slack is used in STA. Instead, a general figure of merit can be derived using a variety of statistical and sensitivity metrics. These issues

---

are addressed in (soon to be) Dr. Christos' Georgakidis PhD thesis who has been my colleague throughout conducting all SET related research in UTH CASlab.

# Chapter 6

## Experimental Results

### 6.1 Delay Calculation

The proposed methodologies can be evaluated using a random stage generation method, which is capable of creating both the required inputs, as well as the appropriate testing data. The required inputs are the driver and receiver cells of the stage, as well as the interconnect between them, the input slew and output load data. These must be within reasonable limits for the library used, which can be predictively achieved as described in [51]. The appropriate testing data are produced by providing SPICE with a deck, which contains the transistor-level stage representation as well as measurement commands for driver and receiver delay and slew. For such purpose, it is convenient to use both Tau Workshop Contest 2020 and 2021 Delay Calculation Toolkit (DCTK) frameworks. The former generates fanout of one stages, with interconnection in form of pi-model, while the latter generates arbitrary fanout stages with arbitrary RC interconnect. Both cases contain a generator, which creates delay calculation stages as described above, for the predictive ASU ASAP 7nm FinFet PDK [51]. The produced results were obtained on a physical machine with an Intel Xeon E5-1620 v4 @3.5GHz CPU and 16GB RAM. Root Mean Square Error (RMSE) is used as a metric of comparison between calculations and SPICE measurements and speedup over SPICE execution time.

Traditional CeFF (1) vs Proposed Algorithm (2)												
Test	# Stages	Configuration 1										
		Driver Delay RMSE (1)	Driver Slew RMSE (1)	Receiver Delay RMSE (1)	Receiver Slew RMSE (1)	Speedup Over SPICE (1)	Driver Delay RMSE (2)	Driver Slew RMSE (2)	Receiver Delay RMSE (2)	Receiver Slew RMSE (2)	Speedup Over SPICE (2)	SPICE Elapsed Time (s)
		PI1K	1000	1.86%	2.16%	21.73%	8.30%	2318.06	1.06%	1.09%	1.08%	1.12%
PI5K	5000	1.82%	2.22%	22.36%	8.49%	2098.11	1.02%	1.06%	1.07%	1.12%	1487.49	188.90
PI10K	10000	1.71%	1.99%	19.92%	7.72%	1859.05	1.11%	1.12%	1.14%	1.13%	1584.26	375.41
PI25K	25000	1.56%	1.86%	18.62%	7.18%	2302.91	1.12%	1.13%	1.15%	1.17%	1543.07	946.82
PI50K	50000	1.88%	1.98%	19.99%	7.58%	2016.96	1.06%	1.07%	1.07%	1.10%	1554.73	1937.41
PI100K	100000	1.72%	1.82%	18.33%	7.17%	2052.76	1.08%	1.09%	1.11%	1.10%	1596.57	3956.36
PI250K	250000	1.72%	1.83%	18.35%	7.15%	2168.33	1.10%	1.12%	1.15%	1.18%	1687.41	10820.48
<b>AVG</b>		<b>1.75%</b>	<b>1.98%</b>	<b>19.90%</b>	<b>7.65%</b>	<b>2116.60</b>	<b>1.08%</b>	<b>1.10%</b>	<b>1.11%</b>	<b>1.13%</b>	<b>1507.81</b>	

**Table 6.1.1:** Traditional Effective Capacitance [1] VS Proposed Methodology for Stages with Balanced Pi-Model Interconnect.

### 6.1.1 Pi-models, TAU2020 Benchmarks

To begin with, a good point of reference to evaluate the exact proposed methodology for pi-models is an effective capacitance methodology which is based in traditional principles while also being modern enough to handle the currently adopted timing models, such as CCS. Such a methodology is proposed in [1]. This was implemented to first evaluate its accuracy, before deriving a more elaborate method. It is obvious that the golden reference to compare against is always SPICE measurements. Both benchmarks and SPICE decks generation and error evaluation is performed by DCTK 2020 framework. It is worth noting that the DCTK 2020 framework contains an error threshold mechanism, to ignore delay and slew errors below a pre-specified threshold value. For fairness and accuracy of the reported results, this thresholding mechanism is disabled.

Another important potential source of inaccuracy and ambiguation of the produced results, is the presence of stages, which cause extreme extrapolation, either on driver or receiver side. To cope with this, the core DCTK 2020 code has been modified, in order to exclude stages, for which extrapolation is observed in any step of the delay calculation methodology. This way, accuracy and performance of both proposed and traditional methodology is performed within characterization limits, without any interference from any employed extrapolation method.

The produced results are also indicative to successfully prove the speculation that the traditional Effective capacitance method is able to provide the required accuracy for use within any sign-off timing engine, only for the case where the pi-model is balanced, *i.e.*

$C_1 = C_2$ . The aforementioned speculation is presented as a proof that the significantly arbitrary shapes of modern interconnections can not be handled by the traditional effective capacitance methodology, as it can not even handle the slightest imbalance in simple pi-model interconnects.

Such imbalance is artificially injected by generation code modification of the DCTK 2020 framework, to create 3 benchmark configurations:

- **Configuration1**:  $C_1 = C_2$
- **Configuration2**:  $C_1 = 0.1C_2$
- **Configuration3**:  $C_1 = 0.01C_2$

For these configurations, 7 benchmarks were created with different number of stages in order to evaluate performance scaling and accuracy conservation.

Table 6.1.1 presents all results for stages with balanced pi-model interconnect and a single receiver (Configuration 1). Both the traditional methodology and the proposed one provide adequate accuracy on the driver side, *i. e.* less than 2% RMSE compared to SPICE, however the proposed methodology provides better accuracy than the traditional one. On the other hand, when it comes to receiver input delay and slew calculation, the traditional methodology, which uses Elmore delay and a geometric, slew rate and slope based approach for slew propagation, fails to provide accurate results. However the proposed methodology again provides around 1% error against SPICE. Furthermore, the traditional methodology is faster. This is expected and makes perfect sense as it computes only 3 voltage regions while the proposed one calculates as many as characterization data; definitely more than 10. Thus, the proposed methodology poses higher computational complexity.

Table 6.1.2 presents results for stages with non balanced pi-model interconnect, where  $C_1$  is 10 times smaller than  $C_2$  (Configuration 2). While the traditional Effective Capacitance is again faster than the proposed methodology, it lacks the required accuracy, as the discontinuity we presented in Figure 3.1.1 highly impacts slew computation. Also, on the receiver side, the traditional methodology is again unable to provide the desired accuracy.

Finally, Table 6.1.3 presents results for stages with even less balanced pi-model interconnect,

Traditional Ceff (1) vs Proposed Algorithm (2)												
Test	# Stages	Configuration 2										
		Driver Delay	Driver Slew	Receiver Delay	Receiver Slew	Speedup Over	Driver Delay	Driver Slew	Receiver Delay	Receiver Slew	Speedup Over	SPICE Elapsed
		RMSE (1)	RMSE (1)	RMSE (1)	RMSE (1)	SPICE (1)	RMSE (2)	RMSE (2)	RMSE (2)	RMSE (2)	SPICE (2)	Time (s)
PI1K	1000	2.16%	13.09%	18.96%	20.48%	1954.34	1.56%	1.91%	1.61%	2.07%	1167.69	38.63
PI5K	5000	2.15%	14.19%	21.98%	22.69%	2192.02	1.62%	1.46%	1.63%	1.58%	1510.80	192.82
PI10K	10000	2.18%	13.87%	21.60%	21.02%	1780.40	1.61%	1.37%	1.69%	1.41%	1533.04	366.22
PI25K	25000	2.15%	13.90%	20.90%	20.87%	1673.25	1.60%	1.46%	1.64%	1.59%	1501.31	941.33
PI50K	50000	2.08%	13.83%	21.84%	21.16%	1830.18	1.64%	1.47%	1.71%	1.53%	1544.36	1851.09
PI100K	100000	2.11%	13.94%	21.85%	20.96%	1825.27	1.62%	1.48%	1.65%	1.59%	1586.83	3913.29
PI250K	250000	2.10%	13.87%	19.91%	21.22%	2005.98	1.62%	1.43%	1.70%	1.54%	1685.77	10446.38
<b>AVG</b>		<b>2.13%</b>	<b>13.81%</b>	<b>21.01%</b>	<b>21.20%</b>	<b>1894.49</b>	<b>1.61%</b>	<b>1.51%</b>	<b>1.66%</b>	<b>1.62%</b>	<b>1504.26</b>	

**Table 6.1.2:** Traditional Effective Capacitance [1] VS Proposed Methodology for Stages with Non Balanced Pi-Model Interconnect,  $C_1 = 0.1C_2$

Traditional Ceff (1) vs Proposed Algorithm (2)												
Test	# Stages	Configuration 3										
		Driver Delay	Driver Slew	Receiver Delay	Receiver Slew	Speedup Over	Driver Delay	Driver Slew	Receiver Delay	Receiver Slew	Speedup Over	SPICE Elapsed
		RMSE (1)	RMSE (1)	RMSE (1)	RMSE (1)	SPICE (1)	RMSE (2)	RMSE (2)	RMSE (2)	RMSE (2)	SPICE (2)	Time (s)
PI1K	1000	2.39%	16.31%	23.03%	18.61%	1954.34	1.72%	1.04%	1.79%	1.13%	1158.66	30.68
PI5K	5000	2.44%	15.83%	24.02%	18.71%	2192.02	1.77%	1.36%	1.77%	1.49%	1514.89	121.18
PI10K	10000	2.48%	16.74%	23.65%	19.30%	1780.40	1.82%	1.18%	1.88%	1.29%	1404.62	230.47
PI25K	25000	2.53%	16.67%	25.86%	21.48%	1673.25	1.84%	1.52%	1.91%	1.56%	1480.44	585.05
PI50K	50000	2.50%	16.39%	23.19%	18.23%	1830.18	1.77%	1.38%	1.85%	1.45%	1496.93	1184.34
PI100K	100000	2.48%	16.55%	24.79%	19.03%	1825.27	1.78%	1.36%	1.83%	1.47%	1531.39	2473.27
PI250K	250000	2.50%	16.59%	26.11%	19.41%	2005.98	1.77%	1.38%	1.80%	1.42%	1643.49	6590.70
<b>AVG</b>		<b>2.47%</b>	<b>16.44%</b>	<b>24.38%</b>	<b>19.25%</b>	<b>1894.49</b>	<b>1.78%</b>	<b>1.32%</b>	<b>1.83%</b>	<b>1.40%</b>	<b>1461.49</b>	

**Table 6.1.3:** Traditional Effective Capacitance [1] VS Proposed Methodology for Stages with Non Balanced Pi-Model Interconnect,  $C_1 = 0.01C_2$

where  $C_1$  is 100 times smaller than  $C_2$  (Configuration 3). Again, the traditional Effective Capacitance fails to accurately capture the driver and receiver waveforms delay and slew, while being faster. Thus, for all cases, the proposed methodology always provides less than 2% RMSE compared to SPICE, while still being adequately fast. The traditional Effective Capacitance methodology fails to provide adequate accuracy when the pi-model is not balanced.

### 6.1.2 Arbitrary RC Interconnects, Approximate Method

When dealing with stages with complex arbitrary RC interconnects, we created 7 benchmarks to see how well the proposed approach scales in performance. Comparison with the traditional method is not performed because the traditional method can not provide accurate results, even for the basic case of pi-models. So, it is reasonable to

Proposed Algorithm vs SPICE								
Test	# Stages	Driver Delay RMSE	Driver Slew RMSE	Receiver Delay RMSE	Receiver Slew RMSE	Elapsed Time (s)	SPICE Elapsed Time (s)	Speedup Over SPICE
DN1K	1000	0.89%	1.10%	0.18%	0.98%	0.71	566.61	797.06
DN5K	5000	0.81%	1.14%	1.01%	1.42%	3.31	2724.1	822.00
DN10K	10000	0.98%	1.15%	0.22%	1.02%	6.72	5494.45	817.06
DN25K	25000	0.88%	1.16%	1.01%	1.52%	16.29	12872.40	790.32
DN50K	50000	0.82%	1.12%	1.06%	1.52%	32.67	23527.21	720.15
DN100K	100000	0.90%	1.13%	1.07%	1.49%	65.50	50078.88	764.57
DN250K	250000	0.85%	1.13%	1.04%	1.47%	162.42	167726.13	1032.70
<b>AVG</b>		<b>0.87%</b>	<b>1.13%</b>	<b>0.80%</b>	<b>1.35%</b>			<b>820.55</b>

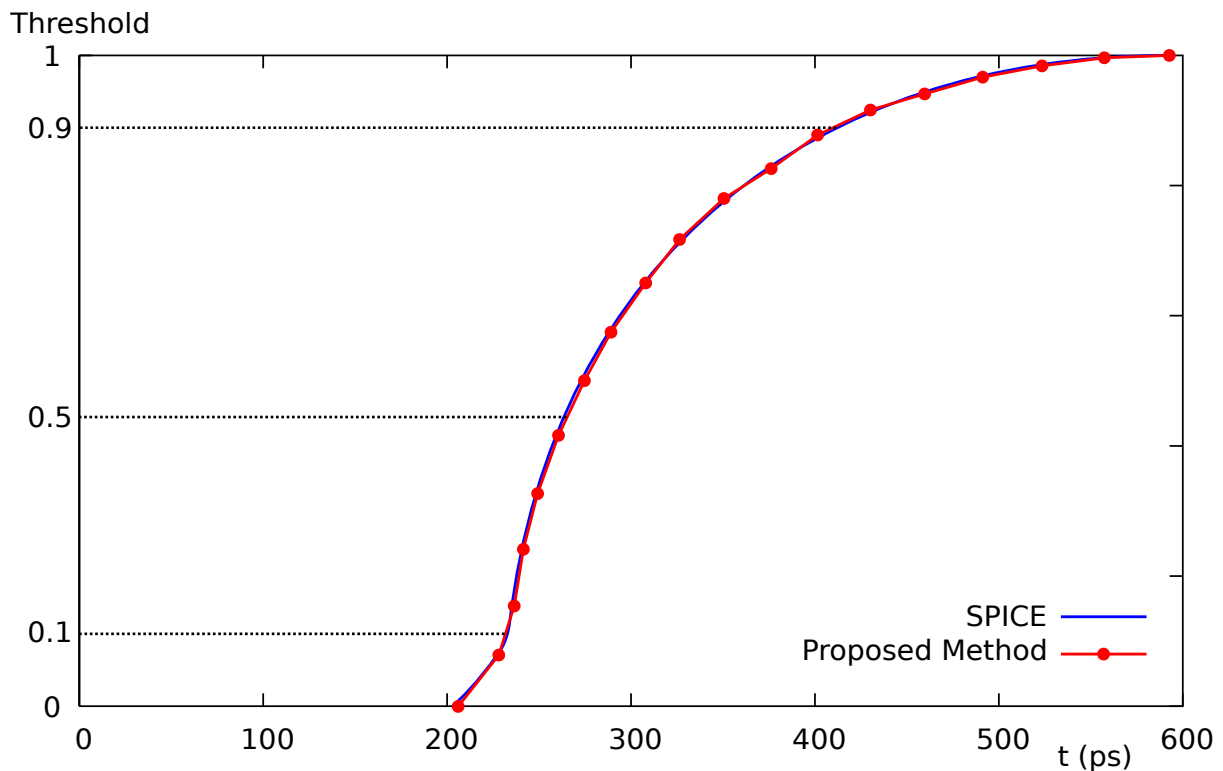
**Table 6.1.4:** Proposed Methodology VS SPICE for Stages with Arbitrary RC Interconnect

assume that it can not provide accurate results for more complex interconnect shapes.

DCTK 2021 was used to carefully generate these benchmarks and evaluate the results. The Table 6.1.4 presents these results. Noteworthy is the observation that our methodology yields an approximate 1% Root Mean Square error (RMSE) for both driver delay and slew, with receiver delay RMSE also within the 1.5% margin. Average speedup over SPICE is less than the speedup observed with the proposed pi-models methodology, which is expected as the computational complexity of the generalized method is directly dependent on the number of interconnect nodes.

Even though the approximate method is a bit slower than the direct one, it still achieves adequate speedup over SPICE simulation. This renders it suitable for use with Static Timing Analysis engines, where a good balance between accuracy and performance is needed. Of course it is worth noting again, that accuracy is always prioritized over performance, as not meeting timing requirements is destructive for ASICs.

Finally, for clarity purposes, Figure 6.1.1 depicts a comparison between the voltage signal waveform created by the proposed approximate methodology and SPICE electrical simulation, for a stage with 38 nodes and 12 receivers. It is evident that these waveforms are almost indistinguishable, which is another way of proving the suitability of the proposed method for STA.



**Figure 6.1.1:** Proposed Methodology Driver Waveform VS SPICE for Stage with Arbitrary RC Interconnect.

## 6.2 SET Generation And Propagation

This section presents the experimental methodology and results of the proposed SET analysis flows.

### 6.2.1 Industrial Tools PBA-based Flow

The proposed industrial tool based scriptware methodology was assessed through an analysis of the top 20 critical paths in a PID controller, synthesized in  $0.13 \mu\text{m}$  technology. The design netlist and parasitics were created with the aid of standard commercial Electronic Design Automation (EDA) tools, through all steps of Synthesis, Place & Route, Clock Tree Synthesis, In-Place Optimisation, and Extraction. For each critical path, 20 configurations of SET (Single-Event Transient) double exponential current sources were utilized, resulting in a total of 400 scenarios. The SET current source parameters were produced through Monte-Carlo simulations that applied distributions derived from experimental data on particle strikes to vary the SET charge. For illustrative purposes, the outcomes of the Monte-Carlo simulation for 40,000 samples are presented in Figure

6.2.1. Additionally, the selection of parameters Tau1 and Tau2 was based on existing literature.

The proposed scriptware methodology yielded 1.6% average relative error for Pulse Width at the path endpoint when compared to SPICE simulations, while also being orders of magnitude faster. This level of error is considered acceptable as it falls within the typical error range between Static Timing Analysis and SPICE, as supported by existing studies [1, 52]. Out of 400 Single-Event Transient (SET)-induced glitches generated, 371 propagated to the timing path endpoints, whereas the remaining 29 were either fully filtered or attenuated below the endpoint pin's voltage threshold. The PIPB effect was quantified and evaluated by measuring the percentage increase in Pulse Width (PW) from the SET-impacted net to the timing path endpoint. On average, the experiments resulted to 63.5% PW increase. Furthermore, the largest observed PW increase was 143.4%.

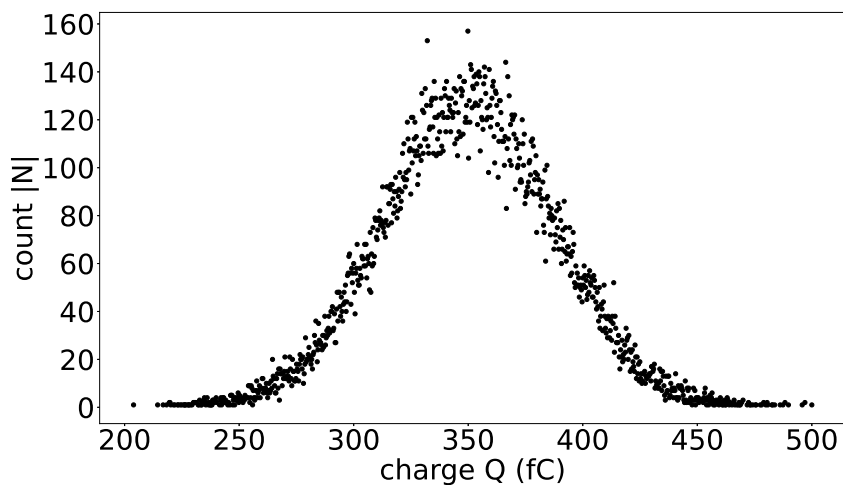


Figure 6.2.1: SET Monte-Carlo Charge Distribution

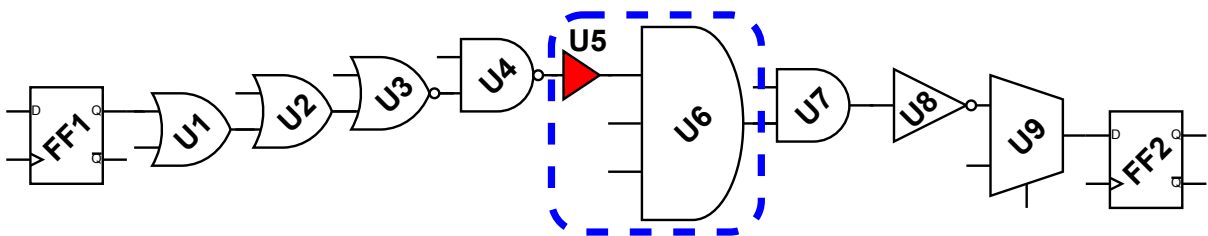
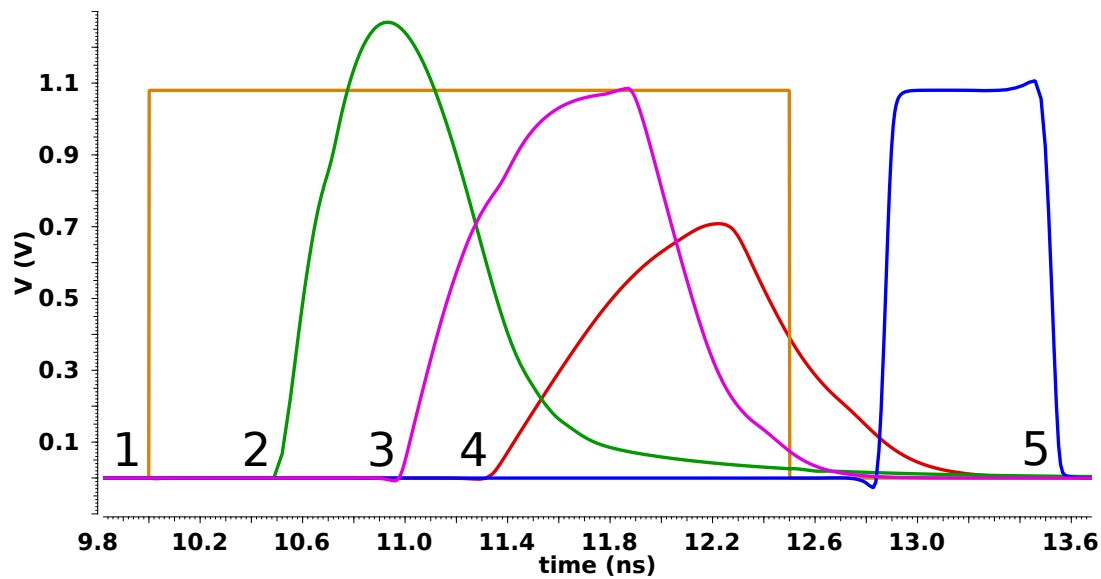


Figure 6.2.2: 90<sup>th</sup> Worst Delay Timing Path

An interesting point here is that the scriptware methodology exhibits a considerable error, approximately 70%, when processing non-full swing pulses. This issue was observed in the 90th slowest path of the design, as shown in Figure 6.2.2. The underlying reason for

this discrepancy is that backend Electronic Design Automation (EDA) tools prioritize optimization of critical paths for timing and signal integrity, while the remainder of the design is optimized for area and power efficiency. The path under consideration is not critical, leading to generally slow signal slews. In particular, component U5, marked in red, is a low drive buffer, which is heavily loaded by component U6, a high drive AND3 standard cell. This in turn imposes a significant load on U5. The outcome of this configuration is depicted in Figure 6.2.3. The green waveform (2) represents the waveform of the Single-Event Transient (SET)-induced pulse, occurring at the input of U1.



**Figure 6.2.3:** 90<sup>th</sup> Worst Delay Timing Path SPICE Waveform Plots

This pulse is transmitted through the path to the input of the highlighted buffer U5, magenta waveform (3), Figure 6.2.3. The output signal of U5 is shown in red (4). This scenario serves as a classic illustration of how a high-frequency glitch fails to reach full rail voltage when passing through a substantially loaded low drive gate .

The blue waveform (5) illustrates the pulse as it arrives at the input of FF2, which is the timing path endpoint. Additionally, for context, the clock waveform for FF1 and FF2 is shown in yellow (1). The inaccuracy arises from the red waveform not reaching full swing voltage. Typically, standard Static Timing Analysis (STA) engines assume the propagation of clean, full-swing signals. Therefore, when non-full swing signals are propagated, the industrial STA tool tends to calculate more conservative arrival times, leading to a more pessimistic Pulse Width (PW) estimate compared to SPICE simulations. Consequently, these inaccuracies are challenging to address with standard STA methodologies, as they

may occur not only with SET-induced glitches but also with any type of high-frequency glitch. Finally, it is worth mentioning that the most recent Standard Library CCS Noise (CCSN) timing model may be able to handle such issues, however typically older technology nodes are used for RADHARD chips and CCSN libraries are not always available for these.

### 6.2.2 Integrated GBA-based Flow

This section details the integrated proposed SET Analysis methodology experimental results. The experiments were conducted on a post Place & Route design, covering all possible combinations of particle profiles and generation target standard cells. For these experiments, four particle profiles and fifteen benchmarks (including ISCAS'85 and OpenCores projects) were investigated. The characteristics of these profiles and benchmarks are documented in Tables 6.2.1 and 6.2.2, respectively. For generating the designs used for these experiments, IHP's 130nm technology node was again utilized and the experiments were performed on a CentOS7 machine with a 12-core Intel<sup>®</sup> Xeon CPU E5-2620 @ 2.10GHz and 126GB memory.

**Table 6.2.1:** Examined Particle Profiles

Profile Name	$q$ (fC)	$t_r$ (ps)	$t_f$ (ps)
<b>p1</b>	34	10	100
<b>p2</b>	66	10	100
<b>p3</b>	99	10	100
<b>p4</b>	132	10	100

At this point, it is important to define the overall metric of merit for this flow. This metric is circuit sensitivity and it can be defined using the following rationale. After the SET analysis is complete, the number of pulses reaching each endpoint per particle strike scenario is known. Also the total number of generated pulses is known, thus the probability of any pulse reaching a single endpoint is known. To tally up, the global number of Accumulated SETs (ASET) can be derived using the following equation:

$$ASET = \sum_e^{endpoints} \sum_s^{SET_{scenarios}} P_{SET}(e, s), \quad (6.2.1)$$

where  $s$  is the particle strike scenario,  $e$  is the examined endpoint, while  $P_{SET}(e, s)$

**Table 6.2.2:** Benchmarks Characteristics

Design	#Cells	#PIs	#Endpoints	Max Logic Depth	AVG Reachable Endpoints
<b>c432</b>	123	37	7	49	4.6
<b>c499</b>	177	42	32	29	17.7
<b>c880</b>	214	61	26	45	3.0
<b>c1355</b>	175	42	32	29	17.0
<b>c1908</b>	218	34	25	47	12.6
<b>c2670</b>	326	234	140	35	4.5
<b>c3540</b>	739	51	22	59	8.0
<b>c5315</b>	751	179	123	47	7.9
<b>c6288</b>	1711	33	32	165	15.4
<b>c7552</b>	921	208	108	77	11.0
<b>lpffir</b>	487	17	96	43	7.6
<b>pid</b>	3734	53	463	43	10.4
<b>aes</b>	7496	260	799	71	15.6
<b>aes192</b>	9204	324	927	79	18.9
<b>aes_ip</b>	9349	41	948	65	23.1

represents the probability of any SET pulse of the current scenario reaching endpoint  $e$ .

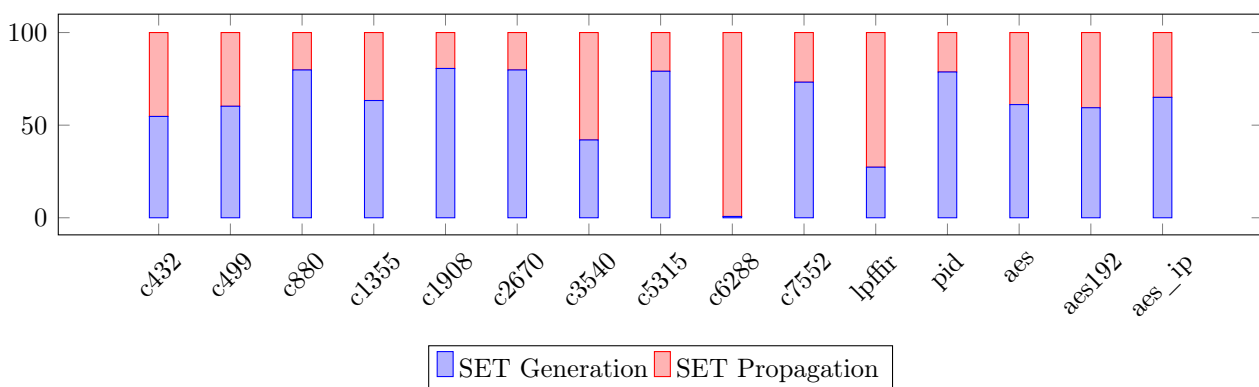
**Figure 6.2.4:** SET Analysis Runtime Distribution

Table 6.2.3 presents the Total ASET and Runtime for the designs under investigation, including averages for each endpoint and SET scenario. Figure 6.2.4 depicts the SET Analysis runtime distribution between the SET Generation and Propagation phases. On average, 62.71% of the time spent for SET analysis is dedicated to the SET Generation phase. However, design **c6288** presents an exceptional case, with 99.3% of its total runtime being spent for SET propagation. This outlier behaviour can be attributed to the Average ASET per endpoint, indicating a high number of SETs impacting each endpoint, and the

Maximum Logic Depth, which significantly exceeds the typical logic path depth found in other designs.

**Table 6.2.3:** Exhaustive SET Analysis Results

<b>Design</b>	<b>Total ASET</b>	<b>AVG ASET</b>	<b>Total Runtime (s)</b>	<b>AVG Runtime (s)</b>
<b>c432</b>	103.56	14.79	3.778	0.008
<b>c499</b>	88.31	2.76	4.689	0.007
<b>c880</b>	109.98	4.23	3.742	0.004
<b>c1355</b>	82.88	2.59	4.045	0.006
<b>c1908</b>	106.21	4.25	4.372	0.005
<b>c2670</b>	210.69	1.51	7.142	0.005
<b>c3540</b>	264.35	12.02	53.951	0.018
<b>c5315</b>	575.41	4.68	26.302	0.009
<b>c6288</b>	1034.46	32.33	6086.102	0.889
<b>c7552</b>	398.44	3.69	41.262	0.011
<b>lpffir</b>	298.88	3.11	19.818	0.015
<b>pid</b>	1764.35	3.81	303.634	0.025
<b>aes</b>	4841.14	6.06	957.549	0.035
<b>aes192</b>	6671.71	7.20	1314.464	0.039
<b>aes_ip</b>	9460.09	9.98	1151.782	0.034

# Chapter 7

## Conclusion

This dissertation has explored two general topics of great significance and scientific interest, one being fast and SPICE-accurate delay calculation for STA, and the other being the SET circuit sensitivity analysis. For the former, the proposed stage delay calculation methodology which employs the CCS Timing standard library model and a novel full waveform propagation technique is able to provide SPICE accurate results in reasonable execution time. The proposed technique, which utilizes the old yet proven and powerful Asymptotic Waveform Evaluation, is able to propagate any waveform shape through the interconnect, by effectively tackling the old method's weaknesses while extending it, to make it suitable for nowadays applications. The methodology is currently integrated within UTH CASlab's in-house EDA tool. For the latter, two methodologies were proposed, one being based on an industrial SPICE engine and Sign-Off STA engine, which employs PBA-based STA to propagate glitches through timing paths and analyze the PIPB effect. The other methodology is again integrated within UTH CASlab's in-house EDA tool and it uses a less elaborate yet effective simulation model for SET generation, and a custom STA-based propagation flow which attempts to support the masking mechanisms. The following subsection briefly presents the resultant publications.

## 7.1 Publications

### 7.1.1 Delay Calculation

1. *Full Stage Delay Calculation Using Full Waveform Propagation and Standard Library CCS Model* [52]

- This paper presents the proposed delay calculation methodology.

2. *Gate delay estimation with library compatible current source models and effective capacitance* [1]

- This publication documents the traditional effective capacitance methodology which was used throughout this dissertation as a benchmark. It is a joint publication with our colleagues from UTH Electronics Research Lab and it presents the approach we used to win the TAU 2020 Contest.

### 7.1.2 Single Event Transients Analysis

1. *Towards a Comprehensive SET Analysis Flow for VLSI Circuits using Static Timing Analysis* [53]

- This paper contains the proposed integrated SET analysis methodology. It presents the results of our joint research effort with The Leibniz Institute for High Performance Microelectronics (IHP) towards a complete radiation hardened ASIC design flow.

2. *Single Event Transients Generation and Propagation Flow using Commercial EDA Tools* [54]

- This publication presents the proposed industrial tools based SET analysis methodology, which was implemented to initially prove the suitability of STA for the purpose of SET analysis. It is a joint publication between UTH CASlab and our colleagues at The Leibniz Institute for High Performance Microelectronics (IHP).

### 7.1.3 Miscellaneous

1. *RADPlace-MS: A Timing-Driven Placer and Optimiser for ASICs Radiation Hardening* [55]

- This paper presents RADPlace, an academic timing-driven detailed placement algorithm which can guarantee spacing constraints for Triple Modular Redundancy Flip-Flops, in order to mitigate simultaneous faults due to particle strikes area effect.

2. *Investigation on Performance, Power, Area Trade-Offs using Deterministic and Monte-Carlo Process Variation Aware Synthesis Flows* [56]

- This investigation paper is exploring the possibility of reaping the benefits of process variation impact awareness at the synthesis step of the ASIC flow, *i.e.* early on before passing the synthesized netlist to the ASIC backend tools.

3. *Static Timing Analysis Induced Simulation Errors for Asynchronous Circuits* [57]

- This publication investigates the digital simulation errors that occur in simulation of cyclic or asynchronous circuits, when the Standard Delay Format (SDF) file used was generated either by an industrial STA tool that breaks loops, or by the in-house Asynchronous STA engine.

## 7.2 Awards

1. *Outstanding Student Paper Award* [53]

- Best Research Paper Award titled "*Towards a Comprehensive SET Analysis Flow for VLSI Circuits using Static Timing Analysis*", which proposes the first comprehensive STA-based method for Single Event Transients generation and propagation, to enable SET-aware RADHARD ASIC backend design and optimization. - Issued by 36th IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)

## 2. *Best Paper Award* [54]

- Best Paper Award titled "*Single Event Transients Generation and Propagation Flow using Commercial EDA Tools*", on Special Session Devices and Systems in Radiation Environment - 2021 IEEE 32nd International Conference on Microelectronics (MIEL)

## 7.3 Future Work

All presented methodologies can be extended to support a variety of new and interesting features. For example, in order to address the path full waveform propagation issue, the proposed methodology can be modified to support the CCS Noise model, which enables full waveform propagation not only for a single stage, but through standard cells as well. Another interesting area of research could be the domain of extending the delay calculation methodology to support Statistical STA data by employing any On-Chip Variation model. Finally, several issues on the SET analysis domain are already under investigation by the CASlab team. Some of these issues consist of probabilistic SET analysis, SET Generation models, RADHARD P&R and optimization.

# Bibliography

- [1] D. Garyfallou, S. Simoglou, N. Sketopoulos, C. Antoniadis, C. P. Sotiriou, N. Evmorfopoulos, and G. Stamoulis, “Gate delay estimation with library compatible current source models and effective capacitance,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 5, pp. 962–972, 2021.
- [2] J. Bhasker and R. Chadha, *Static Timing Analysis for Nanometer Designs: A Practical Approach*. Springer Publishing Company, Incorporated, 1st ed., 2009.
- [3] J. M. Rabaey, A. P. Chandrakasan, and B. Nikolić, *Digital integrated circuits: a design perspective*. Pearson Education, Incorporated., 2003.
- [4] R. Ecoffet, “Overview of In-Orbit Radiation Induced Spacecraft Anomalies,” *IEEE Transactions on Nuclear Science*, vol. 60, no. 3, 2013.
- [5] N. Miskov-Zivanov and D. Marculescu, “MARS-C: modeling and reduction of soft errors in combinational circuits,” in *Proceedings of annual Design Automation Conference (DAC)*, 2006.
- [6] S. Krishnaswamy, G. Viamontes, I. Markov, and J. Hayes, “Accurate reliability evaluation and enhancement via probabilistic transfer matrices,” in *Design, Automation & Test in Europe (DATE)*, 2005.
- [7] S. Z. Shazli and M. B. Tahoori, “Using Boolean satisfiability for computing soft error rates in early design stages,” *Microelectronics Reliability*, vol. 50, no. 1, 2010.
- [8] G. Bany Hamad, O. Ait Mohamed, S. Rafay Hasan, and Y. Savaria, “Identification of soft error glitch-propagation paths: Leveraging SAT solvers,” in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2012.
- [9] C. Liu, L. Zhang, X. He, and Y. Guo, “Analysis of SET Reconvergence and Hardening in the Combinational Circuit Using a SAT-Based Method,” *IEEE Access*, vol. 6, 2018.
- [10] G. Asadi and M. Tahoori, “An analytical approach for soft error rate estimation in digital circuits,” in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2005.
- [11] H. Asadi, M. B. Tahoori, M. Fazeli, and S. G. Miremadi, “Efficient algorithms to accurately compute derating factors of digital circuits,” *Microelectronics Reliability*, vol. 52, no. 6, 2012.
- [12] L. Chen, M. Ebrahimi, and M. B. Tahoori, “CEP: Correlated Error Propagation for Hierarchical Soft Error Analysis,” *Journal of Electronic Testing*, vol. 29, 2013.
- [13] F. Kriebel, S. Rehman, D. Sun, *et al.*, “ACSEM: Accuracy-configurable fast soft error

- masking analysis in combinatorial circuits,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2015.
- [14] M. Zhang and N. Shanbhag, “Soft-Error-Rate-Analysis (SERA) Methodology,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, 2006.
- [15] R. Rajaraman, J. Kim, N. Vijaykrishnan, *et al.*, “SEAT-LA: a soft error analysis tool for combinational logic,” in *International Conference on VLSI Design (VLSID)*, 2006.
- [16] J. Li and J. Draper, “Accelerated Soft-Error-Rate (SER) Estimation for Combinational and Sequential Circuits,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 22, no. 3, 2017.
- [17] M. Anglada, R. Canal, J. L. Aragón, and A. González, “Fast and Accurate SER Estimation for Large Combinational Blocks in Early Stages of the Design,” *IEEE Transactions on Sustainable Computing*, vol. 6, no. 3, 2021.
- [18] Synopsys Inc., “CCS timing. Technical White Paper, Version 2.0,” 2006.
- [19] Synopsys Inc., *PrimeTime GCA Tool Commands Manual*, Version L-2016.06-SP2 ed., September 2016.
- [20] Synopsys Inc., *Liberty User Guides and Reference Manual Suite*, Version 2017.06 ed., 2017.
- [21] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd ed., 2009.
- [22] “IEEE Standard for Integrated Circuit (IC) Open Library Architecture (OLA),” *IEEE Std 1481-2019 (Revision of IEEE Std 1481-2009)*, pp. 1–641, 2020.
- [23] Synopsys Inc., *StarRC User Guide and Command Reference*, Version R-2020.09-SP4 ed., 2021.
- [24] W. E. Boyce, R. C. DiPrima, and D. B. Meade, *Elementary differential equations and boundary value problems*. John Wiley & Sons, 2021.
- [25] C. K. Alexander, *Fundamentals of Electric Circuits*. McGraw-Hill, 2013.
- [26] A. V. Oppenheim, A. S. Willsky, S. H. Nawab, and J.-J. Ding, *Signals and Systems*. Prentice hall Upper Saddle River, NJ, 1997.
- [27] E. Chiprout, M. S. Nakhla, E. Chiprout, and M. S. Nakhla, *Asymptotic Waveform Evaluation*. Springer, 1994.
- [28] J. S. Kauppila, *Layout-Aware Modeling and Analysis Methodologies for Transient Radiation Effects on Integrated Circuit Electronics*. Vanderbilt University, 2015.
- [29] M. Andjelkovic, *A Methodology for Characterization, Modeling and Mitigation of Single Event Transient Effects in CMOS Standard Combinational Cells*. PhD thesis, Universität Potsdam, 2021.
- [30] Y.-C. Wu and Y.-R. Jhan, *3D TCAD Simulation for CMOS Nanoelectronic Devices*. Springer, 2018.

- [31] Synopsys Inc., *Sentaurus™ Device User Guide*, Version K-2015.06 ed., 2015.
- [32] M. Andjelkovic, A. Ilic, Z. Stamenkovic, M. Krstic, and R. Kraemer, “An overview of the modeling and simulation of the single event transients at the circuit level,” in *2017 IEEE 30th International Conference on Microelectronics (MIEL)*, pp. 35–44, IEEE, 2017.
- [33] R. Puri, D. S. Kung, and A. D. Drumm, “Fast and accurate wire delay estimation for physical synthesis of large asics,” in *Proceedings of the 12th ACM Great Lakes symposium on VLSI*, pp. 30–36, 2002.
- [34] P. Feldmann, S. Abbaspour, D. Sinha, G. Schaeffer, R. Banerji, and H. Gupta, “Driver waveform computation for timing analysis with multiple voltage threshold driver models,” in *Proceedings of the 45th annual Design Automation Conference*, pp. 425–428, 2008.
- [35] R. P. Brent, “An algorithm with guaranteed convergence for finding a zero of a function,” *The computer journal*, vol. 14, no. 4, pp. 422–425, 1971.
- [36] W. C. Elmore, “The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers,” *Journal of Applied Physics*, vol. 19, no. I, pp. 55–63, 1948.
- [37] C. L. Ratzlaff and L. T. Pillage, “Rice: Rapid interconnect circuit evaluation using awe,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 6, pp. 763–776, 1994.
- [38] L. T. Pillage and R. A. Rohrer, “Asymptotic waveform evaluation for timing analysis,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 4, pp. 352–366, 1990.
- [39] D. F. Anastasakis, N. Gopal, S.-Y. Kim, and L. T. Pillage, “Enhancing the stability of asymptotic waveform evaluation for digital interconnect circuit applications,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 6, pp. 729–736, 1994.
- [40] H. Jiang, H. Zhang, J. Kauppila, L. Massengill, and B. Bhuvu, “An empirical model for predicting SE cross section for combinational logic circuits in advanced technologies,” *IEEE Transactions on Nuclear Science*, vol. 65, no. 1, pp. 304–310, 2017.
- [41] R. Rajaraman, J. Kim, N. Vijaykrishnan, Y. Xie, and M. J. Irwin, “SEAT-LA: A soft error analysis tool for combinational logic,” in *19th International Conference on VLSI Design held jointly with 5th International Conference on Embedded Systems Design (VLSID’06)*, pp. 4–pp, IEEE, 2006.
- [42] S. Kiamehr, M. Ebrahimi, F. Firouzi, and M. B. Tahoori, “Chip-level modeling and analysis of electrical masking of soft errors,” in *2013 IEEE 31st VLSI Test Symposium (VTS)*, pp. 1–6, IEEE, 2013.
- [43] G. B. Hamad, S. R. Hasan, O. A. Mohamed, and Y. Savaria, “Modeling, analyzing, and abstracting single event transient propagation at gate level,” in *2014 IEEE 57th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 515–518, IEEE, 2014.

- [44] J. Li and J. Draper, “Accelerating Soft-Error-Rate (SER) Estimation in the Presence of Single Event Transients,” in *2016 53rd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2016.
- [45] B. Narasimham, B. L. Bhuva, R. D. Schrimpf, L. W. Massengill, M. J. Gadlage, O. A. Amusan, W. T. Holman, A. F. Witulski, W. H. Robinson, J. D. Black, *et al.*, “Characterization of digital single event transient pulse-widths in 130-nm and 90-nm cmos technologies,” *IEEE Transactions on Nuclear Science*, vol. 54, no. 6, pp. 2506–2511, 2007.
- [46] M. Glorieux, A. Evans, V. Ferlet-Cavrois, C. Boatella-Polo, D. Alexandrescu, S. Clerc, G. Gasiot, and P. Roche, “Detailed set measurement and characterization of a 65 nm bulk technology,” *IEEE Transactions on Nuclear Science*, vol. 64, no. 1, pp. 81–88, 2016.
- [47] V. Ferlet-Cavrois, P. Paillet, D. McMorrow, N. Fel, J. Baggio, S. Girard, O. Duhamel, J. Melinger, M. Gaillardin, J. Schwank, *et al.*, “New insights into single event transient propagation in chains of inverters—Evidence for propagation-induced pulse broadening,” *IEEE Transactions on Nuclear Science*, vol. 54, no. 6, pp. 2338–2346, 2007.
- [48] A. S. Householder, *The numerical treatment of a single nonlinear equation*. McGraw-Hill, 1970.
- [49] G. Messenger, “Collection of charge on junction nodes from ion tracks,” *IEEE Transactions on nuclear science*, vol. 29, no. 6, pp. 2024–2031, 1982.
- [50] S. Gangadharan and S. Churiwala, *Constraining Designs for Synthesis and Timing Analysis*. Springer, 2013.
- [51] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, “Asap7: A 7-nm finfet predictive process design kit,” *Microelectronics Journal*, vol. 53, pp. 105–115, 2016.
- [52] S. Simoglou, I. Lilitsis, N. Blias, and C. Sotiriou, “Full Stage Delay Calculation Using Full Waveform Propagation and Standard Library CCS Model,” in *2024 International Symposium on Quality Electronic Design (ISQED)*, IEEE, 2024.
- [53] C. Georgakidis, D. Valiantzas, S. Simoglou, I. Lilitsis, N. Chatzivangelis, I. Golfos, M. Andjelkovic, C. Sotiriou, and M. Krstic, “Towards a Comprehensive SET Analysis Flow for VLSI Circuits using Static Timing Analysis,” in *2023 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 1–6, IEEE, 2023.
- [54] S. Simoglou, C. Georgakidis, I. Lilitsis, C. Sotiriou, M. Andjelkovic, and M. Krstic, “Single Event Transients Generation and Propagation Flow using Commercial EDA Tools,” in *2021 IEEE 32nd International Conference on Microelectronics (MIEL)*, IEEE, 2021.
- [55] C. Georgakidis, S. Simoglou, and C. Sotiriou, “RADPlace-MS: A Timing-Driven Placer and Optimiser for ASICs Radiation Hardening,” in *2022 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 1–6, IEEE, 2022.

- 
- [56] N. Blias, I. Lilitsis, S. Simoglou, E. Bakas, and C. Sotiriou, "Investigation on Performance, Power, Area Trade-Offs using Deterministic and Monte-Carlo Process Variation Aware Synthesis Flows," in *2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 1–6, IEEE, 2022.
- [57] S. Simoglou, C. Sotiriou, and N. Blias, "Static Timing Analysis Induced Simulation Errors for Asynchronous Circuits," in *2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 1–4, IEEE, 2021.