“Study of Timing Analysis of Digital Intergrated Circuits Under Process Varation”

Diploma Thesis Project

Athanasios G. Koltsidas

University of Thessaly
School of Enginering
Department of Electrical and Computer Enginering

Volos, July 2014

University of Thessaly
School of Engineering
Department of Electrical and Computer Engineering

"Study of Timing Analysis of Digital Integrated Circuits under Process Variation"

Diploma Thesis Project

Athanasios G. Koltsidas

Supervisors:

Georgios Stamoulis
Professor
University of Thessaly

Nestor Evmorfopoulos
Assistant Professor
University of Thessaly

Πανεπιστήμιο Θεσσαλίας
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών

«Μελέτη Ανάλυσης Χρονισμού Ολοκληρωμένων Ψηφιακών Κυκλωμάτων υπό την Επίδραση της Διακύμανσης Κατασκευαστικών Παραμέτρων»

Διπλωματική Εργασία

Αθανάσιος Γ. Κολτσίδας

Επιβλέποντες Καθηγητές:

Γεώργιος Σταμούλης
Καθηγητής
Πανεπιστήμιο Θεσσαλίας

Νέστωρ Ευμορφόπουλος
Επίκουρος Καθηγητής
Πανεπιστήμιο Θεσσαλίας

Εγκρίθηκε από τη διμελή εξεταστική επιτροπή την 11/7/2014

....	....
Γεώργιος Σταμούλης	Νέστωρ Ευμορφόπουλος
Καθηγητής	Επίκουρος Καθηγητής

....

Αθανάσιος Κολτσίδας

Διπλωματούχος Μηχανικός Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας

Table of Contents

List of Figures

List of Acronyms

AT	Arrival Time
BDD	Binary Decision Diagram
BFS	Breadth-First Search
CDF	Cumulative Distribution Function
CDT	C/C++ Development Tooling
CPU	Central Process Unit
DAG	Directed Acyclic Graph
DFS	Depth-First Search
DSTA	Deterministic Static Timing Analysis
EDA	Electronic Design Automation
FiFo	First in, First out
GCC	GNU Compiler Collection
IC	Integrated Circuit
LTS	Long Term Support
PDF	Probability Density Function
RAT	Required Arrival Time
RC	Resistance Capacitance
RV	Random Variable
SSTA	Statistical Static Timing Analysis
STA	Static Timing Analysis
VLSI	Very-Large-Scale Integration

List of Algorithms

Acknowledgments

As this long journey as an undergraduate student draws to an end, I would like to seize the opportunity and acknowledge the people who helped me and supported me throughout this process and made my academic years in Volos interesting, illuminating but also quite joyful.

First of all, I would like to thank my advisor, Professor George Stamoulis for his encouragement, support and insight. Through our perennial relationship since I was a freshman, I obtained knowledge not only related to the field of Computer Science, but also the wisdoms of life. The blessing, help and guidance provided by him from time to time shall carry me a long way in the journey of life on which I am about to embark.

My deepest gratitude is due to the co-advisor, Assistant Professor Nestoras Evmorfopoulos, without whose knowledge and assistance this thesis project would not have been successful.

Furthermore, I am so grateful to all the graduate students in the VEDA Labs, for making the group a dream-place to work. Special thanks are in line to Babis Antoniadis and Panagiotis Giannakou who stood by me throughout my undergraduate studies not only as invaluable contributors but also as precious friends. Last but not least, I am deeply thankful to Konstantis Daloukas for his tireless aid and for his kindness and tolerance throughout our coexistence in VEDA Labs. He became during the last years of my studies one of the closest collaborators as well as a trusted and consistent friend.

I would also like to express my truehearted appreciation to my closest friends in life, Michalis, George, Noah and Giorgos for their love and encouragement. My life would be much less joyful and interesting without her, so I owe to my person, Marina, all my gratefulness for providing the smile and warmth to my everyday life and I am truthfully thankful for that.

Finally, I would like to thank my parents, my sister and my brother, for their uninterrupted support and trust, their loyalty, their unconditional love and most of all their patience. I would be nothing without them and I will always keep them in the most special place in my heart. My eternal gratitude is due to my parents for providing me everything and for being present in all the important events of my life. I own a lot of thanks especially to my father, who triggered my competitive nature from early in my life and condemned me never to settle, to endlessly strive to excel in all the aspects of my life. Lastly, I could not even express in words all the sacrifices my mother had to make in personal time and hard work in order to be successful in her profession, but also as a mother and wife. She will always

be a role model for me for her hard-working nature and her resourcefulness.

Ευχαριστίες

Καθώς αυτό το μακρύ ταξίδι ως προπτυχιακός φοιτητής φτάνει στο τέλος του, δράττομαι της ευκαιρίας να ευχαριστήσω τους ανθρώπους που με στήριξαν και με βοήθησαν καθ’ όλη τη διάρκεια αυτής της διαδικασίας και γέμισαν τα χρόνια μου στο Βόλο με χαρούμενες και ενδιαφέρουσες στιγμές κι εμπειρίες.

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα της διπλωματικής μου, Καθηγητή κ. Γιώργο Σταμούλη για την υποστήριξη και την αρωγή του. Μέσα από τη μακρόχρονη επαφή μαζί του από το πρώτο έτος των σπουδών μου, απέκτησα γνώσεις τόσο σχετιζόμενες με την Επιστήμη των Υπολογιστών καθώς και εμπειρίες σχετιζόμενες με τη ζωή εν γένει. Η βοήθειά και η καθοδήγηση που μου παρείχε κατά καιρούς στη φοιτητική μου ζωή είναι εφόδια που θα με συνοδεύουν στην μετέπειτα πορεία της ζωής μου.

Θα ήθελα επίσης να ευχαριστήσω από τα βάθη της καρδιάς μου τον δεύτερο επιβλέποντα της διπλωματικής μου, Επίκουρο Καθηγητή κ. Νέστορα Ευμορφόπουλο, χωρίς τις γνώσεις και τη βοήθεια του οποίου η διπλωματική εργασία αυτή δε θα μπορούσε να στεφθεί με επιτυχία.

Επιπροσθέτως, δε θα μπορούσα να μην εκφράσω την ευγνωμοσύνη μου προς όλους τους μεταπτυχιακούς και διδακτορικούς φοιτητές στο Εργαστήριο Ηλεκτρονικής, οι οποίοι κατέστησαν το εργαστήριό μας το ιδανικό εργασιακό περιβάλλον. Ειδικότερα, θα ήθελα να ευχαριστήσω τους μεταπτυχιακούς φοιτητές Μπάμπη Αντωνιάδη και Παναγιώτη Γιαννακού, οι οποίοι στάθηκαν δίπλα μου ως ανεκτίμητης αξίας συνεργάτες και φίλοι. Επίσης, να εκφράσω τις ευχαριστίες μου και στον διδάκτορα του Τμήματος Ηλεκτρολόγων Μηχανικών & Μηχανικών Η/Υ Κωνσταντή Νταλούκα για την αδιάκοπη βοήθεια, την ευγένεια και την ανοχή του καθ’ όλη τη διάρκεια της συνύπαρξής μας στο εργαστήριο. Μετά το πέρας των σπουδών μου νιώθω ότι πέρα από έναν στενό συνεργάτη, κέρδισα έναν καλό κι έμπιστο φίλο.

Θα ήθελα επίσης να πω ένα τεράστιο και ειλικρινές ευχαριστώ στους φίλους μου, το Μιχάλη, το Γιώργο, το Νώε και το Γιώργο, για την αγάπη και την συμπαράστασή τους όλα αυτά τα χρόνια. Η ζωή μου θα ήταν πολύ λιγότερο ενδιαφέρουσα και ευχάριστη εάν δεν είχαμε συναντηθεί, χρωστάω λοιπόν στον άνθρωπό μου, στη Μαρίνα, την αναγνώριση για όλα όσα μου παρείχε, το χαμόγελο και την κατανόηση στην καθημερινότητά μου, πράγματα που τόσο πολύ χρειαζόμουνα και γι’ αυτό την ευχαριστώ.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου, τα αδέρφια μου και τους γονείς μου, για τη συνεπή στήριξη και την εμπιστοσύνη τους, την

πίστη τους σε εμένα, την άνευ όρων αγάπη τους και κυρίως την υπομονή τους. Χωρίς αυτούς δε θα ήμουνα τίποτα και για αυτό θα έχουν πάντα μια ξεχωριστή θέση στην καρδιά μου. Χρωστάω αιώνια ευγνωμοσύνη στους γονείς μου, οι οποίοι μου παρείχαν τα πάντα, βρισκόντουσαν εκεί σε όλα τα σημαντικά γεγονότα της ζωής μου. Ειδική μνεία οφείλω στον πατέρα μου Γιώργο, ο οποίος προκαλούσε πάντα την ανταγωνιστικότητά μου, γεγονός που με οδήγησε στο να μη συμβιβάζομαι ποτέ και να προσπαθώ αδιάκοπα για το καλύτερο σε όλους τους τομείς της ζωής μου. Κράτησα για το τέλος το δυσκολότερο, καθώς δεν υπάρχουν λόγια για να περιγράψω τις θυσίες και τη σκληρή δουλειά που η μητέρα μου Αναστασία, κατέβαλε ώστε να καταφέρει να επιτύχει σε όλους τους ρόλους που είχε αναλάβει, τόσο στο επάγγελμά της όσο και ως μητέρα και σύζυγος. Θα είναι για πάντα ένα παράδειγμα προς μίμηση για τη σκληρή δουλειά και την αυταπάρνηση που τη διακρίνει.

Abstract

Timing Analysis is an integral part of any integrated circuit (IC) chip design-closure flow, and is utilized at several stages of the flow, including pre/post-route timing optimization and timing signoff. Even though accurate timing analysis is important, at the same time the run-time of the analysis is evenly crucial with growing chip design sizes and complexity (for example, growing number of clocks domains, voltage islands etc.). Furthermore, the rising importance of variability in the chip manufacturing process along with environmental variability, demands the use of variation aware techniques for chip timing analysis which significantly affects the analysis run-time.

In this diploma thesis project, we study the process variation parameters that impact the process of timing analysis, then we categorize and review some previous approaches in variation aware timing analysis. Next in line, we present the models and the implementation stages of a tool we developed for timing analysis under process variations. The innovative element of our approach is the extensive use of a BFS-like algorithm during the stage of arc-delay-information propagation as well as the use of DFS algorithm through the interconnect delay calculation phase, in order to minimize the iterations that were necessary so as to discover all the paths included in the specific net. Finally, we evaluate the results of our approach, by estimating the yield on certain nodes of the benchmark netlists, utilizing their PDF and CDF. Another evaluation metric we used to verify the validity of our implementation, was the examination for slack constraints violations by applying a worst-case slack estimation based on the slack information we gathered.

Keywords: Timing Analysis; Variability; Process Variations; Timing Graph; Yield Estimation; Slack Violation.

Περίληψη

Η *Ανάλυση Χρονισμού* αποτελεί αναπόσπαστο κομμάτι της διαδικασίας σχεδίασης οποιουδήποτε ολοκληρωμένου κυκλώματος και χρησιμοποιείται μάλιστα σε αρκετά από τα στάδια της σχεδίασης. Παρόλο που μια ακριβής ανάλυση χρονισμού είναι ζωτικής σημασίας, ταυτόχρονα ο χρόνος εκτέλεσης της ανάλυσης είναι εξίσου σημαντικό στοιχείο, δεδομένου του συνεχώς αυξανόμενου μεγέθους και της πολυπλοκότητας των αρχιτεκτονικών των ολοκληρωμένων κυκλωμάτων. Επιπλέον, η αυξανόμενη σημασία της μεταβλητότητας στη διαδικασία παραγωγής ολοκληρωμένων κυκλωμάτων σε συνδυασμό με την μεταβλητότητα περιβάλλοντος, απαιτούν τη χρήση τεχνικών ανάλυσης χρονισμού που θα λαμβάνουν υπόψη τη διακύμανση των παραμέτρων, η οποία επηρεάζει σημαντικά το χρόνο εκτέλεσης της ανάλυσης.

Στη συγκεκριμένη διπλωματική εργασία, μελετούμε τις παραμέτρους οι οποίες επηρεάζουν το στάδιο της ανάλυσης χρονισμού, κατηγοριοποιούμε και στη συνέχεια εξετάζουμε μερικά παραδείγματα πρότερων εργασιών στο συγκεκριμένο αντικείμενο έρευνας. Στη συνέχεια, παρουσιάζουμε τα μοντέλα και τα στάδια της υλοποίησης του εργαλείου που υλοποιήσαμε. Το πρωτότυπο στοιχείο της προσέγγισής μας είναι η εκτενής χρήση του αλγορίθμου της θεωρίας γράφων Αναζήτησης κατά Πλάτος (BFS) κατά το στάδιο της μετάδοσης της πληροφορίας για την καθυστέρηση των στοιχείων του κυκλώματος, καθώς επίσης και η χρήση του αλγορίθμου Αναζήτησης κατά Βάθος (DFS) στο στάδιο του υπολογισμού των καθυστερήσεων καλωδίωσης του κυκλώματος. Τέλος, αξιολογούμε τα αποτελέσματα της επίδοσης του εργαλείου που υλοποιήσαμε επικεντρώνοντας στο yield που παρουσιάζουν συγκεκριμένοι κόμβοι που εξετάσαμε από τα netlists, εφαρμόζοντας στις πληροφορίες καθυστέρησης που λάβαμε κατά την ανάλυσή μας τη συνάρτηση πυκνότητας πιθανότητας καθώς και την αθροιστική συνάρτηση κατανομής.

To my family & my friends

1. Introduction

Timing analysis, within ***Electronic Design Automation*** (**EDA**), refers to the process of determining timing information as signal transitions propagate throughout a digital circuit, commonly described by a netlist of circuit elements. Every signal transition that arrives at the input of a circuit element will be available at its outputs some time later. Thus, each element introduces a ***delay*** on signal transition propagation. Moreover, we assume that signal transitions are defined by a ***slew***. Circuit elements alter the signal transitions at their inputs by modifying their slew when shown at the outputs. A graphical representation of the terms ***delay*** and ***slew*** are displayed in ***Figure 1***.

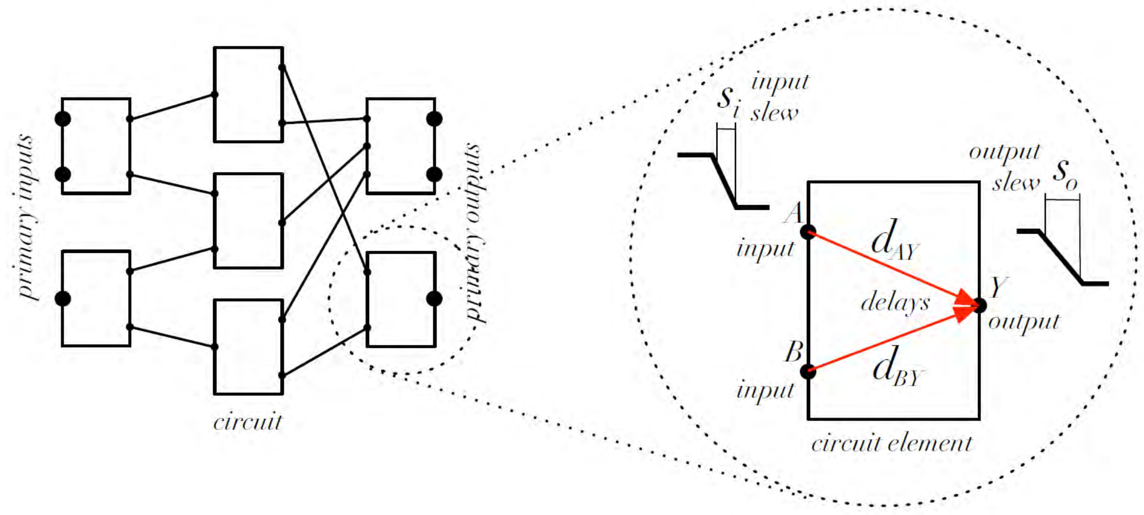


Figure 1: Circuit and circuit element characterization

Arrival times (at), represent the earliest or the latest moment in time that a signal transition is about to reach a specific node in the circuit, travelling from a circuit input. The meaning of the arrival time depends on whether we consider the ***early*** or the ***late*** mode. In early mode, we are interested in the earliest moment that a signal transition can reach any given circuit node. Vice versa, in late mode we are interested in the latest moment that a signal transition can reach any given circuit node. As a result, arrival times are determined by adding edge delays throughout a specific path and computing the min or max (depending on the mode of operation we choose - early or late) of such delays when they converge at a certain circuit node. For instance, assuming at_A^{early} and at_B^{early} to symbolize the early arrival times at pins A and B respectively of the circuit element illustrated in ***Figure 1***, then the early arrival time at the output pin Y is calculated as follows:

$$at_Y^{early} = \min\left(at_A^{early} + d_{AY}, at_B^{early} + d_{BY}\right) \quad (1) \ .$$

On the other hand, the late arrival time at the output pin Y will be:

$$at_Y^{late} = \max\left(at_A^{late} + d_{AY}, at_B^{late} + d_{BY}\right) \quad (2) \ .$$

Required arrival times **(rat), are limits applied on the arrival times in specified nodes of the circuit. These kind of limits are usually imposed in order to secure appropriate circuit operation. Again assuming either early or late mode, when a rat is defined for a specific circuit node, the following restrictions must be valid:**

$$at^{early} \geq rat^{early} \quad (3)$$

$$at^{late} \leq rat^{late} \quad (4)$$

Slacks **($slack$) are the disparity between arrival times and required arrival times, and estimate how well the constraints of rat are met.**

$$slack^{early} = at^{early} - rat^{early} \quad (5)$$

$$slack^{late} = rat^{early} - at^{late} \quad (6)$$

Slacks are positive in the event the required arrival time constraints are met and negative in other case.

Slew **(s_o) propagation is also vital for timing analysis, as cell and interconnect delays are a function of the input slew. We will assume worst-slew propagation, which means that we propagate either the smallest or the largest slew, when we examine either early or late mode.**

$$s_{o\,Y}^{early} = \min\left(s_{o\,AY}^{early}\left(s_{i\,A}^{early} \right), s_{o\,BY}^{early}\left(s_{i\,B}^{early} \right)\right) \quad (7)$$

$$s_{o\,Y}^{late} = max\left(s_{o\,AY}^{late}\left(s_{i\,A}^{late} \right), s_{o\,BY}^{late}\left(s_{i\,B}^{late} \right)\right) \quad (8)$$

We should point out that slew propagates regardless of delay propagation: we can propagate, for instance the delay from input A and slew from input B (*Figure 1*).

From the early 1990s, *Static Timing Analysis* (STA) has been widely adopted as a common tool in the process of *very-large-scale-integration* (VLSI) design. Static timing analysis is not only the universal timing tool but also lies at the core of numerous timing optimization tools. The main advantage of STA over vector-based timing simulation is the fact that it does not rely

on input vectors, which can be difficult to construct and can easily miss an obscure path in the circuit. The extensive use of **STA** can be associated with various factors:

- The basic **STA** algorithm is linear time with circuit size, allowing analysis of designs in excess of 10 million instances;
- The basic **STA** is conservative in the sense that it will overestimate the delay of long paths in the circuit and underestimate the delay of short paths accordingly. This provides a "safe" analysis, guaranteeing that the design will function *at least* as fast as predicted and will not suffer from hold-time violations; and
- The **STA** algorithms have matured over time, addressing crucial timing issues such as interconnect analysis and accurate delay modelling.

Conventional **STA** tools are deterministic (**STA** is often called **DSTA** – *deterministic static timing analysis*) and compute the circuit delay for a *specific* process condition. Hence, all parameters that affect the delay of a circuit, such as device gate length and oxide thickness, as well as operating voltage and temperature, are presumed to be fixed and they are uniformly applied throughout the devices in the design.

In **DSTA**, process variation is modeled by running the analysis multiple times, each at a different process condition. Therefore, by analyzing an adequate number of process conditions the delay of the circuit under process variation can be confined.

The fundamental weakness of **STA** is that while shifts in the process (referred to as *die-to-die* variations) can be approximated by creating multiple corner files, there is no statistically strict method for modeling variations across a die (referred to as *within-die* variations). Despite that, with process scaling progressing well into the nanometer status quo, process variations have become significantly more distinct and within-die variations became a non-negligible component of the total variation. As a result, the clear inability of **STA** to model within-die variation may result either in an over – or underestimate of the circuit delay, relying solely on the circuit topology. Consequently, **STA**'s desirable quality of being conservative might no longer hold for specific circuit topologies while, at the same time, **STA** may be overly pessimistic for others. For that reason, **STA**'s accuracy in advanced processes is a major concern.

In addition to the increasing importance of within-die process variations, the total number of process parameters that exhibit significant variation has also increased [1]. Hence, even the modeling of only die-to-die variations in **STA** now requires an unsustainable number of corner files, which could lead to increase the effective runtime of **STA** by one order of magnitude or more.

1.1. Problem Description and Challenges

Problem Description

Traditional **STA** methods extract a ***timing graph*** from a circuit, as shown in ***Figure 2***. The nodes of the graph represent primary inputs/outputs of the circuit as well as input/output pins of the circuit's gates. Its edges represent timing elements of the circuit such as the gate input-pin-output-pin delay and wire delay from one node to the ones adjacent in the timing graph. Device parameters, like metal thickness and gate length must be treated as ***random variables*** (**RVs**) as a result of process variation. Thus, the delay of each edge, since it is a function of these parameters, turns out to be an **RV** too. This induction grants the transformation of the traditional **STA** timing graph into a statistical timing graph which is described as follows:

Definition: A timing graph $G = \{N, E\}$ is a directed acyclic graph (**DAG**), where N is a set of nodes and E is a set of edges. The weight associated with an edge reciprocates to either the gate or the interconnect delay. The timing graph is said to be a ***statistical timing graph*** if the $i-th$ edge weight d_i is an **RV**.

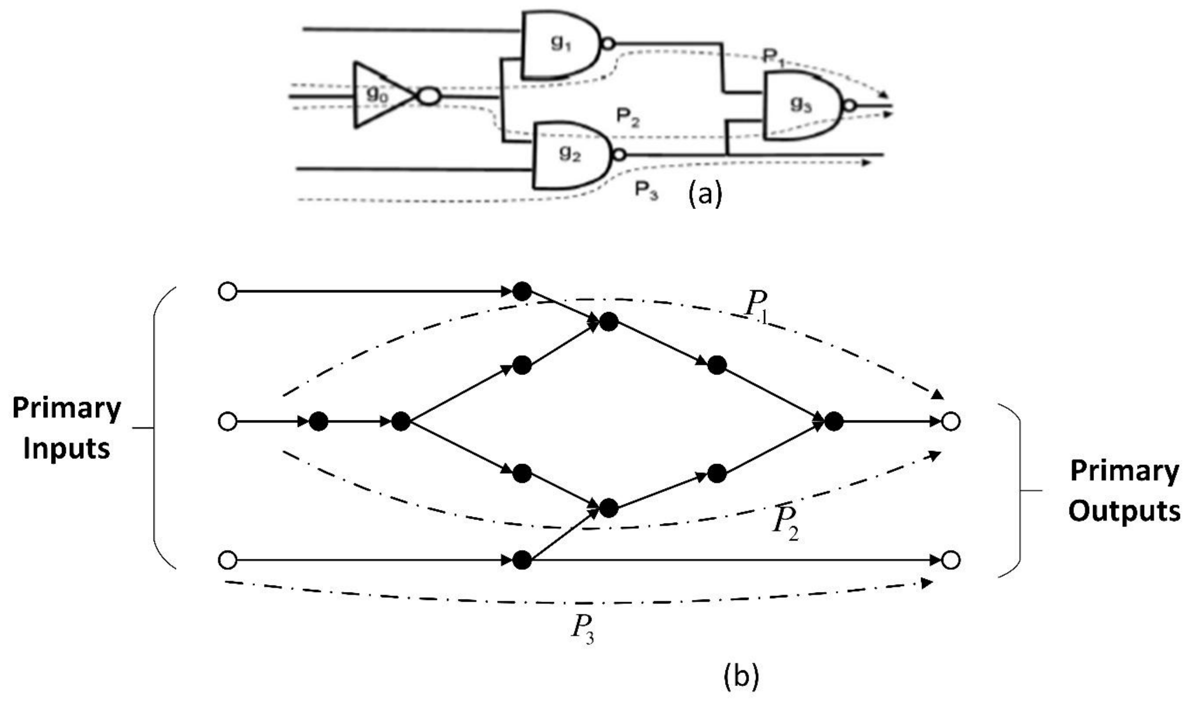


Figure 2: Example circuit (a) and its timing graph (b)

The arrival times at the source nodes of the timing graph (primary inputs of the circuit), typically have a deterministic zero value. In STA, the fundamental goal of the analysis is to locate the critical path (path with the maximum delay between a primary input node and a primary output node in the graph).

When modelling process-induced delay variations, the sample space is set of all manufactured dies, in which case the device parameters will have different values across the specific sample space and as a result the *critical path* (and its delay) will vary from one die to another. Consequently, the delay of the circuit is also an RV, and the primary task of *Statistical Static Timing Analysis* is to estimate the characteristics of this RV. This is achieved by computing its *probability-distribution function* (PDF) or *cumulative-distribution function* (CDF) (*Figure 3*). At this point, we should remind that the CDF and the PDF can be derived from each other through differentiation and integration [2].

Definition: **Let a path p_i be a set of ordered edges from the primary input nodes to the primary output nodes in G, and let D_i be the path-length distribution of p_i, computed as the** *sum* **of the weights d for all edges k on the path. Finding the distribution of $D_{max} = \max(D_1, D_2, \dots, D_i, D_{n-paths})$ among all paths (indexed from 1 to n paths) in the graph G is referred to as the *statistical static timing analysis* (SSTA) problem of a given circuit.**

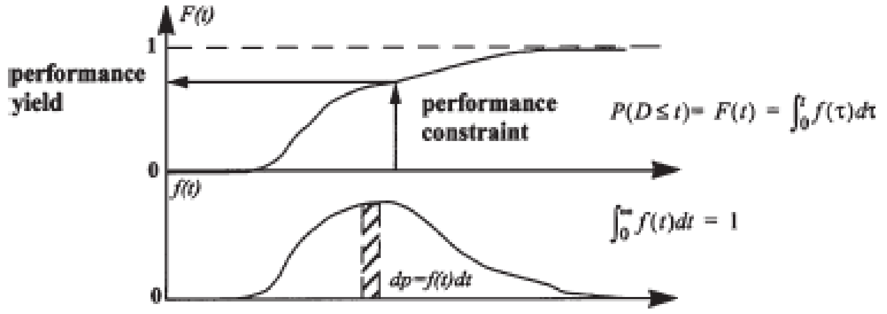

F(t)
1
performance
yield
performance
constraint
$P(D \leq t) = F(t) = \int_0^t f(\tau)d\tau$
0
f(t)
$\int_0^\infty f(t)dt = 1$
dp=f(t)dt
0

Figure 3: PDF and CDF

Similar again to traditional STA, the SSTA problem can be formulated as the procedure of finding the latest arrival-time distribution at any of the primary output nodes in G. The latest *at* **distribution at the primary output nodes can be calculated by propagating the** *at* **from the primary input nodes through the timing edges of the graph, while at the same time we compute the latest arrival time at every node in topological order. As a result of this process, the latest arrival-time distribution at any of the primary output nodes symbolizes the circuit-delay distribution.**

Despite the problem of finding the delay of the circuit, which we have suggested as the primary **SSTA** problem, statistical approach in timing analysis is also an answer to the problem of improving the delay in the event that timing requirements are not met.

Conventional **STA** approaches usually report the *slack* at each node in the graph, besides the circuit delay and critical paths. As a reminder to what we already mentioned during the introduction, the slack associated with each node, is the difference between the latest a signal can arrive at the specific node, such that the timing constraints of the circuit are satisfied. Hence, correspondingly to the circuit delay, the slack of a node is formulated as an **RV** in **SSTA** methodology.

Challenges in SSTA

While **SSTA** has proven to be quite useful in the task of handling properly and effectively process variation parameters in comparison to traditional **STA**, the statistical formulation of timing analysis introduced various novel modeling and algorithmic issues that make **SSTA** a complex as well as durable topic for research. In this subsection, we present some of these issues along with some related terminology.

1) *Topological Correlation*

Paths that start with one or more shared edges after which paths separate and join at a later node are called ***reconvergent paths***. The node at which these paths reconverge is called the ***reconvergent node***. For example, in ***Figure 2***, the two paths P_1 and P_2 share the same first edge and reconverge at the output of gate g_3. In a case like that, the arrival times at the reconvergent node become dependent on each other because of the common's edge delay. This specific dependence leads to so-called ***topological correlation*** between the arrival times and complicates the maximum operation at the reconvergent node. The challenge here for **SSTA** methods is to capture and propagate this correlation in order to be properly accounted for during the computation of the *max* function.

2) *Spatial Correlation*

Within-die variation of the physical device parameters usually exposes *spatial correlation*, triggering correlation between the gate delays. Therefore, if the gates that involve two paths have spatially correlated device parameters they will consequently have correlated path delays. Thus, correlation is possible to be introduced amongst paths that do not share timing edges. For instance, again in ***Figure 3***, the paths P_1 and P_3 do not share any common delay edges, nevertheless if gates g_1 and g_2 are within close proximity on the die, their spatially correlated delays can arise correlation between the two path delays. Hence, spatial

correlation (as topological) of the arrival times must be captured and propagated during SSTA so that it is correctly calculated for the *max* function.

While topological correlation impacts solely the *maximum* operation, spatial correlation influences both the *sum* operation as well as the *maximum* operation. This brings up two fundamental issues for SSTA approaches:

- **How to form gate delays and arrival times so that spatial correlation of the underlying device parameters can be formulated**
- **Granted a model of the spatial correlation, how to propagate and keep the correlation information at the same time that performing the *sum* and *maximum* operations.**

3) *Non-Normal Process Parameters & Nonlinear Delay Models*

Normal or Gaussian are undoubtedly the most broadly observed distributions for random variables and numerous elegant analytical results are presented in the statistics literature. As a consequence, most of the primary works published in the field of SSTA adopted normal distributions to model physical device parameters, electrical device parameters, gate delays as well as arrival times. Nevertheless, some physical device parameters may present significantly non-normal distributions.

Even if the parameters are indeed normally distributed, the dependence between the electrical device parameters and the gate delay on these physical parameters may as well be non-linear, causing non-normal gate delays to rise. Original work in modeling spatial correlations in [3] – [5], presented a delay model that assumed a linear dependence of the gate delay on physical device parameters. In case the variations are insignificant, this linear approximation is proven right, as the error introduced by overlooking higher order terms is negligible. Nevertheless, with reduction of geometries, process variation is becoming more notable and the linear approximation may not be precise for some parameters.

Both non-normal delay and arrival time distributions set forth vital challenges for efficient SSTA. While this may be considered as a novel area of research, several members of SSTA research community have suggested approaches to address this issue [6], [7] and [8]. At this point, we should note that besides the difficulty of modeling, the non-normality of an individual RV, the dependence between two non-normal RVs is no longer expressed through a sole correlation factor. This complicates

even more the proper handling of both topological and spatial correlations.

4) *Skewness as a result of Maximum Operation*

Even supposing that gate delays are normal, **SSTA** has to deal with the fact that *maximum* operation is an inherently nonlinear function. The maximum of two normal arrival times will result in a non-normal arrival time that is typically positively *skewed*. Moreover, the non-normal arrival time distribution generated at one node is the input to the *maximum* computation at downstream nodes. Hence, we need a *maximum* operation that is able to operate even in non-normal arrival times.

In probability theory, *skewness* is a measure of the asymmetry of the probability distribution of a real-valued **RV** about its mean. The skewness value can be positive, negative, zero or even undefined. For further insight, the reader is encouraged to go through [9].

The majority of existing works overlook the skewness introduced by the *maximum* operation and calculate the arrival times with normal distributions. The fault in this approximation grows larger in case the input arrival times have comparable means and dissimilar variances [10]. Namely, the error is more distinct when two converging paths have formally balanced path delays, but one of them has a tighter delay distribution in comparison to the other. This is possible to happen in a circuit when two paths with equal nominal delay consist of a different number of gates, or when the correlation among their gates varies. Another example, is when one path is dominated by interconnect delay while the other is dominated by gate delay.

To conclude, it is safe to say that the aforementioned problems shape four basic challenges in **SSTA**, which have received massive attention in the literature. Despite that, numerous other crucial challenges to reach the development of a mature **SSTA** tool remain standing.

1.2. Previous Work

To provide an insight of previous attempts in the field of **SSTA**, we demonstrate a few examples both from earlier approaches as well as some state-of-the-art research works. As it is pointed out in [1], there have been noted original works related to **SSTA** that date back to the 1960s [11], the period of introduction of timing analysis itself, but also later, in the early 1990s, [12] – [14].

In [14], the authors proposed a method for performing statistical timing analysis which involves structural Boolean properties of a combinational circuit. The approach suggested the use of a **PDF** and encoded the delay as well as the logic behavior of the circuit into a Boolean expression that was afterwards simplified using a *binary decision diagram* (**BDD**) representation. Despite the fact that the results of this technique were only shown in small circuits, a remarkable observation was related to the computation of the signal probability (i.e., the probability that the signal is at logic 1) at the output of a gate.

The approach in [12] suggested a symbolic simulation procedure for statistical timing analysis. A particularly notable contribution that has since been used in other work is the idea of using interval analysis to generate trimming strategies in order to remove paths that can never be (or exhibit a very low possibility of being) critical. Unlike [14], this method demonstrated results on large benchmark circuits, although under interval-based delay models.

Since these premature efforts were published, **SSTA** approaches have evolved and from early 2000s the community of **SSTA** researchers has grown immensely. Most of the research work related to **SSTA** date from the last decade, with well over a hundred papers published in this research field from 2001 to 2008. This becomes more obvious when the numbers are invoked: the vast majority of research work on **SSTA** date from the last decade, with well over a hundred papers published since 2001 [1]. Some representative examples of state–of–the–art approaches that solve the **SSTA** problem include [3] – [5] and [15] – [20]. The methods used in these works can be classified according to the categorization that is provided in *Chapter 2*. Authors in [16], [17] and [20] suggest techniques to bound the delay distributions rather than calculate the exact distributions using path–based and block–based methods. In [18] an approximation approach is proposed, which is based on a generic path analysis rather than evaluating every path statistically.

1.3. Outline

In this thesis project, we provide a general idea about **SSTA** and exhibit some representative works from various approaches. Furthermore, we unfold its role in today's design flow process, and consequently we focus more on *Variation Aware Timing Analysis* by giving a description of our own approach to **SSTA** and the challenges we coped with.

In *Chapter 3*, we give an explanation of the models used in the context of the development of our tool, as well as a rough image of the representation.

In *Chapter 4*, we break down the stages of the implementation and give a brief description of the procedures taking place during each step.

Finally, in *Chapter 5*, we present the experimental results coupled with a short evaluation and also some statistical metrics that we based our conclusion on. Moreover, we try to conceive the progress that might take place in the field of **SSTA** and the directions that the research community might head towards.

2. Timing Analysis

2.1. STA in Design Flow

As we mentioned numerous times throughout the introductive chapter, **STA** is the predecessor of any **SSTA** research effort. Modern approaches that involve statistical methods are simply attempting to ameliorate the precision of timing analysis through the design flow stages that it is required.

Timing analysis in general, and **STA** more specifically, plays a crucial part in modern design–closure flow. In addition to determining the longest and shortest timing propagation paths, **STA** (and as a result **SSTA**) can be used to compute arrival times, required arrival times and slacks at all the points of the circuit. Hence, every design stage (***Figure 4***), from floorplaning, logical synthesis to placement and routing, employs timing analysis in order to assess circuit performance, and afterwards modify the design accordingly.

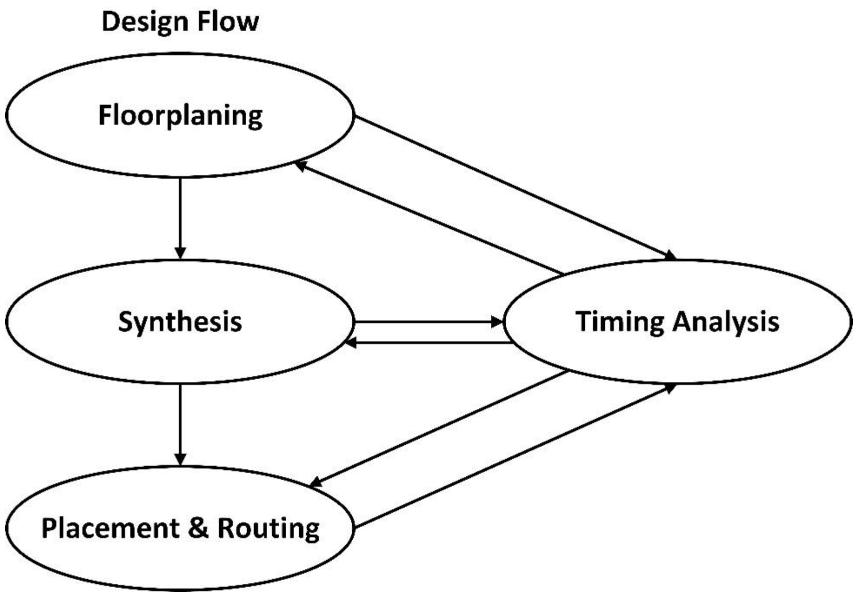


Figure 4: Timing analysis in the design flow

Most modern–day companies specialized in **EDA**, develop engines, such as Prime Time of Synopsys™ [21] and Encounter Timing System of Cadence™ [22]. These timing engines are utilized throughout the synthesis/place–and–route flow.

With that being said, it becomes crystal clear that an efficient timing analysis method is the key to a successful design process. Widely used **STA** approaches, provide limited accuracy as a result of the use of simplified delay models. The lack of attention regarding the function of the circuit, establishes **STA** as an error prone approach. For instance, the inclusion of false paths (paths that are not logically existing in the circuit) jeopardize the optimization process which follows the timing analysis, as it might illogically focus on false paths and neglect the real critical path. Another known issue with traditional **STA** methods, is their conservative nature, which naturally leads to over-design and thus to an increase in the product cost. Taking into consideration all the facts that we mentioned above, the improvement of the accuracy of the timing analysis stage was accomplished by turning to **SSTA** methods. Despite that, it is now the research community's responsibility not to let the efficiency deteriorate for the sake of accuracy.

2.2. Variation Aware Timing Analysis

In accordance to the aforementioned, with the technology of semiconductors shrinking under 65 nanometers, the need for an efficient modeling of process variations throughout a **VLSI** chip manufacturing process, has led to extensive research in *Variation Aware Timing Analysis*. Manufacturing sources of variability include device front-end variability (e.g. variations in channel length, oxide thickness etc.) and back-end-of-line variability (such as metal variability). Moreover, environmental sources of variation like voltage and temperature strongly impact circuit timing. Variability may be classified into different categories like intra-chip variability and inter-chip variability. Each of these can be further sub-classified as systematic and random variability (*Figure 7*). In general, sources of variation that impact circuit timing are termed *parameters*. An easy way to understand previous work done in the field of statistical timing analysis is to try to classify roughly the techniques proposed so far in recent works:

I. *Numerical Integration Methods*

The most straightforward **SSTA** approach results immediately from the problem definition we provided in the previous section. Basically, a numerical integration over the process parameter space is applied to estimate the yield of the circuit for a specific delay. Usually, we express the delay of a set of critical paths as a linear function of the physical device parameters. As a result, an attainable region in parameter space is specified from the desired delay of the circuit. Later, this region we specified is numerically integrated, examining any possible permutations of physical device parameter values located in the specified region. Efficient numerical-integration methods were proposed in [23]. The key advantage of this approach is that it is entirely generic and as a result process variation with any type of distribution and correlation can be represented. Nevertheless, there is a possibility that this type of approaches can be quite costly in run-time and especially for more balanced circuits that consist of multiple critical paths.

II. *Monte Carlo Methods*

The second general approach, which is also the one used on this study, performs a statistical sampling of the sample space using *Monte Carlo* simulation. The basic idea is to determine the regions with significant probability and to sample adequately these regions. By using the **PDF** of each physical device parameter, it is possible to extract a sufficient number of samples. Utilizing traditional **STA** methods makes it possible to calculate the circuit delay by employing the **PDF** of the physical device parameters. From there on, an estimation of timing yield is acquired, by evaluating a portion of samples that meet the timing constraint. If an acceptable number of samples is drawn, then the prediction error is small. After that, it is feasible to determine the delay distribution for the entire circuit by sweeping the timing constraint and determining the yield for each value. Moreover, it has been observed that the performance of Monte Carlo techniques can be improved using methods like importance sampling [24], [25].

Similarly to numerical integration methods, the Monte Carlo approach holds the advantage of being totally generic. Moreover, while based on existing **STA** traditional methods, Monte Carlo methods perform notably better than the numerical integration-based approaches. At the same time, it has been noted that Monte Carlo methods handle expertly the complexities of variations.

III. *Probabilistic Analysis Methods*

Both aforementioned approaches are based on sample-space enumeration, while probabilistic methods specifically model both gate delay and arrival times with random variables. These approaches, usually propagate arrival times through the timing graph, by performing statistical *sum* and *max/min* operations. It is viable to categorize this type of approaches in two wide-ranging groups:

1) ***Path-based approaches***: In path based algorithms, a group of circuit paths most likely to eventually be critical is determined and a statistical analysis is carried out over this set of paths to approximate the circuit delay distribution. Initially, we determine the delay distribution of each path by summing the delay of the path's edges. If assumed normal gate delays, the path-delay distribution can be analytically calculated, as presented in [19], [26] and [27]. Finally, a

statistical *maximum* operation is performed over all the path delays in order to find the overall circuit delay distribution.

The basic advantage of this approach is the fact that the analysis is distinctly divided into two parts – the computation of path delays coupled with the statistical *max* operation over these path delays. Thus, it is clear why many of the initial research attempts in **SSTA** were focused on path-based approaches [13], [19], [20], [26], [28] and [29]. However, the obvious obstruction with this approach is how to precisely determine the subset of candidate paths in a way that no path that displays a notable probability of being critical will be omitted. Moreover, when it comes to balanced circuits, the number of candidate paths under consideration can be very high. Hence, based on what we mentioned above, it is clear why most of the later research works have focused on block-based approach.

2) *Block-based approaches*: Block-based methods tend to follow more strictly the deterministic **STA** algorithm and traverse the circuit graph in a topological way. Arrival times at each node is then calculated based on two fundamental operations:

    i. For all fan-in edges of a specific node, the edge delay is added to the arrival-time at the source node of the edge using the *sum* operation, and

    ii. given the resulting arrival times, the concluding arrival time at the node is computed using the *maximum* operation.

    Thus, block–based methods propagate exactly two arrival times (a rise and a fall arrival time) at each circuit node, resulting in a runtime that is linear with the circuit size. The computation of the *sum* function is usually a straightforward process. Yet, determining the statistical maximum of two associated arrival times is not insignificant.

    As a result of its runtime advantage, many modern research and commercial works have chosen the block-based over other approaches.

2.3. Sources of Variation and Sensitivity

In this section we explore the main sources of variation through the process of timing prediction that makes timing analysis a demanding task when it comes to nanoscale digital circuits. We will refer firstly to the different types of uncertainties that rise as a design goes from specification to implementation and final field operation. Nevertheless, we will focus on process variations and specifically on the distinction between die-to-die and within-

die variations. Lastly, we will address the impact that different types of process variations have on the timing quantities of a circuit.

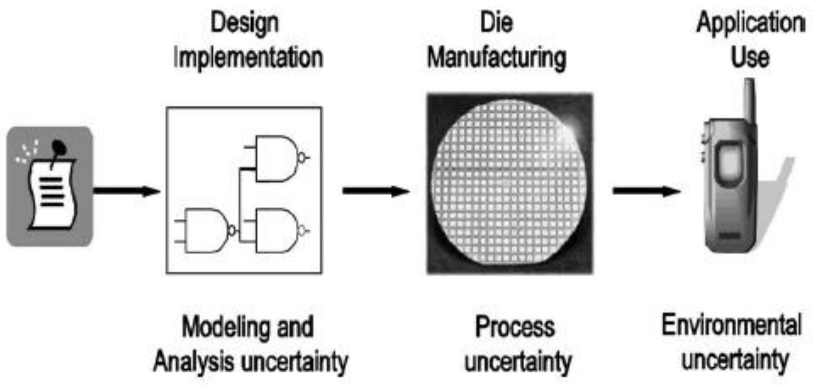


Figure 5: Steps of the design procedure and the resulting timing uncertainties

The unpredictability in the timing estimation of a design lies into three main categories:

I. ***modeling and analysis errors*** **- inaccuracy in device models in extraction and reduction of interconnect parasitics as well as in timing analysis algorithms,**

II. ***manufacturing variations*** **- uncertainty in the parameters of fabricated devices and interconnects from die-to-die and within-die, and**

III. ***operating-related variations*** **- uncertainty in the operating environment of a particular device during its lifetime (temperature, supply voltage, mode of operation etc.).**

All these three steps that produce potentially timing uncertainties, are illustrated in *Figure 5*. Since each of the three variabilities represents orthogonal sample spaces, it is an uphill situation to perform a unified analysis. Environmental uncertainty and uncertainty due to modeling and analysis error are usually modeled utilizing worst-case margins, while uncertainty in process is commonly handled statistically. Thus, most variation aware timing analysis works, this thesis project included, focus only on modeling process variations.

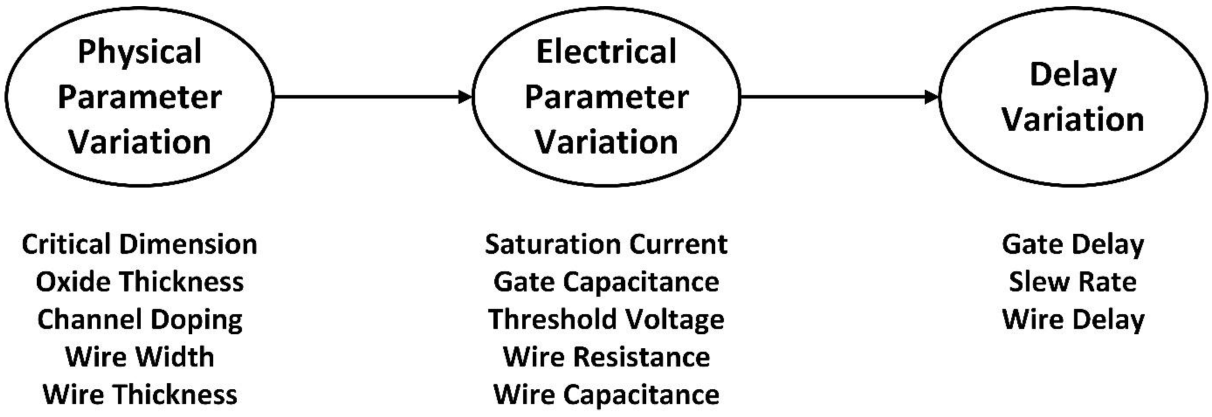


Figure 6: Parameter variations domino effect

The semiconductor manufacturing process has become more complicated while simultaneously process control precision is striving to stay relatively accurate with continued process scaling. This leads to many steps throughout the manufacturing process to become prone to variations. The main physical parameters affected are the gate length, the interconnect thickness and height, as illustrated in *Figure 6* and extendedly explained in [1].

Variations in these physical parameters result in variations in electrical device characteristics, like threshold voltage, resistance and capacitance of interconnects. Ultimately, the variations in electrical characteristics of circuit elements affect the delay variations of the timing characteristics of the circuit.

It is notable to mention that more than one electrical parameter may have a dependence on a specific physical parameter. For instance, both resistance and capacitance of an interconnect network are influenced by variation in wire width. In *Figure 7*, a general taxonomy of process variations is provided in order to classify them and give a notion of the spatial scale over which each one operates.

For the sake of simplicity, in our research work we take into consideration the following global inter-chip sources of variability and assume that each timing quantity may be sensitive to them:

- environmental: **voltage (V)**, **temperature (T)**
- front end of line process: **channel length (L)**, **device width (W)**, **voltage threshold (H)**
- back end of line: **metal (M)**

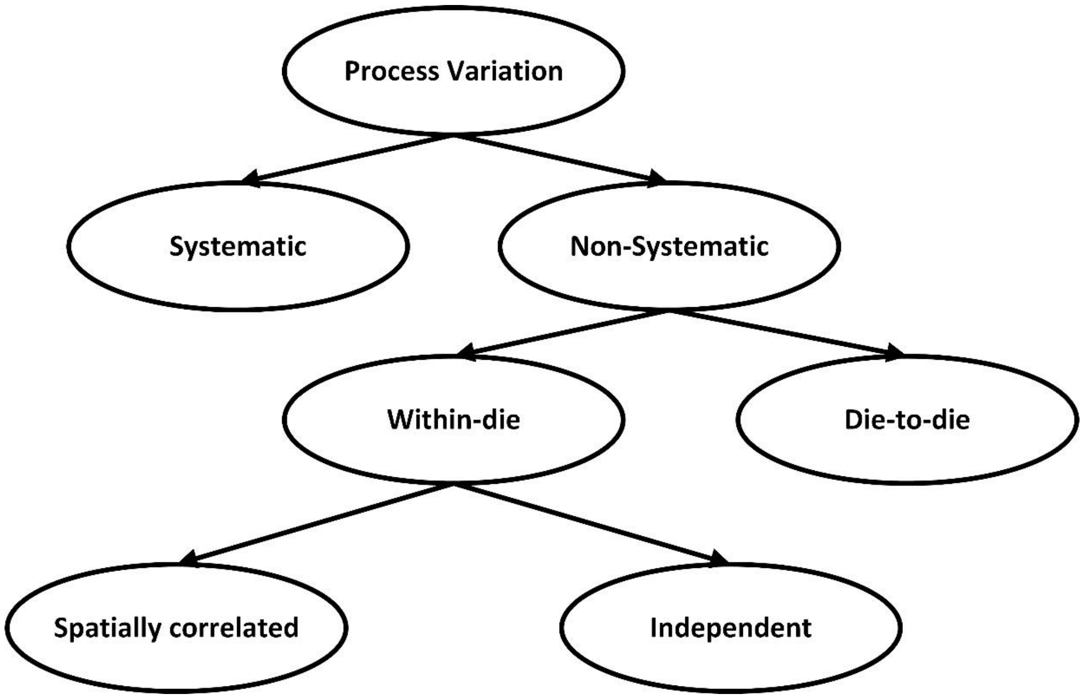


Figure 7: Taxonomy of process variations

For homogeneity, we suggest only a single parameter *M* for all metal layers. These parameters indicate systematic chip-to-chip (or inter-chip) sources of variation. To provide an example, variations in parameter temperature suggest that the chips would be dependent to different environmental temperature conditions. This parametric variation does not imply intra-chip systematic temperature differences. For the sake of simplicity, we ignore systematic intra-chip variations in all parameters. Besides the global inter-chip parameters we mentioned above, each one of the timing quantities might enclose an independent *random source of variability* (R) that symbolizes both random inter-chip as well as random intra-chip variation. Any timing quantity may therefore be expressed in the following notation:

$$\mu + \alpha_v \Delta V + \alpha_t \Delta T + \alpha_l \Delta L + \alpha_w \Delta W + \alpha_h \Delta H + \alpha_m \Delta M + \alpha_r \Delta R \quad (9),$$

where μ signifies the nominal value of the quantity (in other words, its value in the absence of variability).

Each parameter may diverge from -3 to $+3$ sigmas. Parameter sensitivities are expressed as time units per sigma values (e.g. $\alpha_v = 5$ picoseconds per sigma), and are captured either as *asserted* values, or via *finite differencing*. In the former case, the sensitivity of a timing quantity to a parameter is attainable directly as an input (e.g. voltage sensitivity for cells are available as warranted values in the cell library). Finite differencing in context of any parameter X suggests that the value of the timing

quantity Q is feasible (or can be computed) for at least two sigma corners of X.

Assuming that two sigma corners of X as $+3$ and -3 sigma, the finite differenced sensitivity α_X is computed as:

$$\alpha_X = \frac{Q_{|X=+3\sigma} - Q_{|X=-3\sigma}}{3-(-3)} \quad \textbf{(10).}$$

For the purposes of this thesis project, we assume all parameters (except metal (M) to be finite differenced (if required) between $+3$ and -3 sigma values. The metal parameter should be finite differenced between $+3$ and 0 sigma values, as it benefits faster analysis (as we will exhibit in the following chapter). All timing quantities mentioned in this section are assumed to be a function of variational parameters.

3. Models and Representation

For the purposes of this thesis project, we assume that for each benchmark circuit we have two files available: a netlist file and a library file[1], which are presented in *Appendix A.1*. The former, includes circuit information, topology and other circuit-related data, that will be modeled in accordance to what will be explained in the following sections. The netlist, consists of a set of interconnected elements, particularly cell instances as well as interconnecting circuitry. The latter, on the other hand, contains timing information in relation to the cell elements as well as variability information.

3.1. Interconnect

The primary ingredient of interconnection is a *net*, which assumingly has an input pin, called *port* and one or many output pins, called *taps* (*Figure 8*). For each net, the netlist of its parasitic RC tree is given in the netlist file. Parasitic RC trees only include grounded capacitors and resistors located between nodes in the tree (there are no coupling capacitors or grounded resistors).

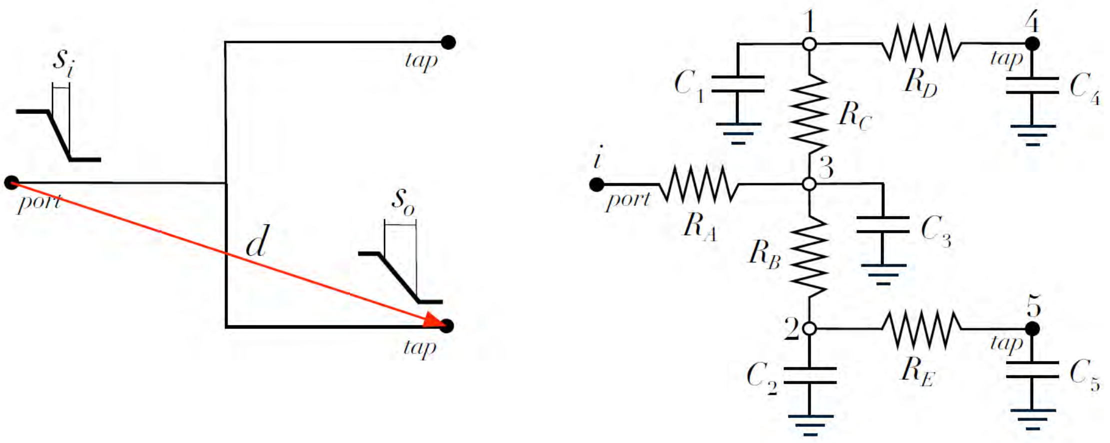


Figure 8: Interconnection representation

The calculation of the delays from port to each potential tap is performed using a very straightforward delay model, the ***Elmore delay model*** [1], according to which the delay is estimated by the value of the first moment of the impulse response. For RC tree networks, we utilize the method described in [31], as summarized below.

[1] The files are formatted according to the TAU Variation Aware Timing Analysis Contest (2013) [32] standards.

Consider any two given nodes e and k, where the lumped capacitance in node k is C_k. The resistance R_{ke} is the resistance of the common sub-path between the paths from the port to k and e respectively. Furthermore, R_{ee} is the resistance between the port and node e. For the example net tree demonstrated in *Figure 8* (right), we have $R_{15} = R_A$, since the common sub-path between nodes 1 and 5 includes solely resistor R_A. The Elmore delay, for a given node e, is given by the sum:

$$d_e = \sum_k R_{ke} C_k \quad \textbf{(11)}$$

where the summation extends over all nodes in the network. We can easily calculate this value by properly traversing the netlist of the parasitic RC tree. Once more, considering the example given in *Figure 8*, we have:

$$d_5 = R_A(C_1 + C_3 + C_4) + (R_A + R_B)C_2 + (R_A + R_B + R_E)C_5 \quad \textbf{(12)}$$

This value calculated above, provides the nominal or mean wire delay between the port and the tap. For variation aware parametric delay computation, we perform finite differencing to compute the sensitivity of delay to the metal parameter *M*. Provided corner specific metal resistance m_C^σ scalar values (from the cell library) are used to acquire the updated resistance and capacitance values of the interconnect network when the metal parameter is set to a given corner ($\Delta M = \sigma$). Each interconnect resistance and capacitance is scaled by the provided scalar and another deterministic delay computation is performed to compute the delay when ΔM is at the σ corner.

For the example we examined above, considering that the tap capacitance C_5 comes partly from the cell pin capacitance $C_{p,5}$ connected to node 5, the remaining capacitance $(C_5 - C_{p,5})$ is part of the interconnect network. Adopting similar notation for tap capacitance C_4, the delay at this corner can be calculated as follows:

$$\begin{aligned} d_{5|\Delta M=\sigma} = {} & m_R^\sigma R_A\left(m_C^\sigma\left[C_1 + C_3 + C_4 - C_{p,4}\right] + C_{p,4}\right) \\ & + (m_R^\sigma[R_A + R_B])m_C^\sigma C_2 \\ & + (m_R^\sigma[R_A + R_B + R_E])\left(C_{p,5} + m_C^\sigma\left(C_5 - C_{p,5}\right)\right) \quad (13) \end{aligned}$$

We perform the above calculation using the provided sigma corner of the metal parameter (usually +3 sigma). We then compute the sensitivity parameter ΔM utilizing finite differencing method between this corner and the nominal corner:

$$a_{m,5}^{D} = \frac{d_{5|\Delta M=\sigma} - d_{5|\Delta M=0}}{\sigma - 0} = \frac{d_{5|\Delta M=\sigma} - d_5}{\sigma} \quad (14)$$

Hence, the parametric delay model of the interconnection between the port and the tap node 5 includes two deterministic delay calculations and finally approximates to a linear model as demonstrated below:

$$d_5 + a_{m,5}^{D} \Delta M \ .$$

As a result, wire delays will not involve any sensitivity to other parameters (including random variation).

The nominal value of output slew on any given tap node o, can be approximated by a two-step procedure, which is illustrated below. First we calculate the nominal output slew of the impulse response on o, which can be approximated by the following formula, as noted in [30], [35]:

$$\hat{s}_0 \approx \sqrt{2\beta_0 - d_0^2} \quad (15)$$

Where β_o is the second moment of the impulse response at node o and d_o is the equivalent Elmore delay computed from (11) for node o. The value of β_o is computed by replacing all capacitance values C_k by $C_k d_k$, where d_k is the Elmore delay computed from (11). In *Figure 9*, the modified parasitic RC tree is demonstrated for the example of *Figure 8*.

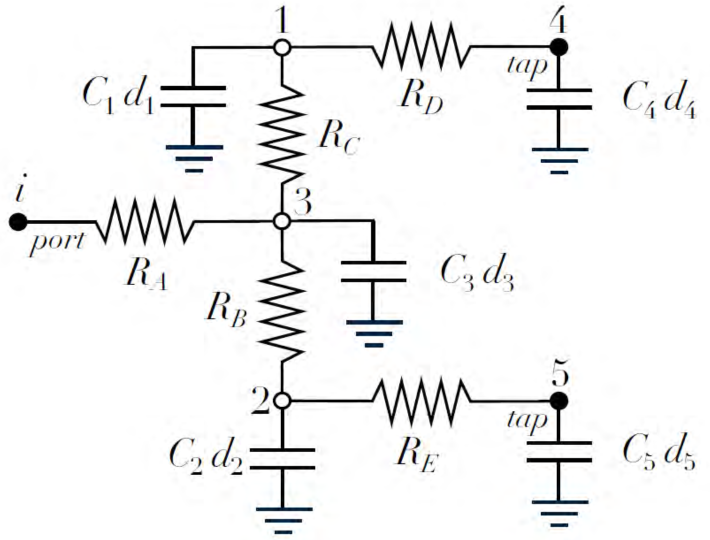


Figure 9: Modified RC tree for computing the second moment of impulse response

We followed the same procedure in order to calculate β_e:

$$\beta_e = \sum_k R_{ke} C_k d_k \quad (16)$$

Consequently, for the parasitic RC tree illustrated in figure 3 we get:

$$\beta_5 = R_A(C_1d_1 + C_3d_3 + C_4d_4) + (R_A + R_B)C_2d_2 + (R_A + R_B + R_E)C_5d_5 \quad \textbf{(17)}$$

Since we already computed $\hat{s}_o$, we go on to calculate the mean slew of the response to the input ramp, s_o, for which a valid approximation is reached by the expression:

$$s_o \approx \sqrt{s_i^2 + \hat{s}_o^2} \quad \textbf{(18)},$$

where s_i is the nominal or mean input slew.

Parametric output slew calculation comprises a bit of complicated finite differencing. Considering metal variability, $\hat{s}_o$ is a function of parameter ΔM only since both β_o and d_o are dependent on metal scalars. For a specific metal sigma corner σ:

$$\hat{s}_{o|\Delta M=\sigma} \approx \sqrt{2\beta_{o|\Delta M=\sigma} - d_{o|\Delta M=\sigma}^2} \quad \textbf{(19)},$$

where $d_{o|\Delta M=\sigma}$ is the Elmore delay value of the tap at node o at the specified metal corner, and $\beta_{o|\Delta M=\sigma}$ can be calculated by scaling the interconnect resistance and capacitance values similar values similar to the way we described earlier as in (13). The sensitivity to the metal parameter may now be computed via finite differencing the value of $\hat{s}_0$ between the two sigma corners for ΔM:

$$a_m^{\hat{S}_0} = \frac{\hat{s}_{0|\Delta M=\sigma} - \hat{s}_{0|\Delta M=0}}{\sigma - 0} = \frac{\hat{s}_{o|\Delta M=\sigma} - \hat{s}_o}{\sigma} \quad \textbf{(20)}.$$

We assume that to compute β_o at the metal corner σ an additional computation is required to calculate the parametric output slew $\hat{S}_o$ to an impulse response. Hence, we conclude to:

$$\hat{S}_o = \hat{s}_o + a_m^{\hat{S}_0}\Delta M \quad \textbf{(21)}.$$

3.2. Combinational Cells

Now we presume that cell delay, D, and output slew, S_o, can be approximated for a given combinational cell input/output pin pair by the following equations:

$$D = a\left(1 + k_{d,v}\Delta V + k_{d,l}\Delta L + k_{d,w}\Delta W + k_{d,h}\Delta H + k_{d,r}\Delta R\right) + bC_L + cS_i \quad \textbf{(22)}$$

$$S_o = x(1 + k_{s,v}\Delta V + k_{s,t}\Delta T + k_{s,l}\Delta L + k_{s,w}\Delta W + k_{s,h}\Delta H + k_{s,r}\Delta R) + yC_L + zS_i \quad \textbf{(23)},$$

where a, b, c, x, y, z and each k_i term are cell-dependent constants; C_L and S_i are the parametric output load and parametric input slew, respectively. The constants for each cell are provided in the cell library file, for both rise and fall transition. It should be noted that the k_i terms are a representation of the sensitivity to front end parameters as a fraction of a or x.

As demonstrated in ***Figure 10***, C_L refers to the downstream capacitance seen from the output of the cell. Quite a few complicated models have been suggested in order to compute C_L.

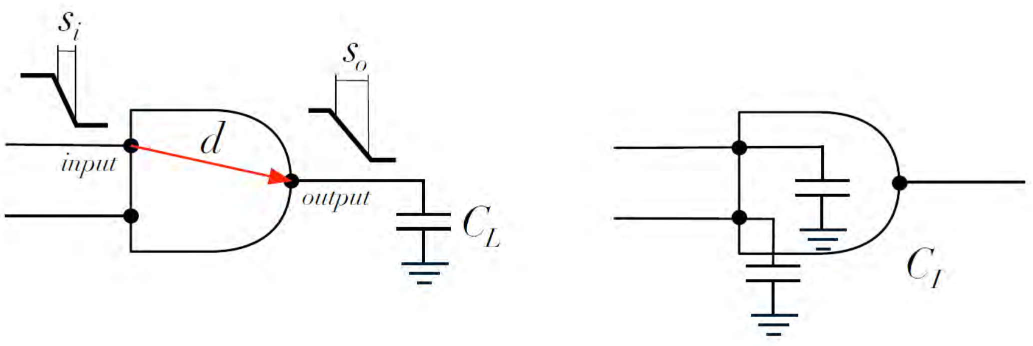


Figure 10: Combinational cell illustration

For the sake of simplicity, we adopted a rather straightforward model for that purpose. Thus, C_L is assumed to be the summation of all the capacitances in the RC parasitic tree, including cell pin capacitances at the taps of the interconnection:

$$C_L = \sum_k C_k \quad (24).$$

For the example presented in ***Figure 8***, we have:

$$C_{L|\Delta M=0} = C_1 + C_2 + C_3 + C_4 + C_5 \quad (25).$$

Due to the fact that the interconnect capacitances are depending on ΔM, consequently C_L is also a function of ΔM. In the above example we have:

$$C_{L|\Delta M=\sigma} = m_C^\sigma[C_1 + C_2 + C_3 + (C_4 - C_{p,4}) + (C_5 - C_{p,5})] + C_{p,4} + C_{p,5} \quad (26),$$

where $C_{p,4}$ and $C_{p,5}$ symbolize the cell pin capacitance contributions in the tap capacitances C_4 and C_5 respectively, and m_C^σ denotes the interconnect capacitance scalar value acquired from the cell library and it represents the impact of the metal variation at corner σ. Cell input pin capacitances are provided also in the cell library file, as fixed values for each pin and each transition state (rise or fall).

Finally, we compute the effective load's sensitivity to ΔM using finite differencing:

$$l_m = \frac{C_{L|\Delta M=\sigma} - C_{L|\Delta M=0}}{\sigma - 0} = \frac{C_{L|\Delta M=\sigma} - C_{L|\Delta M=0}}{\sigma} \quad (27).$$

We can then express the effective load C_L in a linear parametric form as follows:

$$C_{L|\Delta M=0} + l_m \Delta M \quad (28),$$

and afterwards replace it into delay and slew models in equations (22), (23).

3.3. Flip-Flops

Sequential circuits are composed of combinational logic blocks interrupted by registers, most commonly implemented with ***flip-flops***. Normally, they consist of several stages, where a register attains data from the outputs of a combinational block and infuses it into the inputs of the combinational block that follows as the next stage of combinational logic. Register operation is adjusted by clock signals generated by one or multiple clock sources. Clock signals that reach different flip-flops (represented by sinks in the clock tree), are delayed from the clock source by a known ***clock latency*** that will be symbolized as l.

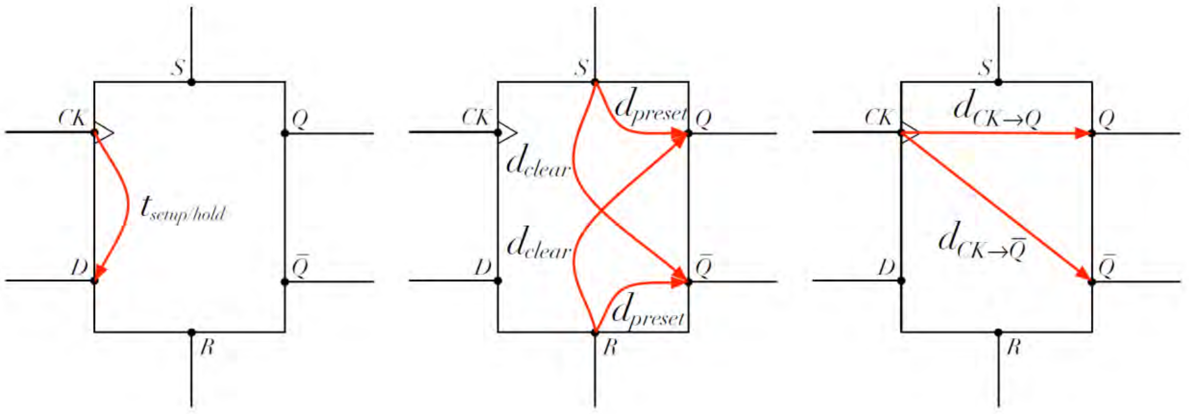


Figure 11: Flip-flop description

A ***flip-flop*** (and especially a ***D flip-flop***) is a storage element that captivates a given logic value at its input data pin D, when a given clock edge is identified at its clock pin CK, and thus propagates the captured value and its complement at the output pins Q and $\bar{Q}$. The flip-flop is also capable to enable asynchronous preset (set) and clear (reset) of the output pins, utilizing the S and R input pins.

Conventional operation of flip-flop demands the logic value of the input data pin to be stable for a specific period of time ***before*** the capturing clock edge. This specific period of time is denoted as ***setup time*** and will be represented by t_{setup} . Moreover, the logic value of the input data pin must be simultaneously available and stable for a specified period of time ***after*** the capturing clock edge. The specific period of time is denoted as ***hold time*** and we designate it as t_{hold}. We should point out that both setup and hold times are two of the usual performance figures provided in cell specification libraries for storage elements such as flip-flops. Other conventional figures include delay from clock to output, $d_{CK\to Q}/d_{CK\to\bar{Q}}$ and asynchronous preset and clear delays, d_{preset} and d_{clear}. A visualization of the standard performance metrics we just mentioned is provided in ***Figure 11***.

Observe now that setup as well as hold time are modeled as functions of the input slews at both the clock pin, CK, and the input pin, D, respectively:

$$t_{setup} = g + h\, S_i^{CK} + j\, S_i^{D} \quad (29)$$

$$t_{hold} = m + n\, S_i^{CK} + p\, S_i^{D} \quad (30)$$

Notice that in the above equations the input slews are parametric and, as a result, so are the setup and hold times.

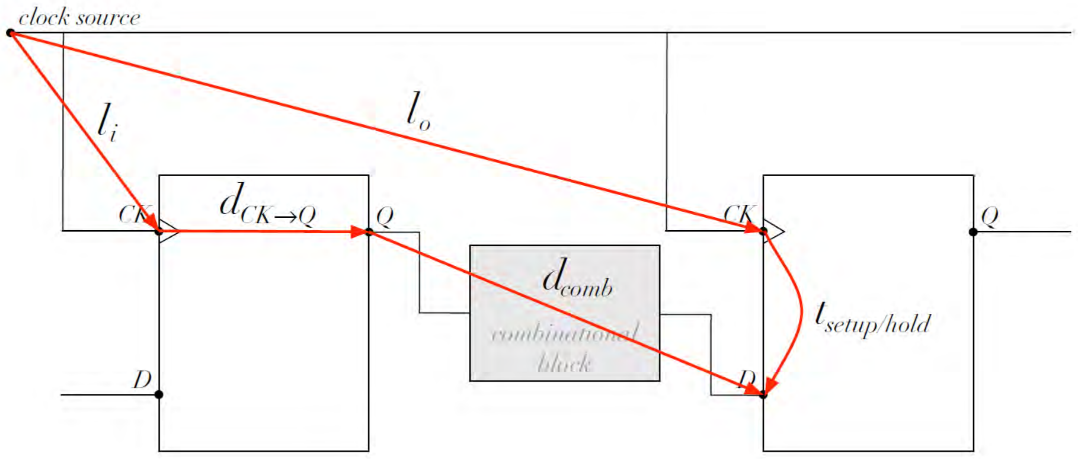


Figure 12: Signal propagation between two flip-flops

Let us consider now the standard case of signal propagation between two flip-flops as it is presented in *Figure 12.* Taking for granted that the clock edge is generated in the clock source at time 0, then it will reach the injecting flip-flop at time l_i, enabling the data at the input of the combinational block $d_{CK \to Q}$ time later. If the propagation delay in the combinational block is d_{comb}, then the data will be available at the input of the capturing flip-flop at time $l_i + d_{CK \to Q} + d_{comb}$. Presuming that the clock period is a deterministic constant T, the next clock edge will reach the capturing flip-flop at time $T + l_o$. To achieve proper operational conditions, the data must be available at the input of the flip-flop t_{setup} earlier than the next clock edge. Consequently, at the data input pin, D:

$$at_D^{late} = l_i^{late} + d_{CK \to Q} + d_{comb}^{late} \quad \textbf{(31)}$$

$$rat_{setup} = rat_D^{late} = T + l_o^{late} - t_{setup} \quad \textbf{(32)}$$

An analogous condition can be extracted from the necessity to ensure that the hold time is respected. Thus, the data input of the capturing flip-flop must remain stable for at least t_{hold} after the clock edge reaches the corresponding CK pin. Hence, at the data input pin, D, we have:

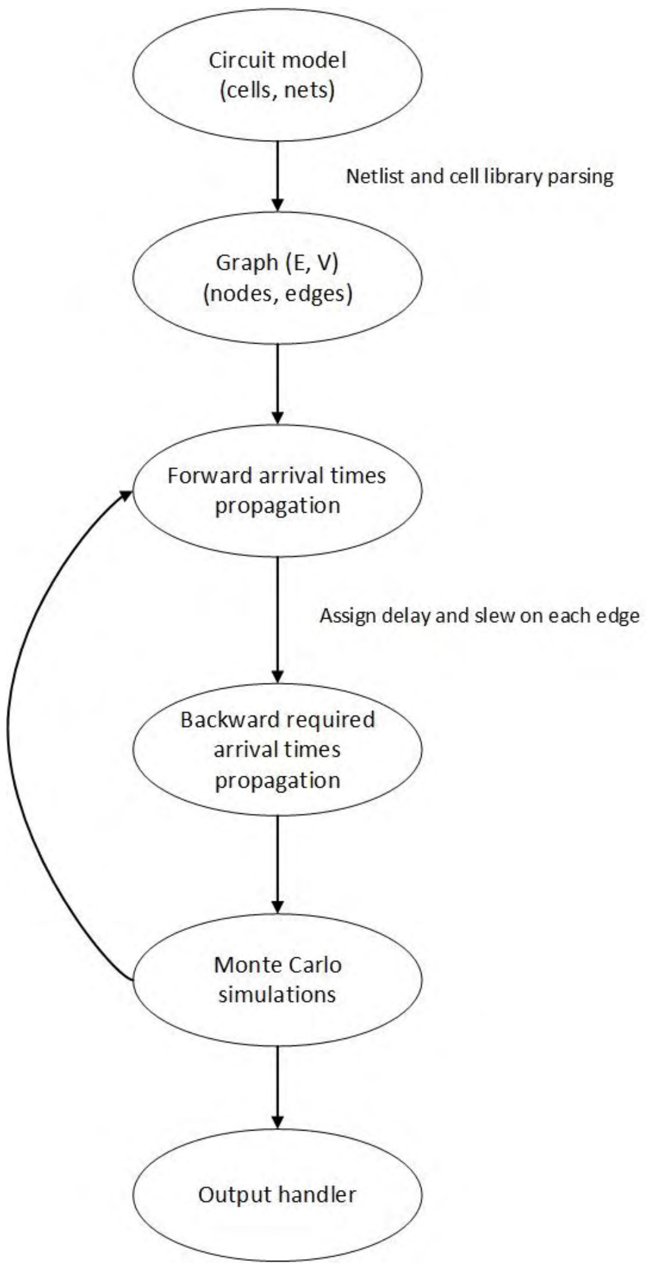
$$at_D^{early} = l_i^{early} + d_{CK \to Q} + d_{comb}^{early} \quad \textbf{(33)}$$

$$rat_{hold} = rat_D^{early} = l_o^{early} + t_{hold} \quad \textbf{(34)}$$

We should note that the arrival and required arrival times included in the equations above, can be computed from equations (5) and (6) in *Chapter 1.*

4. Stages of Implementation

In this chapter, we give a clear image of the technique we proposed for this thesis project and of the stages involved in the process. After providing a thorough description of the models and the input files used we will proceed by breaking down each stage of the procedure and explaining the diverging parts which distinct our work from previous approaches already presented. To begin with, we provide a flowchart in *Figure 13* with the path that was followed during the process.



Circuit model (cells, nets)
Netlist and cell library parsing
Graph (E, V) (nodes, edges)
Forward arrival times propagation
Assign delay and slew on each edge
Backward required arrival times propagation
Monte Carlo simulations
Output handler

Figure 13: Flowchart including the steps we followed during the approach proposed

4.1. Netlist and Cell Library Parsing

Our first goal was to extract all the necessary data from the input files and form it accordingly into the desirable data representation. This primary objective was achieved through the parsing of the netlist and the cell library files. For the netlist file, we parsed all the available information, stored it appropriately and formed the timing graph we described earlier, by modeling primary inputs, primary outputs and cell pins as nodes and nets and intra-cell circuitry as edges. A rough sketch of the netlist parser is provided below:

```
BEGIN
  while(!empty(netlist))
    parse next line L
    parse next token

    switch (token)
      case "input":
        insert in primary inputs hash table
        initialize slew and at for current node

      case "output":
        insert in primary inputs hash table

      case "instance":
        while(!L.empty)
          if(output pin)
            type = "outPin"
            if(internal node)
              insert in internal nodes hash table
            else
              insert in primary outputs hash table
          else
            type = "inPin"
            if(internal node)
              insert in internal nodes hash table
            else
              insert in primary inputs hash table
      case "slew":
        find node in primary inputs hash table
        assign fall and rise slews to node

      case "at":
        find node in primary inputs hash table
        assign early/late fall, early/late rise arrival times to node

      case "clock":
        find node in primary inputs hash table
        assign clock period to node

      case "wire":
        find node in primary inputs OR internal nodes hash tables
        makeUpNet(node)

      case "cap":
        assign capacitance to node

      case "res":
        determine starting and ending point of resistance
        assign resistance value to starting node

      case "rat":
        find node in primary outputs OR internal nodes hash tables
        assign operation mode to node

        if(opMode == "late")
          assing late fall/rise values to node
        else
          assing early fall/rise values to node
END
```

Algorithm 1: Netlist parser main function

As far as the cell library parser is concerned, besides the main parsing function, which basically operates as the netlist parsing function we illustrated thoroughly. We utilized some functions which we describe hereinafter:

- `classifyCell`: Classifies the parsed cell <key> into the cell library dictionary and returns the correct position of the dictionary that the cell structure should be stored.
- `readSlewDelay`: Collects the fall/rise slew values and fall/rise delay values.
- `readFallRiseConst`: Correspondingly to the previous function, parses the fall/rise constraints information contained in the line we examine.

- **`findCell`**: As its name states clearly, used to search through the parsed cell-types and locate the cell with the corresponding `<key>`.

4.2. Interconnect Delay Calculation

As aforementioned in *Chapter 3*, where we described the models, the basic instance of interconnect wire is a *net*, which basically consists of an input pin, namely *port*, and one or multiple output pins, namely *taps* and we illustrate it appropriately in *Figure 8*.

```
INPUT: pin -> start of a net

OUTPUT: delay/slew info @ the taps -> end nodes of a net

BEGIN

MAIN STEPS:

->    Compute delay
->    Compute slew
->    Write delay/slew info to the corresponding nodes


Elmore Delay Calculation using DFS algorithm:

      Apply DFS to reach the taps
      Store path in FIFO
      for every FIFO(i)
           if(branch root)
                SUM(capacitances of the other branches)
           dequeue()
           SUM(cap*TOTAL(res))
      END for
END


SUM is a controlled addition using a stack, in order to sum up the
correct resistances and capacitances (include only the on-path
resistances)
```

Algorithm 2: Main algorithm of interconnect delay calculation

The computation of port-to-tap delays can be accurately performed through electrical simulation using a similar software like Synopsys™ HSpice simulator. Nevertheless, as we mentioned before we used a much simpler and adequately accurate model, the Elmore delay model. To provide you with an insight of our approach, we demonstrate above the main steps of the algorithm we implemented.

Besides the mathematical operations that are performed and are modeled in equations (11) – (20), we employed the **DFS** algorithm in order to reach the potential taps of a net wire and avoid multiple traversals of the same net to discover all the potential taps. As illustrated in *Algorithm 3*, we utilized a **FiFo** stack structure in order to store temporarily the discovered paths and then, recursively, proceed with the application of the mathematical operations on each path.

4.3. Delay information propagation

During this stage, we describe the process of propagating the delay information both forward (from primary inputs to the primary outputs) propagating *arrival times* as well as backwards (from primary outputs to the primary inputs) propagating *required arrival time* throughout the circuit. The procedure is based on the following steps:

Step 1: Assign delay information to all timing arcs, while ignoring wire delays at the moment.

Step 2: Perform forward *arrival time* propagation.

Step 3: Perform backwards *required arrival time* propagation.

All of the above steps are based on an algorithm created on top of *breadth-first-search* (**BFS**) algorithm, which is a mainstream strategy in graph theory for searching in a graph. The **BFS** begins at a root node and inspects all of its adjacent nodes. Thereafter, for each of these adjacent nodes, it inspects in turn their neighbor nodes which are unvisited, and so on. The characteristics of this method provide an obvious explanation for the suitability on the task that we utilize a BFS-like algorithm. Providing as input a graph G and a root v of G, the following algorithm implements the **BFS** algorithm:

```
procedure BFS($G$,$v$) is
    create a queue $Q$
    create a set $V$
    enqueue $v$ onto $Q$
    add $v$ to $V$
    while $Q$ is not empty loop
        $t$ ← Q.dequeue()
        if $t$ is what we are looking for then
            return $t$
        end if
        for all edges $e$ in G.adjacentEdges(t) loop
            $u$ ← G.adjacentVertex($t$,$e$)
            if $u$ is not in $V$ then
                add $u$ to $V$
                enqueue $u$ onto $Q$
            end if
        end loop
    end loop
    return $none$
end BFS
```

Algorithm 3: Breadth-first search algorithm

4.4. Monte Carlo Simulations

Following the steps we described above, intervenes the statistical aspect of our approach, through the Monte Carlo simulations. After calculating and propagating all the delay information throughout the timing graph we apply the Monte Carlo simulations procedure in order to model the variation on the deterministic values we estimated previously.

Monte Carlo **implies the utilization of random numbers in scientific computing. To be more specific, it implies the utilization of random numbers to compute something that is not random. For example, let X be a random variable and denote its expected value as $A = E[X]$. Provided that we can generate $X_1, \dots, X_n$, n independent random variables with the same distribution, then we can make the approximation:**

$$A \approx \hat{A}_n = \frac{1}{n}\sum_{k=1}^{n} X_k$$

The ***strong law of large numbers*** **[33] states that $\hat{A}_n \to A$ as $\to \infty$. The X_k and $\hat{A}_n$ are random and (depending on the random number generator that is utilized) could be different every time we run the program. Nevertheless, the target number, A, is not random.**

We will emphasize at this point to the distinction between Monte Carlo and ***simulation.*** **Simulation implies producing random variables with a certain distribution just to examine them. The reason for this distinction is that there may be other ways to define A that make it easier to estimate. This process, is called** ***variance reduction*****, since most of the error in $\hat{A}$ is** ***statistical.*** **By reducing the variance of $\hat{A}$ we achieve also the reduction of the statistical error.**

We often have a choice between Monte Carlo and deterministic methods. Although this sounds appealing, the general rule is that deterministic are better than Monte Carlo in any situation where the deterministic method is a ***viable*** **option.**

This leads to the conclusion that quite often, we are driven to resort to Monte Carlo due to the ***"curse of dimensionality".*** **This curse, insinuates that the work to solve a multi-dimensional problem may grow exponentially with the dimension. Suppose, for example, that we want to compute an integral over ten variables, an integration in ten dimensional space. Supposing that we attempt to approximate the integral using twenty points in each coordinate direction, the total number of integration**

points would be $20^{10} \approx 10^{13}$, which is on the edge of what a computer is able to calculate in a whole day. For the same example, Monte Carlo could reach the same accuracy with approximately, say, 10^6 points.

One beneficial feature of Monte Carlo is that it is possible to estimate the order of magnitude of statistical error, which is the dominant error in most Monte Carlo computations. Another advantageous characteristic of Monte Carlo is that simple but clever ideas can lead to immense practical improvements in efficiency and accuracy. Hence, to rephrase what we stated earlier, while A is given, *the algorithm for estimating it is not*. Therefore, the quest for more accurate alternative algorithms is often called "*variance reduction*", with the most common variance reduction technique being *importance sampling* [34].

4.5. Output Handling

For the sake of simplicity, we encourage the more thorough reader to address the *Appendix A.3.* for more insight on the output files format, as it is beyond the purposes of this work to go to deep with technical details.

5. Evaluation and Conclusion

5.1. Computational Infrastructure

The variation aware timing analysis tool of this thesis project was developed and evaluated on a machine with the following specifications:

- Intel® Core™ i7-3770 CPU @ 3.40GHz, 8MB cache memory and integrated graphics processor Intel® HD Graphics 4000
- 8GB of RAM

The installed software that was utilized:

- Kubuntu 14.04 LTS (Trusty Tahr), 64-bit
- Gcc 4.9.0 20140422
- Eclipse CDT (C/C++ Development Tooling) 8.4.0, 64-bit

5.2. Evaluation

In this section, we present the features of the tool we developed in this master thesis project, by initially outlining the test cases we utilized and afterwards provide some experimental results in order to demonstrate the basic uses of our tool. As an important remark, we should point out that in any variation-aware approach, the runtime of the process is not the key metric and this is the reason we neglect it during the evaluation.

We selected various benchmarks amongst the ISCAS benchmarks, like the c*499* and the s*400*. We picked specific primary output nodes from the c*499* benchmark, to perform yield estimation (by calculating the *at* and *slew* at the specified node) and random internal nodes from s*400*, *s27* and other suitable benchmarks to examine slack violations.

We set the yield percentage at 95%. The function that is illustrated as the label for the y-axis in ***Figure 14***, is the formula for the normal distribution, parametrized in terms of the mean and the variance:

$$f(x;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

Benchmark	Node	Op. Mode	Transition	Metric
c499	nod23	late	fall	at

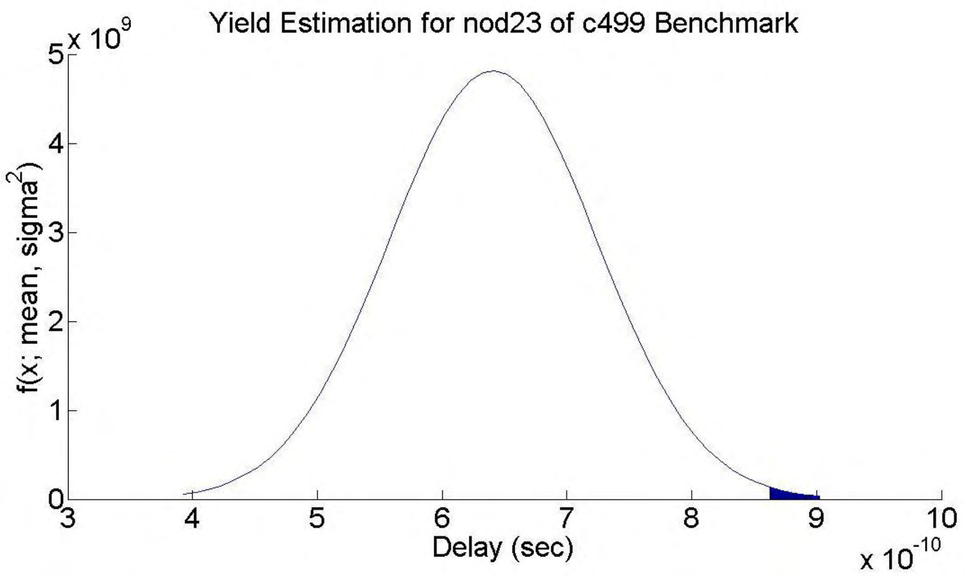


Figure 14: Yield estimation for nod23 of c499 (at)

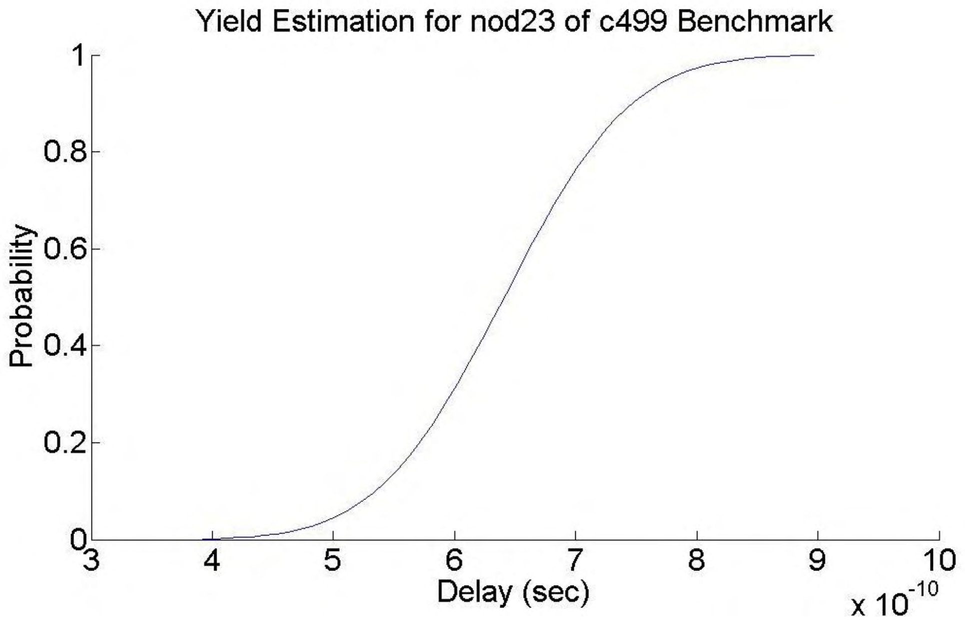


Figure 15: Yield estimation for nod23 of c499 using the CDF

In the same manner as illustrated above, we estimate yield in another node of the c499 benchmark:

Benchmark	Node	Op. Mode	Transition	Metric
c499	nod30	late	fall	at/slew

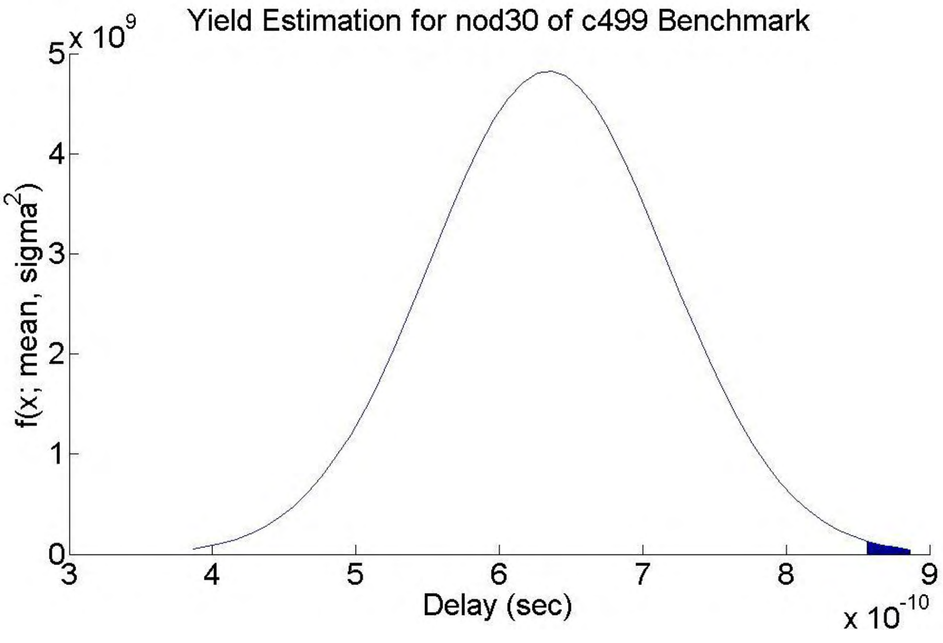


Figure 16: Yield estimation for nod30 of c499 (at)

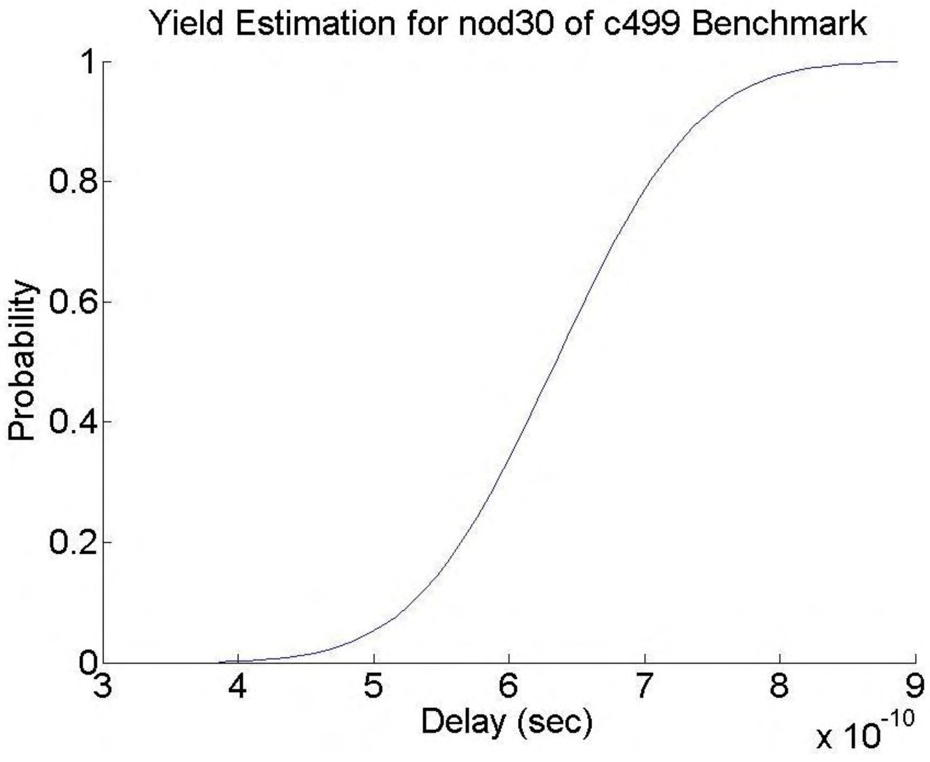


Figure 17: Yield Estimation for nod30 of c499 using CDF

For nod30, we will demonstrate that it is possible to also use *slew* in order to estimate the yield:

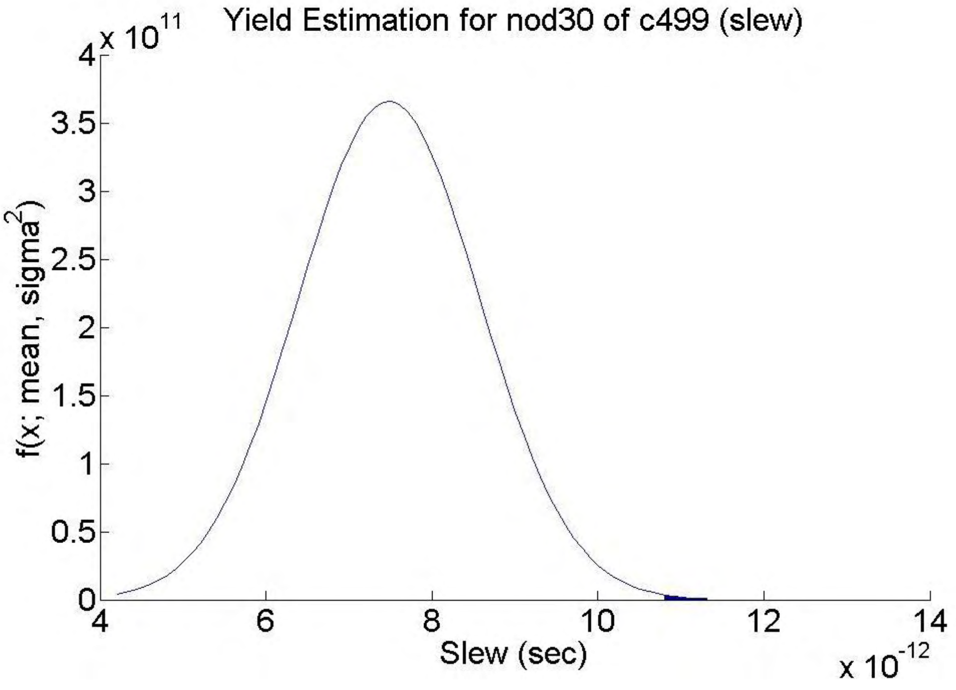


Figure 18: Yield Estimation for nod30 of 499 (slew)

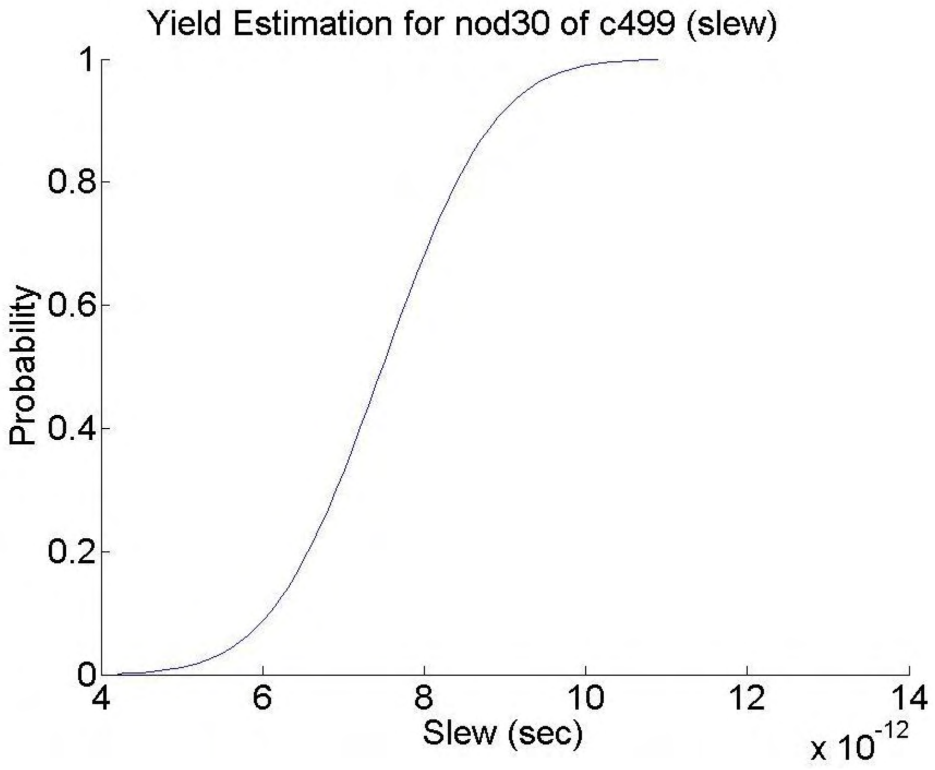


Figure 19: Yield Estimation for nod30 of c499 using CDF (slew)

At this point we should note that we can easily estimate the total yield from both *at* and *slew* metrics by selecting the minimum yield as the total. Generally, for all the metrics we could possible take into consideration for yield estimation we can say that:

$$Total_{yield} = min(of\ all\ metrics)$$

We now proceed to examine slack violations, or also known as setup/hold time violations (as we described in *Chapter 1*), depending on the operation mode, late or early respectively. We examined some of the ISCAS that include filp-flops in order to check the slacks, such as s400, s27 and others.

Benchmark	Node	Op. Mode	Transition	Metric
s400, s27	Various	early/late	Fall/rise	slack

It is accustomed for designers to set a "blind" guardband in order to ensure that no setup or hold time violations will occur. This may be the safest path, nevertheless it is not always optimal, as it may be quite pessimistic.

After evaluating the slack information we obtained at many different nodes of various benchmarks, for both early and late operational mode, we came to the conclusion that there is no need for us to set a "blind" guardband as the worst-case hypothesis we used in the evaluation:

$$WORST_{CASE} = (MEAN + 3 * SIGMA)$$

is a realistic hypothesis. We came to this conclusion after converging to the same probability, resulting from the CDF application on our hypothesis for every particular node (namely 99,9% or more precisely 99,87%) using the slack information we acquired through the examination of every benchmark netlist.

5.3. Future Work

Deterministic STA has developed enormously over the last two decades and handles several technology-scaling-related issues, like resistive and inductive shielding, crosstalk noise, clock skew and many more. However most researchers have, up to now, concentrated on the basic statistical timing analysis operations, the *sum* and *max/min* operations required for propagating the delay information through the timing graph.

As far as our own research plan is to innovate and expand the current approach. Our primary goal will not differ from the

current trend, which as we mentioned is to improve the accuracy and execution runtime of our tool. This will be pursued by parallelizing the most demanding operations, which are, as described earlier, the *sum* and the *min/max* operations that take place during the delay information propagation both forward and backwards through the timing graph. This can be achieved by implementing our algorithm using state-of-the-art parallel programming techniques such as CUDA and hence utilizing GPU cores for these fundamental operations.

Another goal that will be pursued and that potentially would widen the horizons for our work, is the adaptation of our approach with the view to process industrial standard cell libraries. An expansion like this would mean eventually that we will be able to statistically approximate even better to real conditions by completing the following two individual tasks:

I. Build empirical models for performance functions based on the insight gained from data analysis of cell performance functions for several standard industrial cell libraries. This will help to determine the optimal polynomial order for each variation parameter to find the general form of a compact performance function.

II. Formally classify transition waveforms based on the statistical analysis of switching transitions for several standard industrial cell libraries. The anticipated results can be used to speed up vastly our timing analysis engine considering real waveform shapes based on lookup tables. This research aspect could potentially change the way timing analysis is performed by enabling simulation at a higher level of abstraction, with the accuracy of circuit-level simulation. The main applications could be statistical performance simulation as well as variation-aware placement and routing.

In general, we could say that the fundamental challenge that **SSTA** research community has to face, is to prove itself worthy of the trust of both the designers as well as of the EDA-tools major companies. In other words, it is the duty of the research community to bring variation aware timing analysis to maturity.

5.4. Conclusion

Statistical timing analysis has gained extensive interest in recent years. Various research findings have been published and at the same time commercial efforts are underway. Nevertheless, the obstructions to widespread adoption of statistical approaches in industry remain challenging. The main issue is that even the current state-of-the-art variation-aware timing analysis methods

still do not address many of the issues that are taken for granted in traditional deterministic static timing analysis.

In this diploma thesis project, we focused on the process variations that impact the procedure of timing analysis, we classified and then review some previous approaches in variation aware timing analysis. Later, we described the models and the implementation stages of the tool we developed for timing analysis under process variations. The novel element of our method is the extensive use of a BFS-based algorithm during the stage of arc-delay-information propagation as well as the use of DFS algorithm through the interconnect delay calculation phase, in order to minimize the iterations that were necessary so as to discover all the paths included in the specific net. Finally, we evaluated the results of our approach, by estimating the yield on certain nodes of the benchmark netlists. Another evaluation metric we used to verify the validity of our implementation, was the examination of slack constraints violations by applying a worst-case slack estimation based on the slack information we gathered. We came to the conclusion thanks to the experimental results, that our approach to determine the worst case at $mean + 3 * sigma$ **was a successful choice by estimating nearly the optimal worst case.**

References

[1] David Blaauw, Kaviraj Chopra, Ashish Srivastava and Lou Scheffer, "Statistical Timing Analysis: From Basic Principles to State of the Art", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* pp. 589-607, April 2008.

[2] S. Bernd, "Cumulative Distribution Functions", College of Engineering and Science, Louisiana Tech. University, http://www2.latech.edu/˜schroder/slides/stat/cdf.pdf.

[3] C. Visweswariah, K. Ravindran, K. Kalafala, S. Walker and S. Narayan, "First-Order incremental block-based statistical timing analysis", *DAC*, 2004.

[4] A. Agarwal, D. Blaauw and V. Zolotov, "Statistical timing analysis for intra-die process variations with spacial correlations", *ICCAD*, 2003.

[5] H. Chang, S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single PERT-like traversal", *ICCAD*, 2003.

[6] H. Chang, V. Zolotov, S. Narayan and C. Visweswariah, "Parametererized block-based statistical timing analysis with non-Gaussian parameters, nonlinear delay functions", *DAC*, 2005.

[7] Y. Zhan, A. Strojwas, X. Li, T. Pileggi, D. Newmark and M. Sharma, "Correlation-aware statistical timing analysis with non-Gaussian delay distributions", *DAC*, 2005.

[8] V. Khandelwal and A. Srivastava, "A general framework for accurate statistical timing analysis considering correlations", *DAC*, 2005.

[9] Lai, N. Balakrishnan and Chin-Diew, "Continuous Bivariate Distributions", Springer, 2009.

[10] C. Clark, "The greatest of a finite set of random variables", *J. Oper. Res., vol. 9, no. 2,* pp. 145-162, March/April 1961.

[11] T. Kirkpatrick and N. Clark, "PERT as an aid to logic design", *IBM J. Res. Develop., vol. 10, no. 2,* pp. 135-141, March 1966.

[12] H. Jyu, S. Malik, S. Devdas and K. Keutzer, ″Statistical timing analysis of combinational logic circuits″, *IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 1, no. 2,* pp. 126-137, June 1993.

[13] R. Brashear, N. Menezes, C. Oh, L. Pillage and M. Mercer, ″Predicting circuit performance using circuit-level statistical timing analysis″, *DATE*, 1994.

[14] Y. Deguchi, N. Ishiura, and S. Yajima, ″Probabilistic ctss: Analysis of timing error probability in asynchronous logic circuits″, *ACM/IEEE Design Automation Conference*, 1991.

[15] J. A. G. Jess et al., ″Statistical timing for parametric yield prediction of digital integrated circuits″, *DAC*, 2003.

[16] M. Orshansky and A. Bandyopadhyay, ″Fast statistical timing analysis handling arbitrary delay correlations″, *DAC*, 2004.

[17] A. Agarwal, V. Zolotov and D. Blaauw, ″Statistical timing analysis using bounds and selective enumeration″, *IEEE Trans. on CAD, vol. 22, no. 9,* pp. 1243-1260, September 2003.

[18] F. N. Najm and N. Menezes, ″Statistical timing analysis based on a timing yield model″, *DAC*, 2004.

[19] A. Gattiker, S. Nassif, R. Dinakar and C. Long, ″Timing yield estimation from static timing analysis″, *ISQED*, 2001

[20] M. Orshansky and K. Keutzer, ″A general probabilistic framework for worst case timing analysis″, *DAC*, 2002.

[21] Synopsys, Synopsys PrimeTime™,

http://www.synopsys.com/Tools/Implementation/SignOff/Pages/PrimeTime.aspx.

[22] Cadence, Cadence Encounter Timing System™,

http://www.cadence.com/products/di/edi˙system/pages/default.aspx .

[24] L. Scheffer, ″The count of Monte Carlo″, *TAU Int. Workshop Timing*, 2004.

[25] R. Kanj, R. Joshi and S. Nassif, ″Mixture importance sampling and its application to the analysis of SRAM designs in the presence of rare failure events″, *DAC*, 2006.

[26] A. Agarwal, et al., ″Statistical delay computation considering spatial correlations″, *ASP-DAC*, 2003.

[27] C. Amin, ″Statistical static timing analysis: How simple can we get?″, *DAC*, 2005.

[28] R. Lin, M. Wu, ″A new statistical approach to timing analysis of VLSI circuits″, *Int. Conf. VLSI Design*, 1998.

[29] B. Choi and D. Walker, ″Timing analysis of combinational circuits including capacitive coupling and statistical process variation″, *Symp. VLSI Test*, 2000.

[30] W. C. Elmore, ″The transient response of damped linear networks with particular regard to wideband amplifiers″, 11 April 1947.

[31] P. Penfield Jr. and J. Rubinstein, ″Signal delay in RC tree netwroks″, *Design Automation Conference*, 1981.

[32] D. Sinha et al., ″TAU 2013 Variation Aware Timing Analysis Contest″, 2013.

[33] Snell, Charles M. Grinstead and J. Laurie, ″Introduction to Probability″, American Mathematical Society (AMS), 2006.

[34] Peter W. Glynn and Donald L. Iglehart, ″Importance Sampling for Stochastic Simulations″, *Management Science, Vol. 35, No. 11,* pp. 1367-1392, November 1989.

[35] R. Gupta, B. Tutuianu and L. T. Pileggi, ″The Elmore Delay as a Bound for RC Trees with Generalized Input Signals″, *IEEE Transactions on Computer-Aided Design for Integrated Circuits and Systems*, Vol. 16, No. 1, pp. 95-104, January1997.

Appendix

A.1. Input Files Formats

Netlist File

In this appendix, we introduce to the reader some more practical issues concerning the tool developed for this diploma thesis project. As an introduction to this process, we provide you an insight to the input files' structure.

Firstly, we will break down the netlist file, which as we mentioned before, encloses the description of the circuit topology:

```
input <node>
output <node>
instance <cell name> <pin name>:<node> ... <pin name>:<node>
wire <post node> <tap node> ... <tap node>
       res <node> <node> <resistance>
       ...
       cap <node> <capacitance>
       ...
slew <node> <slew fall> <slew rise>
clock <node><period>
at <node> <at fall early> <at fall late> <at rise early> <at rise late>
rat <node> <mode of operation> <rat fall> <rat rise>
```

The keywords included in this netlist type files are the following:

- `input`, primary input node;
- `output`, primary output node;
- `instance`, cell instance;
- `wire`, interconnect net;
- `res`, `cap`, resistor and capacitor of a parasitic RC tree (possible to appear in any order);
- `slew`, input slew at the primary inputs;
- `clock`, clock input constraint (only used for primary input nodes);
- `at`, arrival time constraint (only used for primary input nodes);

- `rat`, required arrival time constraint.

The corresponding variable fields denoted in the above generic netlist file are presented below:

- `<node>`, `<port node>` and `<tap node>` are node names, of up to 64 characters, which can contain alphanumeric characters, the underscore or the dash (the first character must be a letter);
- `<cell name>` is the name of the library cell (exactly as it will appear in the cell library file), of up to 32 alphanumeric characters (the first character must be a letter);
- `<pin name>` is the name of a pin of the cell (exactly as it will appear in the cell library file), of up to 32 alphanumeric characters;
- `<resistance>` is the value of the resistance in Ohm, represented in scientific notation;
- `<capacitance>` is the value of the capacitance in Farad, represented in scientific notation;
- `<slew fall>` and `<slew rise>` are the fall and rise of the corresponding primary input, in seconds, represented in scientific notation. Early and late slews at the inputs are assumed to be identical;
- `<period>` is the clock period in seconds, denoted in scientific notation;
- `<at fall early>`, `<at fall late>`, `<at rise early>` and `<at rise late>` are real numbers, represented in scientific notation, which represent arrival time constraints for fall/rise transitions in early/late mode, at the primary inputs (in seconds);
- `<mode of operation>` is the mode of operation and can be either `early` or `late`;
- `<rat fall>` and `<rat rise>` are real numbers, represented in scientific notation, which represent required arrival time constraints for fall/rise transitions and early/late mode in seconds.

It should be noted that if no input slew is defined for any primary input, we assumed $1e-12$, for both fall and rise transitions. The designs described in any netlist of this form, will have only one clock input pin (one clock domain) at most.

Cell Library File

Alongside the netlist file we present the format of the cell library file, which includes the timing information of any cell used and is formatted as described as follows:

```
metal <sigma corner> <resistance scale factor> <capacitance scale factor>
      ...
cell <cell name>
      pin <pin name> input <fall capacitance> <rise capacitance>
      pin <pin name> output
      pin <pin name> clock <fall capacitance> <rise capacitance>
      ...
      timing <input pin name> <output pin name> <timing sense> \
               <fall slew> <rise slew> <fall delay> <rise delay>
      setup <clock pin name> <input pin name> <edge type> \
              <fall constraint> <rise constraint>
      hold <clock pin name> <input pin name> <edge type> \
             <fall constraint> <rise constraint>
      preset <input pin name> <output pin name> <edge type> <slew> <delay>
      clear <input pin name> <output pin name> <edge type> <slew> <delay>
```

In accordance to the presentation we provided for the netlist file, we describe below the keywords and the variable fields utilized in the cell library files. Firstly, the keywords:

- **`metal`, metal parameters scalars at specific sigma corners;**
- **`cell`, start of cell definition;**
- **`pin`, start of pin definition;**
- **`input`, `output` and `clock`, pin type;**
- **`timing`, delay;**
- **`setup`, setup time;**
- **`hold`, hold time;**
- **`preset`, preset time (output node is set to *high*);**
- **`clear`, clear time (output node is set to *low*).**

Similarly to the netlist file presentation, we exemplify the variables presented in the above generic cell library file:

- **`<sigma corner>` is the sigma corner value (σ) of metal parameter ΔM for which resistance and capacitance scale factors are provided;**

- **`<resistance scale factor>`** is the value (m_R^σ) by which the nominal interconnect resistance provided in the netlist should be scaled at the given metal sigma corner;
- **`<capacitance scale factor>`** is the value (m_C^σ) by which the nominal interconnect capacitance provided in the netlist should be scaled at the given metal sigma corner;
- **`<cell name>`** is the name of the cell, of up to 32 characters in length, which can contain solely alphanumeric characters;
- **`<pin name>`** is the name of a pin of the cell, of up to 32 characters in length, which can contain exclusively alphanumeric characters, also;
- **`<fall capacitance>`** and **`<rise capacitance>`** are values of the pin's input capacitances in Farad, for rise/fall transitions, represented in scientific notation;
- **`<input pin name>`**, **`<output pin name>`** and **`<clock pin name>`** are the names of the input, output and clock pins respectively, of a given delay or constraint specification, of up to 32 characters in length, which is allowed to contain (once more) only alphanumeric characters;
- **`<timing sense>`**, can be any of:
    - **`positive_unate`**, transition direction is preserved from input to output (rise → rise, fall → fall);
    - **`negative_unate`**, transition direction is reversed from input to output (rise → fall, fall → rise);
    - **`non_unate`**, transition direction cannot be presumed from a single input (take the worst, among rise/fall);
- **`<slew>`**, **`<fall slew>`**, **`<rise slew>`**, are each given by 9 real numbers separated by white spaces, which match the parameters x, y, z, $k_{s,v}$, $k_{s,t}$, $k_{s,l}$, $k_{s,w}$, $k_{s,h}$, and $k_{s,r}$ of equation calculating S_o (fall/rise refers to the transition direction in the output pin);
- **`<delay>`**, **`<fall delay>`** and **`<rise delay>`**, are each given by 9 real numbers separated by white spaces, which correspond to the parameters a, b, c, $k_{d,v}$, $k_{d,t}$, $k_{d,l}$, $k_{d,w}$, $k_{d,h}$, and $k_{d,r}$ of equation computing D;
- **`<edge type>`**, can be one of:
    - **`falling`**, constraint that applies to the falling clock edge;
    - **`rising`**, constraint applying to the rising clock edge;

- **`<fall constraint>` and `<rise constraint>` are each given by three real numbers separated by white spaces, which correspond accordingly to the parameters g , h and j of equation for the t_{setup} or m, n and p of the equation for the t_{hold} , depending whether we are dealing with setup constraints or hold constraints, respectively;**

We should note that both preset and clear values are represented for entirety and are ignored for the timing analysis approach we propose.

A.2. Example Input Files

The netlist file which includes the circuit topology is formatted as follows:

```
input in_1
input in_2
input in_3
input in_4
output out
instance AND2X1 A:in_1 B:in_2 Y:w
instance XOR2X1 A:u B:in_4 Y:v
instance NOR2X1 A:k B:h Y:out
wire w k
      res w r 0.355
      cap r 1.23423e-13
      res r k 0.7884
      cap h 0.8e-14
wire v h
      res v h 0.5
      cap h 1.37e-13
wire in_3 u
      res in_3 h 0.75
      cap u 1.44e-13
at in_1 0 0 0 0
at in_2 0 0 0 0
at in_3 0 0 0 0
at in_4 0 0 0 0
rat k late 1e-13 2e-13
```

In the next figure (*Figure 14*), we illustrate the example circuit that is described in the above netlist.

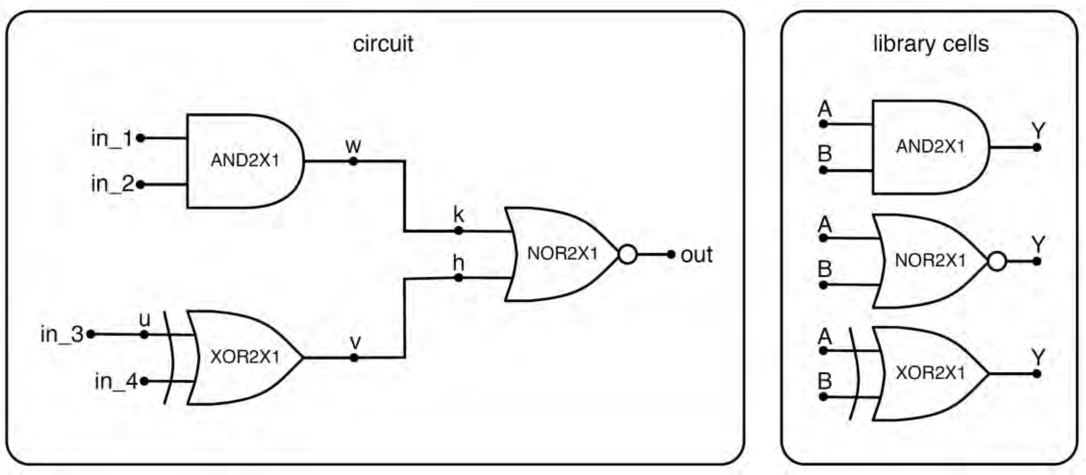


Figure 20: Example circuit

And the cell library file:

```
metal 0 1.0 1.0
metal 3 0.85 1.24
cell AND2X1
  pin A input 5.14e-16 5.34e-16
  pin B input 5.35e-16 5.7e-16
  pin Y output
  timing A Y positive_unate 4.78193e-12 4807.29 0.000196751 -0.00602524
    0.00585616 0.00021047 -0.0000985957 0.000195775 0.00019961 6.88656e-12
    11138.6 0.0000843378 -0.012127 0.037587 0.000227912 -0.0000493878
    0.0000838057 0.00019257 6.25087e-11 6032.36 0.00247343 -0.0026644
    0.00159043 0.0000694437 -0.00371931 0.0000911961 0.000157218 5.56981e-11
    11789 -0.000183877 -0.003468 0.00371931 0.0000911961 -0.00012835
    0.00023857 0.00022949
  timing B Y positive_unate 5.98666e-12 4868.63 0.000112529 -0.0055235
    0.00511724 0.000158928 -0.000179324 0.0000354006 0.000151742 6.69766e-12
    11142.7 0.0000819104 -1.23305 0.03842 0.0000047514 -0.0000607217
    0.0000343079 0.00020144 7.51493e-11 6005.01 0.00261237 -0.00222054
    0.00135711 0.0000391698 -0.000100236 0.0000324476 0.0000272022
    5.93502e-11 11798.7 -0.000224446 -0.0039269 0.00354033 0.000249731
    -0.0000545642 0.000128233 0.000209778
cell XOR2X1
  pin A input 4.15e-16 4.35e-16
  pin B input 4.34e-16 4.7e-16
  pin Y output
  timing A Y non_unate 5.78193e-12 5807.29 0.000196741 -0.00602425 0.00484616
    0.00021057 -0.0000984947 0.000194774 0.00019961 6.88646e-12 11138.6
    0.0000853378 -0.012127 0.0374874 0.000227912 -0.0000593878 0.0000838047
    0.00019247 6.24087e-11 6032.36 0.00257353 -0.0026655 0.00149053
    0.0000695537 -0.000138592 0.000119359 0.000147218 4.46981e-11 11789
    -0.000183877 -0.003568 0.00371931 0.0000911961 -0.00012834 0.00023847
    0.00022959
  timing B Y non_unate 4.98666e-12 5868.63 0.000112429 -0.0044234 0.00411725
    0.000148928 -0.000179325 0.0000345006 0.000141752 6.69766e-12 11152.7
    0.0000819105 -0.0123304 0.03852 0.0000057415 -0.0000607217 0.0000353079
    0.00020155 7.41593e-11 6004.01 0.00261237 -0.00222045 0.00134711
    0.0000391698 -0.000100236 0.0000325576 0.0000272022 4.93402e-11 11798.7
    -0.000225556 -0.0039269 0.00345033 0.000259731 -0.0000454652 0.000128233
    0.000209778
cell NOR2X1
  pin A input 5.13e-16 5.33e-16
  pin B input 5.35e-16 5.7e-16
  pin Y output
  timing A Y negative_unate 3.78193e-12 3807.29 0.000196761 -0.00602623
    0.00686616 0.00021037 -0.0000986967 0.000196776 0.00019961 6.88666e-12
    11138.6 0.0000833378 -0.012127 0.0376876 0.000227912 -0.0000393878
    0.0000838067 0.00019267 6.26087e-11 6032.36 0.00237333 -0.0026633
    0.00169033 0.0000693337 -0.000138392 0.000119339 0.000167218 6.66981e-11
    11789 -0.000183877 -0.003368 0.00371931 0.0000911961 -0.00012836
    0.00023867 0.00022939
  timing B Y negative_unate 6.98666e-12 3868.63 0.000112629 -0.0066236
    0.00611723 0.000168928 -0.000179323 0.0000363006 0.000161732 6.69766e-12
    11132.7 0.0000819103 -0.012330 0.03832 0.0000037613 -0.0000607217
    0.0000333079 0.00020133 7.61393e-11 6006.01 0.00261237 -0.00222063
    0.00136711 0.0000391698 -0.000100236 0.0000323376 0.0000272022
    6.93602e-11 11798.7 -0.000223336 -0.0039269 0.00363033 0.000239731
    -0.0000636632 0.000128233 0.000209778
```

A.3. Output File Format

```
at <node> <at early fall> <at early rise> <at late fall>
   <at late rise> <slew early fall> <slew early rise>
   <slew late fall> s<slew late rise>
...
slack <node> early <slack early fall> <slack early rise>
slack <node> late <slack late fall> <slack late rise>
...
```

All numerical results will be given in seconds and printed scientific notation, with 5 decimal places (eg. $1.23456e-10$).