UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Timing analysis of VLSI circuits in advanced technology nodes

Diploma Thesis

Vagenas Anastasios-Ioulios

Supervisor: George Stamoulis

Ιούλιος 2022

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Timing analysis of VLSI circuits in advanced technology nodes

Diploma Thesis

Vagenas Anastasios-Ioulios

Supervisor: George Stamoulis

Ιούλιος 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάλυση χρονισμού κυκλωμάτων πολύ μεγάλης κλίμακας ολοκλήρωσης σε προηγμένες τεχνολογίες

Διπλωματική Εργασία

Βαγενάς Αναστάσιος-Ιούλιος

Επιβλέπων: Γεώργιος Σταμούλης

Ιούλιος 2022

Approved by the Examination Committee:

Supervisor **George Stamoulis**
Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Nestor Evmorfopoulos**
Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Christos Sotiriou**
Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

To my friends and family.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, 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

Vagenas Anastasios-Ioulios

Diploma Thesis

Timing analysis of VLSI circuits in advanced technology nodes

Vagenas Anastasios-Ioulios

Abstract

The rapid scaling of process technology and the widespread use of integrated circuits in our daily lives render the task of gate-level timing analysis highly challenging. The resistivity of on-chip interconnects, signal non-linearity, crosstalk, and driver-receiver interdependencies introduce challenges in accurate timing estimation. Moreover, timing analysis engines have to be highly efficient in order to scale for the ever-increasing design sizes and to be integrated in timing-driven optimization flows for placement/routing. In this thesis, we introduce a novel iterative methodology for gate-level stage timing estimation using current source models and the concept of multiple slew and effective capacitance values. The iterative approach utilizes piecewise linear waveforms to approximate the driver waveform, computationally fast closed-form formulas, and efficient topological traversals of the RC interconnect of the stage to propagate the slew and effective capacitance values. Unlike previous works, our methodology considers the resistive shielding effect, the Miller effect, and takes into account the driver-receiver interdependencies existing in deep nanometer technology stages. Experimental evaluation on 2.2 million <driver, interconnect, receiver(s)> gate-level stages implemented using the ASU ASAP 7 nm FinFET Predictive PDK show that our approach can achieve up to 5.40% mean relative error against SPICE. Furthermore, it consumes only 0.38 MB of memory while also providing timing estimates in 8.16 s for 2.2 million stages. Thus, it may be easily integrated into iterative timing-driver optimization flows to provide signoff accuracy and accelerate timing closure.

Keywords:

Timing analysis, current source models, effective capacitance, resistive shielding, Miller effect

Διπλωματική Εργασία

Ανάλυση χρονισμού κυκλωμάτων πολύ μεγάλης κλίμακας ολοκλήρωσης σε προηγμένες τεχνολογίες

Βαγενάς Αναστάσιος-Ιούλιος

Περίληψη

Η συνεχής κλιμάκωση της τεχνολογίας των τρανζίστορ και η ευρεία χρήση των ολοκληρωμένων κυκλωμάτων στις καθημερινές μας ζωές, έχουν κάνει εξαιρετικά απαιτητικό το έργο της ανάλυσης χρονισμού σε επίπεδο πύλης. Η αντίσταση των διασυνδέσεων, η μη-γραμμικότητα των σημάτων, το φαινόμενο του θορύβου, και η αλληλοεξάρτηση στην ανάλυση πυλών και διασυνδέσεων οδηγεί σε νέες προκλήσεις στην ακριβή εκτίμηση χρονισμού. Επιπλέον, η ανάλυση χρονισμού θα πρέπει να είναι αποδοτική ώστε να κλιμακώνει για κυκλώματα ολοένα και αυξανόμενου μεγέθους και να μπορεί να ενσωματωθεί σε επαναληπτικές ροές βελτιστοποίησης στα στάδια της χωροθέτησης πυλών και δρομολόγησης διασυνδέσεων. Σε αυτή τη διπλωματική, παρουσιάζουμε μια νεα επαναληπτική μεθοδολογία για ανάλυση χρονισμού σταδίων σε επίπεδο πύλης, χρησιμοποιώντας μοντέλα πηγών ρεύματος και την έννοια των πολλαπλών κλίσεων σήματος και ισοδύναμων χωρητικοτήτων. Η επαναληπτική διαδικασία χρησιμοποιεί τμηματικά γραμμικές κυματομορφές για να προσεγγίσει τα μη γραμμικά σήματα, υπολογιστικά αποδοτικές φόρμουλες, και γρήγορες τοπολογικές διαπεράσεις του RC μοντέλου για να προωθήσει τις πολλαπλές κλίσεις και ισοδύναμες χωρητικοτήτες. Σε αντίθεση με προηγούμενες προσεγγίσεις, η μεθολογία μας λαμβάνει υπόψη το φαινόμενο προστάσιας αντίστασης, το φαινόμενο Miller, και τις αλληλοεξαρτήσεις πυλών-διασυνδέσεων σε τεχνολογίες χαμηλών νανομέτρων. Η πειραματική μας αξιολόγηση σε 2.2 εκατομμύρια <οδηγος, διασύνδεση, δέκτες> στάδια επιπέδου πύλης, υλοποιημένα σε τεχνολογία 7 νανομέτρων FinFET του ASU ASAP Predictive PDK, δείχνει ότι η υλοποίησή μας μπορεί και να έχει εως και 5.40% μέσο σφάλμα συγκριτικά με SPICE προσομοίωση. Επιπροσθέτως, καταναλώνει 0.38 MB μνήμης ενώ υπολογίζει τους χρονισμούς σε 8.16 δευτερόλεπτα για 2.2 εκατομμύρια στάδια. Εν κατακλείδι, η προτεινόμενη μεθοδολογία μπορεί να επιταχύνει επαναληπτικές ροές βελτιστοποίησης ενός ολοκληρωμένου κυκλώματος, πα-

ρέχοντας ακριβή εκτίμιση χρονισμού.

Λέξεις-κλειδιά:

Ανάλυση χρονισμού, μοντέλα πηγών ρεύματος, ισοδύναμη χωρητικότητα, προστασία αντίστασης, φαινόμενο Miller

Table of contents

List of figures

List of tables

List of Algorithms

Abbreviations

ASIC	Application-Specific Integrated Circuit
CAD	Computer-Aided Design
CCS	Composite Current Source
CSM	Current Source Model
DTA	Dynamic Timing Analysis
EDA	Electronic Design Automation
GBA	Graph-Based Analysis
HDL	Hardware Description Language
IC	Integrated Circuit
LUT	Look-Up Table
NLDM	Non-Linear Delay Model
PBA	Path-Based Analysis
PPA	Power, Performance, and Area
RTL	Register Transfer Level
SI	Signal Integrity
SPICE	Simulation Program with Integrated Circuit Emphasis
STA	Static Timing Analysis
SSTA	Statistical Static Timing Analysis
VLSI	Very Large Scale Integration
VRM	Voltage Response Model

Chapter 1

Introduction

Over recent years, Integrated Circuits (ICs) have become an integral part of our daily life, with many applications in low-power devices (e.g., watches, smartphones, sensors), medical devices, home appliances, and automobiles. The requirements for massive production of ICs, coupled with the increasing customer needs for performance and accessibility of next-generation technology, pose a challenge for the Electronic Design Automation (EDA) industry. In addition, the desire for higher performance and lower power consumption in modern ICs brought rapid scaling of technology to deep nanometer nodes (45 nm and below). Consequently, there is an increase in design complexity and size (transistor count), the resistivity of parasitic interconnects, and process variation.

Most EDA design flows for an Application-Specific Integrated Circuit (ASIC) include timing analysis steps as a verification method. The main objective of timing analysis is to analyze the circuit for timing violations (e.g., setup, hold) and guarantee correct operation at a target clock frequency. Furthermore, timing analysis is essential in timing-driven optimization flows for placement/routing [1, 2] and the final signoff that determines whether the IC is ready for manufacturing. Recent studies show that timing analysis can take up to 60% of the total design time [3].

Static Timing Analysis (STA) [2] and Dynamic Timing Analysis (DTA) [4] engines are employed to tackle the problem of timing verification of contemporary ICs. STA tools need to be orders of magnitude faster than SPICE while being highly accurate (2-3% over SPICE). For that to happen, industrial STA tools have to scale to accommodate larger designs and be capable of modeling the complex electrical phenomenona (e.g., interconnect resistivity, signal non-linearity, crosstalk) during circuit delay calculation. In order for timing analysis to

be feasible during the ASIC design flow, the existence of an efficient STA tool is necessary.

1.1 Motivation

Circuit delay calculation is performed at the gate-level stage [2], where each stage consists of a driver gate, an RC interconnect, and multiple receiver gates. Due to the interdependence between driver and receivers, which is very prevalent in advanced technology nodes, an iterative approach is mandatory for stage timing estimation.

1.1.1 Gate timing

Gate modeling uses Lookup Table-based (LUT) models, such as the Current Source Model (CSM) [5, 6] and the less accurate Voltage Response Model (VRM), to effectively estimate the gate output delay and slew. However, these models are pre-characterized into standard cell libraries using lumped capacitive loads, posing a significant challenge for highly resistive interconnects in advanced technology nodes. Traditional techniques approximate the driving point admittance of an interconnect using a single effective capacitance (C_{eff}) [7], which fails in capturing the non-linear characteristics of the driver waveform. Moreover, most works ignore the Miller effect introduced by the receivers to the interconnect [7, 8, 9]. Instead, C_{eff} values computed in multiple voltage regions are required to capture these effects using CSMs [10, 11].

1.1.2 Interconnect timing

Interconnect timing has become crucial, as interconnect delay takes up a large portion of the total stage delay [12]. SPICE simulation of RC interconnects is devoid of expensive transistor parameter evaluation but requires substantial resources (i.e., runtime, memory) and is prohibitive for optimization-based flows [13, 14]. On the other hand, fast moment-based timing metrics [15, 16, 17, 18, 19] are computationally more efficient and easily scale on large designs but might be highly inaccurate as they rely on simplistic assumptions. Even higher-order moment-based metrics, such as AWE and RICE [12, 20], suffer from numerical instability and do not capture the effects in modern RC interconnects. More specifically, moment-based methods/metrics do not take into account the resistive shielding effect present in highly resistive interconnects, assume step inputs (instead of non-linear waveforms), and

might not be applicable for some interconnect topologies (π-models). In contrast, another approach in [21] used closed-form formulas and C_{eff} to account for resistive shielding but ignored the signal non-linearity and the Miller effect.

1.2 Contributions

This work focuses on the timing analysis of gate-level stages, using the concept of C_{eff} and CSMs, with the main contributions being:

- A fast iterative algorithm for stage timing calculation that approximates the Piece-Wise Linear (PWL) voltage waveforms using multiple C_{eff} values while considering the Miller effect. Compared to previous works, it considers the driver-interconnect interdependencies and does not require expensive computation of moments or a reduction to a π-model.

- Experimental evaluation on representative stages implemented on 7 nm FinFet technology, where timing and C_{eff} calculation is challenging. The iterative algorithm is compared against methods based on established techniques (i.e., π-model reduction, moments computation). Results show that our methodology achieves up to 5.40% mean Relative Error (RE) against SPICE while requiring extremely low amount of memory (0.6 MB).

Part of this work was published in [22], which also won the first place in the ACM TAU 2021 student contest in timing analysis.

1.3 Thesis organization

The rest of the thesis is organized as follows:

- **Chapter 2 (Background)**: The basics of the ASIC design flow, timing analysis, STA, and the problems that arise during stage timing estimation.

- **Chapter 3 (Proposed methodology)**: The proposed iterative algorithm for stage timing estimation, including explanation of each step and the implementation details.

- **Chapter 4 (Experimental evaluation)**: The evaluation of the proposed iterative algorithm against established methods in terms of accuracy, runtime, and memory consumption.
- **Chapter 5 (Conclusions)**: Conclusions of this work, potential enhancements and future directions.

Chapter 2

Background

In this section, we present the theoretical background needed for gate-level stage timing estimation. Firstly, we describe the top-level steps of the ASIC design flow and the significant role timing analysis plays during each step. Secondly, we delve into the details of STA, how timing paths are formed, examples of timing constraints, and the different methods of STA used. Finally, we describe the problem of stage timing analysis with the difficulties arising in gate and interconnect timing.

2.1 Design flow

Modern VLSI circuits contain billions of transistors and require significant amount of resources and time for manufacturing. Furthermore, the customer needs for lower power, reduced chip area, and higher clock frequencies have exponentially increased the complexity and the number of verification methods required during the design of an ASIC. These needs have led to the multi-step flow performed for the complete manufacturing of an ASIC, which can be broadly split into the frontend and backend parts of the flow.

2.1.1 Frontend

The frontend part of the design flow, as shown in Figure 2.1, is responsible for the transformation of the ASIC design specification into an equivalent Register Transfer-Level (RTL) circuit description.

Design Specification

Initially, the customer (or the market trend) describes the problem and the product specifications (e.g., Power, Performance, and Area [PPA]). Then, the designer formulates a solution in the form of a top-level description, including the protocols, the package die, and the power supply voltage used.

Architecture

After the design specification, the designer creates the architecture that characterizes the ASIC. The architectural choices for the ASIC can be some of the following:

- Arithmetic units (i.e., adder) architecture
- Chip and functional module hierarchy
- Module connections and relationships
- Design time and resource allocation for each module

Logic Design and Verification

The last step of the frontend involves describing the data flow of each functional module in a Hardware Description Language (HDL) like Verilog, VHDL, or System Verilog. Before moving to the backend part of the flow, the behavioral circuit is simulated logically to check for correct operation.

2.1.2 Backend

The backend part of the design flow, as shown in Figure 2.1, is responsible for the transformation of the RTL circuit description into the final physical chip.

Synthesis

Synthesis transforms the RTL circuit description into a standard cell gate-level representation using a technology library (.lib file) based on the design constraints (e.g., timing). This representation, also called netlist, is simulated for logical equivalency with the behavioral circuit and initial PPA estimation.

Floorplanning

After synthesis, a rough estimate of the total area required for the circuit is possible. Combining this estimate along with the package die restrictions, the designer creates the floorplan where the cells will be placed. This phase usually includes power planning, meaning the creation of the power/ground network.

Placement

Following the floorplanning step, the standard cells are placed inside legal positions of the floorplan while keeping the connections (wires) lengths minimal. Even though connections between cells do not exist at this step, the distances are approximated using the total wire length and routing congestion.

Routing

The final physical step of the backend is the wire routing. The pins of the cells are connected using multiple metal layers (up to 15), with each metal situated at a specific track (height). Furthermore, the router aims to minimize the length (along with timing) and reduce the congestion of the metals in the floorplan.

Signoff

After the physical design of the ASIC, the final signoff-level verification follows. The routed design goes through verification checks for timing (e.g., setup, hold), Design Rule Violations (DRV), electromigration phenomena, and more. Once the design passes all of the final verification checks, it is ready for manufacturing.

2.2 Timing analysis

Timing analysis is one of the numerous and most commonly used verification methods in the design flow of an ASIC. The primary goal of timing analysis is the timing and functional verification of the design. Timing verification examines the circuit for timing violations to guarantee stable operation at a target clock frequency, which is usually the major constraint in a design. Functional verification incorporates the inspection of the logical operation of the design, which is carried out by sensitizing the circuit's input(s) with several input vectors.

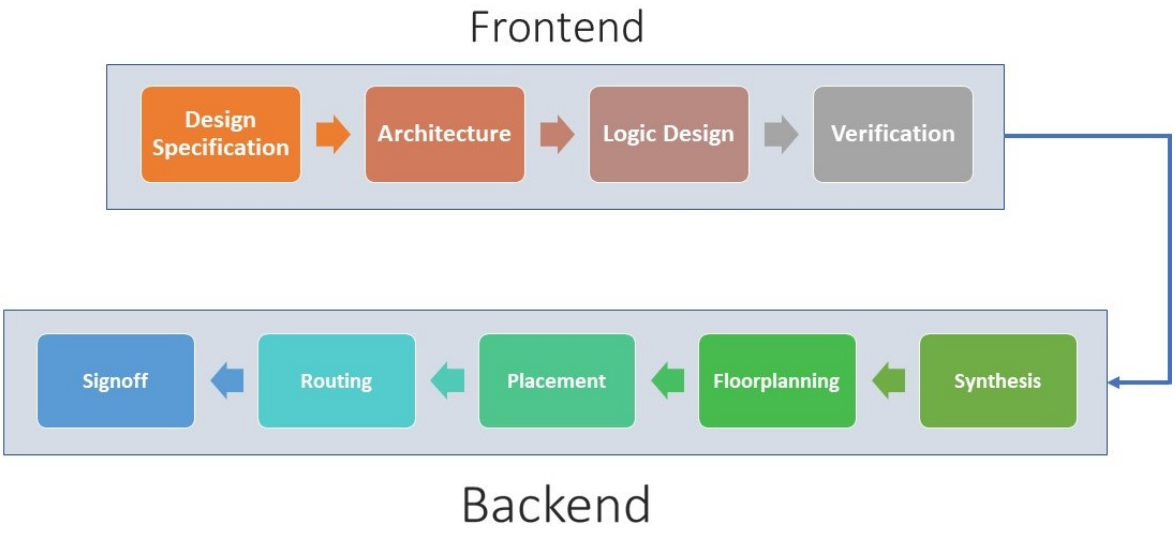

Figure 2.1: The ASIC design flow.

On the one hand, timing analysis needs to be highly accurate in order to achieve timing closure of the design and reduce unnecessary loops in the design flow (e.g., from routing back to synthesis). The complex electrical phenomena during delay calculation, such as interconnect resistivity, signal non-linearity, and crosstalk, heavily disturb the task of an EDA timing tool since they greatly affect its accuracy. On the other hand, timing analysis needs to be efficient (in terms of memory/runtime) since iterative optimization flows, which operate during each step of the design flow, utilize it to ensure that changes do not impact the timing or functionality of the design. This, coupled with the sheer complexity of VLSI circuits of billions of transistors (or millions of gates), further complicates the task of timing analysis.

It is apparent that timing analysis needs to be quick, accurate, and scale for the the ever increasing circuit sizes. A single timing analysis methodology is insufficient to cover all the timing and functional verification needs of the design, hence multiple types exist and are employed during the flow. The main variants of timing analysis methodologies used are i) Static Timing Analysis (STA), which verifies the timing integrity of the design using a fast wost-case vectorless approach, and ii) Dynamic Timing Analysis (DTA) that also covers the functional verification aspect with a slower input-vector dependent approach. Finally, during the signoff step of the flow, more complex and computationally expensive timing methodologies that are extensions of the above can be employed, such as Statistical STA (SSTA) [23], IR-drop aware timing analysis, and Signal Integrity (SI) aware timing analysis.

2.2.1 Static Timing Analysis

STA is the cornerstone of timing analysis and the core of any EDA tool since it is the most commonly used analysis throughout the entire design flow of an ASIC. The analysis is ”static” since no input vectors are required to excite the circuit input pins. Moreover, it can produce pessimistic (worst delay/slew) and optimistic (best delay/slew) results for the target circuit, providing an upper and lower bound for the design timing.

A circuit typically contains sequential (typically Flip-Flops [FF]) and combinational elements connected together (e.g., NOR, AND, XOR gates), which form timing paths [1]. STA is performed on these paths, which are made up of the following: [24]:

- **Startpoint**: The path starting point, where data is propagated by a clock edge or is constrained by a specific timing. Startpoints can be either input ports or register clock pins.
- **Combinational logic network**: Combinational logic, meaning elements that do not have any state or memory.
- **Endpoint**: The path ending point, where data is captured by a clock edge or is constrained by a specific timing. Endpoints can be either register data input pins or output ports.

Figure 2.2 and Table 2.1 show a design example and the possible timing paths that might exist in a design. Each ”logic bubble” represents a combinational network with each path starting from a data launch point and ending at a data capture point. The typical STA flow breaks down the design into timing paths to check for the specified timing constraints, with the most common being:

- **Setup** specifies the time needed for the data to be available at the input pin of a sequential cell before the capturing clock edge. During setup analysis the longest path has to meet the setup time constraint.
- **Hold** specifies the time needed for the data to be stable at the input pin of a sequential cell after the capturing clock edge. During hold analysis the shortest path has to meet the hold time constraint.

[1] For brevity, only data timing paths are considered.

A single combinational network can potentially contain multiple paths, as illustrated in Figure 2.3, and each one can be further broken down into <driver-connection(s)-receiver(s)> timing stages. Each stage is a self-contained entity where delay/slew calculation occurs, therefore the STA unit cell for timing analysis is the stage. It is important to note that a stage's connection (e.g., net) delay is initially calculated using LUTs or empirical wire-delay models. However, after the placement/routing step of the design flow, the net connection becomes an RC parasitic interconnect (Figure 2.3).

The main task of an STA tool is to find the propagation delay/slew at the endpoints of the timing paths and perform the setup/hold violation checks. Initially, the slew constraints are set at the startpoint of each timing path and they are propagated across the stages that form the combinational network up to the corresponding endpoint. At each successive stage, the delay is accumulated in order to find the total propagation delay and the input slew at the endpoint, which are both used to perform the setup/hold violation checks. For that reason, the majority of runtime during STA delay calculation is spent in stage timing estimation, which also heavily depends on the complexity and size of the RC interconnect itself. Finally, depending on the analysis mode of the STA tool (Path-Based Analysis [PBA] or Graph-Based Analysis [GBA]) the way the slew/delay values are propagated changes appropriately.

Graph-Based Analysis

GBA is the STA-mode that is mostly used during the design flow. In this mode, only the worst (or best) input slew is propagated from the input pin of each gate to the output, in order to produce the gate delay/slew (Figure 2.4). This approach is computationally efficient since it runs in polynomial time (i.e., proportional to the number of gates on the design), but adds unnecessary pessimism to the design's timing.

Path-Based Analysis

PBA is the second, more accurate, STA mode used during the design flow. In this mode, the actual slew is used on an input pin of the gate, with the real slew/delay value being propagated across the stages of each timing path (Figure 2.4). This mode of analysis is the most accurate gate-level analysis, but requires potentially exponential time (i.e., proportional to the number of paths on the design).

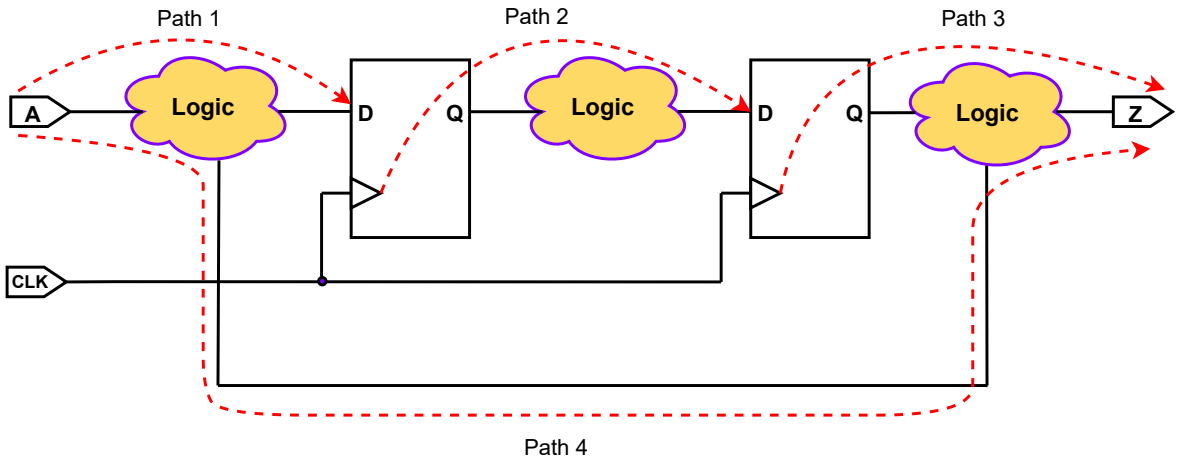

Figure 2.2: Timing paths on a simple design.

Path	Startpoint	Endpoint
Path 1	Input port	Data pin of FF
Path 2	Clock pin of FF	Data pin of FF
Path 3	Clock pin of FF	Output port
Path 4	Input port	Output port

Table 2.1: Timing paths examples for Figure 2.2

2.3 Stage timing estimation

2.3.1 Problem formulation

Consider the stage of Figure 2.5, where the driver gate output (i.e., the source node 0) is connected to the input of the receiver gates via a distributed interconnect. Each node i of the parasitic RC tree has a capacitance to ground C_i and is connected with a resistance $R_{i \to j}$ to each fanout node j. We denote the slew at node i by S_i, its total downstream capacitance (i.e., the sum of all the capacitances of the subtree starting at node i) by C_{tot_i}, and the corresponding effective capacitance by C_{eff_i}. More specifically, the resistance value ($R_{1 \to 5}$) from node 1 to fanout node 5 is 5 $k\Omega$, the capacitance value (C_6) of node 6 is 1 fF, and the downstream capacitance value (C_{tot_2}) of node 2 is 2 fF. A special case can be made for the downstream capacitance of node 0, since it is the entire RC tree's capacitance sum and is denoted as the total capacitance (C_{total}) of the stage.

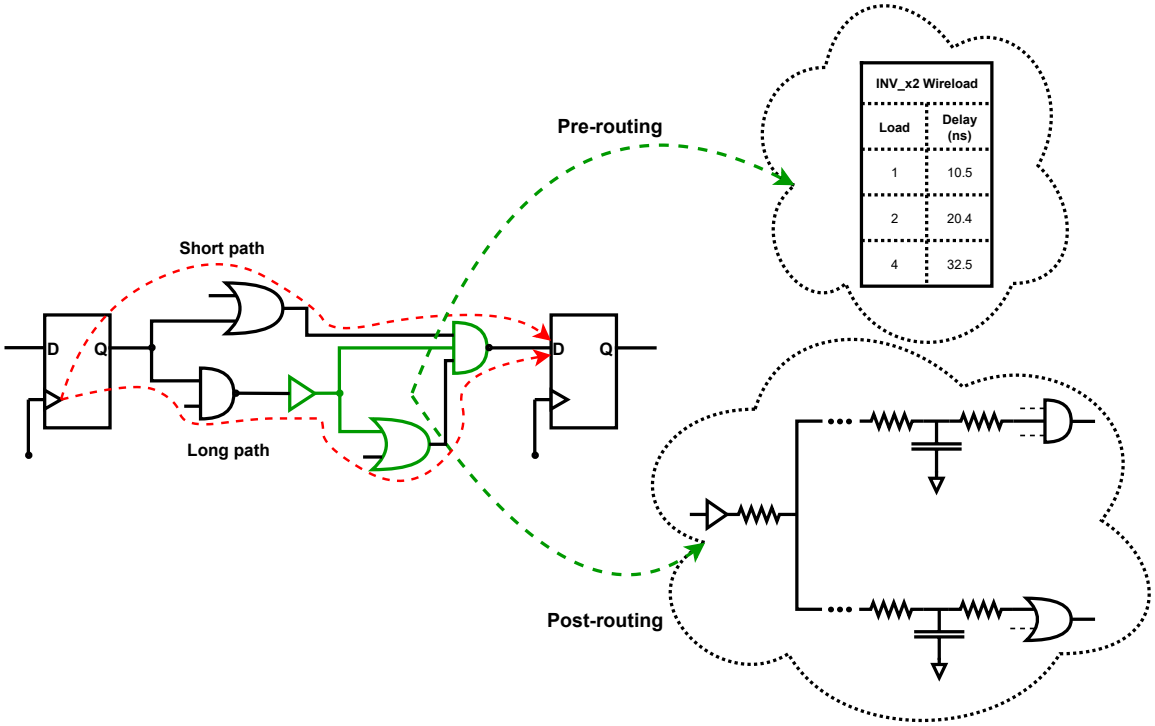

Figure 2.3: Combinational network example along with potential timing paths.

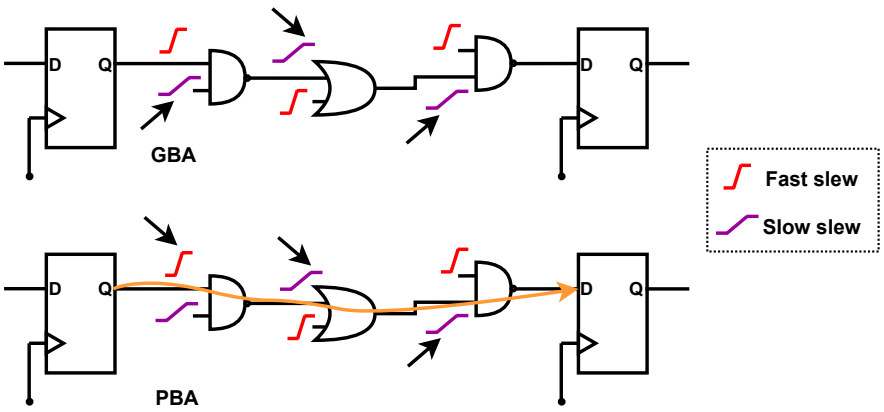

Figure 2.4: GBA and PBA methods for STA.

Problem: The main objective of this work is to accurately estimate the driver delay (D_{drv}), the driver output slew (S_{drv}), the interconnect delay to the target receiver (D_{rcv}), and the target receiver input slew (S_{rcv}), given the driver input slew ($S^{\text{in}}_{\text{drv}}$) and the output capacitance values ($C^{\text{out}}_{\text{rcv}}$) at the receivers [typically set to C_{total} of the respective fanout stage].

Typical solving approach: First, the interconnect input admittance is approximated by

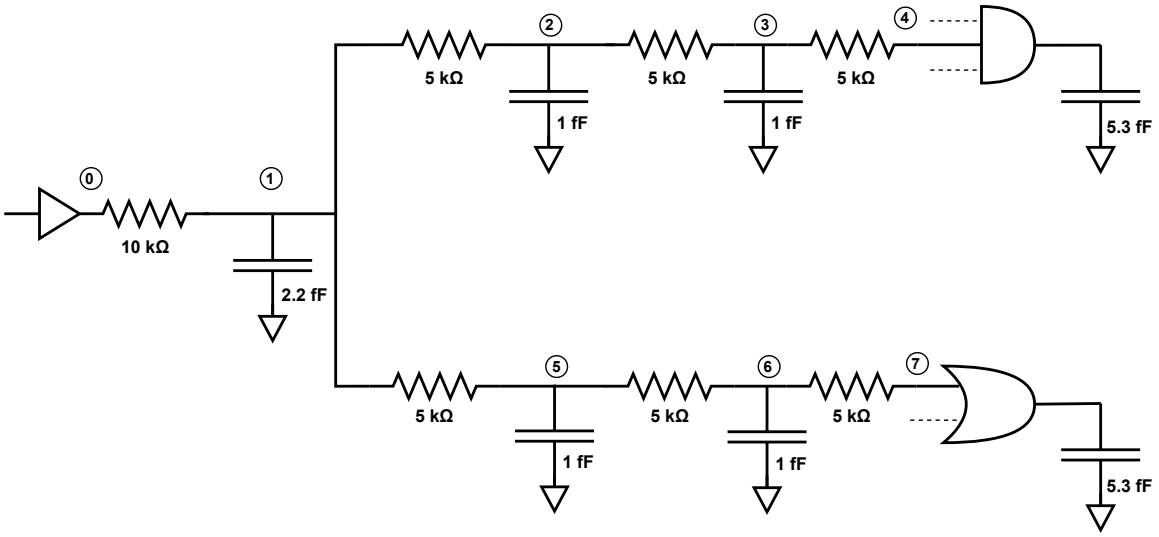

Figure 2.5: A timing stage example with multiple receivers and annotated RC values.

C_{eff} and the driver output voltage waveform is computed in order to estimate driver delay and output slew. Then, this waveform (or the estimated slew value(s)) is used for interconnect delay estimation as well as for estimation of the receiver input voltage waveform (or slew value(s)). The receiver input slew can then used as input in the analysis of the next stage.

2.3.2 Gate timing

Library compatible CSMs - VRMs

Gate modelling uses LUT-based models, which are precharacterized into 1D/2D tables of technology libraries to be used for stage delay/slew estimation, such as the CSM and the VRM model. Our methodology primarily focuses on the Composite Current Source (CCS) model [5] for CSMs and Non-Linear Delay Model (NLDM) on the side of VRMs. Below we provide a demonstration of the NLDM and CCS driver/receiver models.

NLDM driver model captures the output slew/delay as a function of driver input slew ($S^{\text{in}}_{\text{drv}}$) and output load ($C^{\text{out}}_{\text{drv}}$). A single value is used for the slew/delay estimation, hence, non-linearities of the driver waveform cannot be effectively modeled. As shown in Figure 2.6, this information is stored in `cell_rise/fall` and `rise/fall_transition` LUTs, for gate delay and slew, respectively.

NLDM receiver model uses a single input pin capacitance (C_p) value to model the non-linear receiver transistor input capacitance. As depicted in Figure 2.6, the NLDM receiver

capacitance value is stored in `rise/fall_capacitance` fields of each standard cell. Unfortunately, this way of modeling the receiver capacitance does not capture the Miller effect or the dependency of C_p to the input slew (S_{rcv}) and the output load ($C_{\text{rcv}}^{\text{out}}$) of the receiver cell.

CCS driver model captures the output current as a function of driver input slew ($S_{\text{drv}}^{\text{in}}$) and output load ($C_{\text{drv}}^{\text{out}}$). As shown in Figure 2.6, this information is stored in `output_current_rise/fall` LUTs. The time instant when the corresponding driver input voltage crosses the delay threshold (usually $0.5V_{\text{dd}}$), which is necessary to calculate gate delay, is also stored as `reference_time` in these LUTs.

CCS receiver model typically uses two different input C_{p} values, $C1$ and $C2$, to model the non-linear receiver transistor input capacitance and the Miller effect. This is very important since the load seen by the driver depends both on the interconnect RC network and the input pin capacitance of the multiple receiver gates. As depicted in Figure 2.6, CCS receiver capacitance values are stored in `receiver_capacitance_1/2_rise/fall` LUTs, and are also dependent on the S_{rcv} and $C_{\text{rcv}}^{\text{out}}$ of the receiver cell.

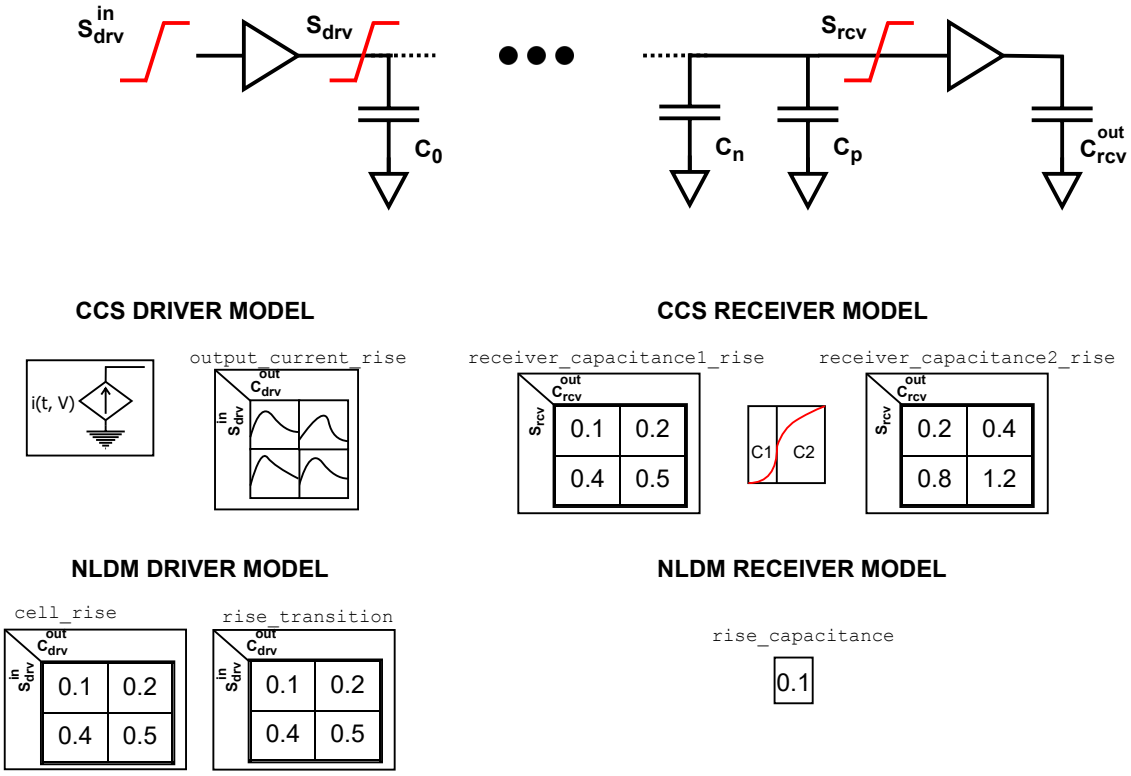

Figure 2.6: Comparison of the NLDM and CCS timing model.

Modeling the RC interconnect load with C_{eff}

In advanced technology nodes, on-chip RC interconnect loads become increasingly resistive, which complicates their modeling with a single capacitance value for gate timing estimation with CSMs. The simplest approximation of the driving point admittance of an RC interconnect is C_{total}. However, this results in pessimistic gate timing estimation, as it totally ignores the resistive shielding effect, where interconnect resistance shields a part of C_{total}.

To address this limitation, the concept of C_{eff} is introduced in [7], which is an equivalent capacitance such that gate timing is the same when using (i) the driving point admittance and (ii) the estimated C_{eff} as a load. The traditional approach for C_{eff} estimation includes two steps [7]. First, the distributed interconnect is approximated by a π-model, proposed by O'Brien and Savarino [25]. Then, C_{eff} is computed by equating the charge absorbed by the π-model and the charge absorbed by C_{eff}, up to the delay threshold (i.e., the time instant when driver output voltage crosses $0.5V_{\text{dd}}$). However, a single C_{eff} value focuses only on delay estimation and is inadequate to approximate slew, as it assumes that driver output voltage waveform is a linear ramp. Moreover, it does not consider the Miller effect, neglecting the impact of receiver input capacitance (C_{p}) on C_{eff}, delay, and slew estimation.

In order to accurately approximate the non-linear driver waveform, C_{eff} must be computed in multiple voltage regions. Recently, authors in [10] presented a CSM-based method to compute driver output C_{eff} and slew in multiple regions, considering the Miller effect, for a π-model load approximation. The main drawback of [10] is that the reduction of the distributed interconnect to a π-model induces significant error in advanced technologies while introducing memory overhead, as we demonstrate in Section 4.

Another interesting C_{eff} estimation technique [21] handles the distributed interconnect as connected π-models and iteratively refines C_{eff} (along with delay and slew) on each RC-tree node by repeated forward-backward traversals. However, it does not exploit CSMs and still approximates signal waveforms by linear ramps, assuming a single C_{eff} on each node while ignoring the Miller effect.

To support our methodology, we extended [10, 21] to compute multiple C_{eff} values, avoiding a π-model reduction. In our modeling, for each node i of the RC-tree (see Figure 2.6),

C_{eff_i} for each region ($\alpha V_{\text{dd}} - \beta V_{\text{dd}}$), where $\alpha, \beta \in [0.0, 1.0]$ and $\alpha < \beta$, is calculated by:

$$C_{\text{eff}_i}^{\alpha-\beta} = C_i + \sum_{\text{each fanout } j} K_j^{\alpha-\beta} * C_{\text{tot}_j}, \tag{2.1}$$

$$\text{where } K_j^{\alpha-\beta} = 1 - \frac{X_j^{\alpha-\beta}}{(\beta-\alpha)}\left(e^{-\frac{\alpha}{X_j^{\alpha-\beta}}} - e^{-\frac{\beta}{X_j^{\alpha-\beta}}}\right) \text{ and } X_j^{\alpha-\beta} = \frac{R_{i\to j} C_{\text{eff}_j}^{\alpha-\beta}}{S_i^{\alpha-\beta}}.$$

In the above formula, $S_i^{\alpha-\beta}$ is the slew of node i while $C_{\text{eff}_j}^{\alpha-\beta}$ is the C_{eff} of the fanout node j, corresponding to region ($\alpha V_{\text{dd}} - \beta V_{\text{dd}}$). As can be seen, $C_{\text{eff}_i}^{\alpha-\beta}$ accounts for a different shielding factor ($K_j^{\alpha-\beta}$) for each fanout node j, which shields C_{tot_j} and depends on $R_{i\to j}$, $C_{\text{eff}_j}^{\alpha-\beta}$, and $S_i^{\alpha-\beta}$ to properly account for the resistive shielding. Note that only for receiver input pins, C_i also incorporates the receiver input capacitance (including the Miller component) computed in the corresponding region ($C_p^{\alpha-\beta}$).

The main advantage of using multiple C_{eff} regions for each RC-tree node is that the non-linear driver output waveform can be accurately approximated by a PWL ramp, exploiting library compatible CSMs to compute the slew of each region using the corresponding C_{eff} value. As a result, our approach achieves superior accuracy compared to [21, 10], especially for driver output slew, while it is also essential for accurate interconnect analysis.

2.3.3 Interconnect timing

The most accurate interconnect timing estimation is obtained by transient analysis using SPICE. Although SPICE provides golden results and is devoid of expensive transistor parameter evaluation, it fails to meet the performance and memory requirements for full-chip analysis. This is due to the fact that it involves detailed computations with runtime proportional to the number of time steps and the RC network complexity. On the contrary, several lighter analytical models, based on the first few moments of the impulse response, have been proposed for fast timing analysis of interconnects [15, 16, 17, 18, 19]. However, these models assume step inputs while ignoring the resistive shielding and the interdependence between slew and C_{eff}. For each node i of the RC-tree, the j-th ($m_j^{(i)}$) moment can be recursively calculated by:

$$m_j^{(i)} = \sum_{\text{each node } k} R_{\text{ki}} C_k m_{\text{j-1}}^{(k)}, \qquad where\ m_0^{(k)} = 1, \tag{2.2}$$

where R_{ki} is the common resistance of paths $0 \to i$ and $0 \to k$, C_k the node capacitance, and for each node i the initial condition m_0^i is set to 1.

In contrast to conventional delay and slew models [15, 16, 17, 18, 19], the iterative refinement-based method proposed in [21] exploits the C_{eff} concept for accurate gate-level interconnect timing estimation. More specifically, interconnect delay is computed by an extended version of Elmore delay, which uses C_{eff_i} instead of C_{tot_i} on each interconnect node i. For the RC-tree of Figure 2.6, the delay from driver output (node 0) to each node i is computed by:

$$D_{0\rightarrow i} = \sum_{\text{each node } k} R_{ki} C_{\text{eff}_i}^{0-0.5}, \tag{2.3}$$

where R_{ki} is the common resistance of paths $0 \rightarrow i$ and $0 \rightarrow k$. On the other hand, in [21], a single slew value is computed on each node, using $C_{\text{eff}}^{0-0.5}$ and ignoring the Miller capacitance, which is insufficient to accurately approximate the non-linear signal waveform.

To enable accurate stage timing analysis, we generalized the slew metric of [21] to any voltage region, considering the respective C_{eff} [computed by Eq. (2.1)]. In our method, for each node i, the slew of region ($\alpha V_{\text{dd}} - \beta V_{\text{dd}}$) is propagated to each fanout node j as follows:

$$S_{\text{j}}^{\alpha-\beta} = \frac{S_{\text{i}}^{\alpha-\beta}}{1 - X_j^{\alpha-\beta}\left(1 - e^{-\frac{1}{X_j^{\alpha-\beta}}}\right)} \tag{2.4}$$

Chapter 3

Proposed methodology

In this section, we present our iterative stage timing estimation algorithm that considers the strong interdependencies of Eq. (2.1, 2.3, 2.4), along with its implementation details.

3.1 Iterative algorithm

The details of our initial iterative algorithm for post-PnR stage timing estimation are described in Algorithm 1. First, the C_{eff_i} of each node i is initialized to the downstream capacitance, for each voltage region, using the NLDM capacitance for receivers (step 0). In the first step of the main iterative loop, D_{drv} and S_{drv} are estimated by computing the driver output PWL voltage ramp using CSMs and multiple C_{eff_0} values (step 1). In step 2, during a forward Breadth-First Search (BFS) traversal of the RC-tree, the modified Elmore delay ($D_{0\to i}$) from driver output to node i is computed using Eq. (2.3), while the multiple slew values (S_i) of node i are propagated to each fanout node j using Eq. (2.4). When the multiple slew values of the PWL voltage ramps have been propagated to the target and side receivers, their C_{eff} values for all regions are updated using the CSM receiver model, which considers the Miller effect (step 3). Finally, in step 4, the RC-tree is traversed backward using BFS to update C_{eff_i} per region based on Eq. (2.1). This iterative refinement of C_{eff}, delay, and slew values is performed until the multiple driver output C_{eff_0} values converge within a specified tolerance (e.g., 0.01 fF).

The overall time complexity of Algorithm 1 is $O(k|V||E|)$, where k is the number of iterations needed for convergence, $|V|$ is the number of voltage regions used, and $|E|$ is the number of edges on the RC interconnect tree. The computation of the CSM PWL voltage

ramp can be considered a constant-time operation, as they may differ across CSM variants and do not depend on $|E|$ or $|V|$. In contrast, the most prominent costs during the operation of Algorithm 1 are the two BFS traversals across the edges of the RC tree and the multiple ($|V|$) S_i/C_{eff_i} values that are propagated across each edge (i.e., π-model). Finally, the iterative procedure (steps 1-4) is performed for a number of iterations, typically 5 iterations, which sets the overall complexity to $O(k|V||E|)$.

Algorithm 1: Stage timing estimation using CSM

Function `stage_CSM_timing` (*stage, multiple ($\alpha V_{dd} - \beta V_{dd}$) regions*) **:**

0. Initialization:

   - Set all $C_{\text{eff}_i}^{\alpha-\beta}$ values for each RC-tree node i to C_{tot_i} (using NLDM capacitance for receivers)

while *driver output $C_{eff_0}^{\alpha-\beta}$ values not_converged* **do**

1. Compute driver CSM PWL voltage ramp using $C_{\text{eff}_0}^{\alpha-\beta}$ values
2. Forward Traversal (driver-to-receivers):
   - Compute Elmore delay ($D_{0\to i}$) to node i using Eq. (2.3)
   - Compute $S_j^{\alpha-\beta}$ values for each fanout node j using Eq. (2.4)
3. Update $C_{\text{eff}_i}^{\alpha-\beta}$ values on receivers using CSM receiver model
4. Backward Traversal (receivers-to-driver):
   - Compute $C_{\text{eff}_i}^{\alpha-\beta}$ values of node i using Eq. (2.1)

end

End Function

It is worth noting that several other moment-based metrics have been evaluated for delay (e.g., D2M [17], WED [16]) and slew (e.g., SS2M [17], LnS [19]) propagation, along with PERI [26] to extend metrics to saturated ramp inputs. However, our metrics employed in step 2 of our proposed Algorithm 1 lead to better accuracy results, while they do not require expensive computation of moments.

3.2 Implementation details

3.2.1 Algorithm setup and modifications

Voltage regions

Algorithm 1 can be configured in arbitrary voltage regions but the most common configuration would be using 3 regions with 4 thresholds $V = \{0, V_{\text{low}}, V_{\text{delay}}, V_{\text{high}}\}$, which match the gates' library predefined thresholds. Depending on the complexity of the RC interconnect and the accuracy needed, the algorithm can be enhanced by increasing the voltage thresholds in between those. Nevertheless, for the rest of the thesis we use a configuration of V=$\{0, 0.1V_{\text{dd}}, 0.5V_{\text{dd}}, 0.9V_{\text{dd}}\}$, that matches the technology library's thresholds used in the experimental evaluation of the methodology in Section 4.

NLDM integration

The CCS model is the default model used in the configuration of Algorithm 1, as shown during the algorithmic steps. However, simple modifications can be made to the voltage regions used and steps (1, 2, 3), in order to support only the NLDM driver/receiver model instead of CCS.

Firstly, Algorithm 1 should be configured with a single voltage region using 2 thresholds $V = \{V_{\text{low}}, V_{\text{high}}\}$, since NLDM will not benefit from more regions. Secondly, the driver delay D_{drv} and output slew S_{drv} are calculated by accessing the NLDM `cell_rise/fall` and `rise/fall_transition` LUTs (step 1), respectively. Then, Eq. (2.3) used in step 2 is modified and uses $C_{\text{eff}}^{\text{low-high}}$ instead of $C_{\text{eff}}^{\text{0-50}}$. Finally, in step 3 the updating of the $C_{\text{eff}}^{\text{low-high}}$ receiver value is done using the NLDM receiver model.

3.2.2 Driver delay and output slew computation

To calculate driver delay D_{drv} and output slew S_{drv} in step 1 of Algorithm 1, we first access the CCS `output_current_rise/fall` LUTs to find the four closest output current waveforms for the current $(S_{\text{drv}}^{\text{in}}, C_{\text{eff}_0}^{0-10})$, $(S_{\text{drv}}^{\text{in}}, C_{\text{eff}_0}^{10-50})$,$(S_{\text{drv}}^{\text{in}}, C_{\text{eff}_0}^{50-90})$ breakpoints. Then, we transform them into CSM PWL voltage waveforms by integrating using the Trapezoidal rule, and compute the time instants when the actual voltage waveform crosses $0V_{\text{dd}}$, $0.1V_{\text{dd}}$, $0.5V_{\text{dd}}$, and $0.9V_{\text{dd}}$, using interpolation. To compute output slew in three regions

(S_{drv}^{0-10}, S_{drv}^{10-50}, S_{drv}^{50-90}), we use the corresponding time instants, while S_{drv} is calculated by $S_{\text{drv}} = S_{\text{drv}}^{10-50} + S_{\text{drv}}^{50-90}$. As for D_{drv} estimation, we use the `reference_time` corresponding to an input voltage waveform having slew equal to $S_{\text{drv}}^{\text{in}}$, which is computed by interpolating on the `reference_times` of the closest CCS output current waveforms.

3.2.3 Receiver input pin capacitance computation

To compute receiver C_{eff} values in step 3 of Algorithm 1, we use the input slew per region (S_{rcv}^{0-10}, S_{rcv}^{10-50}, S_{rcv}^{50-90}) and fixed receiver load $C_{\text{rcv}}^{\text{out}}$, to access the CCS `receiver_capacitance_1/2_rise/fall` LUTs and compute the corresponding input capacitance C_{p} (i.e., $C1$ or $C2$).

Chapter 4

Experimental evaluation

In this section, we present the experimental results of our proposed methodology against established approaches in the literature using different configurations of Algorithm 1.

4.1 Setup

For the evaluation of the proposed approach, we integrated Algorithm 1 into the C++ framework of the ACM TAU 2021 contest [27], which generates representative <driver, interconnect, receiver(s)> stages of VLSI circuits. Each stage contains one or more receivers, with each stage having a designated ”target” receiver of which we attempt to estimate the timing of (i.e., D_{rcv}, S_{rcv}) and ”side” receivers which are the remaining ones in the stage. The framework is capable of generating different topologies of stages and assessing the accuracy of each methodology employed against the Xyce electrical simulator [28]. The types of stages generated, as illustrated in Figure 4.1, are the following:

- **`standard`**: The general case meant to test typical signal propagation across stages. It is an intermediary type as it is a combination of all the other following testcases.
- **`stress_0`**: The simplest case of stage, a single receiver-driver pair connected with a single-layer wire. Tests short signal propagation effects.
- **`stress_1`**: A single receiver-driver pair connected with a long two-layer wire that transitions multiple times between the layers. Tests long signal propagation effects.
- **`stress_2`**: A multi-receiver case of a stage, where the side receivers are located very close to the driver of the wire. The target receiver rests at the end of the longest branch.

Tests strong side load effects followed by long signal propagation.

- **`stress_3`**: A multi-receiver case of a stage, where the side receivers are situated very close to the target receiver (end of the wire). A long wire routed on the highest layer connects all together. Tests long signal propagation followed by strong side load effects.

In order to evaluate our approach, we compare it against established methodologies, which are modifications of Algorithm 1, namely:

- **Moments**: During the forward traversal of the RC tree in step 2, we utilize moment-based metrics, Elmore for delay and SS2M [17] for slew computation. Furthermore, we use PERI [26] to extend the metrics from step to ramp inputs. Finally, Algorithm 1 is configured with two voltage thresholds V=$\{0.1V_{\text{dd}}, 0.9V_{\text{dd}}\}$.

- **O'Brien**: During the backwards traversal of the RC tree in step 4, we approximate the input admittance of the interconnect by a π-model per region [25] and then compute C_{eff} using [10].

Xyce was configured using a maximum timestep of 0.1 ps for the duration of the simulation, a direct method of solving, and other options that guarantee the highest possible accuracy. For our experiments we generated a million `standard` stages and 300k of each `stress_xx` case, totalling 2.2 million stages of VLSI circuits.

The driver/receiver gates where chosen from the ASU ASAP 7 nm FinFET Predictive PDK [29], with only the inverter/buffer gates being used. Table 4.1 summarizes some of the SPICE parameters of the technology library used in the evaluation. Moreover, the library's voltage thresholds $V = \{0, V_{\text{low}}, V_{\text{delay}}, V_{\text{high}}\}$ are set to V=$\{0, 0.1V_{\text{dd}}, 0.5V_{\text{dd}}, 0.9V_{\text{dd}}\}$, hence the Algorithm configuration described in Section 3.2.1. The stages were generated so that the driver output C_{eff}, the driver input slew, and the receiver output capacitance cover the entire range of the library's precharacterized LUTs and even beyond that.

Finally, we ran all experiments on a Linux workstation with a 3.80 GHz 12-thread AMD® Ryzen 3600X CPU and 16 GB of memory.

SPICE Parameter	Value
Structure Selector ($GEOMOD$)	1 (triple-gate)
Channel Length (L)	21 nm
Fin Height (H_{fin})	32 nm
Fin Thickness (T_{fin})	6.5 nm
Threshold Voltage ($V_{th,n}, V_{th,p}$)	0.25 V, −0.2 V
Oxide Permittivity (ϵ_{ox})	34.53 pF/m
Physical Oxide Thickness (T_{oxp})	21 nm

Table 4.1: ASU ASAP 7 nm SPICE MOSFET parameters

4.2 Accuracy evaluation

In the first part of the accuracy evaluation, we compare the aforementioned methods against our proposed approach. We measured the mean Relative Error (RE) and the mean Absolute Error (AE) against SPICE for each timing metric (i.e., D_{drv}, S_{drv}, D_{rcv}, S_{rcv}), across all testcases cumulatively and for each testcase individually. As shown in Table 4.2, the established approaches fail to achieve signoff-level accuracy, with the `O'Brien` and `Moments` approaches inducing significant error due to the π-model and moments' approximation, respectively. More specifically, these approaches fail in capturing the non-linear effects of the driver waveform, with large errors in the driver's timing estimates (e.g., 22.15% and 26.17% for D_{drv}). In contrast, our methodology is superior across all metrics compared to the other approaches, with the best RE in D_{drv} (5.98% over 11.10%/11.89%) and the worst in D_{rcv} (5.40% over 8.31%/8.91%). This trend is also apparent in the mean AE of all the approaches, where our method achieves extremely low AE (e.g., 5 ps for D_{drv}) against SPICE.

Method	**Mean RE against SPICE**				**Mean AE against SPICE**			
	$D_{\textbf{drv}}$	$S_{\textbf{drv}}$	$D_{\textbf{rcv}}$	$S_{\textbf{rcv}}$	$D_{\textbf{drv}}$	$S_{\textbf{drv}}$	$D_{\textbf{rcv}}$	$S_{\textbf{rcv}}$
O'Brien	22.15%	11.10%	8.31%	8.57%	16.3 ps	20.6 ps	8.3 ps	19.1 ps
Moments	26.17%	11.89%	8.91%	13.18%	19.1 ps	21.3 ps	8.9 ps	26.5 ps
Ours	**6.71%**	**5.98%**	**5.40%**	**5.78%**	**5.0 ps**	**11.5 ps**	**5.3 ps**	**12.6 ps**

Table 4.2: Mean RE/AE against SPICE of the examined approaches across all testcases

In the second part of the accuracy evaluation, we examine the results of our approach

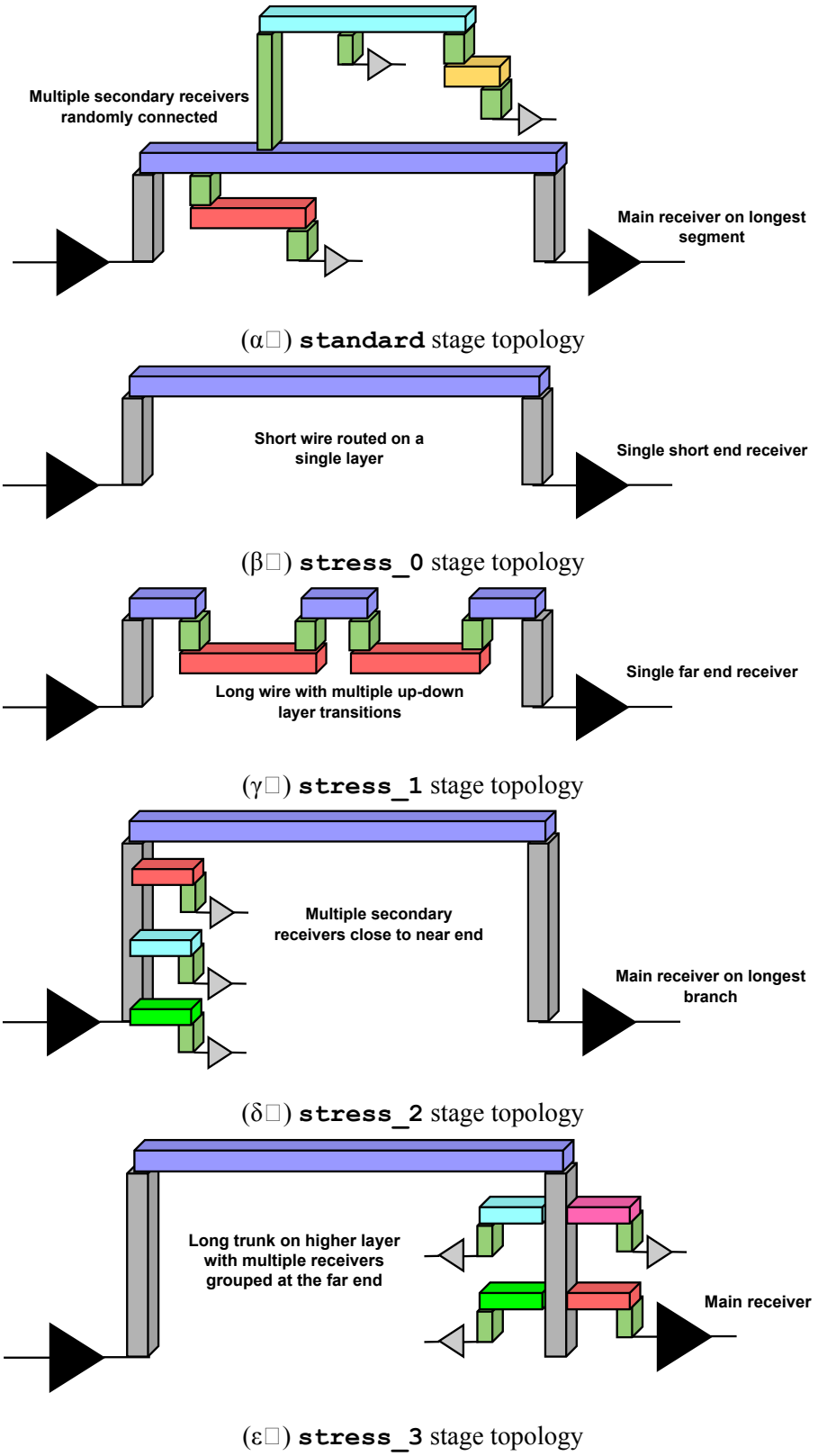

Figure 4.1: The testcases used for the experimental evaluation.

on a per-testcase basis, as shown in Table 4.3. Initially, our proposed methodology performs well, with similar RE on most metrics for the `standard`, `stress_0`, and `stress_3`

testcases. Out of these testcases, the best performance of our approach is in D_{rcv} (4.8%) for `stress_3` while the worst is in D_{drv} (7.44%) for `stress_0`. In contrast, as the testcases get increasingly more complex (`stress_1` and `stress_2`), our method performs far worse, especially in D_{drv} for both the testcases, and S_{rcv} for `stress_1` specifically. In the case of `stress_1`, the slew values are propagated across a long chain of π-models (up to 25) and they degrade rapidly as each consecutive approximation in each π-model adds a slight error to S_{rcv}. The loss of accuracy in D_{drv}, which is also apparent for the rest of the testcases, is due to the fact that the non-linear effects of the driver waveform are not captured effectively. While the waveform total time is approximated well (i.e., S_{drv}), the timepoints required for the estimation of delay are not calculated as precisely.

Testcase	Mean RE against SPICE				Mean AE against SPICE			
	D_{drv}	S_{drv}	D_{rcv}	S_{rcv}	D_{drv}	S_{drv}	D_{rcv}	S_{rcv}
standard	5.36%	6.10%	5.30%	4.97%	5.1 ps	14.3 ps	5.2 ps	11.8 ps
stress_0	7.44%	6.27%	6.11%	4.66%	5.6 ps	10.6 ps	6.1 ps	7.6 ps
stress_1	10.07%	6.73%	6.54%	12.53%	4.0 ps	6.0 ps	6.5 ps	32.5 ps
stress_2	8.76%	5.46%	5.03%	4.91%	6.3 ps	10.5 ps	5.0 ps	9.4 ps
stress_3	5.31%	5.42%	4.80%	5.91%	2.5 ps	6.0 ps	4.7 ps	8.3 ps

Table 4.3: Mean RE/AE against SPICE of our methodology for different testcases

4.3 Runtime and memory results

The runtime and memory of the examined approaches are shown in Table 4.4. As can be seen, our approach has the largest runtime out of all the approaches. This increase in runtime is due to the non-linear operations required for the computation of C_{eff}, compared to the linear operations (only additions/multiplications) of the `O'Brien` and the `Moments` methods, during the forwards/backwards traversals in Algorithm 1. Also, the `Moments` method requires the moments of each node in the RC-tree to be computed only once, taking approximately 33% less time than the `O'Brien` method. Nevertheless, these simpler approaches do not capture the complexity of the interconnect or the non-linear effects of the driver/receivers as effectively as our methodology On the other hand, our approach is the most efficient in terms of memory (0.38 MB), since it does not require any saving of moments or admittance

parameters but instead calculates everything on-the-fly during the traversals of the RC-tree.

Method	Memory (MB)	Runtime (s)
O'Brien	2.08	2.80
Moments	0.67	4.19
Ours	0.38	8.16

Table 4.4: Runtime and memory of the examined approaches across all testcases

Chapter 5

Conclusion

In this work, we presented a novel iterative approach for gate-level stage timing estimation using CSMs and the concept of multiple slew and C_{eff} values. This iterative procedure uses PWL waveforms and closed-form formulas to propagate the slew/delay values across the stage in order to approximate the driver/receiver waveform. The procedure, initially propagates forward the driver waveform across the RC interconnect of the stage up to the receivers cells. Using that waveform, the receiver input pin capacitance is calculated and a backwards traversal is performed to recalculate the new C_{eff} values of the driver. Unlike previous works, this methodology considers the non-linearity of the driver waveform by using PWL ramps to approximate it, does not make simplistic assumptions like other approaches, and considers the gate-interconnect interdependence. Experimental results on 2.2 million <driver, interconnect, receiver(s)> stages implemented using the ASU ASAP 7 nm FinFET Predictive PDK indicate that our method exhibits good correlation with SPICE, achieving 6.71% for D_{drv}, 5.98% for S_{drv}, 5.40% for D_{rcv}, and 5.78% for S_{rcv}. Moreover, the proposed method is highly efficient, since it requires only 8.16 s and 0.38 MB of memory to complete the timing estimation for 2.2 million stages. This feature renders the approach very attractive for iterative timing-driven optimization steps in placement or early stages of routing.

Bibliography

[1] Andrew B Kahng, Jens Lienig, Igor L Markov, and Jin Hu. *VLSI Physical Design: From Graph Partitioning to Timing Closure*. Springer Science & Business Media, 2011.

[2] J. Bhasker and R. Chadha. *Static Timing Analysis for Nanometer Designs: A Practical Approach*. Springer, 2009th edition, 2009.

[3] R. Goering. What's Needed to "Fix" Timing Signoff? . In *Panel of the 50th Design Automation Conference (DAC)*, 2013.

[4] Dimitrios Garyfallou, Ioannis Tsiokanos, Nestor Evmorfopoulos, Georgios Stamoulis, and Georgios Karakonstantis. Accurate Estimation of Dynamic Timing Slacks using Event-Driven Simulation. In *Proc. of the 21st International Symposium on Quality Electronic Design (ISQED)*, pages 225–230, 2020.

[5] Synopsys - Composite Current Source (CCS). `http://www.opensourceliberty.org/ccspaper/`. Accessed: Sep. 15, 2021.

[6] Cadence - Effective Current Source Model (ECSM). `https://www.cadence.com/en_US/home/alliances/standards-and-languages/ecsm-library-format.html`. Accessed: Sep. 15, 2021.

[7] Jessica Qian, Satyamurthy Pullela, and Lawrence Pillage. Modeling the "Effective Capacitance" for the RC Interconnect of CMOS Gates. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 13(12):1526–1535, 1994.

[8] Ying Zhou, Zhuo Li, Rouwaida N Kanj, David A Papa, Sani Nassif, and Weiping Shi. A More Effective C_{eff} for Slew Estimation. In *Proc. of the International Conference on Integrated Circuit Design and Technology (ICICDT)*, pages 1–4, 2007.

[9] Soroush Abbaspour and Massoud Pedram. Calculating the effective capacitance for the RC interconnect in VDSM technologies. In *Proc. of the 2003 Asia and South Pacific Design Automation Conference, ASP-DAC '03, Kitakyushu, Japan, January 21-24, 2003*, pages 43–48, 2003.

[10] Dimitrios Garyfallou et al. Gate Delay Estimation With Library Compatible Current Source Models and Effective Capacitance. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 29(5):962–972, 2021.

[11] Dimitrios Garyfallou. Novel techniques for timing analysis of VLSI circuits in advanced technology nodes. *Ph.D. thesis*, 2021.

[12] Curtis L. Ratzlaff and Lawrence T. Pillage. RICE: rapid interconnect circuit evaluation using AWE. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 13(6):763–776, 1994.

[13] Synopsys - HSPICE®. `https://www.synopsys.com/verification/ams-verification/hspice.html`. Accessed: Sep. 15, 2021.

[14] D. Garyfallou, C. Antoniadis, N. Evmorfopoulos, and G. Stamoulis. A Sparsity-Aware MOR Methodology for Fast and Accurate Timing Analysis of VLSI Interconnects. In *Proc. of the 16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, pages 89–92, 2019.

[15] Rohini Gupta, Bogdan Tutuianu, and Lawrence T. Pileggi. The Elmore delay as a bound for RC trees with generalized input signals. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 16(1):95–104, 1997.

[16] Frank Liu, Chandramouli V. Kashyap, and Charles J. Alpert. A delay metric for RC circuits based on the Weibull distribution. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 23(3):443–447, 2004.

[17] Kanak Agarwal, Dennis Sylvester, and David T. Blaauw. Simple metrics for slew rate of RC circuits based on two circuit moments. In *Proc. of the 40th Design Automation Conference (DAC)*, pages 950–953, 2003.

[18] Halil B Bakoglu. *Circuits, Interconnections, and Packaging for VLSI.* Addison-Wesley Pub. Co., 1990.

[19] C.J. Alpert, F. Liu, C. Kashyap, and A. Devgan. Delay and slew metrics using the lognormal distribution. In *Proc. of the 40th Design Automation Conference (DAC)*, pages 382–385, 2003.

[20] Lawrence T Pillage and Ronald A Rohrer. Asymptotic Waveform Evaluation for Timing Analysis. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 9(4):352–366, 1990.

[21] Ruchir Puri, David S Kung, and Anthony D Drumm. Fast and accurate wire delay estimation for physical synthesis of large ASICs. In *Proc. of the 12th Great Lakes Symposium on VLSI (GLSVLSI)*, pages 30–36, 2002.

[22] Dimitrios Garyfallou, Anastasis Vagenas, Charalampos Antoniadis, Yehia Massoud, and George Stamoulis. Leveraging machine learning for gate-level timing estimation using current source models and effective capacitance. In *Proceedings of the Great Lakes Symposium on VLSI 2022*, page 77–83, 2022.

[23] Charalampos Antoniadis, Dimitrios Garyfallou, Nestor Evmorfopoulos, and Georgios Stamoulis. EVT-based worst case delay estimation under process variation. In *Proc. of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1333–1338, 2018.

[24] Synopsys - What is Static Timing Analysis (STA). `https://www.synopsys.com/glossary/what-is-static-timing-analysis.html`. Accessed: Mar. 31, 2022.

[25] Peter R. O'Brien and Thomas L. Savarino. Modeling the driving-point characteristic of resistive interconnect for accurate delay estimation. In *Proc. of the International Conference on Computer Aided Design (ICCAD)*, pages 512–515, 1989.

[26] Chandramouli Kashyap, C.J. Alpert, Frank Liu, and Anirudh Devgan. Closed-form expressions for extending step delay and slew metrics to ramp inputs for RC trees. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 23(4):24–31, 2004.

[27] TAU 2021 timing contest - Stage Delay Calculator using Current Source Models. https://sites.google.com/view/tau-contest-2021. Accessed: Sep. 15, 2021.

[28] Xyce - Parallel electronic simulator. Accessed: May 30, 2022. https://xyce.sandia.gov/.

[29] ASU - ASAP7 PDK. Accessed: Sep. 15, 2021. https://github.com/The-OpenROAD-Project/asap7/.