NORTHWESTERN UNIVERSITY

Analysis and Optimization under Crosstalk and Variability in Deep Sub-Micron VLSI Circuits

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Electrical and Computer Engineering

By

Debjit Sinha

EVANSTON, ILLINOIS

June 2006

ABSTRACT

Analysis and Optimization under Crosstalk and Variability in
Deep Sub-Micron VLSI Circuits

Debjit Sinha

With very large scale integrated (VLSI) circuit fabrication entering the deep sub-micron era, devices are scaled down to finer geometries, clocks are run at higher frequencies, and more functionality is integrated into one chip. All these bring a great promise of "system-on-a-chip", but also introduce challenging new issues in the design process.

As a result of the increasing frequency and density, coupling effects or *crosstalk* between neighboring wires are increased. These effects can cause functionality and timing failures in a circuit. The dynamic power consumption in charging or discharging coupling capacitances is timing dependent, and contributes significantly to a circuit's power consumption. In addition, manufacturing process variations (e.g. V_T, L_e), and environmental variations (e.g. V_{dd}, Temperature) contribute to uncertainties that deeply impact the timing characteristics of a circuit. This *variability* makes timing verification, and consequently, timing driven circuit optimization extremely difficult. Although worst case analyses for circuit optimization are simpler, they are not desirable since they severely over-constrain the optimization problem, and result in designs that have excessive penalties in terms of area or power consumption.

In this research, we investigate the essential problems of timing verification, power estimation, and circuit (area or power) optimization under crosstalk and variability. We show that a circuit optimization problem under constraints on the maximal induced noise on each wire is equivalent to a fixpoint computation problem in a complete lattice. An optimal algorithm to solving this problem is developed, and is extended to handle variations. Under explicit timing constraints, we solve this problem in a Lagrangian Relaxation framework. We present a timing yield driven circuit optimization algorithm that considers variability and is based on statistical timing methodologies. Approaches to fast and approximation error aware statistical timing analysis are developed that also consider effects due to coupling as well as variability. Multiple input switching effects are considered for improved timing accuracy. We signify the importance of the timing dependence of dynamic power consumption in coupling capacitances, and develop an algorithm for accurate and efficient power estimation. Experimental results validate our approaches, and are promising.

To my family

Acknowledgments

Having spent two decades in the process of educating myself, I see this dissertation as yet another milestone in the path of pursuit to unending knowledge. The past four years in graduate school has bolstered my belief in Eric Hoffer's quote: *In a time of drastic change, it is the learners who inherit the future. The learned usually find themselves equipped to live in a world that no longer exists.*

I would like to express my gratitude and thanks to my advisor, Hai Zhou, for guiding this work to fruition. He has taught me meticulousness, perseverance, and essentially, the morality and ethics of research. He has given me moral support whenever I faced rejection, and the courage to keep my standards high, no matter what. Acknowledgment is due for his financial support, and also to the National Science Foundation and Intel Inc. for partly supporting this work.

Narendra V. Shenoy has been that mentor in my life from whom I have learnt to strive for perfection, the art of delivering coherent presentations, and above all, confidence. Having spent two summers working under his guidance at the Advanced Technology Group of Synopsys Inc., I re-kindled a fondness toward mathematics and programming in general.

I would like to thank Prith Banerjee, under whose guidance, I started my doctoral program. None of this would have been possible if he had not offered the Walter P. Murphy fellowship for graduate studies at Northwestern University. Prith has always been very helpful and has given me tremendous encouragement. I express my sincere regards to my other

doctoral committee members – Yehea Ismail, Robert Dick and Seda Memik. I have had the valuable opportunity to work with Yehea on a part of this research. From him, I have learnt the importance of understanding the fundamentals while solving any problem. Although I could not work with Robert or Seda, I have drawn inspiration from them.

Friends at Northwestern have made this journey so much more enjoyable. It is impossible to forget my research group colleagues – Chuan, Debasish, Jia, Nikos and Ruiming; they have always been friendly and have motivated me to work harder. Acknowledgment is due to my close friends around – Arindam, Paramita, Peter, Rubina and Somsubhra; every second spent with them has made me feel loved and special. My old friends, some of whom I have known for more than twenty years – Anil, Vivek, Narayanan, Samir, Tushar, Anuj, Raj, Biplab, Bharath, Manas, Debajyoti, Somdeb and Shirshanka have inspired me with their achievements, and I am proud to have known them all. Dia has helped me with the research in Chapter 7. To all my other friends, whose names I could not include here, I express my sincere acknowledgments.

I also take this opportunity to thank Partha P. Chakrabarti, my mentor at the Indian Institute of Technology, Kharagpur for being a strong and positive influence in my life. It is of little surprise that of the forty courses during my undergraduate studies, I chose to stick to one of his specializations for my doctoral research. I also thank some of my teachers from primary school – Ms. Majumdar for her mathematics courses, Mr. Naidu for introducing me to a *computer*, and Mr. John for teaching me the English language.

This section would be incomplete by all means until I acknowledge the help and support of my family. It is impossible to express the sacrifices made by my parents – Swapna and Subroto, while raising me. My grandparents have imbibed me with boundless love. I will be forever indebted to my uncles and aunts for loving me no less than their own sons – Sujit,

Sarbani, Kanchan, Neela, Sandeep, Sarmistha, Amiya, Monideepa, Kajal and Debarati. My sister, Debika, to her I owe a childhood filled with unmatched felicity. I feel fortunate to have come across my alter ego Malinky, who has given me a new reason to live life to the fullest. To my dear and special family, I dedicate this work.

Finally, I thank Lord Almighty for everything; as Shenoy acknowledges in his doctoral dissertation, I agree with the philosophy that it is better to thank God intermittently than to spend one's life as an atheist, only to discover that He does exist !

DEBJIT SINHA

Northwestern University

June 2006

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

With very large scale integrated (VLSI) circuit fabrication entering the deep sub-micron (DSM) era, devices and interconnection resources are scaled down to smaller sizes and placed at an ever increasing proximity. The semiconductor industry has advanced tremendously over the last ten years, with features sizes being downscaled from $0.35\mu m$ (in 1996) to $65nm$ today. It is predicted that feature sizes will continue shrinking to less than $45nm$ in the few years to come.

With the increase in die dimensions, more functions are integrated into one chip. There is also a trend of continuous increase in the number of devices within a chip. Modern day chips target over a billion transistors inside a die. Reduction in transistor switching delays results in faster signal transition times and higher clock frequencies. All these bring a great promise of "system-on-a-chip", but also introduce many new issues in the design process. Crosstalk and variability are two important issues among them.

1.1. Crosstalk

With the progress of deep sub-micron technologies, shrinking geometries have led to a reduction in the self-capacitance of wires. Meanwhile, coupling-capacitances have increased as wires have a larger aspect ratio and are brought closer together. For $90nm$ technologies, the ratio of an interconnect's parasitic coupling capacitance to its parasitic ground capacitance is nearly 5.5 (85% of the total parasitic capacitance) [**1, 2, 3**]. This signifies the increased

dominance of coupling capacitances with technology scaling. Trends indicate that the role of coupling-capacitances will be even more dominant in the future as feature-sizes shrink [**4**].

Crosstalk denotes the effects due to coupling, and can be classified into two types, namely *functional noise* and *delay variation*. Functional noise refers to a spurious signal induced on a quiet (non-switching) net by another coupled net that is switching. This noise causes a glitch which may propagate to a dynamic node or a latch, changing the circuit state and causing a functional failure [**5, 6**]. Simultaneous switchings on multiple coupled nets affect switching and propagation delays on the nets, thereby causing delay variations [**7**] in the circuit. These circuit delay variations may cause timing failures. Crosstalk effects are thus critical in modern high performance designs, and their significance is manifested with the usage of more aggressive and less noise-immune circuit structures like dynamic logic. Furthermore, dynamic power consumption in coupling capacitances is timing dependent, and contributes significantly to a circuit's power consumption.

1.2. Variability

As complementary metal oxide semiconductor (CMOS) feature sizes move into the DSM regime, the effects of process variations become critical with the increase in the variability of process parameters [**8, 9, 10**]. Uncertainties are attributed to manufacturing process variations (e.g. V_T, L_e); environmental variations (e.g. V_{dd}, Temperature); and device fatigue phenomenon. We term these sources of variability as parameters, and refer to the range in which they can collectively vary as the parameter space. Parametric variations translate to variations in circuit component delays and impacts the timing characteristics of a circuit. Nominally non-critical paths may become critical in some regions due to sensitivity to the sources of variation and can cause circuits to fail in meeting their timing constraints.

Consequently, it becomes imperative to improve the parametric yield of a circuit, which denotes the probability that given performance constraints are met under variations. This motivates development of robust circuit optimization approaches.

Traditional static timing analysis (STA) considers circuit component delays as deterministic quantities. These delays are obtained from technology library characterization at various corners in the parameter space, that is, at various assignment of parametric values. Timing analysis is then performed for some corner and is thus unable to account for any parametric variation. Multiple runs of STA performed at various corners in the parameter space is akin to a statistical sampling of circuit delays under variability. This is time consuming and may ignore a critical corner. Further, optimization for the worst case over-constrains the problem, and results in designs that have excessive penalties in terms of area or power consumption. Probabilistic and statistical approaches are therefore considered more appropriate for circuit analysis and optimization, under variations.

1.3. Dissertation overview

A growing importance of the mentioned effects motivate their consideration in the development of robust circuit timing/power analysis and circuit optimization methodologies for modern DSM VLSI designs. In this research, we present the results [**11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23**] obtained in solving some essential problems of timing/power estimation and circuit optimization under crosstalk and variability. The rest of this dissertation is organized as follows.

In Chapter 2, we consider the problem of circuit (area/power) optimization under crosstalk noise and timing constraints. We show that the circuit optimization problem under constraints on the maximal induced noise on each wire is equivalent to a fixpoint computation

problem in a complete lattice. An optimal algorithm to solving this problem is developed. Under explicit timing constraints, we solve this problem in a Lagrangian Relaxation framework.

In Chapter 3, we re-consider the above problem under variability by introducing additional yield constraints. This problem is solved as a fixpoint computation problem, and we employ Monte Carlo simulations to handle variability. We demonstrate that on the average, our approach results in a larger parametric yield in comparison to an alternate approach that attempts to handle variations by using guard-bands.

In Chapter 4, we consider the problem of area constrained circuit timing yield optimization under variability. We employ statistical approaches to delay modeling, timing analysis, and gate-sizing in solving this problem. It is shown that optimization for maximizing the probability of non-negative circuit slack is more effective than optimization for maximizing the worst-case slack, in terms of circuit area. We also provide one reason why statistical timing driven optimization does better than deterministic timing driven optimization.

Under variability, circuit component delays are modeled as random variables (often having a Gaussian distribution). Timing analysis of a circuit requires *max* operations on these random variables, which are slow and involve approximations. In Chapter 5, we develop approaches to fast and approximation-error aware computation of the *max* of a set of Gaussian random variables.

In Chapter 6, we develop a unified framework for statistical timing analysis that considers crosstalk and variability. The framework is amenable to delay variations caused due to simultaneous switching of multiple inputs of a gate as well. Comparisons to variation aware approaches that ignore crosstalk delay variations; and those that only handle these variations deterministically show significant timing accuracy gains of the proposed approach.

In Chapter 7, we signify the timing dependence of a circuit's dynamic power consumption in coupling capacitances. We develop a timing dependent framework to efficiently and accurately computing dynamic power consumption. It is shown that the ignorance of this timing dependence could result in both underestimation or overestimation of power. Consequently, we conclude that using guard-bands during timing independent power estimation is meaningless.

In Chapter 8, we study the impact of modern process technologies like dummy metal fills, chemical mechanical polishing, multiple thin dielectrics and trapezoidal conductor cross-sections on the electrical parameters of interconnects. We aim to quantify systematic variability in interconnect parasitics. Based on the obtained experimental results, we conclude that fills and trapezoidal conductor cross-sections can result in substantial variations in interconnect parasitic capacitances.

CHAPTER 2

Gate-size Optimization under Noise and Timing Constraints

Approaches to coupling-noise (or *functional noise*) reduction include routing and wire perturbations [**24, 25**]; buffer insertion [**26**]; and driver sizing. In the post-routing stage, coupling-noise on some nets may be critical and need to be fixed. In this scenario, gate-sizing is an attractive approach to noise reduction since it may not require re-routing, unlike the other mentioned approaches. Flexibility through scalable libraries and existing fill-space aid incremental gate-sizing, without affecting global routing.

The coupling noise that can possibly be induced on a net is dependent on the size of its driving-gate and the driving-gate-sizes of all its coupled nets. The noise can be reduced if the size of the net's driving-gate is increased, or if driving-gate-sizes of the coupled nets are decreased. However, when the driving-gate of a net is sized up, it may increase the noise it induces on other nets as an aggressor. On the other hand, when the driving-gate-size of a net is reduced, noise induced on itself may increase. Since coupling is symmetric on the coupled nets, it is artificial to classify them as aggressors or victims. A net could be an aggressor and a victim at the same time. Although it is plausible to classify a net either as an aggressor or a victim based on the strength of the noise on itself and other coupled nets, with changing driving-gate-sizes, the role of a net may change. Consequently, we do not classify nets as aggressors and (or) victims in our approach.

Gate-sizing for noise reduction has been shown to be effective by researchers. Xiao *et al.* [**27**] propose a transistor sizing algorithm for noise reduction. A gate-sizing algorithm

for noise optimization is proposed by Hashimoto *et al.* in [**28**], but is limited to sizing driving-gates of aggressor nets only. The post-route gate-sizing algorithm for noise reduction proposed by Becer *et al.* [**29**] does not guarantee an optimal solution. These gate-sizing approaches to noise reduction handle the timing constraints of the circuit as constraints on gate-sizes during optimization. This may require timing-budgeting before the size bound generations and can therefore over-constrain the problem.

In this chapter, we propose iterative algorithms for circuit optimization under constraints on noise, timing and gate-sizes. We do not explicitly consider coupling induced delay variations. A weighted sum of gate-sizes is used as the metric for circuit optimization. The formulated optimization problem is broken into sub-problems of circuit optimization under noise and timing constraints, respectively. The former is a gate-size optimization problem under noise constraints without considering timing requirements explicitly, while the latter is a gate-size optimization problem under given timing constraints only.

The sub-problem of gate-size optimization under noise constraints is solved as a fixpoint computation [**30**] problem on a complete lattice. The proposed algorithm is guaranteed to converge to the optimal solution, provided it exists. If timing constraints of the circuit are translated to gate-size constraints as in previous approaches, the proposed approach to solving this sub-problem yields the optimal solution to the gate-size optimization problem under noise and timing constraints. The sub-problem for circuit optimization under timing constraints is considered as a geometrical programming problem. An optimal algorithm for simultaneous gate and wire size optimization under timing constraints is presented by Chen *et al.* in [**31**]. We adopt their idea to solving the latter sub-problem. The solutions to the two problems are finally combined to solve the original problem in a Lagrangian Relaxation [**32, 33, 34, 35**] framework. Experimental results demonstrating the effectiveness

of the algorithms are reported for the ISCAS'85 benchmarks [**36**] and larger circuits. We compare our results to the approach where successive iterations of gate-sizing are performed for timing and for noise reduction independently. This alternative design approach is driven by the algorithms used to solving the mentioned sub-problems respectively.

The rest of this chapter is organized as follows. We present our problem formulation in Section 2.1. The gate-size optimization sub-problem under noise constraints is solved as a fixpoint computation problem in Section 2.2. We briefly consider the gate-size optimization sub-problem under timing constraints in Section 2.3. An iterative algorithm based on Lagrangian Relaxation to solving the original problem is proposed in Sections 2.4 and 2.5. We present experimental results and draw conclusions in Section 2.6.

2.1. Problem formulation

2.1.1. Motivation

A circuit optimization problem typically attains to minimize the area or power consumption of a circuit under given timing and other constraints. To avoid functional failures due to coupling, optimizations are performed to constrain the maximal induced noise on each net of the circuit. However, these optimizations should not result in a timing constraint violation. Successive iterations of optimization for noise reduction and timing therefore become necessary, and may yield sub-optimal solutions. This motivates the development of an optimization approach for noise reduction which explicitly considers the timing constraints of the circuit, and yields better solutions than the approach of independent iterations of optimization for noise reduction and timing, respectively.

2.1.2. Circuit modeling

We model a circuit as a directed acyclic graph (DAG). Nodes of the DAG represent gates in the circuit and the edges represent the corresponding nets. A pseudo primary-output node (*PO* henceforth) is added to represent a single output node having incoming edges from all output nodes of the DAG. Similarly, a pseudo primary-input node (*PI* henceforth) having outgoing edges to all input nodes of the DAG is added. Nodes of the graph are topologically sorted with *PO* having index 0 and *PI* having the largest index for ease of notation.

Gates and nets are modeled as standard switch level and π-type RC circuits, respectively. G is used to denote the set of all gates (nodes) in the circuit (graph). The size of a gate i is denoted by $s(i)$. Gate-sizes in the circuit are collectively represented as a gate-size vector S, which is formally defined as :

$$S \triangleq [s(i) : \forall i \in G].$$

$\hat{r_i}$ and $\hat{c_i}$ represent unit gate-size output resistance, and input capacitance per unit gate-size, respectively. The output resistance of any gate i is therefore expressed as $r_i = \hat{r_i}/s(i)$ and the capacitance of an input pin of the gate is thus expressed as $c_i = \hat{c_i}s(i) + f(i)$, where $f(i)$ represents the gate perimeter capacitance. For simplicity, we assume input capacitances of all input pins of a gate to be the same and ignore intrinsic gate delay. However, the proposed algorithm can be simply extended to handle them. We use the Elmore delay model for timing estimations in this chapter.

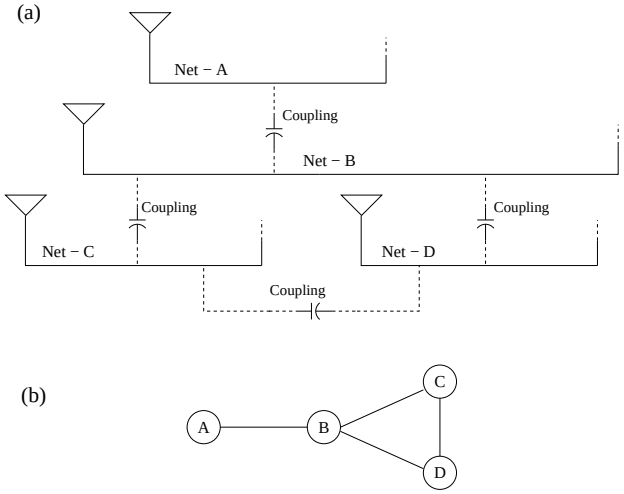

Figure 2.1. (a) A circuit with four coupled nets (b) The corresponding coupling-graph

An undirected *coupling-graph* containing coupling information is superimposed on the DAG. Nodes in the coupling-graph represent the nets of the circuit. Edges in the coupling-graph are called coupling-edges, and are introduced between nodes corresponding to significantly coupled nets. Figure 2.1(a) shows a circuit with nets having significant coupling between them. Figure 2.1(b) shows the corresponding coupling-graph where the nodes represent the nets of the circuit. The coupling-graph model is used for noise estimation and reduction in our proposed approaches.

2.1.3. Problem definition

Given all gate-sizes in a circuit, the noise induced on each net can be calculated using a noise-model. Coupling-noise $N(i)$ on the fanout net of a gate i is formally represented as:

$$N(i) \triangleq f_i(s(i), s(i_1), s(i_2), \ldots, s(i_k))$$

where, $s(i)$ represents the size of gate i, $[i_1, i_2, \ldots, i_k]$ represent driving-gates of nets that are coupled to the fanout net of i, and $[s(i_1), s(i_2), \ldots, s(i_k)]$ represent their sizes, respectively.

Given non-negative weights w_i for each gate i, the objective of the gate-size optimization problem under noise and timing constraints is to find the minimal weighted-sum of gate-sizes under the constraints that the maximal arrival time at the PO with respect to the PI is bounded by a given value A_0, noise $N(i)$ on the fanout net of every gate i is upper bounded by a given value $U(i)$, and the size of each gate i is lower and upper bounded by given values $l(i)$ and $u(i)$, respectively. The noise upper bound $U(i)$ represents the maximum noise a given net can tolerate. Size bounds $l(i)$ and $u(i)$ are given by physical constraints. The timing constraints require that for all paths p from PI to PO, the sum of the delays across components in each path p must be at most A_0. Formally, we attain to solve the following problem.

$$
\begin{aligned}
& min \quad \sum_{i \in G} w_i s(i) \qquad (2.1) \\
& s.t. \\
& \qquad \textstyle\sum_{i \in p} D_i \leq A_0 \qquad \forall p \in P \\
& \qquad N(i) \leq U(i) \qquad \forall i \in G \\
& \qquad l(i) \leq s(i) \leq u(i) \quad \forall i \in G
\end{aligned}
$$

where, D_i is the Elmore delay associated with component i and P is the set of all paths p from PI to PO. Since the total number of paths in P can be exponential, the above formulation is impractical for analysis and optimization. We therefore associate a variable a_i to each component which represents the output arrival time for that component. We now have an

alternate definition of the above problem.

$$\begin{aligned} & min \quad \sum_{i \in G} w_i s(i) \\ & s.t. \end{aligned} \tag{2.2}$$

$$\begin{aligned} a_j &\leq A_0 && \forall j \in input(PO) \\ a_j + D_i &\leq a_i && \forall i \wedge j \in input(i) \\ N(i) &\leq U(i) && \forall i \in G \\ l(i) \leq s(i) &\leq u(i) && \forall i \in G \end{aligned}$$

where, $input(i)$ gives the set of components that are inputs to component i.

2.2. Gate-size optimization under noise constraints

2.2.1. Problem definition

In this section, we consider a sub-problem of the gate-size optimization problem by temporally ignoring the explicit constraints on the arrival times. In addition, we generalize the objective of the problem to determine a gate-size vector S that minimizes any arbitrary cost-function $C(S)$ that is a monotonically non-decreasing function of S. We note that the objective function of our original problem $\sum_{i \in G} w_i s(i)$ is a monotonically non-decreasing

function of S for non-negative weights w_i. The sub-problem is formally defined as the following.

$$min \;\; C(S) \tag{2.3}$$

$$s.t.$$

$$N(i) \leq U(i) \qquad \forall i \in G$$

$$l(i) \leq s(i) \leq u(i) \quad \forall i \in G$$

2.2.2. Gate-sizing transformation

Irrespective of the noise model being used, the noise function $f_i(s(i), s(i_1), s(i_2), \ldots, s(i_k))$ for any gate i, is considered to be monotonically non-increasing on $s(i)$, and monotonically non-decreasing on $s(i_j)$ for any $j \in [1, k]$. This assumption is conservative and is satisfied by any reasonable noise-function. Similar assumptions based on experimental results are used in [**27**]. The monotonic properties for any noise function are formally expressed as the following.

$$s(i) \;<\; s_1(i) \tag{2.4}$$

$$\Rightarrow \qquad f_i(s(i), s(i_1), s(i_2), \ldots, s(i_k)) \;\geq\; f_i(s_1(i), s(i_1), s(i_2), \ldots, s(i_k))$$

$$s(i_j) \;<\; s_1(i_j) \tag{2.5}$$

$$\Rightarrow \quad f_i(s(i), s(i_1), \ldots, s(i_j), \ldots, s(i_k)) \;\leq\; f_i(s(i), s(i_1), \ldots, s_1(i_j), \ldots, s(i_k))$$

These properties are used to formulate a gate-sizing transformation. We define a transformation g_i, which gives the minimal size (within size constraints) of a gate i such that there

are no noise violations on its fanout net. The transformation is a function of the gate-sizes $s(i_1), s(i_2), \ldots, s(i_k)$. Formally, we define g_i as:

$$g_i(s(i_1), s(i_2), \ldots, s(i_k)) \triangleq min\ x \tag{2.6}$$

$$s.t.$$

$$f_i(x, s(i_1), s(i_2), \ldots, s(i_k)) \leq U(i)$$

$$l(i) \leq x \leq u(i).$$

A system of equations is formed when all gate-sizes are optimized with the corresponding g_i transformations as follows.

$$s(i) \ = \ g_i(s(i_1), s(i_2), \ldots, s(i_k)) \quad \forall i \in G \tag{2.7}$$

We use $\mathcal{G} \triangleq [g_i : i \in G]$ to denote the vector transformation of g_i, and write the above equation in vector notation as:

$$S \ = \ \mathcal{G}(S). \tag{2.8}$$

A solution to (2.8) is a gate-size vector S, which is called a fixpoint of $\mathcal{G}$. The following theorem proves that a solution to (2.8) is a necessary condition for a solution to the gate-size optimization sub-problem under noise constraints as defined in (2.3).

Theorem 2.2.1. *If there is a solution to (2.3), there is also a solution to both (2.3) and (2.8).*

Proof. Let S' be the solution to (2.3), and $S'' = \mathcal{G}^k(S')$ such that $\mathcal{G}(S'') = S''$, for some $k > 0$. $\mathcal{G}^k(S)$ represents successive applications of transformation $\mathcal{G}$ on the vector S.

Transformation $\mathcal{G}^k$ on vector S', comprising of gate-sizes $s'(i),\ \forall i \in G$, which satisfies the noise constraints and the size constraints, yields a vector S'' comprising of gate-sizes $s''(i)$ such that $s''(i) \leq s'(i),\ \forall i \in G$. This follows from the definition of g_i in (2.6), which ensures that a gate is never over-sized if it satisfies the noise and size constraints. The obtained relation between the gate-size vectors is expressed as $S'' \leq S'$. It is known that the cost function is monotonically non-decreasing with S. We therefore have the following.

$$C(S'') \ \leq \ C(S')$$

Since S' is a solution to (2.3), $C(S')$ is the minimum value of $C(S)$ for all S which satisfies the noise and size constraints. However, it is known that even S'' satisfies the given constraints and that $C(S'') \leq C(S')$. This implies that

$$C(S'') \ = \ C(S').$$

S'' is thus a solution to (2.3), and also a solution to (2.8) from definition. □

It is, thus, established that if a solution to the gate-size optimization sub-problem exists, so does a solution to the same problem which is also a fixpoint of $\mathcal{G}$. The converse is however not true. An arbitrary fixpoint of $\mathcal{G}$ is not necessarily the solution to the sub-problem since it may not yield the minimum $C(S)$. A fixpoint of $\mathcal{G}$ can be the optimal solution to our sub-problem if and only if it yields the minimal cost-function among all other fixpoints. Consequently, we attain to obtain a fixpoint of $\mathcal{G}$ which is also a solution to the sub-problem defined in (2.3).

We next formally define the relation $\leq$ over two gate-size vectors $X \triangleq [x(i) : i \in G]$ and $Y \triangleq [y(i) : i \in G]$. Since $\leq$ satisfies the properties of reflexivity, transitivity, and antisymmetry, it also forms a partial order on the family of all gate-size vectors.

Definition 2.2.1.

$$X \leq Y \quad \triangleq \quad x(i) \leq y(i) \qquad \forall i \in G$$

Theorem 2.2.2. *$\mathcal{G}$ is a monotonic transformation, that is, if $S_1 \leq S_2$, then $\mathcal{G}(S_1) \leq \mathcal{G}(S_2)$.*

Proof. From Definition 2.2.1, $S_1 \leq S_2$ implies the following.

$$s_1(i) \;\leq\; s_2(i) \quad \forall i \in G \tag{2.9}$$

We use contradiction to prove our claim. If $\mathcal{G}(S_1) \leq \mathcal{G}(S_2)$ is not true, then there must be at least a gate j, the fanout net of which couples to fanout nets of gates $j_1, j_2, \ldots, j_k$ such that

$$m \;>\; n \tag{2.10}$$

where, $m \triangleq g_j(s_1(j_1), s_1(j_2), \ldots, s_1(j_k))$ and $n \triangleq g_j(s_2(j_1), s_2(j_2), \ldots, s_2(j_k))$. From the monotonic property of f_j in (2.5), it is known that

$$f_j(n, s_1(j_1), s_1(j_2), \ldots, s_1(j_k)) \;\leq\; f_j(n, s_2(j_1), s_2(j_2), \ldots, s_2(j_k)). \tag{2.11}$$

From the definitions of n and g_i, we have the following.

$$\begin{aligned} n &= g_j(s_2(j_1), s_2(j_2), \ldots, s_2(j_k)) \\ &\Rightarrow f_j(n, s_2(j_1), s_2(j_2), \ldots, s_2(j_k)) \leq U(j) \\ &\Rightarrow f_j(n, s_1(j_1), s_1(j_2), \ldots, s_1(j_k)) \leq U(j) \quad \text{from (2.11)} \end{aligned}$$

However, from the definitions of m and g_i, we obtain the following.

$$\begin{aligned} m &= g_j(s_1(j_1), s_1(j_2), \ldots, s_1(j_k)) \\ &\Rightarrow m = min\ \{x : (f_j(x, s_1(j_1), s_1(j_2), \ldots, s_1(j_k)) \leq U(j)\} \\ &\Rightarrow m \leq y,\ \forall y : f_j(y, s_1(j_1), s_1(j_2), \ldots, s_1(j_k)) \leq U(j) \\ &\Rightarrow m \leq n, \quad \text{which contradicts (2.10) and proves our claim.} \end{aligned}$$

□

According to lattice theory [**37**], a partially ordered set forms a *complete lattice* if it has a least upper bound and a greatest lower bound on any subset of its elements. A lattice is constructed of all gate-size vectors that satisfy the size constraints. The bottom element of the lattice is a gate-size vector, comprising gate-sizes of which are equal to their individual lower bounds. The top element of the lattice is a virtual gate-size vector, having at least one upper gate-size bound violation. It is a virtual vector as it is a mapping of all gate-size vectors in the partially ordered set with at least one upper size-bound violation. This makes the family of gate-size vectors with the $\leq$ relation a complete lattice.

Given a subset A of elements in a complete lattice L, we use $\bigvee A$ and $\bigwedge A$ to represent the least upper bound and the greatest lower bound of elements in A, respectively. The

existence of a fixpoint in our system is guaranteed by the following theorem due to Knaster and Tarski [**37**].

Theorem 2.2.3 (Knaster-Tarski)**.** *Let L be a complete lattice and $\mathcal{G} : L \rightarrow L$ an order-preserving map. Then $\bigvee\{x \in L : x \leq \mathcal{G}(x)\} \in fix(\mathcal{G})$ where $fix(\mathcal{G})$ is the set of fixpoints of $\mathcal{G}$.*

The above theorem is however, not employed to compute a fixpoint since it is not feasible to compute the set $\{x \in L : x \leq \mathcal{G}(x)\}$. Furthermore, it is not known whether such a fixpoint is also the smallest solution to the gate-sizing problem. We use the iterative method (also called *successive approximation*) instead to find a fixpoint. In this method, one selects an initial solution X_0 and iteratively computes $X_1 = \mathcal{G}(X_0), X_2 = \mathcal{G}(X_1), \ldots$ in the hope of finding an X_n such that $X_n = \mathcal{G}(X_n)$. But the hope may not be fulfilled by starting from any initial point. Fortunately, the bottom and the top elements are good candidates for that.

Following a tradition in lattice theory, $\perp$ and $\top$ are used to represent the bottom and top elements of the complete lattice, respectively. $\perp$ is defined as the vector of lower-bound gate-sizes comprising of $l(i), \forall i \in G$. Since $\perp \leq \mathcal{G}(\perp)$, based on the monotonic property of $\mathcal{G}$, there is an ascending chain $\perp \leq \mathcal{G}(\perp) \leq \mathcal{G}^2(\perp) \leq \cdots$. If the chain has only finite elements, which is true on any finite solution space, the process eventually reaches a fixpoint. The only property that is used of $\perp$ is $\perp \leq \mathcal{G}(\perp)$. Consequently, any solution X_0 such that $X_0 \leq \mathcal{G}(X_0)$ can be used as an initial solution to reach a fixpoint. Since we attain to find the least fixpoint, we start our iterations with the $\perp$ element as our initial solution. The fixpoint obtained is the optimal solution to our sub-problem.

Theorem 2.2.4. *If $fix(\mathcal{G}) = \{S_{f_1}, \ldots, S_{f_l}\}$ denotes the set of fixpoints of $\mathcal{G}$, then there exists a least fixpoint $S_{f_L} \in fix(\mathcal{G})$, defined as $S_{f_L} \triangleq [s_{f_L}(i) = min(s_{f_1}(i), \ldots s_{f_l}(i)) : i \in G]$, such that $S_{f_L} \leq S_{f_j},\ \forall j \in [1, l]$.*

Proof. From the definition of S_{f_L}, there must exist a fixpoint S_{f_j} with the same size for an arbitrary gate k, such that $s_{f_L}(k) = s_{f_j}(k)$ and $s_{f_j}(i) \geq s_{f_L}(i),\ \forall i \in G,\ i \neq k$. Since S_{f_j} is a fixpoint, the noise constraint on all its gates are satisfied. Thus, given the monotonic properties of the noise function in (2.5), S_{f_L} must also satisfy the noise constraints. Additionally since it is the lower bound on all the fixpoints, transformation $\mathcal{G}$ on S_{f_L} yields S_{f_L}. It is thus a fixpoint of $\mathcal{G}$. □

Corollary 2.2.4.1. *The fixpoint reached starting from the $\perp$ element of the lattice is the least fixpoint and is the optimal solution to the gate-size optimization problem for noise reduction, provided the fixpoint $\neq \top$, which implies no solution.*

Corollary 2.2.4.2. *The solution to the gate-sizing problem minimizes any cost-function $C(S)$, which is monotonically non-decreasing with S. It can therefore minimize a set of cost-functions simultaneously. Individual gate-sizes can be consider to be a cost-function each, since they are monotonically non-decreasing with S. This implies that the solution to the gate-sizing problem yields a gate-size vector S which consists of the smallest possible sizes of individual gates, such that coupling-noise on every net is upper bounded and the circuit satisfies given physical constraints.*

Corollary 2.2.4.3. *If timing constraints are incorporated in the gate-size bounds after timing budgeting, the solution reached using the proposed approach is the optimal gate-size*

vector that minimizes the weighted sum of gate-sizes and satisfies both the noise and timing constraints.

The least fixpoint reached is the lower bound on the gate-size vectors that are fixpoints of $\mathcal{G}$, and thus for non-negative weights $w(i)$ yields the minimal weighted sum of gate-sizes. It is thus the solution to the gate-size optimization sub-problem defined in (2.3). The least fixpoint obtained is the optimal solution and is independent of the given weights $[w(i) : i \in G]$.

2.2.3. Scheme of chaotic iterations

Before a good iterative order for updates is defined, it is important to establish its theoretical validity. The scheme of *chaotic iterations* [**38**] ensures that the process will always converge to the same fixpoint, irrespective of the order being used. Transformation $\mathcal{G}$ is composed of a set of partial transformations. In each step, one or more partial transformations are applied to update gate-sizes of some gates, while sizes of all other gates are kept the same. $\mathcal{G}_A$ is used to represent such a partial transformation done in one step, where A represents the points where gate-sizes are updated.

Lemma 2.2.1.

$$(X \leq \mathcal{G}(X)) \Rightarrow (X \leq \mathcal{G}_A(X) \leq \mathcal{G}(X)) \tag{2.12}$$

$$\wedge \quad (X \geq \mathcal{G}(X)) \Rightarrow (X \geq \mathcal{G}_A(X) \geq \mathcal{G}(X))$$

The above lemma states that no matter what evaluation order is used, the generated sequence is monotonic in the same direction and it will not overshoot the fixpoint generated

by $\mathcal{G}$. This guarantees reaching the same fixpoint irrespective of the iteration scheme, starting with the same initial point in the lattice.

2.2.4. Iterative sizing algorithm

We present an algorithm to solving the sub-problem of gate-size optimization under noise constraints. The algorithm is transparent to the timing constraints being incorporated in the size bounds or not. In the former case, the optimal solution attained also guarantees that the timing constraints of the circuit are met, given that the optimal solution does exist. Timing constraints can be incorporated in the size bounds by timing budgeting as in [**27**]. In the latter case, gate-size bounds are given by physical constraints only.

Layout extraction is performed on a given circuit and is used to construct its corresponding DAG and coupling-graph. The size of each gate is initialized to its lower size-bound ($s(i) = l(i),\ \forall i \in G$). Updates ($\mathcal{G}$ transformations) are performed iteratively until all noise violations are eliminated, or it is found that gate-sizing cannot remove all violations. In the latter case, no optimal solution to gate-sizing under noise constraints is declared. For the updates, the noise on the fanout net of node i is calculated based on a noise-model. Driving-gate-sizes of nets coupled to the fanout net of a gate i under consideration are used to determine the value of $N(i)$. If the $N(i)$ exceeds $U(i)$, i is sized up optimally such that $N(i) \leq U(i)$, else no update is performed. If the update in the former case violates the size constraints on i, no solution to the sub-problem is declared. Iterations may optionally be continued to remove other noise-violations. Binary search (for continuous gate-sizing) or linear search (for discrete gate-sizing) is used to find the optimal gate-size during an update. Given n gates in a circuit, and that there exist at most k alternative gates that can be mapped to the same node, the theoretical upper bound on the number of gate-sizing

- Algorithm: Optimal gate-size optimization under noise constraints
- **Input:** Layout extraction results
- **Output:** Optimized gate-size vector, if solution exists
- begin
  (1) construct coupling-graph based on layout extraction
  (2) $s(i) = l(i),\ \forall i \in G$
  (3) do
  (4) for each gate $i \in G : N(i) > U(i)$
  (5) $s(i) = g_i(s(i_1), s(i_2), \ldots, s(i_k))$
  (6) while $((\exists i \in G : N(i) > U(i)) \ \wedge \ (s(i) \leq u(i), \forall i \in G))$
- end

Figure 2.2. Optimal algorithm for gate-size optimization under noise constraints

iterations is nk. The pseudo-code of the proposed algorithm is shown in Figure 2.2. The order of the iterations may only alter the rate of convergence, but it is certain to reach the optimal solution in any case, provided it exists. We present two possible iterative schemes for node updates next.

- **List traversal :** A list of all nodes is maintained and necessary updates are performed sequentially. This is repeated until no updates are found necessary during the entire traversal to yield the optimal solution. However, if a gate exceeds given size-constraints, iterations are stopped and no solution for the circuit is declared. Alternately, that node is ignored in future iterations and the algorithm tries to eliminate noise violations on other nodes. In either case, no solution to complete noise violation elimination by gate-sizing is declared.
- **Queue traversal :** In this scheme, a queue is initially filled with nodes whose fanouts have noise violations. As a node i is popped from the queue, it is sized up to eliminate the noise violation on its fanout net. All nodes with fanout nets having a coupling-edge (in the coupling-graph) from the fanout net of node i are pushed in the queue, if they have noise violations and are not already in the queue.

Iteration stops either when the queue is empty or when a driver-size exceeds the given constraint and no solution is concluded. As in List traversal, iterations may be continued optionally to remove remaining violations.

2.2.5. Results of gate-size optimization under noise constraints

Experimental results of the proposed algorithm are presented for the ISCAS'85 benchmarks and three larger circuits, namely *CKT_1*, *CKT_2*, and *CKT_3*. The larger circuits and the parameters for all benchmarks are randomly generated with realistic parameters from a $0.18\mu m$ technology library. The 2π model [**39**] is used as the noise-model for all simulations. For the test circuits, the driver resistance R_d is from $20 - 2000\Omega$, loading capacitance C_l is from $4 - 50$fF, and the slew is from $10 - 300$ps. Gate-size bounds used do not incorporate timing constraints.

Table 2.1 presents the results obtained. The queue traversal method is used in the experiments. Optimization using list traversal yields similar results, but the queue traversal method is faster on the average. For all benchmarks, we present the number of nodes, the number of coupling-edges, and the number of nets having noise violation before and after optimization. We present noise reduction results as a percentage of nets with noise violations fixed with reference to the initial number of nets with violations. The number of initial violations in the table represents the the number of nets having violations in the original circuit. The increase in the weighted sum of gate-sizes for unit weights is found to be less than 2% as compared to the original circuit. In some cases, and under tighter noise bounds, an optimal solution to the sub-problem does not necessarily exist. The noise reduction result in this case reflect the number of violations the algorithm removes when it quits.

Table 2.1. Noise reduction results : $U(i) = 0.2V_{dd}$

Circuit	# of Nodes	# of C Edges	# of Initial Violations	# of Final Violations	% Noise Reduction
C432	198	553	2	1	100
C499	245	621	3	0	100
C880	445	1240	3	0	100
C1355	589	1653	12	0	100
C1908	915	2655	10	0	100
C2670	1428	3851	32	1	97
C3540	1721	5086	16	3	81
C5315	2487	7076	17	0	100
C6288	2450	7239	15	0	100
C7552	3721	10823	33	0	100
CKT_1	15K	42017	67	0	100
CKT_2	18K	53114	83	1	99
CKT_3	20K	59389	140	0	100

Run time for the benchmarks range between $0.1 - 10.9$ secs using the queue traversal, and between $0.5 - 24$ secs for list traversal. The algorithm and the framework are written in the C language. XVCG [**40**] is integrated with the simulation environment for coupling-graph visualizations. Results presented are obtained by simulations on a Pentium 2.4GHz Xeon processor server, having 1GB RAM and running RedHat Linux 8.0.

2.3. Gate-size optimization under timing constraints

Based on the notations defined earlier, the gate-size optimization sub-problem under timing constraints is defined as the following.

$$min \quad \sum_{i \in G} w_i s(i) \tag{2.13}$$

$$s.t.$$

$$a_j \leq A_0 \qquad \forall j \in input(PO)$$

$$a_j + D_i \leq a_i \qquad \forall i \wedge j \in input(i)$$

$$l(i) \leq s(i) \leq u(i) \quad \forall i \in G$$

This problem is often considered to be a geometrical programming problem. An extended problem is the gate and wire-sizing problem under timing constraints, which is solved optimally in [**31**]. We use their approach to solving this sub-problem and combine our approach to gate-size optimization under noise constraints in the next section to solve the circuit optimization problem under noise and timing constraints defined in (2.2), in a Lagrangian Relaxation framework.

2.4. Optimization under noise and timing constraints by Lagrangian Relaxation

2.4.1. Lagrangian Relaxation

Lagrangian Relaxation (LR) is a widely acclaimed technique for solving constrained optimization problems. In LR, "troublesome" constraints are "relaxed" and incorporated into the objective function after multiplying them with non-negative constants called *Lagrange multipliers*, one multiplier for each constraint. For each fixed vector λ of the *Lagrange multiplier* introduced, we obtain an easier sub-problem called the Lagrangian Relaxation sub-problem associated with λ ($\mathcal{LRS}/\lambda$). The problem of finding a λ such that the solution to the sub-problem is also the solution to the original constrained problem is called the Lagrangian Dual problem ($\mathcal{LDP}$). Kuhn-Tucker conditions [**34**] are used to simplify $\mathcal{LRS}/\lambda$ to a simpler sub-problem denoted by $\mathcal{LRS}/\mu$. We use the method of sub-gradient optimization [**32, 34**] to solve the $\mathcal{LDP}$.

We relax the constraints on the arrival time of the gates since they are difficult to handle. Analytical noise models are complex in form and cannot be expressed as posynomials. We leave the noise and size bound constraints un-relaxed. This facilitates the approach of gate-sizing under noise constraints as a fixpoint computation, which we described earlier. In addition, the objective function of $\mathcal{LRS}/\lambda$ can be expressed as a posynomial.

Following the Lagrangian Relaxation procedure, we introduce a Lagrange multiplier for each constraint on arrival time. For all $j \in input(PO)$, we introduce λ_{j0} for the constraint $a_j \leq A_0$. For all $j \in input(i) \wedge i \neq 0$, we introduce λ_{ji} for the constraint $a_j + D_i \leq a_i$. We assume that the arrival time of PI is 0. We denote the vector of Lagrange multipliers introduced by λ. The vector of arrival times is denoted by $A \triangleq [a_i : i \in G]$.

The LR sub-problem associated with λ ($\mathcal{LRS}/\lambda$) is defined as the following.

$$min\ L_\lambda(S, A) \tag{2.14}$$

$$s.t.$$

$$N(i) \leq U(i) \qquad \forall i \in G$$

$$l(i) \leq s(i) \leq u(i) \quad \forall i \in G$$

where, $L_\lambda(S, A)$ is given by

$$\sum_{i \in G} w_i s(i) + \sum_{j \in input(PO)} \lambda_{j0}(a_j - A_0) + \sum_{i \in G} \sum_{j \in Input(i)} \lambda_{ji}(a_j + D_i - a_i).$$

Kuhn Tucker [**34**] conditions imply the following optimality condition on λ. The proof is presented in [**31**].

$$\sum_{i \in output(k)} \lambda_{ki} = \sum_{j \in input(k)} \lambda_{jk} \quad \forall k$$

The above conditions on λ are used to simplify $\mathcal{LRS}/\lambda$. We define $\Omega_\lambda = (\lambda \geq 0 : \lambda)$ to denote the set of λ that satisfies the above conditions. For any $\lambda \in \Omega_\lambda$, $\mathcal{LRS}/\lambda$ is equivalent to solving a new sub-problem (proof in [**31**]) called $\mathcal{LRS}/\mu$, which is defined formally as the

following.

$$min\ L_\mu(S) \tag{2.15}$$

$$s.t.$$

$$N(i) \leq U(i) \qquad \forall i \in G$$

$$l(i) \leq s(i) \leq u(i) \quad \forall i \in G$$

where, vector $\mu \triangleq [\mu_i : i \in G]$, $\mu_i \triangleq \sum_{j \in input(i)} \lambda_{ji}$, and $L_\mu(S) \triangleq \sum_{i \in G} \mu_i D_i + \sum_{i \in G} w_i s(i)$. We solve $\mathcal{LRS}/\lambda$ by solving $\mathcal{LRS}/\mu$ instead. Vector A is evaluated by topologically considering each variable a_i from the PI and setting it to the smallest possible value that satisfies the timing constraints.

2.4.2. Solving the $\mathcal{LRS}/\mu$

A local refinement function is derived to solve the Lagrangian sub-problem and is applied to gates iteratively until convergence. We define $upstream(i)$ to be the set of resistor indexes on the path(s) from gate i to the nearest upstream gate(s) or input driver(s). Let R_i ($\triangleq \sum_{j \in upstream(i)} \mu_j r_j$) and C_i denote the weighted upstream resistance and the downstream capacitance of gate i, respectively.

We define h_i to be a function of gate-sizes in a circuit that returns the optimal size of gate i that locally minimizes $L_\mu(S)$, given other gate-sizes. This is similar to the local refinement function derived for timing optimization in [**31**] and is formally represented as the following.

$$h_i(S) \ \triangleq \ (min\ s(i) :\ L_\mu(S)) \tag{2.16}$$

Based on the Elmore delay model, $L_\mu(S)$ can be expressed as a function of $s(i)$ assuming all other gate-sizes are fixed. Mathematically, we have

$$L_\mu = A_i(S) \cdot s(i) + \frac{B_i(S)}{s(i)} + E_i(S) \tag{2.17}$$

where, $A_i(S)$, $B_i(S)$, and $E_i(S)$ are functions of gate-sizes of the circuit, but independent of $s(i)$. $A_i(S)$ is given by $\hat{c}_i R_i + w_i$, and $B_i(S)$ is given by $\mu_i \hat{r}_i C_i$ [**31**]. From the above equation, the optimal value of $s(i)$ is obtained that minimizes L_μ. Mathematically, we evaluate h_i as follows.

$$h_i(S) \;=\; \sqrt{\frac{B_i(S)}{A_i(S)}} \;=\; \sqrt{\frac{\mu_i \hat{r}_i C_i}{\hat{c}_i R_i + w_i}} \tag{2.18}$$

We observe that L_μ is a convex function in $s(i)$, and achieves the minimum at $s(i) = h_i(S)$. However, the size bounds on gate i constrain the feasible region to $s(i) \in [l(i),\ u(i)]$. If $h_i(S)$ lies outside the feasible region, the optimal value of $s(i)$ that minimizes L_μ under the size bounds is one of the size bounds depending on the side of the feasible region that $h_i(S)$ lies in. To satisfy the noise constraint in addition to the size bounds, we further constrain the feasible region to the right of our previously derived transformation g_i for noise reduction. From the monotonic property of noise functions, it is known that any $s(i)$ larger than g_i satisfies the noise constraint on the fanout of i. The optimal value of $s(i)$ that minimizes L_μ under the noise and size constraints is therefore given by the larger of g_i and h_i but constrained within the size bounds.

We next define $\phi_i(S)$ to be the a refinement function for a gate i which returns its optimal size that minimizes L_μ, attains to satisfy the noise constraint on its fanout net and is within its size constraints. The noise constraint is satisfied if $g_i \leq s(i)$. Formally, we express ϕ_i as

- Algorithm: *Solve* $\mathcal{LRS}/\mu$
- **Output:** Optimized gate-size vector S
- begin
  (1) $s(i) = l(i),\ \forall i \in G$
  (2) do
  (3) for each gate $i \in G$ in topological order
  (4) evaluate C_i
  (5) for each gate $i \in G$ in reverse topological order
  (6) evaluate R_i
  (7) $s(i) = \phi_i(S)$
  (8) until (no further improvement)
- end

Figure 2.3. Algorithm to solve the $\mathcal{LRS}/\mu$

the following.

$$\phi_i(S) \ \stackrel{\triangle}{=}\ \min\Big(u(i),\ max\big(l(i),\ g_i(S),\ h_i(S)\big)\Big) \tag{2.19}$$

$\mathcal{LRS}/\mu$ is solved by a greedy algorithm based on iteratively resizing the gates. Local refinements of $\phi_i(S)$ are performed on every gate iteratively assuming all other gate-sizes are fixed. R_i and C_i are evaluated for each gate by traversing the circuit in a reverse topological order and in a topological order, respectively. g_i for every gate is evaluated based on the noise model used. We present the pseudo code of the iterative algorithm in Figure 2.3.

A system of equations is formed when locally refined driving-gate-sizes for all nets are combined as $s(i) = \phi_i(S), \forall i \in G$. The transformation is denoted in vector form as $\Phi \stackrel{\triangle}{=} [\phi_i : i \in G]$ and we represent the system of equations in a vector form as the following.

$$S = \Phi(S) \tag{2.20}$$

A solution to (2.20) is a gate-size vector S, called a fixpoint of Φ. We prove in the following theorem that Φ is monotonic, and under constraints, a convergent transformation.

Theorem 2.4.1. *$\Phi(S)$ is a monotonic transformation over the partial order $\leq$, that is, for two gate-size vectors S_1 and S_2, given $S_1 \leq S_2$ implies $\Phi(S_1) \leq \Phi(S_2)$.*

Proof. We use contradiction to assert the theorem. If $\Phi(S_1) \leq \Phi(S_2)$ is not true, then there must be at least a gate j, such that

$$\begin{aligned}
&\phi_j(S_1) \ > \ \phi_j(S_2) \\
\Rightarrow \quad &\min(u(j), \max(l(j),\ g_j(S_1),\ h_j(S_1))) > \min(u(j), \max(l(j),\ g_j(S_2),\ h_j(S_2))) \\
\Rightarrow \quad &\max(g_j(S_1),\ h_j(S_1)) > \max(g_j(S_2),\ h_j(S_2)) \\
\Rightarrow \quad &(g_i(S_1) > g_i(S_2)) \ \vee \ (h_i(S_1) > h_i(S_2)). \qquad (2.21)
\end{aligned}$$

However, g_i and h_i are monotonic as shown earlier and in [**31**], respectively. This is formally expressed as follows.

$$(g_i(S_1) \ \leq \ g_i(S_2)) \ \wedge \ (h_i(S_1) \ \leq \ h_i(S_2)) \ \ \forall i \in G$$

The monotonic properties of g_i and h_i contradict the results obtained above (2.21). This proves that $\Phi(S)$ is a monotonic transformation. □

2.4.3. Solving the $\mathcal{LDP}$

Let the function $Q(\lambda)$ be the solution to $\mathcal{LRS}/\lambda$. We define the $\mathcal{LDP}$ as the following.

$$max \ Q(\lambda) \qquad (2.22)$$

$$s.t.$$

$$\lambda \geq 0$$

$Q(\lambda)$ is not differentiable in general. Methods like steepest descent, which depend on gradient directions, are not necessarily applicable. The sub-gradient optimization method is used instead, and can be viewed as a generalization of the steepest descent method in which the gradient direction is substituted by a sub-gradient based direction [**34**]. This method however, cannot guarantee the optimal solution in all cases.

Starting with an arbitrary value of λ, the method iteratively moves from the current point to a new point following the sub-gradient direction. At step k, we solve the Lagrangian sub-problem using the *Solve* $\mathcal{LRS}/\mu$ algorithm. Subsequently, for each relaxed constraint, we define the sub-gradient to be the right side minus the left hand side of the constraint, evaluated at the current solution. The sub-gradient direction is the vector of all the sub-gradients. We move to a new point by multiplying a step size ρ_k to the sub-gradient direction and adding it to λ. Finally, λ is projected back to the nearest point in Ω_λ, so that we can solve $\mathcal{LRS}/\mu$ instead of $\mathcal{LRS}/\lambda$ in the next iteration. This is repeated until convergence.

If the step size sequence ρ_k satisfies the two conditions $lim_{k\rightarrow\infty}\rho_k = 0$ and $\sum_{k=1}^{\infty}\rho_k = \infty$, then the sub-gradient optimization method will always converge to the optimal solution for a convex problem. The approach to solving $\mathcal{LDP}$ by sub-gradient optimization is a standard approach and is similar to the approach in [**31**]. The pseudo code of the *Solve* $\mathcal{LDP}$ algorithm is given in Figure 2.4.

2.5. Gate-size optimization algorithm

Based on the problem formulation and the Lagrangian Relaxation procedure described in the previous section, we present the pseudo code of a gate-size optimization algorithm that minimizes the weighted sum of gate-sizes under given noise, timing and physical gate-size constraints in Figure 2.5.

- Algorithm: *Solve* $\mathcal{LDP}$
- **Output:** λ that maximizes $\mathcal{LRS}/\lambda$
- begin
  (1) $k = 1$
  (2) λ = arbitrary initial vector in Ω_λ
  (3) $\mu_i = \sum_{j \in input(i)} \lambda_{ji}, \quad \forall i \in G$
  (4) call *Solve* $\mathcal{LRS}/\mu$
  (5) evaluate $A = [a_i : i \in G]$
  (6) for each gate $i \in G$
  (7) for all $j \in input(i)$
  (8) $$\lambda_{ji} = \begin{cases} \lambda_{ji} + \rho_k(a_j - A_0) & \text{if } i = PO \\ \lambda_{ji} + \rho_k(a_j + D_i - a_i) & \text{otherwise} \end{cases}$$
  (9) project λ to the nearest point in Ω_λ
  (10) $k = k + 1$
  (11) Repeat steps 3-10 until $(\sum_{i \in G} w_i s(i) - Q(\lambda)) \leq$ Error bound.
- end

Figure 2.4. Algorithm to solve the $\mathcal{LDP}$

- Algorithm: Gate-size optimization under noise and timing constraints
- **Input:** Layout extraction results
- **Output:** Optimized gate-size vector S
- begin
  (1) construct DAG for circuit
  (2) superimpose coupling-graph from layout extraction
  (3) call *Solve* $\mathcal{LDP}$ to get optimized λ
  (4) evaluate μ from λ $(\mu_i = \sum_{j \in input(i)} \lambda_{ji}, \forall i \in G)$
  (5) call *Solve* $\mathcal{LRS}/\mu$ to get optimized S
- end

Figure 2.5. Algorithm for gate-size optimization under noise and timing constraints

Lemma 2.5.1. *The solution S_{opt} obtained by the gate-size optimization algorithm under noise constraints (Figure 2.2) is a lower bound to the optimal solution of the gate-size optimization problem under noise and timing constraints.*

The developed algorithm for gate-size optimization under noise constraints is used to obtain the updated lower size bound of every gate in the original gate-size optimization

problem under noise and timing constraints. This is because S_{opt} gives the minimum size of each gate for which noise violations do not exist. Here we assume that the initial size bounds used in both the algorithms are the same.

Lemma 2.5.2. *If the optimal solution S^* to the gate-size optimization sub-problem under timing constraints (2.13) has a partial order over the relation $\leq$ with the optimal solution S^{**} to our gate-size optimization problem under noise and timing constraints, the converged solution of our proposed algorithm is guaranteed to be the optimal solution.*

We define $H(S) \triangleq [h_i(S) : i \in G]$ as the vector transformation of the local refinement function $h_i(S)$ defined in (2.16). Since S^* is the optimal solution to the gate-size optimization sub-problem under timing constraints, we have $S^* = H(S^*)$ under size constraints. Given that $S^* \leq S^{**}$, the monotonic property of H implies that $H(S^*) \leq H(S^{**})$. We therefore obtain $S^* \leq H(S^{**}) \leq S^{**}$. In addition, it is intuitive that $\mathcal{G}(\mathcal{S}^{**}) \leq S^{**}$. This implies that the optimal solution S^{**} is a fixpoint of Φ, that is $\Phi(S^{**}) = S^{**}$, since S^{**} yields the minimum sum of gate-sizes. Given the monotonic property of $\Phi(S)$, it is therefore proved that our algorithm yields the optimal solution.

2.6. Experimental results and conclusions

We present results of the proposed algorithm for gate-size optimization under noise and timing constraints on the ISCAS'85 benchmarks [**36**] and three larger benchmarks, parameters for which are randomly generated with realistic values in a 0.18μ technology. We use the 2-π noise model [**39**] for noise estimations. The upper noise bound $U(i)$ on the nets is set to $0.2V_{dd}$. For comparison, we consider four optimization stages, namely *Unoptimized*, *Timing*, *Timing* $\rightarrow$ *Noise*, and *Timing* + *Noise*.

Table 2.2 presents the number of nodes (*# of Nodes*), number of coupling-edges in the coupling-graph for the circuit (*# of C Edges*) and results obtained for the four stages mentioned above. Results for each stage include the weighted-sum of all gate-sizes (*Area*), the arrival time at the *PO* of the circuit (*Ta*), and number of noise violations in the circuit (NV).

Unoptimized represents a circuit stage satisfying a given timing constraint, but without area optimization or noise consideration. For ease of comparison, we normalize the *Area* and *Ta* values for this stage and the corresponding values for the other stages are given with respect to this stage.

Timing represents the circuit stage after it is optimized for minimal weighted sum of gate-sizes under given timing constraints. The approach is similar to the work in [**31**]. As expected, this stage is usually the optimal in terms of area, but has the most number of noise violations. Since reaching the exact optimal using the LR approach is time consuming, we accept a small error bound for the loop termination in the *Solve* $\mathcal{LDP}$ algorithm. This makes the final solution close to the optimal, but not necessarily the exact optimal.

Timing $\rightarrow$ *Noise* represents the circuit stage after gates of circuit in *Timing* are sized up using the algorithm for gate-size optimization under noise constraints (Figure 2.2). We constrain the size bounds such that the timing constraints are not violated during sizing. Results demonstrate increase in area during this stage, and decrease in number of noise-violations, with little effect on timing. In a few cases, the circuit is timed faster, but experiments do not show any timing violation caused by the optimal coupling-noise reduction algorithm.

Timing + *Noise* represents the circuit stage after the *Unoptimized* circuit is optimized with our proposed algorithm for gate-size optimization under noise and timing constraints. Though the area obtained is found to be more than the *Timing* stage on the average, it

Table 2.2. Noise reduction results : $U(i) = 0.2V_{dd}$

Circuit	Unoptimized			Timing			Timing → Noise			Timing + Noise		
	Area	Ta	NV	Area	Ta	NV	Area	Ta	NV	Area	Ta	NV
C432	1.00	1.00	3	0.54	0.99	2	0.60	0.99	1	0.54	0.99	0
C499	1.00	1.00	4	0.43	0.99	3	0.49	0.99	0	0.43	0.99	0
C880	1.00	1.00	2	0.75	0.99	3	0.76	0.99	0	0.73	1.00	0
C1355	1.00	1.00	6	1.11	0.99	12	1.13	0.99	0	1.12	0.99	0
C1908	1.00	1.00	10	0.66	0.95	10	0.67	0.94	0	0.66	0.96	0
C2670	1.00	1.00	14	0.58	1.00	32	0.60	1.00	1	0.60	1.00	1
C3540	1.00	1.00	10	0.78	0.96	16	0.79	0.93	3	0.80	0.92	3
C5315	1.00	1.00	15	0.50	1.00	17	0.50	1.00	0	0.50	1.00	0
C6288	1.00	1.00	16	0.86	0.99	15	0.87	0.99	0	0.87	0.99	0
C7552	1.00	1.00	20	0.49	1.00	33	0.50	1.00	0	0.50	1.00	0
CKT_1	1.00	1.00	53	0.21	1.00	67	0.22	1.00	0	0.22	1.00	0
CKT_2	1.00	1.00	73	0.23	0.98	83	0.23	0.98	1	0.24	1.00	0
CKT_3	1.00	1.00	116	0.28	0.99	140	0.28	0.99	0	0.28	0.99	0

is found to be very effective in reducing noise violations and has lower area penalty in comparison to *Timing* → *Noise*.

Table 2.3 presents similar results for a further constrained noise upper bound ($U(i) = 0.1V_{dd}$). We observe that gate-sizing is inadequate in removing all noise violations in this case. This implies that gate-sizing for noise reduction is a good technique for post-route noise fixes on some nets, and it is important to consider alternate noise reduction techniques in earlier stages of design. Experimental results prove the effectiveness of our algorithm over the approach of optimizing for area and then for noise. In comparison, the proposed approach is shown to reduce larger number of noise violations, which is significant in cases of constrained $U(i)$ and is efficient for area optimization. The algorithm takes 0.8 seconds for the smallest benchmark and 1.4 minutes for the largest benchmark on a Pentium 2.4GHz Xeon processor server, having 1GB RAM and running RedHat Linux 8.0.

Table 2.3. Noise reduction results : $U(i) = 0.1V_{dd}$

Circuit	Unoptimized			Timing			Timing → Noise			Timing + Noise		
	Area	Ta	NV	Area	Ta	NV	Area	Ta	NV	Area	Ta	NV
C432	1.00	1.00	24	0.54	0.99	28	0.78	0.97	5	0.70	0.99	3
C499	1.00	1.00	32	0.67	1.00	35	0.84	0.96	12	0.75	1.00	9
C880	1.00	1.00	37	0.75	0.99	52	0.87	0.99	15	0.80	0.99	10
C1355	1.00	1.00	43	1.11	0.99	65	1.22	0.92	21	1.16	0.99	21
C1908	1.00	1.00	88	0.66	0.95	111	0.93	0.94	27	0.64	0.96	25
C2670	1.00	1.00	146	0.58	1.00	172	0.74	1.00	40	0.62	1.00	30
C3540	1.00	1.00	166	0.78	0.96	220	0.88	0.93	61	0.82	0.98	39
C5315	1.00	1.00	264	0.50	1.00	297	0.66	0.99	50	0.54	1.00	38
C6288	1.00	1.00	220	0.86	0.99	282	0.99	0.90	80	0.93	0.94	62
C7552	1.00	1.00	360	0.49	1.00	414	0.65	0.99	72	0.56	1.00	57
CKT_1	1.00	1.00	1136	0.32	1.00	1272	0.42	0.89	108	0.32	1.00	87
CKT_2	1.00	1.00	1544	0.23	0.98	1573	0.34	0.92	100	0.32	0.98	94
CKT_3	1.00	1.00	1783	0.28	0.99	1860	0.40	0.95	158	0.35	0.99	132

CHAPTER 3

Yield Driven Gate-size Optimization under Noise and Variation

In Chapter 2, we presented algorithms for gate-size optimization under noise and timing constraints. However, we did not consider the effects due to variability, which affects coupling between nets and can cause the noise induced on a net to overshoot its given constraint. Yield in this chapter denotes the probability that all nets of a circuit satisfy coupling-noise constraints, under the effects of process variations. The desire to guarantee a given yield for a circuit motivates development of an approach to solving the yield driven gate-size optimization problem under noise and variation.

In this chapter, we propose an iterative algorithm for gate-size optimization that constrains the yield loss under process variations. The algorithm is presented as an extension to the algorithm for gate-size optimization under noise constraints that we presented in Chapter 2.2. We develop a new gate-sizing transformation and prove that the solution to our problem is a fixpoint of this transformation. The effectiveness of the algorithm is validated by simulations on the ISCAS'85 benchmarks [**36**] and three larger circuits. Results are compared with those from the algorithm for gate-size optimization under noise constraints (Figure 2.2) and to an alternative design approach which uses a guard-band during sizing to account for uncertainty due to variation.

The rest of this chapter is organized as follows. The yield driven gate-size optimization problem under noise and variation is formulated in Section 3.1. We develop a transformation to solving the problem as a fixpoint computation problem in Section 3.2. An iterative

yield driven gate-sizing algorithm is proposed in Section 3.3. Experimental results and comparisons to alternative approaches are presented in Section 3.4.

3.1. Problem formulation

3.1.1. Gate-sizes as random variables

Fabricated gate-sizes cannot be accurately predicted due to variations in the manufacturing process. Probabilistic approaches model the uncertainty in the gate parameters like gate-length, dopant concentration and gate-thickness as random variables. We consider the driving gate-size of a net i as a Gaussian random variable $s(i) \sim N(\mu_i, \sigma_i^2)$, having a mean value of μ_i and variance σ_i^2. However, the theory and concepts behind our proposed approach are not limited to a Gaussian distribution and apply to other distributions as well.

3.1.2. Problem definition

The syntax of single notation quantification is used for all mathematical relations and expressions in this chapter. This is similar to the one advocated by Gries and Schneider [**41**]. The general form of a quantification over a binary operator $\star$ is exemplified by

$$(\star x : R : P)$$

where, variable x is called the *bound variable* or *dummy* of the quantification, R is a boolean expression called the *range* of the quantification such that values assumed by x satisfy R, and P is an expression which is called the *body* of the quantification. The above expression denotes the application of the operator $\star$ to the values of P for all x for which range R is

true. For example, $\sum_{i=1}^{n} x_i$ would be represented as $(+i : 1 \leq i \leq n : x_i)$ and $\forall x\ R \rightarrow P$ as $(\forall x : R : P)$.

Given all gate-sizes in a circuit, the noise induced on each net can be calculated using a noise-model. It is not necessary that the noise-model be an analytical one. Based on the coupling-graph model as introduced in Chapter 2.1.2, the noise $\eta(i)$ on a given net i is represented as:

$$\eta(i) \triangleq f_i(s(i), s(i_1), \ldots, s(i_k)) = f_i(N(\mu_i, \sigma_i^2), N(\mu_{i_1}, \sigma_{i_1}^2) \ldots N(\mu_{i_k}, \sigma_{i_k}^2))$$

where, $i_1, \ldots, i_k$ represent nets which couple to i, and $s(i_1), \ldots, s(i_k)$ represent their driving-gate-sizes, respectively. Nominal driving gate-sizes of all nets in the circuit are represented by a nominal gate-size vector M as the following.

$$M \triangleq (i : 0 \leq i < n : \mu_i)$$

A circuit is considered to satisfy the noise constraint if coupling-noise $\eta(i)$ on every net i is upper bounded by a given value U_i. The nominal driving-gate-size μ_i of a net i is bounded by $l(i)$ and $u(i)$, which represent minimum and maximum bounds on the gate-size respectively, and are given by physical and timing constraints.

The objective of the optimal gate-size optimization problem under noise and variation is to determine the minimal gate-size vector M such that a given lower bound on the yield is guaranteed. This optimization problem is approached in two stages. Given that the desired yield for a design (henceforth referred as the *global yield*) is distributed into probabilities of no noise violation on each net (henceforth referred to as *local yield*), the yield driven gate-size optimization problem attains to find minimal nominal gate-sizes that achieve the local yields

throughout the circuit. In the extreme case, a global yield of 100% is distributed as desired local yields of 100% throughout the circuit. This chapter presents a method to determine minimal nominal gate-sizes that achieve the desired yields, under given physical and timing constraints incorporated into the bounds on each gate-size.

3.2. Gate-sizing as a fixpoint computation

Irrespective of the noise model being used, the noise function $f_i(s(i), s(i_1), s(i_2), \ldots, s(i_k))$ for any net i, is considered to be monotonically non-increasing on $s(i)$ and monotonically non-decreasing on $s(i_j)$ for any $j \in [1, k]$. This assumption is conservative and is satisfied by any reasonable noise-function. Similar assumptions based on experimental results are used in [**27**]. The monotonic properties of any noise function are formally expressed as the following.

$$s(i) \; < \; s_1(i) \tag{3.1}$$

$$\Rightarrow \qquad f_i(s(i), s(i_1), s(i_2), \ldots, s(i_k)) \; \geq \; f_i(s_1(i), s(i_1), s(i_2), \ldots, s(i_k))$$

$$s(i_j) \; < \; s_1(i_j) \tag{3.2}$$

$$\Rightarrow \quad f_i(s(i), s(i_1), \ldots, s(i_j), \ldots, s(i_k)) \; \leq \; f_i(s(i), s(i_1), \ldots, s_1(i_j), \ldots, s(i_k))$$

Based on these monotonic properties, we define a new transformation g_i as a function of the driving-gate-sizes of nets that couple to i. It gives the minimal driving-gate-size of i satisfying physical and timing constraints, such that the probability of i satisfying its noise constraint is at least Y_i, which denotes the desired local yield of net i.

Formally, g_i is defined as the following.

$$g_i(\mu_{i_1}, \mu_{i_2}, \ldots, \mu_{i_k}) \triangleq \tag{3.3}$$

$$(min\ x : [Pr(f_i(N(x, \sigma_i^2), s(i_1), \ldots, s(i_k)) \leq U_i)) \geq Y_i] \bigwedge [l(i) \leq x \leq u(i)] : x)$$

where, $Pr(k)$ denotes the probability of event k.

A system of equations is thus formed when driving-gate-sizes for all nets are optimized as:

$$(\forall i : 0 \leq i < n : \mu_i = g_i(\mu_{i_1}, \mu_{i_2}, \ldots, \mu_{i_k})).$$

If nominal gate-sizes and the transformations are represented as vectors $M = (i : 0 \leq i < n : \mu_i)$, and $G = (i : 0 \leq i < n : g_i)$, the above equation can be written as the following.

$$M = G(M) \tag{3.4}$$

A solution to (3.4) is a gate-size vector M, which is called a fixpoint of G. (3.4) is next shown to be a necessary condition for the solution to the yield driven gate-size optimization problem.

Theorem 3.2.1. *The solution to the yield driven gate-size optimization problem is also a solution to (3.4), that is, a fixpoint of the G transformation.*

Proof. Let M' be the solution to the yield driven gate-size optimization problem, and $M'' = G^k(M')$ such that $G(M'') = M''$, for some $k > 0$. $G^k(M)$ represents successive applications of transformation G on the vector M. Transformation G^k on vector M' that satisfies the yield and size constraints, produces vector M'' such that driving-gate-size of every net in M'' is less than or equal to the same in M'. This follows from the definition

of g_i in (3.3), which ensures that a gate is never over-sized if it satisfies the yield and size constraints. The obtained relation between the gate-size vectors is expressed as $M'' \leq M'$. However, M' is the minimal vector of gate-sizes satisfying yield and size constraints. Since M'' also satisfies the yield and size constraints, and $M'' \leq M'$, it is evident that M'' and M' are identical. Fixpoint M'' is therefore shown to be the solution to the yield driven gate-sizing problem. □

It is, thus, established that a solution to the yield driven gate-size optimization problem is also a fixpoint of G. However, the converse is not true. An arbitrary fixpoint of G may not be a minimal solution to the optimization problem. A gate-size vector that satisfies (3.4) can be a solution to the original problem if and only if it yields the minimal gate-size vector among all other fixpoints of G.

Theorem 3.2.2. *G is a monotonic transformation, that is, if $M_1 \leq M_2$, then $G(M_1) \leq G(M_2)$.*

Proof. From Definition 2.2.1, $M_1 \leq M_2$ implies the following.

$$(\forall i : 0 \leq i < n : \mu_{1_i} \leq \mu_{2_i}) \tag{3.5}$$

We use contradiction to prove our claim. If $G(M_1) \leq G(M_2)$ is not true, we must have the following.

$$(\ \exists j : 0 \ \leq \ j \ < \ n : [m \ \triangleq \ g_j(\mu_{1_{j_1}}, \ldots, \mu_{1_{j_k}})] \ > \ [n \ \triangleq \ g_j(\mu_{2_{j_1}}, \ldots, \mu_{2_{j_k}})]\ ) \tag{3.6}$$

From the monotonic property of f_j in (3.2), it is known that

$$Pr(f_j(N(n, \sigma_j^2), s_1(j_1), \ldots) \leq U_j) \ \ \geq \ \ Pr(f_j(N(n, \sigma_j^2), s_2(j_1), \ldots) \leq U_j). \tag{3.7}$$

From the definitions of n and g_i, we have the following.

$$\begin{aligned} n &= g_j(\mu_{2_{j_1}}, \mu_{2_{j_2}}, \ldots, \mu_{2_{j_k}}) \\ &\Rightarrow Pr(f_j(N(n, \sigma_j^2), s_2(j_1), s_2(j_2), \ldots, s_2(j_k)) \leq U_j) \geq Y_j \\ &\Rightarrow Pr(f_j(N(n, \sigma_j^2), s_1(j_1), s_1(j_2), \ldots, s_1(j_k)) \leq U_j) \geq Y_j \quad \text{from (3.7)} \end{aligned}$$

However, from the definitions of m and g_i, we obtain the following.

$$\begin{aligned} m &= g_j(\mu_{1_{j_1}}, \mu_{1_{j_2}}, \ldots, \mu_{1_{j_k}}) \\ &\Rightarrow m = min\ \{x : Pr(f_j(N(x, \sigma_j^2), s_1(j_1), s_1(j_2)\ldots) \leq U_j) \geq Y_j\} \\ &\Rightarrow (\ \forall k : Pr(f_j(N(k, \sigma_j^2), s_1(j_1), s_1(j_2)\ldots) \leq U_j) \geq Y_j : m \leq k\ ) \\ &\Rightarrow m \leq n, \quad \text{which contradicts (3.6) and proves our claim.} \end{aligned}$$

□

According to lattice theory [**37**], a partially ordered set forms a *complete lattice* if it has a least upper bound and a greatest lower bound on any subset of its elements. A lattice is constructed of all gate-size vectors that satisfy the size constraints. The bottom element of the lattice is a gate-size vector, comprising nominal gate-sizes of which are equal to their individual lower bounds given by physical and timing constraints. The top element of the lattice is a virtual gate-size vector, having at least one upper gate-size bound violation. It is a virtual vector as it is a mapping of all gate-size vectors in the partially ordered set with at least one upper size-bound violation. This makes the family of gate-size vectors with the $\leq$ relation a complete lattice. The existence of a fixpoint in the lattice is guaranteed by Knaster and Tarski's theorem [**37**].

Theorem 3.2.3. *If* $fix(G) = \{M_{f_1}, M_{f_2}, \ldots, M_{f_l}\}$ *denotes the set of fixpoints of* G*, then there exists a least fixpoint* $[M_{f_L} \stackrel{\Delta}{=} (i : 0 \leq i < n : \mu_{i_{f_L}})] \in fix(G)$*, where* $\mu_{i_{f_L}} = min(\mu_{i_{f_1}}, \mu_{i_{f_2}}, \ldots \mu_{i_{f_l}})$*, such that* $(\forall j : 0 \leq j \leq l : M_{f_L} \leq M_{f_j})$.

Proof. The proof is similar to the proof of Theorem 2.2.4. □

We use an iterative approach to compute the least fixpoint. The *scheme of chaotic iterations* [**38**] ensures that no matter what evaluation order is used, the generated sequence is monotonic in the same direction and it will not overshoot the fixpoint generated by G.

Corollary 3.2.3.1. *If the solution to the yield driven gate-size optimization exists, then the fixpoint reached starting from the bottom element of the lattice as our initial solution in the iterative approach is the least fixpoint and also the optimal solution.*

The least fixpoint yields the minimal size of gates which achieve the local yield constraints and thus is the solution to the yield driven gate-sizing problem.

3.3. Yield driven gate-size optimization algorithm

We present an iterative yield driven gate-size optimization algorithm in this section. Layout extraction is performed on a given circuit and is used to construct the corresponding coupling-graph. Timing budgeting is performed on the circuit to further constrain the gate-size bounds which are initially given only by physical constraints. This incorporates the timing constraints of the circuit into the optimization process. The nominal driving-gate-size of each net corresponding to a node in the coupling-graph is then initialized to its lower size-bound. Graph traversal is performed using a queue which is initially filled with nodes having $Pr(\eta(i) \leq U_i) < Y_i$. As a node i is popped from the queue, its driver is sized up so that the yield constraint is met. All nodes having an edge from node i are pushed in the

queue, if they violate their yield constraints and are not already in the queue. Iteration stops either when the queue is empty, or when any nominal driver size exceeds the given constraint. In the latter case, no solution is concluded. Alternately, traversal may continue to try fixing other violations. In this way, updates are performed iteratively until the probability of no noise violations on each node is at least the local desired yield, or it is found that gate-sizing cannot achieve the local yield constraints.

The probability of a net satisfying a given noise constraint is evaluated from the distribution of coupling-noise induced on it. Analytical estimation of this distribution is not trivial. Given that the noise distribution is known, it is also needed to determine minimal nominal driving gate-sizes (g_i) of nets for noise-optimization under yield and size constraints. Both the analysis and optimization steps are difficult to solve analytically. Monte Carlo simulations are thus performed to estimate and determine noise distributions and optimal gate-sizes, respectively. For the updates, the noise on a node i is calculated based on the noise-model used. Nodes that have a direct edge to node i induce coupling-noise on node i and are used to determine the value of $\eta(i)$. If the $Pr(\eta(i) \leq U_i) < Y_i$, the driving-gate of i is optimally sized to g_i such that $Pr(\eta(i) \leq U_i) \geq Y_i$. The value of g_i is evaluated using binary search and Monte Carlo simulations are used to determine if the current gate-size satisfies the noise constraints on all its driven nets. No update is performed if the calculated probability is already more than Y_i. If the sizing up violates size constraints, no solution to achieving the desired yield is declared. Optionally, iterations may be further done to achieve the desired yield on other nets.

The order of the iterations may only alter the rate of convergence, but the algorithm is guaranteed to reach the solution in any case, provided it exists. The pseudo-code of the proposed algorithm is shown in Figure 3.1.

- Algorithm: Yield driven gate-size optimization
- **Input:** Layout extraction results
- **Output:** Optimized gate-size vector, if solution exists
- begin
  (1) construct coupling-graph G based on layout extraction
  (2) perform timing budgeting on circuit to obtain size bounds
  (3) $\mu_i = l(i), \quad \forall i \in [0, n-1]$
  (4) do
  (5) for each net $(i \in [0, n-1] : Pr(\eta(i) \leq U_i) < Y_i)$
  (6) $\mu_i = g_i(\mu_{i_1}, \mu_{i_2}, \ldots, \mu_{i_k})$
  (7) while ($(\exists i \in [0, n-1] : Pr(\eta(i) \leq U_i) < Y_i) \ \wedge \ (\forall i \in [0, n-1] : \mu_i \leq u(i))$)
- end

Figure 3.1. Algorithm for yield driven gate-size optimization under noise and variation

3.4. Experimental results and conclusions

Experimental results of the proposed algorithm are presented for the ISCAS'85 benchmarks [**36**] and three larger circuits. Parameters for the circuits are randomly generated with realistic values in a $0.18\mu m$ technology. The 2π model [**39**] is used for noise modeling. Nominal gate-sizes are bounded in each direction by a factor of at most 2. Circuits are simulated for two variance values σ_1^2 and σ_2^2, with 3σ values set to 10% and 15% of the nominal, respectively. Monte Carlo simulations are performed with 1000 and 10000 samples. The error between the results for the two samples is found to be less than 2%. The maximum tolerable noise (U_i) for a net is set to $0.2V_{dd}$.

Obtained results are compared with those obtained by using the algorithm for gate-size optimization under noise constraints presented in Figure 2.2. Results are also presented for a modified algorithm for gate-size optimization under noise constraints where gates are sized more than their optimal values to account for variations. This method employs guard-bands (GB) to size up the driving-gate of a net i for noise bound lower than its actual noise

bound $U(i)$. Since the optimization is conservative, it is expected that the circuit will have a high probability of satisfying the noise constraints on its nets under variability. Results are presented for GB values of 2% and 3% safety. The algorithms which do not consider process variations directly are denoted as *WoPV* algorithms with a given GB value. The proposed yield driven gate-sizing algorithm is denoted as *WPV*.

Table 3.1 shows the the number of nets, the number of edges in the coupling-graph formed, the total number of initial noise violations, and run time for the *WoPV* and *WPV* algorithms (with 1000 Monte Carlo samples). The number of initial violations denote the number of noise violations in the given circuit before the gates are sized down. Run time for the three *WoPV* algorithms are almost identical. The *WoPV* algorithm without any guard band successfully removes noise violations for the circuits under no variation. Tables 3.2 and 3.3 present global yield values obtained by the algorithms for variances of σ_1^2 and σ_2^2, respectively. Global yield in circuits where optimal gate-sizing cannot remove all noise violations would be 0% since even a single violation fails the circuit. The algorithms are thus tested on circuits whose noise violations can be removed by sizing. All algorithms can however, optimize circuits partially when yield constraints cannot be globally met. It is observed that the gate-size vector given by the optimal gate-sizing algorithm is very sensitive to variations. The *WoPV* algorithms with GB perform better but produce lower yields when variations are increased. Larger GB values seem helpful, but do not necessarily produce a solution. This is observed in CKT_3 where the *WoPV* algorithm fails to produce a solution for a 3% value of GB. Higher values of GB also translate to larger gate-sizes which is attained to minimize. The results demonstrate that the *WPV* algorithm produces higher yields and area overheads are under 2%. The *WPV* algorithm is also more efficient in terms of area overhead as compared to *WoPV* with 3% GB. Since the optimization is performed only

Table 3.1. Benchmarks with initial noise violation figures and run time

Circuit	# of Nets	# of C Edges	# of Initial Violations	Run time (s)	
				WoPV	WPV
C432	336	566	3	0.01	2.55
C499	408	653	3	0.01	3.23
C880	729	1253	5	0.01	5.49
C1355	1064	1669	6	0.01	6.27
C1908	1498	2677	9	0.01	9.48
C2670	2076	3854	12	0.01	14.3
C3540	2939	5042	11	0.01	18.78
C5315	4386	7059	8	0.02	23.11
C6288	4800	7213	13	0.01	23.40
C7552	6144	10965	16	0.02	35.80
CKT_1	20K	39155	42	0.09	133.81
CKT_2	32K	62768	61	0.16	185.12
CKT_3	40K	63222	44	0.17	255.94

Table 3.2. Comparison of global yields obtained with variance σ_1^2

Circuit	% Global Yield			
	WoPV			WPV
	0% GB	2% GB	3% GB	$Y_i = 0.98$
C432	2	89	98	100
C499	2	84	93	99
C880	0	82	96	99
C1355	0	95	100	100
C1908	0	75	97	100
C2670	0	83	97	100
C3540	0	64	94	97
C5315	0	91	99	99
C6288	0	84	100	100
C7552	0	73	97	99
CKT_1	0	41	91	100
CKT_2	0	47	93	96
CKT_3	0	62	0	94

once at the post-route stage, the run time increase of the WPV algorithm over the others is acceptable. Results reported are for simulations run on a Dell Latitude laptop with an Intel P-IV 1.6 GHz processor, and 128Mb RAM.

Table 3.3. Comparison of global yields obtained with variance σ_2^2

Circuit	% Global Yield			
	WoPV			WPV
	0% GB	2% GB	3% GB	$Y_i = 0.98$
C432	2	67	95	95
C499	0	51	86	94
C880	0	52	85	88
C1355	0	75	97	94
C1908	0	46	75	92
C2670	0	40	81	82
C3540	0	25	70	84
C5315	0	55	93	92
C6288	0	62	92	92
C7552	0	43	78	86
CKT_1	0	2	59	82
CKT_2	0	2	65	83
CKT_3	0	9	0	78

CHAPTER 4

Statistical Timing Yield Optimization by Gate-sizing

Statistical static timing analysis (SSTA) has emerged as an approach to avoid multiple runs of deterministic static timing analysis and capture the statistical behavior of a circuit's delay (under variability) in a single pass. In statistical timing, parameters are considered as random variables. Circuit component delays are modeled as functions of these variables with a known distribution (often a Gaussian). This allows for an analytical evaluation of the circuit delay as a random variable with a known probability density function (PDF). The probability that the circuit delay exceeds a required value can be computed as the area under the PDF to the right of the required value. For simplicity, we assume that circuit failure is from late arriving signals only (setup condition). The extension to include failures due to early arriving signals (hold condition) is similar. Statistical optimization provides an additional degree of freedom in the solution space where the optimizer tunes the PDFs of circuit delays instead of worst case values. This makes statistical timing optimization an attractive design approach.

Researchers have proposed multiple approaches to statistical static timing analysis. Devgan *et al.* propose an efficient technique for block-based static timing analysis considering uncertainty [**42**]. Agarwal *et al.* present a statistical timing analysis approach which focuses on handling spatial correlations [**43**]. A technique for timing analysis under uncertainty is proposed by Bhardwaj *et al.* in [**44**], which gives a tight bound on circuit delay distributions.

Khandelwal *et al.* propose an efficient statistical timing analysis approach using error budgeting [**45**]. Many recent literature consider circuit component delays as Gaussian random variables since it facilitates fast analytical evaluation. Chang *et al.* propose a statistical timing analysis approach under this assumption which considers spatial correlations [**46**]. A timing analysis algorithm that accounts for correlations and accommodates dominant interconnect coupling is proposed by Le *et al.* in [**47**]. A first order incremental block based statistical timing analyzer is presented by Visweswariah *et al.* in [**48**]. Chao *et al.* propose a statistical timing analysis method for latch-based pipeline designs [**49**].

Multiple approaches to statistical timing analysis driven circuit optimization have emerged recently. Jacobs *et al.* [**50**] propose a gate-sizing approach to minimize a given percentile point in the arrival time distribution of a circuit. However, correlations between gate delay distributions are ignored. Hashimoto *et al.* [**51**] propose a gate-sizing approach to circuit optimization, which is driven by statistical timing analysis. Their approach to statistical timing analysis ignores correlations, and employs numerical methods to evaluate the mean and the variance of the maximum of multiple Gaussians. Raj *et al.* [**52**] propose a statistical gate-sizing methodology for timing yield improvement using a notion of path disutilities. A statistical optimization algorithm for gate-sizing under a linear delay model is proposed by Mani *et al.* [**53**]. Correlations between gate delays are ignored and the methodology for an optimal assignment of required timing yields at the output of each gate to satisfy the required timing yield of the circuit is not discussed. Choi *et al.* [**54**] propose a sizing algorithm for yield improvement using Lagrangian Relaxation. Computation of the mean and variance of the latest arrival time at the output of any multi-input gate is based on the temporal proximity of only the latest two arriving inputs. Under variations, it may not be always possible to select two inputs of a gate as its critical ones. Agarwal *et al.* [**55**] propose

a sensitivity based gate-sizing algorithm and faster approaches that perform sensitivity calculation based on slack computation [**56**], to minimize the 99-percentile point of a circuit's delay distribution. Intra-die variability is considered, and gate delay variations are assumed to be 10% of their nominals. A robust gate-sizing methodology based on geometric programming is proposed by Singh *et al.* [**57**]. They incorporate an uncertainty ellipsoid to model variations and attain to optimize circuit area under worst case timing constraints. Guthaus *et al.* [**58**] propose a gate-sizing algorithm to optimize circuit area while satisfying a given timing yield target. They employ a sensitivity metric based on slack distributions to select gates for resizing. Our experiments conclude that node and edge criticalities [**48**] evaluated in their approach can only be estimated in closed form to be within 20% of those obtained from Monte Carlo simulations. This is due to the assumption of independence between the criticalities of any two paths while evaluating a node or an edge criticality. As a result, they may be inadequate for guiding timing optimization.

In this chapter, we present an approach to area constrained statistical timing yield optimization that involves statistical modeling, statistical timing analysis and gate-sizing. We do not focus on improving a given percentile point of a circuit's delay distribution, but attain to maximize the probability that given timing constraints are met, under variations. Statistical gate delay modeling is performed for a commercial 0.13μ technology library from a foundry. We employ Visweswariah's approach [**48**] for statistical static timing analysis and present a formal proof that validates their variance matching methodology used in the computation of the maximum of two Gaussians. Gate-sizing is performed using a statistical global sizing algorithm. We prove that maximizing the timing yield of a circuit is equivalent to maximizing a simple expression involving the mean and the standard deviation of the

circuit's slack distribution. Experiments performed in an industrial framework show absolute timing yield gains of 30% on the average in comparison to a commercial synthesis tool for an area overhead of at most 10%. We observe that for iso-area solutions, our metric obtains larger timing yields than optimization for the worst case slack. Finally, we present insight into statistical properties of gate delays from a commercial technology library which intuitively provides one reason why statistical timing driven optimization does better than deterministic timing driven optimization.

The rest of this chapter is organized as follows. Sections 4.1 and 4.2 present our approaches to statistical modeling and statistical static timing analysis, respectively. In Section 4.3, we propose our statistical gate-sizing algorithm for timing yield optimization, and present experimental results in Section 4.4. We provide insight into statistical properties of gate delays in Section 4.5, and draw conclusions in Section 4.6.

4.1. Statistical modeling

Statistical delay modeling involves expressing circuit component delays as functions of the parameters of variation, which we model as Gaussian random variables. Based on the work in [**46**] and [**48**], we assume that gate delays are approximated by a linear function of the parameters. In addition, we assume that these parameters are independent, since a dependent set of Gaussian parameters can be transformed into an equivalent set of independent Gaussian parameters using principal component analysis [**46**]. Circuit component delays are therefore expressed as the following.

$$a_0 + \sum_{i=1}^{n} a_i \Delta X_i + a_{n+1} \Delta R_a \tag{4.1}$$

In the above expression, a_0 denotes the mean or nominal value of the delay, ΔX_is ($i = 1, 2, \ldots, n$) represent the variations of n global parameters X_is ($i = 1, 2, \ldots, n$) from their nominal values, and a_is ($i = 1, 2, \ldots, n$) denote the delay sensitivities to their corresponding sources of variation. ΔR_a represents the variation from the nominal of an independent random variable R_a that is associated with each component, and a_{n+1} denotes the delay sensitivity to R_a.

To compute the delay sensitivities for any gate in the circuit, we obtain pre-characterized gate delay values as functions of their loading capacitance and input slews (based on deterministic timing analysis at nominal corner) at multiple corners in the parameter space. The parameters are normalized by subtracting their nominal values followed by a division with their standard deviations. A least-squares fit is finally employed to obtain the desired delay sensitivities that express the gate delay as a linear function of normal random variables, as expressed in (4.1). This procedure is repeated for each gate in the circuit. Figure 4.1 shows pre-characterized delay values for some inverter in a circuit at multiple corners in a two-dimensional parameter space. A least square fit of the obtained points results in a plane, the slope of which in the two coordinate directions give the sensitivities of the inverter delay to the parameters, respectively. The inverter delay is thus obtained as a weighted linear sum of Gaussian random variables.

4.2. Statistical static timing analysis

Statistical static timing analysis requires propagation of delay distributions through the circuit. This involves *add* and *max* operations on the delay random variables. Since we express circuit component delays as a linear combination of Gaussian random variables, the

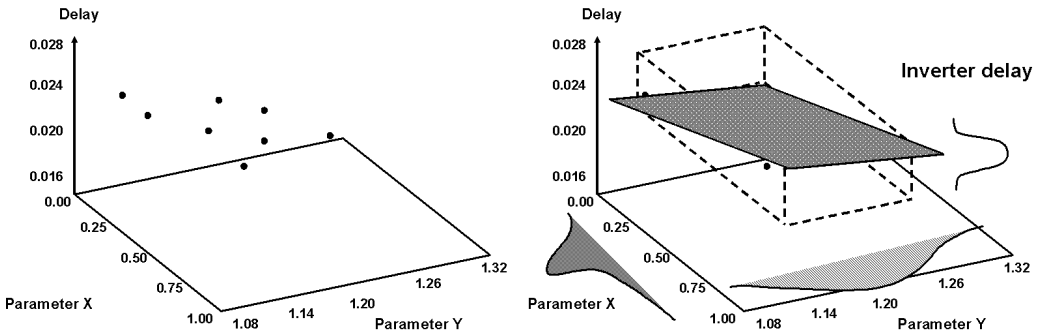

Figure 4.1. Statistical inverter delay modeling example

add operation is performed in a straight forward manner and yields another Gaussian. However, the *max* operation on Gaussians is an intricate operation, and results in a non-Gaussian distribution. This distribution is approximated by a Gaussian using Clark's approach [**59**] which matches the first and second moments of the two distributions.

In this work, we employ Visweswariah's approach [**48**] to computing the maximum of two Gaussian delay random variables A and B, which are expressed as a weighted linear sum of normal random variables as in (4.1). We denote the (mean, variance) of A and B as $(a_0, \sigma_A^2 = \sum_{i=1}^{n+1} a_i^2)$ and $(b_0, \sigma_B^2 = \sum_{i=1}^{n+1} b_i^2)$ respectively, where the a_is and b_is represent delay sensitivities as mentioned earlier. We use $\rho = \frac{\sum_{i=1}^{n} a_i b_i}{\sigma_A \sigma_B}$ to denote the correlation coefficient between A and B, and define the following.

$$\phi(x) \triangleq \frac{1}{\sqrt{2\pi}} exp(-\frac{x^2}{2}) \tag{4.2}$$

$$\Phi(y) \triangleq \int_{-\infty}^{y} \phi(x)dx \tag{4.3}$$

$$\theta \triangleq (\sigma_A^2 + \sigma_B^2 - 2\rho\sigma_A\sigma_B)^{1/2} \tag{4.4}$$

$$\alpha \triangleq \frac{a_0 - b_0}{\theta} \tag{4.5}$$

The mean μ_{max} and variance σ^2_{max} of $max(A, B)$ are computed as follows (Clark's approach [**59**]).

$$\mu_{max} = a_0\Phi(\alpha) + b_0\Phi(-\alpha) + \theta\phi(\alpha) \tag{4.6}$$

$$\sigma^2_{max} = (\sigma^2_A + a_0^2)\Phi(\alpha) + (\sigma^2_B + b_0^2)\Phi(-\alpha) + (a_0 + b_0)\theta\phi(\alpha) - \mu^2_{max} \tag{4.7}$$

Approximation of $max(A, B)$ with a Gaussian C having a canonical form $(c_0 + \sum_{i=1}^{n} c_i\Delta X_i + c_{n+1}\Delta R_c)$ is performed as follows (Visweswariah's approach [**48**]).

$$c_0 = \mu_{max} \tag{4.8}$$

$$c_i = a_i\Phi(\alpha) + b_i[1 - \Phi(\alpha)] \qquad \forall i \in 1, 2, \ldots, n \tag{4.9}$$

$$c_{n+1} = [\sigma^2_{max} - \sum_{i=1}^{n} c_i^2]^{1/2} \tag{4.10}$$

$\Phi(\alpha)$ in the above expression denotes the tightness probability of A over B, that is, the probability that A dominates B. Our first contribution to this approach is that we formally validate the variance matching approach in (4.10). We prove in Appendix A.1 that $(\sigma^2_{max} - \sum_{i=1}^{n} c_i^2)$ is always non-negative. This implies that the variance matching approach never involves the computation of the square root of a negative quantity.

Required time estimation in statistical timing analysis is performed by a backward propagation of delay distributions and involves the *subtract* and *min* operations. Given the canonical forms of the distributions, the *subtract* operation is performed easily. We next

present analytical expressions to compute the mean μ_{min} and variance σ^2_{min} of $min(A, B)$.

$$\mu_{min} = a_0\Phi(-\alpha) + b_0\Phi(\alpha) - \theta\phi(\alpha) \tag{4.11}$$

$$\sigma^2_{min} = (\sigma^2_A + a_0^2)\Phi(-\alpha) + (\sigma^2_B + b_0^2)\Phi(\alpha) - (a_0 + b_0)\theta\phi(\alpha) - \mu^2_{min} \tag{4.12}$$

Approximation of $min(A, B)$ with a Gaussian D having a canonical form $(d_0 + \sum_{i=1}^{n} d_i \Delta X_i + d_{n+1}\Delta R_d)$ is performed as follows.

$$d_0 = \mu_{min} \tag{4.13}$$

$$d_i = a_i\Phi(-\alpha) + b_i\Phi(\alpha) \qquad \forall i \in 1, 2, \ldots, n \tag{4.14}$$

$$d_{n+1} = [\sigma^2_{min} - \sum_{i=1}^{n} d_i^2]^{1/2} \tag{4.15}$$

When a gate has more than two fanins (fanouts), the *max* (*min*) operation for the arrival (required) time distribution calculation is done one pair a time, each step of which involves approximations. We observe that an arbitrary order of these pair-wise operations may accumulate errors and can significantly affect the accuracy of the final solution. We employ a greedy approach for smart pair-wise *max* (*min*) operations based on the approximation error computations [**19**]. The details of this approach is presented in Chapter 5.

Statistical timing analysis is performed in an incremental manner like a standard incremental deterministic static timer. We can incorporate separate PDFs for the rise and fall delays. Slack estimation during timing analysis involves *subtract* operations which can be performed on the canonical forms of the timing distributions. A *min* operation on the slack distributions at the primary outputs gives the circuit slack, the PDF of which is evaluated to obtain the probability that the circuit meets its given timing constraints.

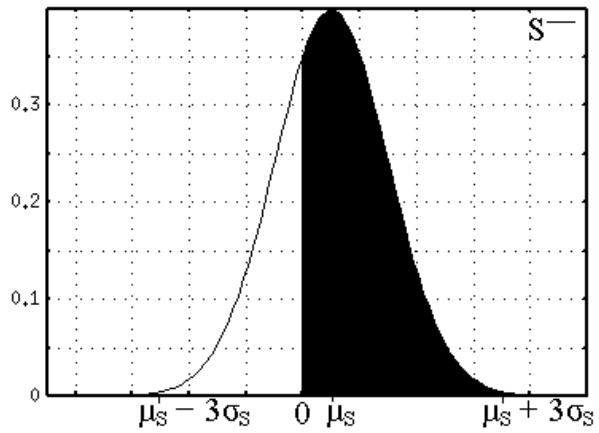

Figure 4.2. Shaded area of the slack PDF representing timing yield

4.3. Statistical gate-sizing

We formally define the *timing yield* of a circuit to be the probability that the slack at its primary output is non-negative. This probability can be computed by integrating the slack PDF from 0 to ∞, and is shown as the area of the shaded region in Figure 4.2. Given the circuit slack (after statistical timing analysis) as a Gaussian random variable S with mean μ_S and standard deviation σ_S, the timing yield P of the circuit is formally defined as the following.

$$P \triangleq \frac{1}{\sqrt{2\pi}\sigma_S}\int_0^{\infty} exp[-\frac{(x-\mu_S)^2}{2\sigma_S^2}]dx \tag{4.16}$$

In this work, we attain to maximize the timing yield of a circuit using gate-sizing, under given area constraints. We next prove that maximizing the timing yield is equivalent to maximizing the ratio of the mean to the standard deviation of the circuit slack distribution.

Theorem 4.3.1.

$$max\ \frac{1}{\sqrt{2\pi}\sigma_S}\int_0^{\infty} exp[-\frac{(x-\mu_S)^2}{2\sigma_S^2}]dx\ \equiv\ max\ \frac{\mu_S}{\sigma_S}$$

Proof. We define $y \triangleq \frac{\mu_S - x}{\sigma_S}$. Under variable transformation,

$$\begin{aligned} & \frac{1}{\sqrt{2\pi}\sigma_S}\int_0^{\infty} exp[-\frac{(x-\mu_S)^2}{2\sigma_S^2}]dx \\ = & \frac{1}{\sqrt{2\pi}} - \int_{\mu_S/\sigma_S}^{-\infty} exp[-\frac{y^2}{2}]dy = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\mu_S/\sigma_S} exp[-\frac{y^2}{2}]dy, \end{aligned}$$

which is strictly increasing with $\frac{\mu_S}{\sigma_S}$. This proves our claim. □

Our statistical gate-sizing approach thus attains to maximize the metric $\frac{\mu_S}{\sigma_S}$, under area constraints. For sake of comparison, we also consider maximizing the metric $\mu_S - 3\sigma_S$, under identical area constraints; such an objective function attains to maximize the worst case slack.

We design a statistical global gate-sizing (SGGS) algorithm for timing yield optimization as an extension to the global gate-sizing algorithm [**60**]. The global sizing algorithm is a mix of an approach that involves multi-dimensional descent based optimization, a perturbed propagation based heuristic that avoids gradient re-computations, and involves a global perturbation technique to get out of local minimums. It is a general purpose algorithm oriented to pure combinatorial optimization. The algorithm addresses non-linear, non-unimodal, constrained optimization which enables it to handle complex cost models in a realistic design environment. Our choice of the global sizing algorithm is motivated by results obtained by Coudert *et al.* [**61**], which show that this algorithm is superior to common greedy or genetic approaches to circuit optimization in terms of performance and power/delay curves.

The proposed algorithm considers the circuit as a network N of nodes with a global cost function *Cost* that is to be maximized under given area constraints $Area_{max}$. The global cost function used in our approach is the metric $\frac{\mu_S}{\sigma_S}$, where μ_S and σ_S denote the mean and the standard deviation, respectively, of the circuit slack distribution S. Each node in the network is implemented using some gate from the given technology library. Multiple gates, each belonging to the same gate class as the node, can be mapped to a given node. We refer to this process as resizing a node. The variation in the global cost due to resizing a node is denoted its gradient for a particular resize operation. We define the local cost of a node as the ratio of the mean to the standard deviation of the slack distribution at its output. Variations in the local cost of a node due to various resizing operations are termed as corresponding local gradients.

We describe the algorithmic flow next. A set *update* maintains a list of nodes whose gradients are to be computed. This set is initialized with all nodes in N. Another set *moves* maintains a list of nodes that can potentially be resized. This set is initially kept empty. For any node n, the gradient computation for each possible resize involves a run of statistical timing analysis on the entire circuit. This makes the gradient evaluation computationally very expensive. In practice, we observe that the impact of a node resize on the local gradients decrease quickly (approximately geometrically [**60**]) with increasing fanin and fanout level. We therefore extract a sub-network N_n for each node n in *update*, which is made out of one or two transitive levels of fanin and fanout around n. The inputs and outputs of the sub-network N_n are annotated with the corresponding arrival and required time distributions respectively from the original network N. Statistical timing analysis is now performed on N_n and the local gradient at the output of this sub-network is used as the metric for evaluation. Unless no possible resize operation on n improves this metric, the new gate involved in a

possible resize that maximizes this metric is termed as the *best-gate* for n. The node n and its *best-gate* are now added as a possible resize operation to the set *moves*. However, the resize is not actually performed at this stage.

Following the above procedure for each node in the set *update*, a *MultiMove* routine picks a sub-set of possible resize operations from the set *moves* that provide maximum cumulative gain in the global cost. These resize operations are then performed and the resized nodes are returned in a new set *moved*. The *MultiMove* routine determines the sub-set for the move based on the descent direction or by a conjugation of directions of the cost gradients [**60**]. In our experiments, we employ a greedy heuristic that chooses the best two nodes for resize in terms of yield improvement in each *MultiMove* operation. A new set of nodes whose gradients need to be recomputed are now derived from *moved* in the function *PerturbedNodes*. In our approach, we choose a node for gradient re-computation only if it is sufficiently perturbed, that is, if one of its close neighbors (within one or two transitive fanin or fanout levels) has been resized. The entire process is repeated till convergence, wherein future iterations do not improve the global cost (timing yield of the circuit) further or till the run-time/area constraints of the design are violated. For comparison, this procedure is repeated starting with the original design, using $\mu_S - 3\sigma_S$ as both the global cost function and the local cost function. The complexity of this algorithm using best-fit polynomial is shown to be $k^{1.2}$, where k denotes the number of internal nodes [**61**]. The pseudo-code of the SGGS algorithm is presented in Figure 4.3.

- Algorithm: **SGGS**(N, $Cost$, $Area_{max}$)
- **Output:** Optimized circuit with improved timing yield
- begin

```
(1)  update = N;
(2)  moves = ∅;
(3)  do {
(4)    old_cost = Cost(N);
(5)    foreach n ∈ update {
(6)      extract sub-network N_n around n;
(7)      find best-gate g for n wrt Cost(N_n);
(8)      if g ≠ gate(n) {
(9)        n.move = g;
(10)       moves = moves ∪ {n};
(11)     }
(12)   }
(13)   moved = MultiMove(N, Cost, moves);
(14)   update = PerturbedNodes(moved);
(15) } until (Converge(old_cost, Cost(N), moved) ∨ (Area ≥ Area_max ))
```

- end

Figure 4.3. Statistical global gate-sizing algorithm

4.4. Implementation and experimental results

The proposed statistical modeling, statistical timing analysis and gate-sizing routines are implemented in an industrial framework, as an addition to a commercial synthesis and optimization tool. Experiments are performed on combinational ISCAS'85 and MCNC benchmarks mapped to a 0.13μ commercial technology library from a foundry.

For our experiments, we choose V_{dd} and temperature as the parameters of variation. We acknowledge that these parameters may have a non-linear impact on delays. However, pre-characterized gate delay values were available for a commercial 0.13μ library that we intended to use in our experiments. It was not immediately possible to re-characterize these gates for other parametric variations, and we did not use artificial values for the same as

done in a majority of other mentioned approaches to statistical optimization. In any case, our approach is not limited to the use of any particular parameters of variation.

We consider V_{dd} variations in the range of $1.08V$ to $1.32V$. The nominal value V_0 is set to $1.2V$ and the standard deviation is set as the following.

$$3\sigma_V = 1.32V - 1.20V$$

Similarly, we consider temperature variations from 0^oC to 125^oC, with nominal temperature T_0 as 25^oC and standard deviation σ_T set to 8.33^oC. For any characterization point X, the delay equation is set up as the following.

$$D_X = D_0 + D_1\frac{T_X - T_0}{\sigma_T} + D_2\frac{V_X - V_0}{\sigma_V}$$

D_0 represents the typical delay obtained from gate characterization at T_0 and V_0. This formulation is scalable to any number of parameters. A least squares fit procedure is employed to obtain the coefficients D_is. The accuracy of this approach is dependent on the number of characterization points that are available in the library.

Following statistical modeling, arrival times and required times are set at the primary inputs and primary outputs of the circuit, respectively. Statistical timing analysis is now performed to obtain the arrival and required time distributions on all nodes of the circuit. A *min* operation is performed on the obtained slack PDFs at the primary outputs to determine the global circuit slack PDF S, with mean μ_S and variance σ_S^2. Timing yield of the circuit is obtained from (4.16). Table 4.1 shows the obtained statistical timing analysis results. For each benchmark, we present the evaluated circuit arrival time (AT) mean (μ), and the circuit arrival time standard deviation (σ). For comparison, we also report corresponding

Table 4.1. Statistical timing analysis results

Circuit	AT μ (ns)		AT σ (ns)		Run time (secs)	
	SSTA	MC	SSTA	MC	SSTA	MC
C432	2.194	2.197	0.165	0.172	0.1	6.5
C499	1.316	1.311	0.095	0.095	1.2	14.6
C880	1.973	1.968	0.143	0.143	0.4	14.0
C1355	1.829	1.831	0.141	0.139	0.8	20.4
C1908	2.208	2.214	0.161	0.160	0.7	14.3
C2670	1.950	1.957	0.177	0.174	1.5	24.7
C3540	3.242	3.234	0.261	0.260	0.8	37.7
C5315	3.029	3.024	0.246	0.242	7.3	63.0
C6288	9.996	9.968	0.779	0.789	0.7	85.1
C7552	3.313	3.305	0.261	0.254	5.1	71.9
cm85a	0.425	0.423	0.029	0.029	0.0	1.6
sct	0.485	0.484	0.030	0.030	0.1	2.8
alu2	2.584	2.590	0.211	0.213	0.2	12.9
too_large	1.048	1.047	0.071	0.073	0.1	13.6
frg2	1.486	1.490	0.097	0.098	1.3	29.3

values obtained from 10000 random samples of Monte Carlo simulations. Benchmark sizes ranged from about 100 to 2000 gates. From Table 4.1, we observe that the average and maximum error in the estimation of the mean and standard deviation of the circuit delay distribution is under 1% and 4.1%, respectively. Our ordering approach for pair-wise *max* and *min* operations reduces the maximal error by up to 5% and is possibly one reason why our error values are lower in comparison to those mentioned in [**46, 62**]. SSTA is found to be faster than Monte Carlo simulations by 42.2X on the average.

For timing yield improvement estimation, we perform deterministic timing optimization on a given circuit using a commercial synthesis tool, which attains to improve the circuit slack under area constraints. Statistical timing analysis is then performed to obtain the slack distribution at the primary output of the circuit, the mean of which we denote as μ_{Static}. We next perform statistical timing optimization using our proposed gate-sizing approach to obtain a new circuit slack distribution. To estimate the relative gain in timing yield, we compute the relative timing yield of the circuit after the deterministic and statistical

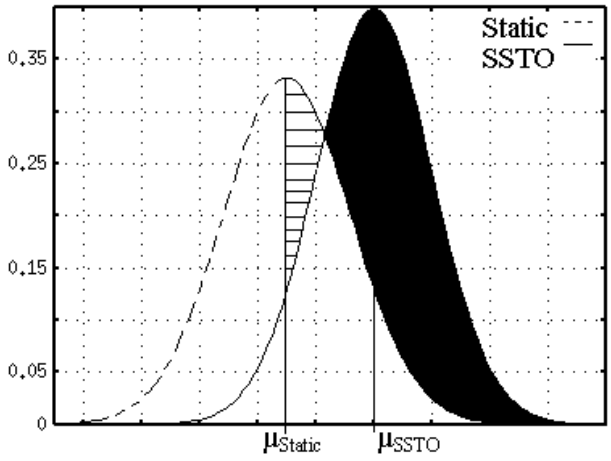

Figure 4.4. Timing yield improvement denoted by area in black − area in stripes

optimization passes as the area under their respective circuit slack PDFs from μ_{Static} to ∞. Figure 4.4 shows this relative timing yield improvement graphically as the area of the black region minus the area of the striped region. We next repeat this procedure using the alternate metric $\mu_S - 3\sigma_S$ as the cost function during statistical optimization instead of our original metric μ_S/σ_S.

Table 4.2 presents obtained relative timing yield improvements for both the optimization objective functions. We observe that statistical timing optimization achieves timing yield improvements of 0.3 on the average, and up to 0.5 with an area overhead of at most 10%. Corresponding average and maximum timing yield improvements using the metric for worst case slack are found to be 0.27 and 0.49, respectively. It is seen that the proposed approach of optimizing the timing yield guides better optimization than that for maximizing the worst case slack for iso-area solutions. For the design *alu2*, the alternate metric worsens the yield.

We next present a special case of timing yield improvement observed for the MCNC benchmark *APEX6*. The three PDFs in Figure 4.5 denote the slack distributions for the

Table 4.2. Relative timing yield improvement results

Circuit	Statistical optimization metric	
	Maximize $\mu_S - 3\sigma_S$	Maximize μ_S/σ_S
C432	0.0105	0.0122
C499	0.3032	0.3158
C880	0.0037	0.0037
C1355	0.4939	0.4963
C1908	0.1319	0.1584
C2670	0.1730	0.2153
C3540	0.4949	0.4977
C5315	0.4923	0.4925
C7552	0.4998	0.4995
cm85a	0.0569	0.2037
sct	0.1821	0.4342
alu2	-0.0570	0.1240
too_large	0.4269	0.4580
frg2	0.4516	0.3774

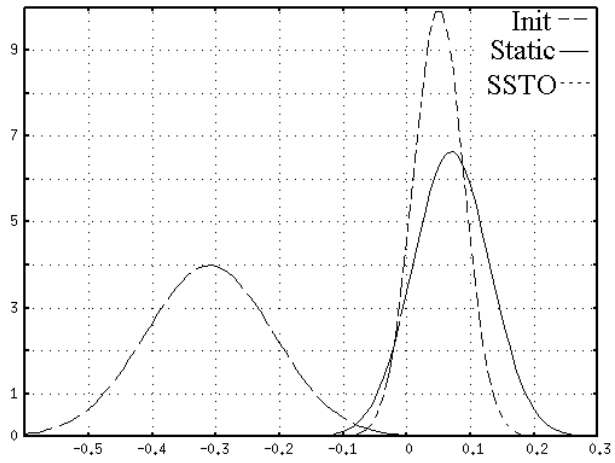

Figure 4.5. Pre and post optimization slack PDFs for the MCNC benchmark APEX6

unoptimized circuit *(Init)*, circuit following deterministic static timing optimization *(Static)* and circuit following statistical timing optimization *(SSTO)*. The reduced variance of the *SSTO* slack PDF improves the timing yield (from 0.87 to 0.89) even though it has a smaller mean as compared to *Static* slack PDF. This example illustrates how statistical optimization uses the additional information on variation to achieve larger timing yields, even for iso-area solutions. The proposed algorithm takes less than 480 mins for the largest benchmarks

on a 400 Mhz Sun Ultra 4 machine with 4 GB RAM. The primary reasons for large run time include multiple calls to statistical timing analysis that performs smart pair-wise *max* operations [**19**]; and an exhaustive search of the best-gate for any node in the inner loop of the algorithm.

4.5. Analysis of statistical properties of gate delays

We perform an analysis of statistical properties of gate delays on different gate classes from our 0.13μ commercial technology library. We select some nodes arbitrarily from a test circuit; and observe the mean and the standard deviation of the arrival time distribution at each of their outputs, while mapping different gates on them (the different gates belong to the gate-class of the node, for example, NAND or NOR). Figure 4.6 presents a plot of the arrival time standard deviation (Sigma) against the arrival time mean for a class of inverters. Dots on the plot represent gates which are sorted on the mean of their output arrival times when mapped to the given node and not in any order of their sizes. Figure 4.7 presents similar graphs for two classes of AND gates.

We observe that though most gates of a class make the plots monotonic, there exist exceptions. In some cases during our experiments, we observe that while the deterministic timing driven optimizer resizes a node to a gate with a smaller mean arrival time ignoring the fact that it may have larger variability, the statistical timing driven optimizer selects a gate with a larger mean arrival time, but a significantly lesser variance. Such a choice is found to increase the overall timing yield of the circuit. This behavior provides one reason why statistical timing driven optimization gains an edge over deterministic timing driven optimization.

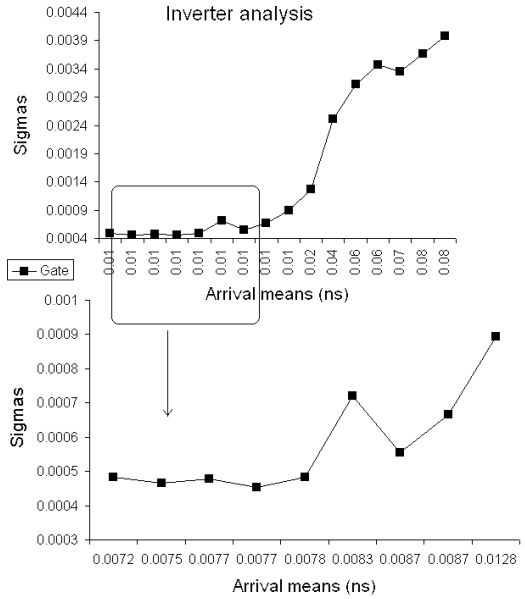

Figure 4.6. Arrival means and standard deviations for a class of inverters

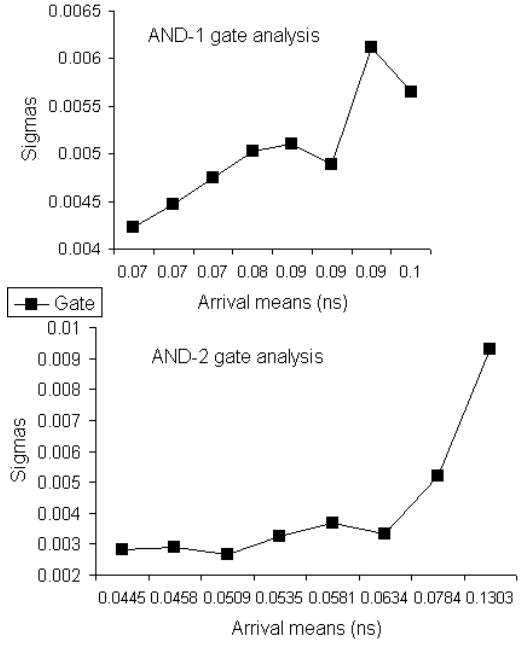

Figure 4.7. Arrival means and standard deviations for two classes of AND gates

4.6. Conclusions

In this chapter, we propose a statistical gate-sizing approach to maximize the timing yield of a given circuit under area constraints. Our approach involves statistical gate delay modeling, statistical static timing analysis and statistical gate-sizing. Experiments performed in an industrial framework on combinational ISCAS'85 and MCNC benchmarks show timing yield gains of 0.3 on the average, over deterministic timing optimization for at most 10% area penalty. It is further shown that circuits optimized using our metric have larger timing yields than the same optimized using a worst case metric, for iso-area solutions. Finally, we present an insight into statistical properties of gate delays for a commercial 0.13μ technology library which intuitively provides one reason why statistical timing driven optimization does better than deterministic timing driven optimization.

Though this work considers delays as a weighted linear sum of Gaussian random variables, the statistical timing yield improvement approach can be extended to handle non-Gaussian parameters and non-linear delay functions as proposed in [**63**]. However, obtaining a simple metric for timing yield optimization would be a challenging problem.

CHAPTER 5

Advances in Computation of the Maximum of a Set of Random Variables

A majority of statistical static timing analysis approaches consider the parameters of variation to be Gaussian random variables. Each circuit component delay is expressed as a weighted linear sum of these parameters, and therefore, has a Gaussian distribution. Propagation of these Gaussian distributions in block-based statistical timing analysis involves operations like *add* and *max*. It is required that the output of these operations be a Gaussian for further propagation. An *add* operation on Gaussians yields another Gaussian and is accurate. However, the *max* of multiple Gaussians is not a Gaussian, and approximating its distribution with a Gaussian introduces inaccuracies. Clark's approach [**59**] is used to approximate the *max* of two Gaussians with another Gaussian by matching the first two moments of their distributions. For multiple Gaussians, the *max* operation is performed a pair at a time. Each of these pair-wise operations introduce errors by approximating the resulting distribution with a Gaussian. These approximation errors can propagate and affect accuracy. We observe that the final loss in accuracy in the *max* of multiple Gaussians is dependent on the order in which pair-wise *max* operations are performed. Prior work does not describe the impact of ordering on the inaccuracy of the process. Our primary contributions in this chapter are summarized as follows.

- We quantify the error in the approximation of the *max* of two arbitrary Gaussians with a Gaussian. The closed form expression for the PDF of the true *max* is derived,

and used to develop an analytical expression which quantifies the approximation error.

- We present a transformation to obtain the *max* of two random variables from the *max* of a new pair of derived random variables, parameters of which can be bounded. In addition, we show that approximation error of the *max* operation is an invariant of our transformation.
- We introduce the idea of using a finite look-up table (LUT) to store quantified approximation errors in the *max* operation on any Gaussian pair.
- We study the approximation errors as functions of the given Gaussians and propose good orderings for pair-wise *max* operations on a given set of Gaussians. The orderings attain to reduce the loss in accuracy, without significant increase in run time.

Due to the symmetry of *max* and *min* operations, the *max* operation is considered in this chapter. The approaches presented are extensible for the case of *min* operations. Experiments results show significant timing estimation accuracy improvements in comparison to the traditional approach. In addition, we present an approach to computing the tightness probability [**48**] of Gaussian pairs with dynamic runtime-accuracy trade-off options. We replace traditionally required numerical computations for these estimations by closed form expressions based on Taylor series expansion that involve table lookup and a few fundamental arithmetic operations. Experiments demonstrate almost an order of speedup when using our approach for computing the *max* of two Gaussians in comparison to the traditional approach, without any accuracy penalty.

The rest of this chapter is organized as follows. Section 5.1 provides some preliminaries to our problem. The error in the approximation of the *max* of two Gaussians is quantified

in Section 5.2. We present our error minimization problem in Section 5.3. Smart orderings for pair-wise *max* operations that aim to reduce approximation errors while computing the *max* of multiple Gaussians are presented in Section 5.4. Experimental results are reported in Section 5.5, and we draw conclusions in Section 5.6.

5.1. Preliminaries

Based on the work in [**46, 47, 48**], we consider circuit delays as Gaussian random variables. A Gaussian random variable X is formally expressed as $N(\mu_X, \sigma_X^2)$, with mean μ_X and variance σ_X^2.

The *add* operation on Gaussian variables is performed easily and yields another Gaussian. The *max* operation, on the other hand, is an intricate operation, and for a given set of Gaussians, is performed a pair at a time. We next show Clark's moment matching approach [**59**] to computing the *max* of two Gaussians X and Y. ρ is used to represent the correlation coefficient between X and Y. We define the following.

$$\phi(x) \triangleq \frac{1}{\sqrt{2\pi}} exp(-\frac{x^2}{2}) \tag{5.1}$$

$$\Phi(y) \triangleq \int_{-\infty}^{y} \phi(x)dx \tag{5.2}$$

$$a \triangleq (\sigma_X^2 + \sigma_Y^2 - 2\rho\sigma_X\sigma_Y)^{1/2} \tag{5.3}$$

$$\alpha \triangleq \frac{\mu_X - \mu_Y}{a} \tag{5.4}$$

The mean μ_Z and variance σ_Z^2 of $Z \triangleq max(X, Y)$ is expressed analytically as follows.

$$\mu_Z = \mu_X\Phi(\alpha) + \mu_Y\Phi(-\alpha) + a\phi(\alpha) \tag{5.5}$$

$$\sigma_Z^2 = (\sigma_X^2 + \mu_X^2)\Phi(\alpha) + (\sigma_Y^2 + \mu_Y^2)\Phi(-\alpha) + (\mu_X + \mu_Y)a\phi(\alpha) - \mu_Z^2 \tag{5.6}$$

The tightness probability of X over Y is denoted by $\Phi(\alpha)$ [**48**]. Analytical evaluations of μ_Z and σ_Z involve computing the exponential $\phi(\alpha)$ and the definite integral $\Phi(\alpha)$, in addition to some fundamental arithmetic operations. Further, if we assume that X and Y are timing random variables at the fanin edges of a node, the corresponding node and edge criticality estimations [**48**] require the value of $\Phi(\alpha)$. It is therefore evident that efficient approaches to computing the exponential $\phi(\alpha)$ and definite integral $\Phi(\alpha)$ can speedup the *max* computation and thereby improve performance of a statistical timer. Faster statistical timing analysis helps improving the run time for traditionally slow statistical timing optimization, even when the criticality concepts are not employed.

Z is traditionally approximated to a Gaussian variable $Z_G \sim N(\mu_Z, \sigma_Z^2)$ for delay propagation. The first and second order moments of Z are matched to obtain Z_G, while the higher order moments of Z are ignored. This is the first and foremost source of inaccuracy in the approach. The non-linearity of the *max* operation causes Z to have an asymmetric density function. However Z_G has a symmetric density function. We quantify the error introduced during the above approximation in the following section.

5.2. Approximation errors in the *max* operation

5.2.1. Error definition

A formal comparison of two given distributions requires a metric that quantifies the dis-similarity (or similarity) between them. Given two random variables X, Y, and their probability density functions (PDFs) φ_X, φ_Y respectively, we quantify the dis-similarity or the error Ξ_{XY} between the variables as the total area under the non-overlapped region of their

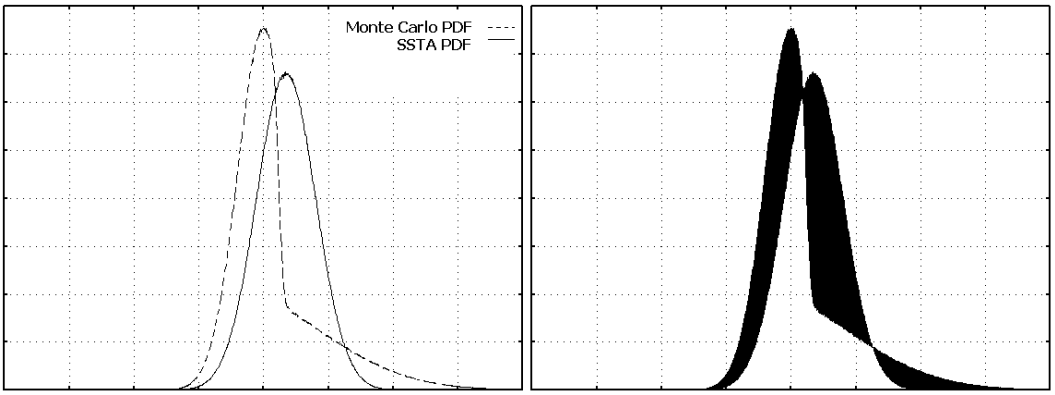

Figure 5.1. Error Ξ between two random variables with given PDFs represented by the area of the shaded region

PDFs. This is formally expressed as follows.

$$\Xi_{XY} \triangleq \int_{-\infty}^{\infty} |\varphi_X(t) - \varphi_Y(t)| dt \tag{5.7}$$

Figure 5.1 shows PDFs of two random variables, the dis-similarity between which we attain to quantify. The area of the shaded region represents the error between them. Since the area under each PDF is 1, the following is immediate.

$$0 \leq \Xi_{XY} \leq 2$$

5.2.2. Error in approximating the *max* of two Gaussians

We consider the *max* of two Gaussians $Z \triangleq max(X, Y)$, where $X \sim N(\mu_X, \sigma_X^2)$, $Y \sim N(\mu_Y, \sigma_Y^2)$, and ρ denotes their correlation coefficient. Z is approximated to a Gaussian $Z_G \sim N(\mu_Z, \sigma_Z^2)$, after computing μ_Z and σ_Z from (5.5)−(5.6). Based on (5.7), we formally

quantify the error introduced in this approximation as:

$$\Xi_{(Z)(Z_G)} \triangleq \int_{-\infty}^{\infty} |\varphi_Z(t) - \varphi_{Z_G}(t)| dt. \tag{5.8}$$

φ_{Z_G} denotes the PDF of the Gaussian Z_G. Mathematically,

$$\varphi_{Z_G}(t) \triangleq \frac{1}{\sqrt{2\pi}\sigma_Z} e^{-\frac{(t-\mu_Z)^2}{2{\sigma_Z}^2}} = \frac{1}{\sigma_Z}\phi(\frac{t-\mu_Z}{\sigma_Z}). \tag{5.9}$$

The joint probability density function (JPDF) $\varphi(x, y)$ for the Gaussians X and Y is defined as:

$$\varphi(x,y) \triangleq \frac{e^{\left\{\frac{-1}{2(1-\rho^2}\left[(\frac{x-\mu_X}{\sigma_X})^2 - 2\rho(\frac{x-\mu_X}{\sigma_X})(\frac{y-\mu_Y}{\sigma_Y}) + (\frac{y-\mu_Y}{\sigma_Y})^2\right]\right\}}}{2\pi\sigma_X\sigma_Y(1-\rho^2)^{1/2}}. \tag{5.10}$$

The CDF $\Psi(t)$ of Z is defined as the probability that Z is at most t. Since $Z = max(X, Y)$, $\Psi(t)$ is given by the probability that both X and Y are not more than t. This is evaluated by integrating the JPDF in the region where both $X \leq t$ and $Y \leq t$. Thus,

$$\begin{aligned}
\Psi_Z(t) &\triangleq Pr(Z \leq t) = \int_{-\infty}^{t}\int_{-\infty}^{t} \varphi(x,y)\, dx\, dy \\
&= \int_{-\infty}^{t}\int_{-\infty}^{t} \frac{e^{\frac{-1}{2(1-\rho^2)}\left[(\frac{x-\mu_X}{\sigma_X})^2 - 2\rho(\frac{x-\mu_X}{\sigma_X})(\frac{y-\mu_Y}{\sigma_Y}) + (\frac{y-\mu_Y}{\sigma_Y})^2\right]}}{2\pi\sigma_X\sigma_Y(1-\rho^2)^{1/2}} dx\, dy \\
&= \int_{-\infty}^{t}\int_{-\infty}^{t} \frac{e^{\frac{-1}{2(1-\rho^2}\left[(\frac{x-\mu_X}{\sigma_X}) - \rho(\frac{y-\mu_Y}{\sigma_Y})\right]^2} e^{\frac{-1}{2}\left[(\frac{y-\mu_Y}{\sigma_Y})^2\right]}}{2\pi\sigma_X\sigma_Y(1-\rho^2)^{1/2}}\, dx\, dy \\
&= \frac{\int_{-\infty}^{t}\phi(\frac{y-\mu_Y}{\sigma_Y})\int_{-\infty}^{t}\phi(\frac{[\frac{x-\mu_X}{\sigma_X}] - \rho[\frac{y-\mu_Y}{\sigma_Y}]}{[1-\rho^2]^{1/2}})\, dx\, dy}{\sigma_X\sigma_Y(1-\rho^2)^{1/2}} \\
&= \frac{1}{\sigma_Y}\int_{-\infty}^{t}\phi(\frac{y-\mu_Y}{\sigma_Y})\Phi\big[\frac{(\frac{t-\mu_X}{\sigma_X}) - \rho(\frac{y-\mu_Y}{\sigma_Y})}{(1-\rho^2)^{1/2}}\big]\, dy.
\end{aligned} \tag{5.11}$$

We next derive a closed form expression for the PDF $\varphi_Z(t)$, which is defined as the derivative of $\Psi(t)$ with respect to t.

$$\begin{aligned}\varphi_Z(t) &\triangleq \frac{d}{dt}[\Psi_Z(t)] = \frac{d}{dt}\big[\int_{-\infty}^{t}\int_{-\infty}^{t}\varphi(x,y)\,dx\,dy\big] \\ &= \int_{-\infty}^{t}\varphi(x,t)\,dx + \int_{-\infty}^{t}\varphi(t,y)\,dy \\ &= \frac{1}{\sigma_Y}\phi(\frac{t-\mu_Y}{\sigma_Y})\Phi\big[\frac{(\frac{t-\mu_X}{\sigma_X})-\rho(\frac{t-\mu_Y}{\sigma_Y})}{(1-\rho^2)^{1/2}}\big] + \frac{1}{\sigma_X}\phi(\frac{t-\mu_X}{\sigma_X})\Phi\big[\frac{(\frac{t-\mu_Y}{\sigma_Y})-\rho(\frac{t-\mu_X}{\sigma_X})}{(1-\rho^2)^{1/2}}\big]\end{aligned} \tag{5.12}$$

$\Xi_{(Z)(Z_G)}$ can now be evaluated from (5.9) and (5.12) using numerical integration.

5.2.3. Errors in canonical form

We consider two properties of a generic *max* operation.

(1) Scaling property

$$max(\lambda X, \lambda Y) = \lambda \cdot max(X,Y) \quad \forall \lambda \geq 0$$

(2) Shift-invariance property

$$max(X+\theta, Y+\theta) = max(X,Y) + \theta$$

We consider Gaussians $X \sim N(\mu_X, \sigma_X^2)$ and $Y \sim N(\mu_Y, \sigma_Y^2)$, with correlation coefficient ρ. Without any loss of generality, we assume $\sigma_X \geq \sigma_Y$. Application of the above properties on $max(X,Y)$ (shifting by μ_X and subsequent scaling by σ_X) results in the following.

$$max(X,Y) = \mu_X + \sigma_X \cdot max(X', Y') \tag{5.13}$$

where,

$$X' \sim N(\mu_{X'}, \sigma^2_{X'}) = \frac{X - \mu_X}{\sigma_X} \sim N(0,1)$$
$$Y' \sim N(\mu_{Y'}, \sigma^2_{Y'}) = \frac{Y - \mu_X}{\sigma_X} \sim N(\frac{\mu_Y - \mu_X}{\sigma_X}, \frac{\sigma^2_Y}{\sigma^2_X})$$

ρ' denotes the correlation coefficient between X' and Y'. Since $\sigma_{Y'} = \sigma_Y/\sigma_X$ and $\sigma_X \geq \sigma_Y$, we have the following.

$$0 \leq \sigma_{Y'} \leq 1 \tag{5.14}$$

Lemma 5.2.1. $\rho' = \rho$

Proof. The covariance (cov) of two Gaussians is independent of their means μ and is directly proportional to their standard deviations σ. Since $\sigma_{X'} = \frac{\sigma_X}{\sigma_X}$ and $\sigma_{Y'} = \frac{\sigma_Y}{\sigma_X}$, we have $cov(X', Y') = \frac{cov(X,Y)}{{\sigma_X}^2}$. From definition, $\rho = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$.

$$\rho' = \frac{cov(X', Y')}{\sigma_{X'}\sigma_{Y'}} = \frac{\frac{cov(X,Y)}{{\sigma_X}^2}}{1 \cdot \frac{\sigma_Y}{\sigma_X}} = \rho$$

□

Based on the definitions in (5.1)-(5.4), we define the following.

$$a' \triangleq (\sigma^2_{X'} + \sigma^2_{Y'} - 2\rho'\sigma_{X'}\sigma_{Y'})^{1/2} \tag{5.15}$$
$$\alpha' \triangleq \frac{\mu_{X'} - \mu_{Y'}}{a'} \tag{5.16}$$

Lemma 5.2.2. $\alpha' = \alpha$

Proof. From (5.3) and (5.15), we have $a' = \frac{a}{\sigma_X}$. Thus,
$\alpha' = \frac{\mu_{X'} - \mu_{Y'}}{a'} = \frac{-\mu_{Y'}}{a'} = \frac{-\frac{\mu_Y - \mu_X}{\sigma_X}}{\frac{a}{\sigma_X}} = \alpha.$ □

We denote the error in approximating $Z' \triangleq max(X', Y')$ with a Gaussian $Z'_G \sim N(\mu_{Z'}, \sigma^2_{Z'})$ to be $\Xi_{(Z')(Z'_G)}$, and prove that it is exactly equal to the error $\Xi_{(Z)(Z_G)}$ in approximating $Z = max(X, Y)$ with Gaussian $Z_G \sim N(\mu_Z, \sigma^2_Z)$. The proof is presented in Appendix A.2. Mathematically, we prove the following.

Theorem 5.2.1.

$$\Xi_{(Z')(Z'_G)} \quad = \quad \Xi_{(Z)(Z_G)} \tag{5.17}$$

The approximation error in the *max* of any two Gaussians can thus be estimated from the approximation error in the *max* of the derived Gaussians, one of which is the unit normal Gaussian. The error is therefore a function of $\mu_{Y'}$, $\sigma_{Y'}$ and $\rho'(= \rho)$. Since $\alpha'(= \alpha)$ is a function of $\mu_{Y'}$, the error can be expressed as a function of α, σ'_Y and ρ. It is known that $\phi(\alpha) \approx 0$ for $|\alpha| \geq 4$, $\Phi(\alpha) \approx 0$ for $\alpha \leq 4$, and $\Phi(\alpha) \approx 1$ for $\alpha \geq 4$. Consequently, for $|\alpha| \geq 4$, $max(X, Y)$ almost identically resembles the dominating Gaussian [**48**]. The approximation error in this case is negligible. Thus, the region of interest for the parameters that affect the approximation error is bounded by the following.

$$\begin{aligned} -4 \leq \quad & \alpha \quad \leq 4 \\ 0 \leq \quad & \sigma_{Y'} \quad \leq 1 \quad \text{from (5.14)} \\ -1 \leq \quad & \rho \quad \leq 1 \quad \text{from definition} \end{aligned}$$

Experiments are performed to study the dependence of the approximation error $\Xi_{(Z')(Z'_G)}$ on the above parameters. It is observed that $\Xi_{(Z')(Z'_G)}$ decreases when either of the Gaussians

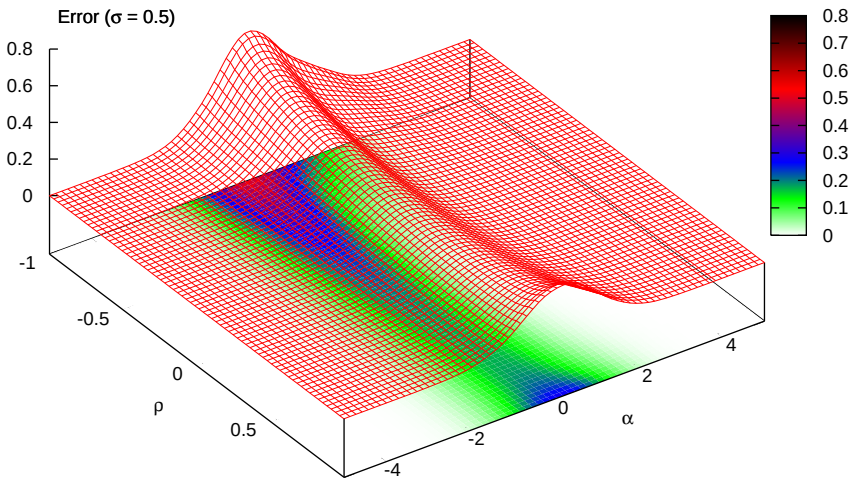

Figure 5.2. $\Xi_{(Z')(Z'_G)}$ as a function of ρ and α ($\sigma_{Y'} = 0.5$)

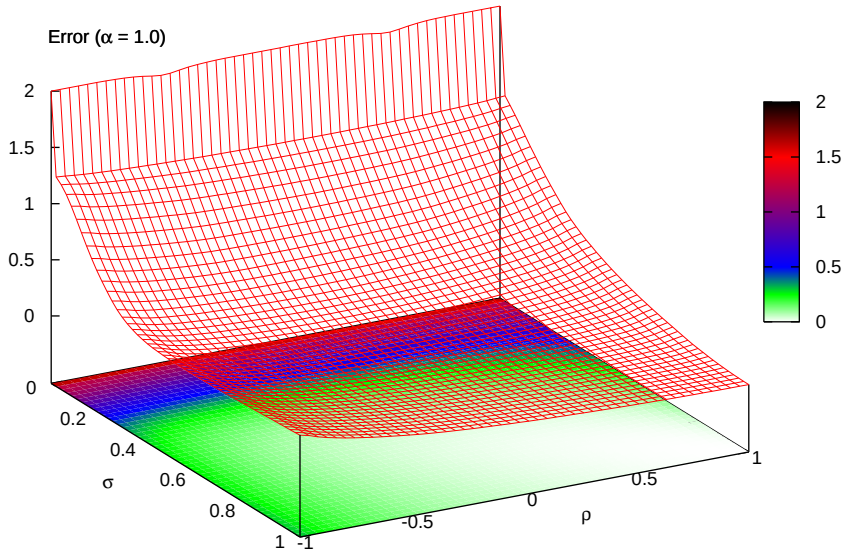

Figure 5.3. $\Xi_{(Z')(Z'_G)}$ as a function of $\sigma_{Y'}$ and ρ ($\alpha = 1.0$)

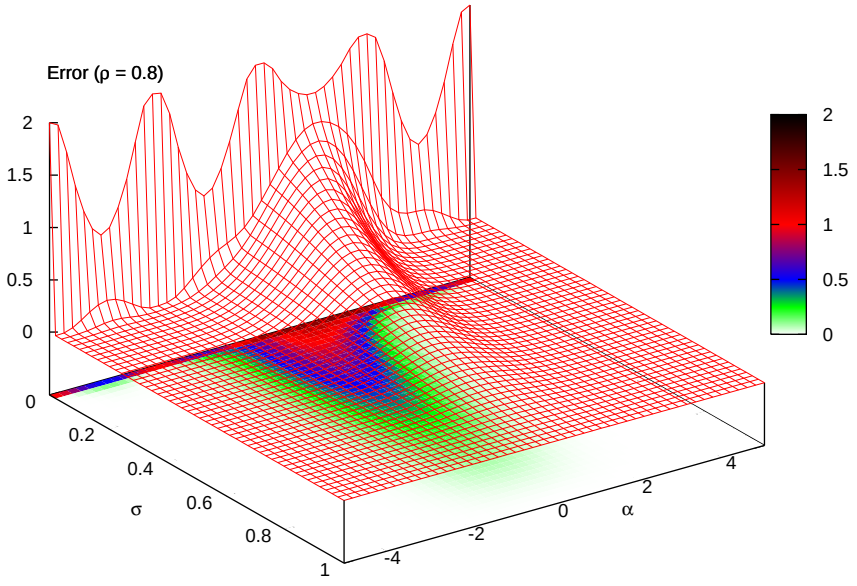

Figure 5.4. $\Xi_{(Z')(Z'_G)}$ as a function of $\sigma_{Y'}$ and α ($\rho = 0.8$)

dominate the other, *i.e.*, $|\alpha| \geq 3$ and increases for Gaussians that contribute almost equally to the *max i.e.*, α in the neighborhood of 0. $\Xi_{(Z')(Z'_G)}$ is found to increase with decreasing $\sigma_{Y'}$ and is convex with respect to the correlation coefficient. Figures 5.2−5.4 show the surface plots of $\Xi_{(Z')(Z'_G)}$ as functions of α, $\sigma_{Y'}$ and ρ. The presented plots reveal that while the *max* of two Gaussians can be very well approximated with a Gaussian in some cases, the approximation in other cases yields large errors.

5.2.4. Look-up table for error storage

The computation of the mean and variance of the *max* of two Gaussians involve the evaluation of a definite integral $\Phi(\alpha)$ and an exponential $\phi(\alpha)$. Numerical computations for their accurate estimation is CPU expensive. We consider the infinite Taylor series expansion [**64**]

about a point k of $\Phi(\alpha)$.

$$\Phi(\alpha) \;=\; \Phi(k) + \Phi^{(1)}(k)(\alpha - k) + \ldots + \frac{\Phi^{(n)}(k)}{n!}(\alpha - k)^n \ldots \tag{5.18}$$

$\Phi^{(n)}(k)$ and $\phi^{(n)}(k)$ represent the nth derivatives of $\Phi(k)$ and $\phi(k)$, respectively. We observe that the nth derivative of $\Phi(\alpha)$ (and $\phi(\alpha)$), for any $n > 0$, is a product of $\phi(\alpha)$ and a polynomial in α as:

$$\begin{aligned}
\Phi^{(1)}(\alpha) &= \phi(\alpha) \\
\Phi^{(2)}(\alpha) &= \phi^{(1)}(\alpha) = \phi(\alpha)(-\alpha) \\
\Phi^{(3)}(\alpha) &= \phi^{(2)}(\alpha) = \phi(\alpha)(\alpha^2 - 1) \\
\Phi^{(4)}(\alpha) &= \phi^{(3)}(\alpha) = \phi(\alpha)(-\alpha)(\alpha^2 - 3) \\
\Phi^{(5)}(\alpha) &= \phi^{(4)}(\alpha) = \phi(\alpha)(\alpha^4 - 6\alpha^2 + 3) \\
&\vdots \\
\Phi^{(n+1)}(\alpha) &= \phi^{(n)}(\alpha) = \phi(\alpha)P_n(\alpha),
\end{aligned}$$

where, $P_n(\alpha)$ is a polynomial of order n in α, $\forall n \geq 0$. It is observed that $P_n(\alpha) = (-1)^n(2)^{-\frac{n}{2}}H_n(\frac{\alpha}{\sqrt{2}})$, where H_n denotes the Hermite polynomial [**65, 66**]. A closed form expression for $\Phi^{n+1}(\alpha)$ is now expressed as: (Proof in Appendix A.3)

$$\Phi^{(n+1)}(\alpha) = \phi^{(n)}(\alpha) = \phi(\alpha)(-1)^n\, n! \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor} \frac{(-1)^i\, \alpha^{n-2i}}{2^i\, i!\, (n-2i)!}. \tag{5.19}$$

We use the above Taylor series expansion to compute $\Phi(\alpha)$ and $\phi(\alpha)$ in the region $|\alpha| < 4$. We propose to pre-compute $\Phi(k)$ and $\phi(k)$ for multiple values of $k \in [\,0, 4)$[1], and store the values in two look-up tables (LUTs). Thereafter, $\Phi(\alpha)$ ($\phi(\alpha)$) is computed by a table lookup on the closest k in the vicinity of α to obtain $\Phi(k)$ ($\phi(k)$), followed by a finite Taylor series expansion about k. For a uniformly sampled LUT with step-size p, it is observed that a very high degree of accuracy can be obtained by expanding few (typically 3 to 4) terms of the Taylor series expansion. We can obtain any desired accuracy in the computation of $\Phi(\alpha)$ and $\phi(\alpha)$ by either decreasing p while keeping n constant or by increasing n for a given p. A discussion on the Taylor series expansion truncation-error bounds and an application is presented in Appendix A.4.

We extend this idea to using LUTs for storing quantified approximation errors next. Given that the approximation error for any two Gaussians is a function of three bounded parameters for all practical purposes, we accurately evaluate approximation errors at discrete points in the bounded space of these parameters. These evaluated errors are stored in a three dimensional finite LUT. Since time is not a constraint in construction of the table, the LUT can be created with as much accuracy as desired. The error estimation for a given point is evaluated from (5.8)−(5.12). Approximation error estimation for a *max* operation is now performed very efficiently by a simple transformation to evaluate $\sigma'_Y = \frac{\sigma_Y}{\sigma_X}$ and a subsequent table lookup.

5.3. Error minimization problem

We consider the *max* operation on n Gaussian random variables $X_0, \ldots, X_{n-1}$, such that

$$X_M \;\triangleq\; max\big(X_0, X_1, \ldots, X_{n-1}\big).$$

[1]It is known that $\Phi(-k) = 1 - \Phi(k)$ and $\phi(-k) = \phi(k)$

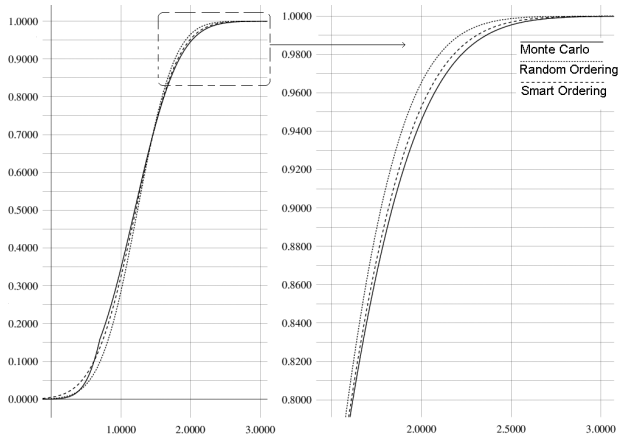

Figure 5.5. CDF comparisons of two orderings

Pair-wise *max* operations are performed on the given Gaussians to yield a Gaussian $X_G \sim N(\mu_{X_G}, \sigma^2_{X_G})$, which is used to approximate X_M. The loss in accuracy of the final result is dependent on the ordering of the pair-wise *max* operations. This is because the introduced inaccuracy for each pair-wise *max* operation is a function of the Gaussian parameters of the pair themselves and can accumulate or get reduced. Figure 5.5 compares resulting Gaussian CDFs from two orderings with the CDF of X_M obtained using Monte Carlo simulations.

The *max* operation on n Gaussians is analogous to the construction of a binary tree with n leaves such that each internal node computes the *max* of its two children. We refer to this tree as a *Max Binary Tree* (MBT) in the rest of this chapter. Given n Gaussians, the *max* (X_M) of which we want to estimate and approximate with X_G, the Error minimization problem is to create an MBT that yields some X_G at its root such that $\Xi_{X_M X_G}$ is minimized.

According to Knuth [**67**], the total number of different *labeled oriented* binary trees with n leaves is $\binom{2n-1}{n-1}(2n-2)!/2^{n-1}$. In an MBT, only the leaves are labeled. Therefore, the

total number of different MBTs is

$$\frac{\binom{2n-1}{n-1}(2n-2)!}{2^{n-1}(n-1)!} > (2n-1)^{n-1}.$$

An exhaustive enumeration is thus, prohibitive in solving this problem. Consequently, we consider good MBT construction approaches for error reduction.

5.4. Intelligent Max Binary Tree construction approaches

In this section, we present novel approaches for constructing good MBTs based on the study in the previous sections. We assume that a *max* operation takes $\Theta(1)$ time in complexity analysis.

5.4.1. Simple Max Binary Tree

This approach constructs the MBT as a skewed binary tree. A *max* is performed on two of the n given Gaussians to yield a new approximated Gaussian. Another *max* operation follows, which evaluates the *max* of the *max* obtained in the previous stage and one of the remaining $(n-1)$ given Gaussians. This process is repeated $(n-1)$ times to obtain a Gaussian, which is used to approximate the *max* of the n given Gaussians. The complexity of this approach is $\Theta(n)$.

5.4.2. Partition Max Binary Tree

The Partition MBT approach attains to reduce the depth of approximation errors accumulated in the $(n-1)$ stages of the previous approach by constructing a balanced binary tree. The given set of Gaussians is randomly partitioned into two subsets. The subsets are then

further bi-partitioned, and the process is done recursively, until the subset contains no more than two Gaussians. A *max* operation is then performed on the Gaussians in each subset. The results are now propagated backward to evaluate the *max* of these values bottom up. The MBT formed using this approach is balanced, and reduces the depth of accumulation of approximation errors from $(n-1)$ to $(log\ n)$. The complexity of this approach is maintained at $\Theta(n)$.

5.4.3. Greedy Max Binary Tree

This MBT construction method involves a greedy approach to reduce approximation errors. Based on the study in the previous sections, the Greedy MBT approach iteratively computes the *max* of two Gaussians from the given set, such that the approximation error for that pair is the least in comparison to all other pairs. The computed *max* is then returned to the original set and the process continues for $(n-1)$ similar iterations. The method is analogous to a graph reduction problem. We consider a fully connected graph with n nodes, each representing a given Gaussian. Edges of the graph contain weights that denote the approximation error in the *max* operation of the pair of nodes it joins. Adjacent nodes of the edge with the least weight are combined using a *max* operation. The corresponding nodes are combined into one and edge weights are incrementally recomputed. This process is repeated $(n-1)$ times, after which the graph is left with a single node, which contains the approximated *max* of the given Gaussians. A LUT is used to evaluate the edge weights in the graph. An alternate metric for any edge weight could be the skewness of the *max* of its adjacent nodes. Ξ for the adjacent nodes of a given edge can also be evaluated analytically from the higher order moments of the *max* in a way similar to that in [**68**]. The Greedy

MBT approach reduces given identical Gaussians (those having same means, variances and $\rho = 1$) to a single one. The complexity of this approach is $\Theta(n^3)$.

5.4.4. Cluster Max Binary Tree

The Cluster MBT approach is constructed as a combination of the Partition MBT and the Greedy MBT approaches. The *max* operation is performed in a greedy way on a Gaussian pair that yields the minimal error in approximation among all other pairs. However, the computed *max* is not sent back to the set of the given Gaussians as in the previous approach. A new pair is selected from the set of given Gaussians for the *max* operation iteratively till at most one Gaussian is left. The process restarts with the computed *max* distributions as the given set of Gaussians henceforth. The approach ensures that the constructed MBT is balanced and tries to reduce accumulation of approximation errors by constraining the maximum depth of the tree to $(log\ n)$. This approach reduces identical Gaussians to fewer ones, but is guaranteed to reducing them to one only when the number of identical Gaussians is 2^k for some positive integral value of k. The complexity of this approach is $\Theta(n^2)$.

5.5. Experimental results

We present experimental results of the proposed MBT construction approaches in this section. We construct a LUT to store approximation errors as presented in Section 5.2.4. Experimental results presented are for a LUT having 2×10^6 entries ($100 \times 100 \times 200$ for $\sigma_{Y'}$, ρ and α, respectively). The simulations to generate the LUT take 4 hours on a Pentium 2.4GHz machine, with 1GB RAM. The Greedy and the Cluster MBT approaches use the LUT to pick Gaussian pairs. The standard deviation of each Gaussian is constrained within 20% of its mean value.

We consider the *max* operation on randomly generated sets of 3 to 100 Gaussians and compare the error of the *max* obtained in different approaches with the distribution obtained from 10^5 Monte Carlo simulations, which we assume golden. 1000 runs are performed for each set to obtain an average. The obtained experimental results reveal that the proposed Partition MBT, Greedy MBT, and the Cluster MBT approaches reduce the loss in accuracy of Ξ by up to 24% with respect to the results obtained from the Simple MBT approach.

In statistical timing, we are concerned about the estimation accuracy of specific probability points in the CDF. We define a probability point $V_{Pr=p}$ for a random variable X as:

$$V_{Pr=p} \triangleq x : (Pr(X \leq x) = p), \quad p \in [0, 1].$$

We compare accuracy gains obtained from the proposed approaches with respect to the Simple MBT approach. The absolute value of the difference in $V_{Pr=p}$ points obtained from the CDF of the constructed MBT and the Monte Carlo (MC) CDF is normalized by $V^{MC}_{Pr=p}$ and multiplied by 100 to denote the error percent. The error percent of the Simple MBT approach is used as a reference for comparisons. For each of our proposed approaches, we denote the value by which the error percent of the Simple MBT method exceeds the error percent of the proposed approach as the *%Gain*. Given a probability point p, the *%Gain* obtained in approach A (could be Partition, Cluster or Greedy) is formally defined as:

$$\%Gain \triangleq (\frac{|V^{Simple}_{Pr=p} - V^{MC}_{Pr=p}|}{V^{MC}_{Pr=p}} \times 100) - (\frac{|V^{A}_{Pr=p} - V^{MC}_{Pr=p}|}{V^{MC}_{Pr=p}} \times 100). \tag{5.20}$$

Figure 5.6 illustrates this definition graphically. The *%Gain* reflects the absolute value of the gain in accuracy and does not reflect the ratio of the gain in accuracy of an approach over

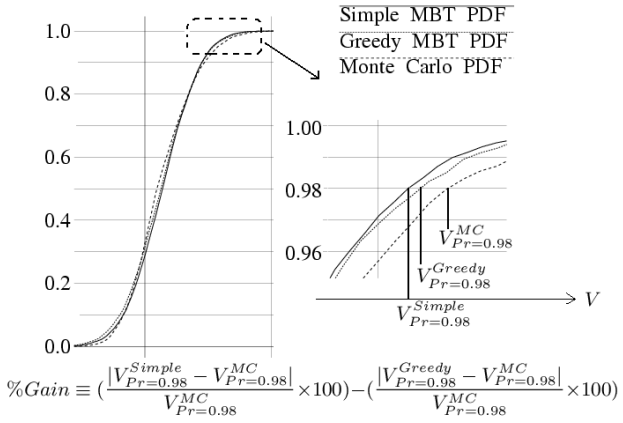

Figure 5.6. % Gain for Greedy MBT with respect to $V_{Pr=0.98}$

the other. Tables 5.1−5.3 present *%Gains* of the proposed approaches for the probability points 0.5 (mean), 0.95 and 0.998, respectively. We present the number of Gaussians (N) in the given set, the average *%Gain* obtained (*Gain*) and the maximum *%Gain* ($MaxG$) obtained in the simulations. Table 5.4 presents the %Gain obtained in the accuracy of estimating the variance of the *max*.

The proposed approaches are found to improve *%Gain*. Since the average error percent in estimation of the $V_{Pr=p}$ points in the Simple MBT approach is experimentally found to be less than 2%, the *%Gain* values obtained are relatively significant. For example a *%Gain* of 0.5% implies a $\frac{0.5}{2} \times 100 = 25\%$ improvement in relative accuracy. The proposed approaches are heuristics and do not guarantee optimality. The Partition MBT approach performs better than the Simple MBT approach on the average and indicates that the depth of cumulative error accumulation causes a difference. The proposed approaches that use the LUT show absolute maximal improvements by up to 5.2% in improving the accuracy of estimation of the critical probability points in the CDF and by up to 50% in estimation of the variance of the *max*. On the average, we find the Greedy approach performs the best. Run time are comparable for sets having up to 30 Gaussians. For the *max* of 50 Gaussians,

Table 5.1. % Accuracy gain results in $V_{Pr=0.5}$ $(mean)$

N	Partition		Cluster		Greedy	
	Gain	MaxG	Gain	MaxG	Gain	MaxG
3	0.00	0.5	0.01	0.5	0.01	0.5
5	0.00	0.5	0.02	1.0	0.02	1.0
7	0.01	0.8	0.03	1.0	0.03	1.0
9	0.03	1.1	0.06	1.1	0.06	1.1
12	0.05	0.8	0.07	1.1	0.07	1.4
15	0.08	0.9	0.11	1.2	0.09	1.4
20	0.14	0.9	0.18	1.3	0.16	1.5
30	0.28	1.1	0.35	1.7	0.31	1.6
50	0.55	1.3	0.70	2.9	0.58	2.9
100	1.04	1.9	1.29	2.7	1.12	2.5

Table 5.2. % Accuracy gain results in $V_{Pr=0.95}$

N	Partition		Cluster		Greedy	
	Gain	MaxG	Gain	MaxG	Gain	MaxG
3	0.02	2.0	0.05	2.0	0.05	2.0
5	0.01	3.1	0.08	3.2	0.09	3.2
7	0.04	2.2	0.07	2.3	0.15	2.7
9	0.03	2.8	0.09	2.3	0.20	2.3
12	0.02	2.7	0.08	2.5	0.23	3.0
15	0.04	2.3	0.14	2.3	0.24	3.1
20	0.05	2.1	0.15	2.8	0.26	2.8
30	0.09	2.8	0.35	3.1	0.31	3.1
50	0.28	3.2	0.33	2.9	0.40	3.2
100	0.57	3.1	0.62	3.4	0.68	3.2

run time for the Simple, Partition, Cluster and Greedy MBT approaches are found to be 0.00007 seconds, 0.0002 seconds, 0.003 seconds and 0.29 seconds, respectively. Though the Cluster and Greedy MBT approaches are computationally more expensive than the Simple MBT approach, given that we intend to use this method for computing the *max* of a finite number of delay distributions in statistical timing, the run time overheads are acceptable. All experiments are performed on a Pentium 2.4GHz, 1GB RAM machine, running RedHat Linux 9.0.

Table 5.3. % Accuracy gain results in $V_{Pr=0.998}$

N	Partition		Cluster		Greedy	
	Gain	MaxG	Gain	MaxG	Gain	MaxG
3	0.01	2.7	0.07	2.7	0.07	2.7
5	0.01	3.8	0.17	3.8	0.18	3.8
7	0.00	3.4	0.10	3.9	0.23	3.8
9	0.06	4.5	0.28	3.6	0.31	4.0
12	0.06	3.7	0.16	4.6	0.31	3.6
15	0.08	3.6	0.14	3.6	0.38	4.3
20	0.10	3.0	0.25	3.6	0.41	4.7
30	0.12	3.3	0.27	4.1	0.50	5.2
50	0.04	3.4	0.23	4.0	0.37	4.3
100	0.08	3.5	0.17	4.1	0.34	3.8

Table 5.4. % Accuracy gain results in variance estimation

N	Partition		Cluster		Greedy	
	Gain	MaxG	Gain	MaxG	Gain	MaxG
3	0.09	42.4	0.54	41.4	0.54	41.4
7	0.08	28.8	0.86	31.6	1.87	31.6
9	0.43	35.7	2.32	27.7	2.54	30.6
12	0.53	36.0	1.35	50.0	2.42	33.1
15	0.68	33.5	1.22	27.6	3.05	34.5
20	0.78	22.6	1.88	33.4	3.06	32.5
30	0.67	24.9	2.04	36.6	3.63	39.8
50	0.24	29.4	2.31	33.9	2.92	32.3
100	0.00	23.3	0.92	28.5	1.50	26.6

5.6. Conclusions

In this chapter, we quantify the approximation error in Clark's approach [**59**] to computing the *max* of Gaussians. We propose approaches to different orderings for pair-wise *max* operations on a set of Gaussians based on error computations. Prior research [**62**] has shown that the average errors in estimating the mean and standard deviation of a circuit's arrival time distribution in comparison to Monte Carlo simulations are $\approx 1.8\%$ and 13.7%, respectively. Similar numbers are shown in [**46**]. The proposed approaches significantly improve the accuracy in variance estimation on the average, errors in which could be large otherwise.

They also improve the estimation accuracy of specific probability points, which could have significant errors (about 4.56% on the average for $V_{Pr=0.99}$ [**46**]).

We believe that the proposed approaches would increase the accuracy in the estimation of node and edge criticalities [**48**], and would thereby guide statistical timing optimization better. In addition, expressions for the CDF and PDF of the true *max* of two Gaussians would help in accurate yield estimation, when considering both timing and power [**62**].

CHAPTER 6

A Unified Framework for Statistical Timing Analysis with Coupling and Multiple Input Switching

An increasing significance of variability with decreasing feature sizes necessitates the consideration of the parametric variations in timing analysis for accurate timing estimation. Statistical timing analysis has thus emerged in an attempt to capture the statistical behavior of circuit delays under parametric variations. In addition, increasing aspect-ratios and decreasing widths of modern day interconnects have resulted in an increased dominance of coupling capacitances with technology scaling. Simultaneous switching on coupled nets in a circuit affects the delay on each net, and cause delay variations in the circuit. The delay variations are often significant (up to 40% stage delay error [**69**]) and should not be ignored. In addition, the mutual dependence of coupling effects and timing make timing analysis a chicken-and-egg problem. A coupling between two wires that lie in the fanin and fanout cone, respectively of a logic gate causes timing dependencies that necessitates iterations during timing analysis. A situation of coupling is initially assumed (often the worst case situation) and the computed timing information is used to modify the coupling situation. This procedure is repeated until convergence.

The gate delay for multiple input CMOS gates often depends on the number of inputs switching at the same time. For example, turning on two transistors in parallel is faster than using only one as the path resistance is lower. The assumption of only single input switchings for timing analysis and consequently ignoring the effect of multiple input switchings (MIS)

can result in significant errors (up to 20% stage delay error [**70**]). Although the probability of several multiple-input switchings adding up along a path in a circuit is small, delay variations due to the temporal proximity of switching inputs should be considered in accurate timing analysis.

The increasing significance of the above factors on timing accuracy motivates considering them in a unified timing analysis framework. Approaches to statistical timing analysis have gained wide acceptance recently [**42, 43, 44, 45, 71, 48, 49**]. Statistical timers treat delays as correlated random variables and propagate their distributions through the circuit. These approaches, however, do not discuss coupling induced delay variations or effects due to MIS. On the other hand, the majority of prevalent timing analysis approaches that consider coupling treat delays as deterministic values and do not discuss the impact of variability [**72, 73, 74, 75, 76, 77, 69, 78, 30**]. A statistical timing analysis algorithm that accounts for correlations and accommodates dominant interconnect coupling is proposed by Le *et al.* in [**47**]. While considering coupling in their proposed approach, they estimate switching window overlaps deterministically, based on worst case values. Our experiments conclude that such an approach is pessimistic and over estimates the switching windows at the outputs. A gate delay model that accounts for the effects of temporal proximity and input transitions is proposed in [**79**]. Another statistical gate delay model that considers MIS is proposed in [**70**]. Dartu *et al.* present results on the significance of the mentioned effects on timing accuracy and recommend increased focus on consideration of these effects in timing analysis for modern circuits [**80**].

In this chapter, we establish a theoretical framework for statistical timing analysis with coupling. Our contributions are summarized as follows.

- We describe the generic problem of timing analysis under uncertainty from the parameters of variation and delay variations due to coupling. We compare and contrast the uncertainty introduced by each of these effects (due to variability and ignorance of functional information, respectively) and propose considering them in a unified framework.
- We introduce the concept of statistical switching windows, which is a novel extension to the idea of switching windows in deterministic static timing analysis.
- We propose an iterative approach for statistical timing analysis with coupling as a fixpoint computation on a lattice and establish its convergence.
- We present a probabilistic view of the overlap between two statistical switching windows. Using a generic coupling model, we discuss the computation of a coupling induced delay pushout as a random variable.
- We consider the implications of the assumption of Gaussian distributions for the parameters of variation in statistical timing analysis with coupling. The concept of a Gaussian switching window is described and we present the computation of a delay pushout under the given assumption.
- We compare and contrast the effect of MIS with coupling. Using a simple MIS model as an example, we illustrate that our framework is amenable to incorporating that effect in a unified way.

We develop a statistical timer that considers coupling based on our proposed approach, and present results obtained for the ISCAS'85 benchmarks [**36**]. We additionally develop the following two timers for comparison.

- A single pass statistical timer based on [**48**], which does not consider the mutual dependence of coupling and timing.
- An iterative statistical timer that considers coupling deterministically, based on [**47**].

Monte Carlo simulations reveal a distinct gain in accuracy (up to 24%) of our approach in comparison to the others.

The rest of this chapter is organized as follows. Section 6.1 describes statistical timing analysis with coupling as a fixpoint computation problem on a lattice. Section 6.2 presents the generalized computation of coupling induced delay pushouts as random variables. We discuss the implications of the assumption of Gaussian distributions for the parameters of variation, and simplify the computation of the delay pushout under the given assumption in Section 6.3. We show that the effects of MIS are amenable to our framework in Section 6.4, and present experimental results and comparisons to other approaches in Section 6.5. Conclusions are drawn in Section 6.6.

6.1. Statistical timing as fixpoints

We consider a combinational circuit and select a set of interconnect wires where timing information needs to be computed. This set includes the primary inputs, the primary outputs, and the inputs and outputs of all gates in the circuit. In statistical timing, we use random variables with known distributions to denote timing information. Depending on the purpose of timing analysis, the timing information could be a delay, a skew, a switching window or a combination of them. Symbolically, we use a variable x_i to denote the timing information on a wire i. Furthermore, we use X to represent the vector $(x_1, x_2, \ldots, x_n)$, that is, the timing information of the whole circuit.

Depending on the actual delay model, the timing information on a wire i depends only on a subset of other wires $i_1, i_2, \ldots, i_k$. These wires may include all inputs of the gate fanning out to i, and the wires coupled to i. This implies that x_i can be computed as:

$$x_i \;=\; t_i(x_{i_1}, x_{i_2}, \ldots, x_{i_k})$$

where, the function t_i is decided by the physical configuration of the circuit and the delay model used. Such a function for computing the timing information on a wire is called a local timing transformation. Putting all local transformations together, we get a global timing transformation T for the entire circuit, which can be written as the following.

$$X \;=\; T(X) \tag{6.1}$$

A solution of timing analysis must be an X satisfying (6.1). Such a solution is also called a fixpoint of T. This formulation is amenable to timing analysis considering variability, coupling, and MIS. Coupling causes mutual dependence of the timing information on coupled wires, and may create cyclic dependencies during timing analysis. For a complex transformation T, an iterative method is perhaps the only possible way to find its fixpoint. First an initial solution X_0 is guessed, then new solutions are iteratively computed from previous solutions $X_1 = T(X_0), X_2 = T(X_1) = T^2(X_0), \ldots$ until we find a fixpoint, that is, $X_m = T^m(X_0)$ such that $T(X_m) = X_m$.

6.1.1. Statistical timing with coupling

Considering functional information in timing analysis involves enumerating all input vectors which is exponential in computational complexity. As a result, static timing analysis does

not use functional information of a circuit. Such a treatment makes timing information on each wire to be a set of possible signal switchings, instead of a single signal switching. The set is represented by a switching window and a set of slew rates such that any signal switching falling in the window and having a slew in the range is in the set. The switching window x_i on a wire i denotes an interval $[x_i^l, x_i^u]$, such that any actual signal switching x_i^* lies in the interval. Thus,

$$\{x_i^* : x_i^l \ \leq \ x_i^* \ \leq \ x_i^u\}$$

denotes the set of all possible signal switchings. The switching window representation is a worst case representation as it only gives bounds (best and worst case) on the elements of the set. It does not provide any information on the distribution of signal switchings within the window.

Statistical timing analysis considers the uncertainty in circuit component delays due to parametric variations. Assumptions on the distribution of the parameters of variation and on the delay model as a function of these parameters provide a known distribution of signal switchings on every wire in the circuit.

Statistical timing with coupling involves uncertainties from both variability and ignorance of functional information. Although both uncertainties prohibit the solution to timing analysis on a wire from being a single signal switching, we do not treat them in the same way. This is because we have information about the delay distribution due to uncertainty from variability, but not so for uncertainty due to ignorance of functional information. Consequently, we cannot obtain the solution to statistical timing analysis with coupling as a known distribution of signal switchings on each wire of the circuit.

The solution to statistical timing analysis on any wire i is a distribution obtained from a statistical sampling of the timing information on i at various corners in the parameter space. For any corner in the parameter space, that is, a given assignment of parametric values, the timing information on i is in the form of a switching window. Consequently, the solution to statistical timing analysis with coupling is a *distribution of switching windows* on each wire of the circuit. This view of the solution is obtained by first considering the uncertainty from ignorance of functional information, and then the uncertainty from variability.

Each window in the above view of the solution is denoted by a best and a worst case value. Consequently, the distribution of the windows contains the distributions of these best and worst case values. The two distributions thus obtained, are represented as the distributions of two correlated random variables. The window formed using these random variables as the best and worst case values contains all possible signal switching distributions in the solution. This transformation of the original view of the solution gives an alternate view of the solution to statistical timing analysis with coupling as a *window of signal switching distributions*.

The latter view of the solution is also obtained by considering the uncertainty from variability first. Temporarily assuming that functional information and input configuration of each gate is known, statistical timing analysis on a wire i computes a distribution of a single signal switching. However, functional information and input configuration is not considered in reality. The timing information on i is therefore a set of signal switching distributions, which is represented by a window of signal switching distributions.

6.1.2. Statistical switching windows

Each view of the solution to statistical timing analysis contains the exact same information. We consider the solution as a window of signal switching distributions, since it allows us to

leverage the theoretical foundations of timing analysis with coupling as a fixpoint computation developed in [**30**]. We cannot use traditional switching windows in our formulation as they represent a set of deterministic quantities, while our set consists of signal switchings as random variables with known distributions.

We introduce a *statistical switching window* as a representation for a set of random variables. For any random variable x_i^* in the set, we consider providing a lower and an upper bound using two correlated random variables x_i^l and x_i^u, respectively. We call these two random variables the lower and upper bounding Random Variable (or *bounding RV*) of the statistical switching window x_i. The bounding RVs are correlated since they are functions of common parameters of variation. Mathematically, the statistical switching window x_i is defined as the following.

$$x_i = [x_i^l, x_i^u] \quad \triangleq \quad \{x_i^* : Pr(x_i^l \leq x_i^* \leq x_i^u) = 1\} \tag{6.2}$$

where, $Pr(k)$ denotes the probability of an event k. A statistical switching window is illustrated graphically in Figure 6.1(a). In this figure, the distributions of various random variables are shown to be bounded within the distributions of x^l and x^u. In the rest of this chapter, we consider the solution to statistical timing analysis with coupling on a wire i as a statistical switching window x_i of signal switching distributions.

6.1.3. Structure of fixpoints

An inclusion relation ($\subseteq$) between two statistical switching windows x_i and x_j is formally defined as:

$$x_i \subseteq x_j \quad \triangleq \quad Pr(x_i^l \geq x_j^l) = 1 \ \wedge \ Pr(x_i^u \leq x_j^u) = 1. \tag{6.3}$$

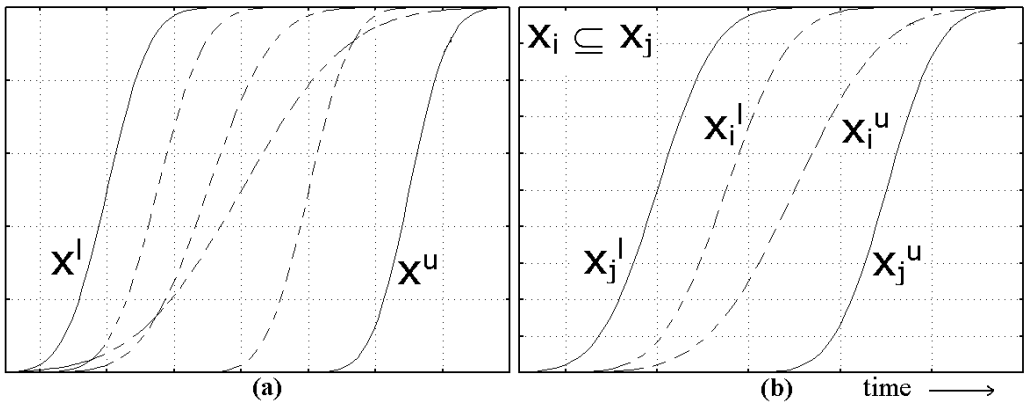

Figure 6.1. (a) A statistical switching window for a set of distributions (b) Inclusion relation between two switching windows

Figure 6.1(b) illustrates the definition graphically. In the figure, the statistical switching window x_i is shown to be contained within the statistical switching window x_j, that is, the distribution of x_i^l is upper bounded by that of x_j^l, while the distribution of x_i^u is lower bounded by that of x_j^u.

We consider the family of all sets of signal switching distributions on a wire. The inclusion relation ($\subseteq$) forms a partial order on the family as it satisfies the properties of reflexivity, transitivity, and antisymmetry [**30**].

A set of signal switching distributions on a wire is actually a subset of the whole set that consists of all possible signal switching distributions. The inclusion relationship between statistical switching windows is the inclusion relationship between these subsets and thus forms a partial order. According to lattice theory [**37**], a partially ordered set forms a *complete lattice* if any subset has a least upper bound and a greatest lower bound of its members. The family of all subsets of a given set with an inclusion relation forms a complete lattice. The proof is provided in [**30**]. Timing information of a circuit is a vector of timing information on all wires. The partial order on each wire can be extended point-wise to get a partial order on vectors. We say that two timing information vectors $X = (x_1, x_2, \ldots, x_n)$

and $Y = (y_1, y_2, \ldots, y_n)$ satisfy $X \subseteq Y$ if and only if $x_i \subseteq y_i$ for all $1 \leq i \leq n$. It can be shown that the vectors with such a partial order also form a complete lattice. We consider the complete lattice structure since the existence of a fixpoint to (6.1) under certain conditions in the lattice is guaranteed.

We now consider the transformation T, which transforms a vector of sets of signal switching distributions to another vector of sets of signal switching distributions. We say that T is a *monotonic* transformation when, for any two vectors X and Y, if $X \subseteq Y$, then $T(X) \subseteq T(Y)$. The monotonicity of T is proved from the monotonicity of its member (or local) timing transformations $t_1, t_2, \ldots, t_n$. We do not restrict our approach to any particular timing model. Theoretically, any timing model must satisfy the monotonic property for each local timing transformation. If a transformation t_i is not monotonic, it implies less possible switching distributions are produced under the condition of the same or more possible switching distributions on the fanin and coupling wires. Such a model is not theoretically sound as it contradicts with the causality of physical effects. Under such a timing model, iterations for timing analysis with coupling may not converge. For such models, iterations are often started with an initial worst case assumption of maximum possible coupling induced delay variations. A finite number of iterations are performed to reduce pessimism in the solution, but not till convergence. However, for a theoretically correct timing model, the existence of a fixpoint of the monotonic transformation T is guaranteed by Knaster-Tarski Theorem [**37**].

6.1.4. Searching for a fixpoint

We use an iterative approach to compute the fixpoint of our transformation. The assumed initial solution in the iterative approach is critical to reaching the desired fixpoint. A random initial solution may keep the iterations going on forever. Based on the monotonicity of the

timing transformation T, the top and bottom elements of our lattice are good candidates for the initial solution. In traditional lattice theory, $\perp$ and $\top$ denote the bottom and the top element of a lattice, respectively. In our lattice, we have $\perp = (\emptyset, \emptyset, \ldots, \emptyset)$ and $\top = (P_1, P_2, \ldots, P_n)$, where $\emptyset$ denotes an empty set while P_i is the set of all possible signal switching distributions on wire i, and can be computed by assuming the effect of coupling everywhere. Since $\perp \subseteq T(\perp)$, based on the monotonicity of T, we have

$$\perp \subseteq T(\perp) \subseteq T^2(\perp) \subseteq T^3(\perp) \quad \ldots$$

which gives us an ascending chain of timing information vectors. Similarly, a descending chain is obtained if we start with $\top$. We next consider the convergence of these chains to a fixpoint.

6.1.5. Convergence to a fixpoint

A discrete delay model has a finite number of statistical switching windows on each wire ideally. For example, we consider a simple coupling model that gives the worst-case additional coupling induced delay on a wire as either of two random variables depending on whether the switching window on that wire overlaps with that of a coupled wire or not. In this case, the maximum possible number of delay random variables for any wire is 2^k, where k is the total number of couplings in the circuit. Under similar discrete models, each wire has ideally a finite number of statistical switching windows. Any chain of timing information for a wire in the solution space has thus a finite number of elements. This implies that the iterative process will always reach a fixpoint in finite steps. The monotonic property of T when we start from $\perp$ or $\top$ is sufficient to prove convergence. In fact, any solution X_0 such that

$X_0 \subseteq T(X_0)$ or $X_0 \supseteq T(X_0)$ can be used as an initial solution to reach a fixpoint. The number of iterations to reach a fixpoint is upper bounded by the sum of the lengths of the longest chains of all timing information variables in the lattice.

Under a continuous delay model, it is possible for the chains generated in the iterative process to have infinite elements. This implies that the generation of an exact fixpoint in finite steps is not guaranteed. However, if the iteration does theoretically converge to a fixpoint, an approximation to the fixpoint can be found. For such an approximated converged fixpoint to be a real fixpoint, stronger properties than the monotonicity are needed for the transformation T.

In general, a statistical gate delay model defines a response waveform distribution as a function of switching waveform distributions on a gate's fanins and wires coupled to its fanout net. In statistical static timing analysis, this model needs to be expanded to compute a set of response distributions as a function of sets of waveform distributions on the fanins and coupling wires. A general way to expand a function from a set to its power set is by natural lifting.

Definition 6.1.1. *Given $f : S \to S$, a natural lifting $F : 2^S \to 2^S$ is*

$$F(A) \;=\; \bigcup_{a \in A} f(a), \qquad \forall A \in 2^S.$$

Given a subset S of elements in a complete lattice L, we use $\bigvee S$ to represent the least upper bound of elements in S. A function $T : L \to L$ is said to be *or-continuous* if for any chain C, $T(\bigvee C) = \bigvee T(C)$. A timing transformation defined by a natural lifting is shown to be or-continuous [**30**]. Given that T is or-continuous, it is known that $\bigvee_{n \geq 0} T^n(\perp)$ is a fixpoint. This establishes that our iterative approach always converges to a fixpoint, given

that we start our iteration from the bottom element of the lattice. A fixpoint may however, not be reached if we start our iterations from the top element [**30**]. This provides a strong reason why timing analysis iterations should start from the bottom element.

6.1.6. Optimal fixpoint

It is known that if fixpoints of a monotonic transformation are found by the iterative method starting from the bottom element, it must be the least fixpoint. It is also known that the fixpoints of a monotonic transformation form a complete lattice. The least fixpoint is therefore the intersection of all the fixpoints. In terms of statistical switching windows, a fixpoint with the smallest window will result from an initial assumption that the probability of any two windows overlapping with each other is zero. Therefore, in order to find the optimal fixpoint, that is the vector of tightest switching windows on the real solution (the one with minimal pessimism), we should start with an initial solution where the probability of an overlap between the statistical switching windows on any two coupled wires is zero. Subsequent application of T until convergence guarantees the optimal solution.

6.2. Coupling induced delay pushouts as random variables

6.2.1. Overlap between statistical switching windows

The overlap between two statistical switching windows is not a deterministic quantity. For two statistical switching windows x_i and x_j, we denote the overlap between the windows by a random variable O_{ij}, which is defined as the following.

$$O_{ij} \triangleq min(x_i^u, x_j^u) - max(x_i^l, x_j^l) \tag{6.4}$$

We consider a probabilistic view of the overlap O_{ij}. If we denote the CDF of O_{ij} by $\Psi_{ij}(t)$, the probability of an overlap between x_i and x_j is given by

$$Pr(O_{ij} > 0) \;=\; 1 - \Psi_{ij}(0). \tag{6.5}$$

6.2.2. Coupling induced delay pushout computation

Simultaneous switchings on a pair of coupled nets cause delay variations on the nets. The amount of additional delay (positive or negative) induced is called the *coupling induced delay pushout* and is a function of the overlap between the switching windows on the coupled nets. Since the overlap between two statistical switching windows is a random variable, the coupling induced delay pushout is consequently a random variable even if the delay pushouts for a deterministic overlap are constants. Considering the timing information on any wire as a set of signal switching distributions, coupling possibly induces additional signal switching distributions in the set. The new set must have an inclusion relationship with the original since any conservative coupling model will not remove any original switching distribution from the set. The delay pushout thus widens the switching window on a wire, that is, if x_i and x_i' denote the switching windows on a wire i when effects due to coupling are not considered and are considered respectively, we have the following relation.

$$x_i \;\subseteq\; x_i' \tag{6.6}$$

Without any loss of generality, we consider the computation of the worst case delay on a wire i when coupling effects are considered. The worst case delay on i is given by the sum of the delay on the wire when coupling effects are not considered and the coupling induced delay pushout due to each wire it couples with.

As an example, we illustrate the computation of a coupling induced delay pushout D as a random variable based on a simple coupling model, which is given by the following.

$$D \triangleq \begin{cases} D^O & \text{overlap} \\ D^N & \text{no overlap} \end{cases} \tag{6.7}$$

where, D^O and D^N are the values assigned to D, depending on whether the statistical switching windows between the coupled wires overlap or not, respectively. We denote the overlap as a random variable O.

Based on prior work in [**71, 48**], we express all random variables as a weighted sum of independent and orthonormal random variables ξ_i, $i = 1, 2, \ldots, n$, such that $E[\xi_i \xi_j] = 0$ if $i \neq j$, and $= 1$ otherwise. The delay pushouts D^O and D^N are random variables, and are expressed as the following.

$$\begin{aligned} D^O &= d_0^O + \sum_{i=1}^{n} d_i^O \xi_i \qquad (6.8) \\ D^N &= d_0^N + \sum_{i=1}^{n} d_i^N \xi_i \\ \Rightarrow D &= d_0 + \sum_{i=1}^{n} d_i \xi_i \triangleq \begin{cases} d_0^O + \sum_{i=1}^{n} d_i^O \xi_i & \text{overlap} \\ d_0^N + \sum_{i=1}^{n} d_i^N \xi_i & \text{no overlap} \end{cases} \end{aligned}$$

We define a new set of random variables $\mathbf{d_i}$s, $i = 0, 1, \ldots, n$ as the following.

$$\mathbf{d_i} \triangleq \begin{cases} d_i^O & \text{overlap} \\ d_i^N & \text{no overlap} \end{cases} \tag{6.9}$$

We therefore have the following.

$$d_0 + \sum_{i=1}^{n} d_i \xi_i \;=\; \mathbf{d_0} + \sum_{i=1}^{n} \mathbf{d_i} \xi_i \tag{6.10}$$

We multiply (6.10) by a new independent and orthogonal random variable ξ_{n+1} and take the expected value ($E[.]$) on both sides. We consider the orthogonality property of ξ_i, and that the overlap is a function of ξ_i, $i = 1, \ldots, n$ but independent of ξ_{n+1}. Using conditional expectation, we obtain the following results.

$$\begin{aligned}
d_0 &= \frac{1}{E[\xi_{n+1}]}\big(E[\mathbf{d_0}\xi_{n+1}] + \textstyle\sum_{i=1}^{n} E[\mathbf{d_i}\xi_i\xi_{n+1}]\big) \\
&= \frac{1}{E[\xi_{n+1}]}\big(E[\mathbf{d_0}] + \textstyle\sum_{i=1}^{n} E[\mathbf{d_i}\xi_i]\big) E[\xi_{n+1}] \\
&= E[\mathbf{d_0}] + \textstyle\sum_{i=1}^{n} E[\mathbf{d_i}\xi_i] \\
&= d_0^O Pr(\text{overlap}) + d_0^N Pr(\text{no overlap}) + \textstyle\sum_{i=1}^{n} E[\mathbf{d_i}\xi_i] \\
&= d_0^O Pr(O > 0) + d_0^N Pr(O \leq 0) + \textstyle\sum_{i=1}^{n} E[\mathbf{d_i}\xi_i], \qquad \text{where} \\
E[\mathbf{d_i}\xi_i] &= d_i^O E[\xi_i |\ O > 0] Pr(O > 0) + d_i^N E[\xi_i |\ O \leq 0] Pr(O \leq 0).
\end{aligned}$$

We multiply (6.10) again by ξ_k for $k = 1, 2, \ldots, n$ and match the expectations on both sides to obtain the following.

$$
\begin{aligned}
& d_k \\
&= \frac{1}{E[\xi_k^2]}\big(E[\mathbf{d_0}\xi_k] + \textstyle\sum_{i=1}^{n} E[\mathbf{d_i}\xi_i\xi_k] - d_0 E[\xi_k]\big), \quad \text{where} \qquad (6.11) \\
& E[\mathbf{d_i}\xi_i\xi_k] \\
&= d_i^O E[\xi_i\xi_k|\ O > 0]Pr(O > 0) + \ d_i^N E[\xi_i\xi_k|\ O \leq 0]Pr(O \leq 0) \qquad (6.12) \\
& E[\mathbf{d_0}\xi_k] \\
&= d_0^O E[\xi_k|\ O > 0]Pr(O > 0) + \ d_0^N E[\xi_k|\ O \leq 0]Pr(O \leq 0). \qquad (6.13)
\end{aligned}
$$

The above illustration of the computation of the delay pushout is extensible to an arbitrary coupling model and does not imply that the proposed approach is limited to a given coupling model. The model can be trivially extended to having M delay pushouts for given ranges of an overlap value which can be expressed as:

$$
D \triangleq \begin{cases} D^N & \text{no overlap} \\ D_i^O & \text{overlap} \in (O_i, O_{i+1}],\ \forall i = 1, 2, \ldots, M-1 \\ D_M^O & \text{overlap} > O_M. \end{cases}
$$

Under such a model, the above equations are modified to obtain the desired coefficients d_is. The conditional expectations are now evaluated for each of the possible overlap intervals. Even for a continuous coupling model (for example, the delay pushout as a exponential function of overlap), a similar approach is used to compute the desired coefficients. The conditional expectation computations in this case involves the PDF of the overlap instead

of probability that the overlap lies in some interval. We do not discuss details of such models as our work is model independent and guarantees convergence under the monotonicity properties of the timing transformation.

6.3. Practical considerations under a Gaussian assumption

6.3.1. Gaussian statistical switching windows

The parameters of variation in statistical timing are often assumed to be random variables having a Gaussian distribution. Gate delays are expressed as a linear weighted sum of these parameters and thereby have a Gaussian distribution too. We consider the following definition.

$$\Phi(t) \triangleq \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} exp(-\frac{x^2}{2})\ dx \tag{6.14}$$

$\Phi(t)$ denotes the CDF of a unit normal Gaussian. It is known that the CDF of an arbitrary Gaussian $X \sim N(\mu, \sigma^2)$ with mean μ and variance σ^2 is equivalent to $\Phi(\frac{t-\mu}{\sigma})$, where $\Phi(t)$ is defined as in (6.14).

The assumption of a Gaussian distribution for the bounding RVs when used in describing statistical switching windows prohibits the generic use of the $\subseteq$ relation between the windows.

Lemma 6.3.1. *No Gaussian can be a bounding RV for a set of Gaussians.*

Proof. We use contradiction to prove our claim and assume that a Gaussian $X \sim N(\mu_x, \sigma_x^2)$ is an upper bounding RV for another Gaussian $Y \sim N(\mu_y, \sigma_y^2)$. From (6.3),

we have the following.

$$\begin{aligned}
& Pr(X \geq Y) && = 1 \\
\Rightarrow \quad & Pr(X - Y \geq 0) && = 1 \\
\Rightarrow \quad & Pr(Z \geq 0) && = 1, \quad \text{where } Z \sim N(\mu_z, \sigma_z^2) \triangleq X - Y \\
\Rightarrow \quad & 1 - \Phi_z(0) && = 1, \quad \text{where } \Phi_z(t) = \Phi(\frac{t - \mu_z}{\sigma_z}) \\
\Rightarrow \quad & \Phi_z(0) && = 0,
\end{aligned}$$

which is a contradiction, since $\Phi(t) \in (0, 1)$ for real t. □

As a heuristic, we now consider a weaker condition on the bounding RVs x_i^l and x_i^u for any statistical switching window x_i as the following.

$$Pr(x_i^l \leq c) \geq Pr(x_i^* \leq c) \geq Pr(x_i^u \leq c), \quad \forall c \in \Re \tag{6.15}$$

It can be trivially proved (using contradiction) that the above condition is a necessary condition to (6.2). Agarwal *et al.* denote such bounding RVs as statistical upper and lower bounds in [**43, 81**]. We observe that the cumulative distribution functions (CDFs) of x_i^l and x_i^u provide a statistical interval for the CDF of the uncertain actual timing variable x_i^*. Similarly, we present a weaker condition on the bounding RVs of two statistical switching windows x_i and x_j to have an inclusion relationship ($x_i \subseteq x_j$) as the following.

$$Pr(x_i^l \leq c) \leq Pr(x_j^l \leq c) \quad \wedge \quad Pr(x_i^u \leq c) \geq Pr(x_j^u \leq c), \quad \forall c \in \Re \tag{6.16}$$

It can be trivially proved (using contradiction) that the above condition is a necessary condition to (6.3). We next show that the assumption of a Gaussian distribution for the

bounding RVs *even under the above weaker conditions* prohibits the generic use of the $\subseteq$ relation between the windows. We discuss the cause of this problem and propose a remedy in the remainder of this section.

Definition 6.3.1. *We say that a random variable X strictly dominates another random variable Y if and only if the CDF of X is upper bounded by the CDF of Y, that is*

$$Pr(X \leq c) \leq Pr(Y \leq c), \qquad \forall c \in \Re.$$

Lemma 6.3.2. *The CDFs of two arbitrary Gaussians with non-equal variances always intersect at exactly one point.*

Proof. We consider two arbitrary Gaussians having means μ_1, μ_2 and variances σ_1^2, σ_2^2, respectively. The point(s) of intersection of the CDFs is(are) found by equating the CDFs and solving for c. We thus solve for c in

$$\Phi(\frac{c-\mu_1}{\sigma_1}) \;=\; \Phi(\frac{c-\mu_2}{\sigma_2}). \tag{6.17}$$

It is known that $\Phi(y)$ is strictly increasing with y, that is $\Phi(x) > \Phi(y)$ if and only if $x > y$. The above equation is therefore equivalent to solving the following.

$$\frac{c-\mu_1}{\sigma_1} \;=\; \frac{c-\mu_2}{\sigma_2} \tag{6.18}$$

(6.18) is a linear equation in c and has a single root at $c^* = \frac{\sigma_2\mu_1 - \sigma_1\mu_2}{\sigma_2 - \sigma_1}$, given $\sigma_1 \neq \sigma_2$. Since both the CDFs are monotonically increasing, c^* denotes the single point of intersection of the CDFs. $\square$

Corollary 6.3.0.1. *For two arbitrary Gaussians with equal variance, the CDF of the Gaussian with a larger mean strictly dominates the CDF of the other.*

Proof. A real solution to (6.18) does not exist if $\sigma_1 = \sigma_2$. Consequently, $\Phi(\frac{c-\mu_1}{\sigma_1}) \geq \Phi(\frac{c-\mu_2}{\sigma_2})$, $\forall c \in (-\infty, \infty)$ if $\mu_1 \leq \mu_2$, and vice-versa. □

Lemma 6.3.3. *An arbitrary Gaussian can never strictly dominate another Gaussian, given that their variances are non-identical.*

Proof. For two arbitrary Gaussians having means μ_1, μ_2 and variances σ_1^2, σ_2^2 respectively, we assume without any loss of generality that $\sigma_2 > \sigma_1$. Based on Lemma 6.3.2 and given the monotonicity of the CDFs, the following results are immediate.

$$\Phi(\frac{c-\mu_1}{\sigma_1}) \geq \Phi(\frac{c-\mu_2}{\sigma_2}) \quad \forall c \in [\frac{\sigma_2\mu_1 - \sigma_1\mu_2}{\sigma_2 - \sigma_1}, \infty) \tag{6.19}$$

$$\Phi(\frac{c-\mu_1}{\sigma_1}) \leq \Phi(\frac{c-\mu_2}{\sigma_2}) \quad \forall c \in (-\infty, \frac{\sigma_2\mu_1 - \sigma_1\mu_2}{\sigma_2 - \sigma_1}] \tag{6.20}$$

Thus, neither of the Gaussians strictly dominate the other. □

Corollary 6.3.0.2. *If the ratio of the means of two arbitrary Gaussians is equal to the ratio of their variances, the CDF of the Gaussian with the smaller mean (or variance) strictly dominates the other in the interval $c \in [0, \infty)$.*

Proof. Given $\frac{\mu_1}{\mu_2} = \frac{\sigma_1}{\sigma_2}$ implies that $\sigma_2\mu_1 = \sigma_1\mu_2$. The proof is immediate when this result is substituted in (6.19). □

Corollary 6.3.0.3. *For two statistical switching windows x_i and y_i, $x_i \subseteq y_i$ implies that x_i^l strictly dominates y_i^l and y_i^u strictly dominates x_i^u.*

Proof. The proof is immediate from (6.3) and Definition 6.3.1. □

Definition 6.3.2. *A statistical switching window x_i is said to be a Gaussian switching window if and only if both x_i^l and x_i^u have a Gaussian distribution.*

Corollary 6.3.0.4. *An arbitrary Gaussian switching window x_i cannot have an inclusion relation to another Gaussian switching window x_j, unless the variances of x_i^l, x_j^l, x_i^u and x_j^u are identical.*

Proof. The result is immediate from Lemma 6.3.3 and Corollary 6.3.0.3. □

Consequently, Gaussian switching windows for statistical timing analysis with coupling may not guarantee convergence. The Gaussian distribution spans the entire range of real numbers and the point of intersection of the CDFs of two Gaussian may lie far away from their mean values (or the region of interest). In addition, realistic parametric variations have a distribution constrained in a finite region. We therefore consider a truncated Gaussian distribution for the bounding RVs in our approach. The CDF of a truncated Gaussian with mean μ and variance σ^2 as a function of c is assumed to be 0 for all $c \leq \mu - k\sigma$ and is assumed to be 1 for all $c \geq \mu + k\sigma$, for a given k (typically $k \in [3, 7]$). Truncated Gaussian distributions have earlier been assumed for parametric distributions in [**82, 70**].

Lemma 6.3.4. *An arbitrary truncated Gaussian with mean μ_1 and variance σ_1^2 can strictly dominate another truncated Gaussian with mean μ_2 and variance σ_2^2 under the following condition.*

$$\begin{array}{rcl} \left(\ max(\mu_1 + k\sigma_1, \mu_2 + k\sigma_2) & \leq & \dfrac{\sigma_2\mu_1 - \sigma_1\mu_2}{\sigma_2 - \sigma_1} \ \right) \ \bigvee \\ \left(\ min(\mu_1 - k\sigma_1, \mu_2 - k\sigma_2) & \geq & \dfrac{\sigma_2\mu_1 - \sigma_1\mu_2}{\sigma_2 - \sigma_1} \ \right) \end{array}$$

The above condition implies that a truncated Gaussian can dominate another if the point of intersection of their CDFs lies outside the range $[\mu - k\sigma, \mu + k\sigma]$ for both the truncated Gaussians. Although we consider delay random variables to be Gaussians, we assume truncated Gaussian distributions for the bounding RVs x_i^l and x_i^u of a statistical switching window x_i to maintain the monotonic property of T for convergence. We denote such a statistical switching window as a truncated Gaussian switching window. A truncated Gaussian switching window x_i can have an inclusion relation with another truncated Gaussian switching window x_j if both pairs of truncated Gaussians x_i^l, x_j^l and x_i^u, x_j^u satisfy the condition mentioned in Lemma 6.3.4 individually. The timing transformation T must ensure this condition for monotonicity and convergence of our approach. It is easy to ensure the condition by adjusting the means (or variances) of the statistical switching window bounds pessimistically after each application of T during iterative timing analysis. This is done by rewriting the condition in Lemma 6.3.4 as inequality conditions on μ_2 (or σ_2). We do not limit our approach to a given delay model. Given a gate and the statistical switching windows on all its fanins and wires coupled to its fanout net, we assume that the delay model provides the switching window at the fanout of the gate. We merely adjust the bounding RVs pessimistically so that the above conditions are satisfied, that is, the upper bounding RV strictly dominates the lower in the interval $[\mu - k\sigma, \mu + k\sigma]$.

6.3.2. Coupling induced delay pushout as a Gaussian

We now consider the implications of the assumption of a unit normal distribution for the independent random variables $\xi_1, \ldots, \xi_n$, and Gaussian switching windows in evaluating the delay pushout D from its definition in (6.8). Using the *add (sub)* and *max (min)* operations

from [**71, 48**], the overlap O_{ab} between two Gaussian switching windows x_a and x_b is approximated to a Gaussian with mean μ and variance σ^2. From the definition of the probability of an overlap between statistical switching windows in (6.5), we have the following [1].

$$Pr(O_{ab} > 0) \;=\; 1 - \Phi(\frac{0-\mu}{\sigma}) \;=\; \Phi(\frac{\mu}{\sigma}) \tag{6.21}$$

Consequently, the probability of no overlap between the Gaussian switching windows is given by the following.

$$Pr(O_{ab} \leq 0) \;=\; \Phi(-\frac{\mu}{\sigma}) \tag{6.22}$$

Evaluating the coefficients $d_1, \ldots, d_n$ of the delay pushout D in (6.11) is non trivial. The conditional expectations and consequently the coefficients d_ks are functions of $\xi_1, \ldots \xi_n$, and not constants. D is therefore not a linear weighted sum of the ξ_is in reality.

For simplicity, we attain to approximate D as a linear weighted sum of the ξ_is. One approach to achieving this is by approximating each random variable d_k to its mean value. We denote the corresponding approximated coefficient as d_k^*. This approach is employed by Wu *et al.* in dynamic range estimations of multiplexing operations in non-linear systems using Polynomial Chaos Expansion [**83**]. Monte Carlo simulations are used to generate samples of the random variables and are classified as to whether they produce the condition in the conditional expectation or not. The probabilities of the two outcomes are thus computed, and within each value, the mean of the expectation is computed in the usual Monte Carlo fashion. However, Monte Carlo simulations are expensive and not suitable for statistical timing analysis.

[1] It is known that $\Phi(-x) = 1 - \Phi(x)$.

From the theory of conditional expectation [**84**], we know the following for any random variables X and Y.

$$E[E[X|\ Y]] = E[X]$$

Consequently, the mean or the expectation of the conditional expectations in (6.11), (6.12), and (6.13) are expressed as the following.

$$E\big[E[\xi_k|\ O > 0]\big] = \quad E\big[E[\xi_k|\ O \leq 0]\big] \quad = E[\xi_k] \tag{6.23}$$

$$E\big[E[\xi_i\xi_k|\ O > 0]\big] = \ E\big[E[\xi_i\xi_k|\ O \leq 0]\big] \ = E[\xi_i\xi_k] \tag{6.24}$$

For $i \neq j$, it is known that $E[\xi_i] = E[\xi_i\xi_j] = 0$ and $E[\xi_i^2] = 1$. The simplified coefficient d_0^* in the representation of the effective delay pushout D from (6.11) is now expressed as the following.

$$\begin{aligned} d_0^* &= E[d_0] \\ &= d_0^O Pr(O > 0) \ + \ d_0^N Pr(O \leq 0) \ + \ E[\textstyle\sum_{i=1}^n E[\mathbf{d_i}\xi_i]] \\ &= d_0^O Pr(O > 0) \ + \ d_0^N Pr(O \leq 0) \ + \ \textstyle\sum_{i=1}^n E[E[\mathbf{d_i}\xi_i]] \end{aligned} \tag{6.25}$$

From (6.11), (6.23) and given that $E[\xi_i] = 0$, we have

$$\begin{aligned} E[E[\mathbf{d_i}\xi_i]] &= 0, \quad \text{and thus} \\ d_0^* &= d_0^O Pr(O > 0) \ + \ d_0^N Pr(O \leq 0) \\ &= d_0^O \Phi(\frac{\mu}{\sigma}) + d_0^N \Phi(-\frac{\mu}{\sigma}). \end{aligned} \tag{6.26}$$

Similarly, from (6.11)−(6.13), (6.23), (6.24), and given that each ξ_k is a unit normal Gaussian, each of the coefficients $d_1^*, \ldots, d_n^*$ in the representation of the effective delay pushout D are consequently simplified to the following.

$$d_k^* \;=\; E[d_k] \;=\; d_k^Q \Phi(\frac{\mu}{\sigma}) + d_k^N \Phi(-\frac{\mu}{\sigma}) \qquad k = 1, \ldots, n \tag{6.27}$$

An alternate approach to reducing D to a weighted linear sum of random variables is by introducing new random variables that approximate the product of the ξ_is and true d_ks (as a function of the ξ_is). Fang *et al.* employ this approach for approximating the product of two *affine* terms into another affine term in static analysis for fixed-point finite-precision effects in DSP designs [**85**].

6.4. Statistical timing with multiple input switching

Delay models considering effects due to MIS compute gate delays as functions of the overlap between the switching windows on the fanin signals of a gate. We consider a simple MIS gate delay model similar to (6.7), where *overlap* denotes the overlap between the statistical switching windows on the fanins of a gate and is a random variable. Given this delay model, we can compute the effects due to MIS in statistical switching windows. Though analogous to the coupling, MIS does not necessitate iterative timing analysis, and statistical timing with MIS is possible in a single pass of a topologically ordered circuit if coupling induced delay variations are not considered.

We therefore conclude that our framework for statistical timing with coupling is amenable to incorporating the effects of MIS. Our reason for this conclusion is based on the assumption that the MIS effect does not change causality and consequently the timing transformation T remains monotonic under its effects. This implies convergence to a fixpoint in our approach.

However, for complex delay models, establishing the delay pushout due to MIS would be non-trivial. We do not consider the computation of these pushouts in this work.

6.5. Experimental results

In this section, we present statistical timing analysis results for the ISCAS'85 benchmarks [**36**], realistic parameters for which are generated from a 0.13μ technology library. Two global sources of variation having a truncated Gaussian distribution (at $\mu \pm 4\sigma$) are considered in addition to a local independent component. Cumulative parametric variations are constrained within 10% of their mean values. We use a simple delay model [**82**] and a delay pushout model (6.8) for our experiments. Arrival time at all primary inputs are set to zero. For comparisons, we develop the following timing analysis approaches.

(1) *SSTA :* denotes an implementation of a statistical timer based on [**48**], which does not consider coupling induced delay variation, that is, the delay pushout due to coupling on each wire is assume to be D^N (6.7).

(2) *STAC :* denotes an implementation of a statistical timer based on the above approach that considers coupling induced delay pushouts assuming a deterministic overlap. In this approach, two statistical switching windows x_i ($[x_i^l, x_i^u]$) and x_j ($[x_j^l, x_j^u]$) are treated as deterministic switching windows given by $[\mu_{x_i^l} - 3\sigma_{x_i^l}, \mu_{x_i^u} + 3\sigma_{x_i^u}]$ and $[\mu_{x_j^l} - 3\sigma_{x_j^l}, \mu_{x_j^u} + 3\sigma_{x_j^u}]$, respectively, to estimate the deterministic overlap between them. This overlap determines the delay pushout. This approach is similar to [**47**].

(3) *STAC* :* denotes an implementation of a statistical timer based on our proposed approach. It considers a probabilistic approach to the overlap between two statistical switching windows, and the delay pushout due to coupling is computed as proposed in Section 6.2.

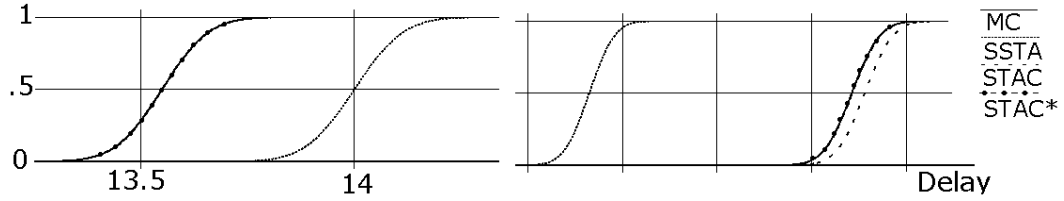

Figure 6.2. Bounding RV distributions for the statistical switching windows obtained at the output of benchmark C432

(4) *Monte Carlo (MC) :* simulations are performed for accuracy comparisons. A static timer is implemented that considers coupling induced delay pushouts and is run for 20000 samples in the parameter space to capture the statistical switching windows on all wires of a given circuit.

We compute the statistical switching window x at the primary output of a given circuit using each of the mentioned approaches. Figure 6.2 shows the the CDFs of the bounding RVs for the switching windows obtained at the output of the ISCAS'85 benchmark C432. It is seen that *SSTA* underestimates the switching window as compared to *MC*. *STAC* is found to give a conservative estimate, while our proposed approach *STAC** accurately estimates the statistical switching window. To give a numerical report, the $(\mu_{x^l} - 3\sigma_{x^l})$ and the $(\mu_{x^u} + 3\sigma_{x^u})$ points obtained from the mentioned approaches are compared. We use the points obtained using *MC* simulations as our reference for accuracy estimations, and formally define the error in any approach A as follows.

$$\%\ \Xi_l^A \triangleq \frac{(\mu_{x_A^l} - 3\sigma_{x_A^l}) - (\mu_{x_{MC}^l} - 3\sigma_{x_{MC}^l})}{(\mu_{x_{MC}^l} - 3\sigma_{x_{MC}^l})} \times 100 \tag{6.28}$$

$$\%\ \Xi_u^A \triangleq \frac{(\mu_{x_A^u} + 3\sigma_{x_A^u}) - (\mu_{x_{MC}^u} + 3\sigma_{x_{MC}^u})}{(\mu_{x_{MC}^u} + 3\sigma_{x_{MC}^u})} \times 100 \tag{6.29}$$

$$\%\ \Xi_{Av}^A \triangleq \frac{|\%\Xi_l^A| + |\%\Xi_u^A|}{2} \tag{6.30}$$

Table 6.1. % Errors in switching window estimation

Circuit	% Ξ_l			% Ξ_u			% Ξ_{Av}		
	SSTA	STAC	STAC*	SSTA	STAC	STAC*	SSTA	STAC	STAC*
C432	3.30	0.00	0.00	-9.2	0.4	-0.01	6.2	0.2	0.01
C499	0.00	0.00	0.00	-1.3	4.7	0.13	0.7	2.4	0.07
C880	0.01	-3.90	-0.01	-13.0	13.0	0.01	6.3	8.3	0.01
C1355	0.01	0.01	0.01	-21.0	2.5	0.01	10.0	1.2	0.01
C1908	6.00	-5.10	-0.01	-23.0	1.6	0.70	14.0	3.4	0.35
C2670	0.00	0.00	0.00	-24.0	7.9	0.01	12.0	4.0	0.01
C3540	9.60	-2.30	-0.01	-17.0	8.9	0.10	13.0	5.6	0.06
C6288	0.01	0.01	0.01	-22.0	0.0	0.00	11.0	0.0	0.01
C7552	0.00	0.00	0.00	-7.0	0.6	0.02	3.5	0.3	0.01

Table 6.2. Run time comparison for the timers

Circuit	Nodes	Edges	# I/N		Run time (s)			
			STAC	STAC*	SSTA	STAC	STAC*	Monte Carlo
C432	198	379	9.1	9.1	0.00	0.01	0.01	21.4
C499	245	481	6.6	5.7	0.00	0.01	0.01	36.8
C880	445	815	10.8	8.8	0.00	0.03	0.03	184.1
C1355	589	1137	14.8	14.4	0.00	0.10	0.15	585.7
C1908	915	1556	14.7	13.9	0.00	0.20	0.26	1018.0
C2670	1428	2449	10.3	9.2	0.01	0.25	0.34	1269.0
C3540	1721	3011	14.9	12.3	0.02	0.46	0.58	2053.0
C6288	2450	4864	50.3	50.9	0.02	2.35	2.90	11960.4
C7552	3721	6459	12.4	11.4	0.03	0.87	1.11	4447.0

Timing analysis accuracy results based on the above metrics are presented in Table 6.1. It is observed that STAC* improves timing estimation accuracy by as much as 24% in comparison to *SSTA* and up to 13% in comparison to *STAC*. On the average, errors in switching window estimation are reduced by 8.5% in comparison to *SSTA* and by 2.8% in comparison to *STAC*.

We present run time for all approaches in Table 6.2. *STAC** is found to be slower by $1.2X$ in comparison to *STAC* on the average. *STAC* and *STAC** are iterative approaches and compute timing information on some wires multiple times till convergence. We report the average number of Iterations per Node ($\#I/N$) obtained in these approaches. Since SSTA does not consider coupling induced delay pushouts, it is non iterative and is a single pass

timer. Consequently, its $\#I/N$ value is always 1. The average number of iterations per node is found to be 16 and 15 for *STAC* and *STAC**, respectively. All experiments are performed on a Pentium 2.4GHz machine, with 1GB RAM, running RedHat Linux 9.0.

6.6. Conclusions

In this chapter, we establish a theoretical framework for statistical timing analysis with coupling. The framework is not constrained to Gaussian distributions for the parameters, but an arbitrary distribution could have a complicated *max* operation and estimation of the conditional expectations would be complicated.

We do not discuss approaches to reducing iterations (or speeding up approaches) for our timer. Clustering the circuit into strongly connected components and then topologically performing timing analysis with coupling is suggested in [**30**]. Breaking up feedback edges in the directed graph representation of the circuit to form clusters is suggested. The feedback edges due to coupling are broken based on a metric of timing proximity. In statistical timing with coupling, we propose to use the probability of an overlap between two windows as the metric for timing proximity, that is, edges having a low probability of overlap are broken. Though we do not discuss details on the approaches to speed up iterations, our formulation of statistical timing analysis with coupling is amenable to each of the mentioned speedup approaches. Since our statistical timing analysis framework is block based, we are amenable to using the concept of node and edge criticalities introduced in [**48**]. This makes our approach attractive in guiding timing optimization [**15**].

CHAPTER 7

Timing Dependent Dynamic Power Estimation considering Coupling

Accurate power estimation is a critical problem in modern integrated circuit (IC) design. It is expected that power dissipation would be a limiting factor for future technologies. Currently, more than 60% of a circuit's power consumption is attributed to charging (or discharging) interconnect capacitances [**86, 87, 88, 89, 90**]. This is due to relatively decreasing gate load capacitances in comparison to increasing parasitic interconnect capacitances. Furthermore, the increased dominance of coupling capacitances with technology scaling makes it evident that the component of power dissipation in parasitic coupling capacitances is significant.

Power consumption estimation for coupling capacitances is more complicated than for ground capacitances. In the latter case, parasitic ground capacitances on the fanout net of a gate are added in parallel to the gate's load capacitance. The ground capacitances are charged and discharged depending on the net's signal transitions in exactly the same way as the load capacitance. They simply introduce additional power consumption components similar to that for the load capacitance. The power dissipation per clock cycle is thus dependent only on the net's switching activity [**91**]. However, the power consumption for a parasitic coupling capacitance (termed *coupling power*) between two interconnects is dependent on the voltage difference across that capacitance. This in turn, is dependent on the

relative switching activities on these interconnects. The voltage across a coupling capacitance can vary in the range of [-V_{dd}, V_{dd}], while the range of voltage variation for a parasitic ground capacitance is [0, V_{dd}]. The worst case voltage variation across a coupling capacitor is therefore $2V_{dd}$ in contrast to V_{dd} for a parasitic ground capacitor. When two coupled wires **a** and **b** with coupling capacitance C_c , simultaneously switch in the same direction, there is no charging or discharging of C_c, and no coupling power is consumed ($P_1 = 0$, Figure 7.1). When only one of the wires switch, C_c charges or discharges with a voltage variation of V_{dd}, and its coupling power consumption is given by $P_2 = 0.5C_cV_{dd}^2$. In the case when the wires simultaneously switch in the opposite direction, C_c may charge or discharge with a voltage variation of $2V_{dd}$, consuming $P_3 = 0.5C_c(2V_{dd})^2 = 2C_cV_{dd}^2$ units of power.

In addition to the dependence of coupling power on the relative switching activities of the coupled interconnects, the power consumed is dependent on the nets' relative switching times [**91**]. The coupling power P_4 (in Figure 7.1) is dependent on some function $\psi(d)$ of the difference d in their switching times. As d increases, the case of simultaneous switchings on the interconnects change to two independent cases of a single interconnect switching while the other is quiet. In this case, the coupling power can vary from 0 to $C_cV_{dd}^2$ depending on the relative delay between their switchings. Relative delays, timing information, and switching activities are therefore, critical to accurate coupling power estimation. Furthermore, the dependence on relative switching activities translate to dependence on the functional information of the circuit. For example, the outputs of an *AND* gate and an *OR* gate have different switching probabilities, even for identical input switching probabilities. This implies that coupling power is dependent on a circuit's functionality (logic implementation).

It is commonly accepted that circuit simulation based approaches to power estimation [**92**] are strongly input pattern dependent and too slow for large circuits. Probabilistic

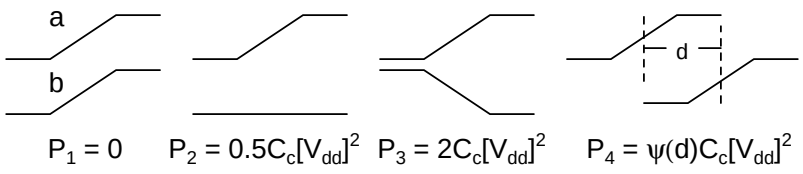

Figure 7.1. Effect of timing on coupling power

approaches to estimate power were first proposed in [**93**]. A zero-delay model was considered which ignored the impact of glitches. Probabilistic simulation approaches [**94, 95, 96**] that did not assume a zero-delay model were later proposed to handle the impact of glitches. To consider temporal correlations, such approaches require the user to specify typical signal behavior at the circuit inputs, often using a sequence of values indicating the probability of switchings at specific time points. The propagation of these signals is similar to event driven logic simulation. An event driven energy-consumption estimation model is proposed in [**97**]. Power estimation approaches that employ BDDs [**98**] are too slow for modern complex circuits. Statistical methods for power estimation [**99**] simulate the circuit repeatedly using some simulator (e.g. HSPICE, Powermill) while monitoring the power consumption. Prior work in power estimation does not couple the problem of power estimation to detailed timing analysis. The effects of relative switching delays are ignored. In most of the power estimation tools, either the coupling power is ignored or is accounted for as a component from coupling capacitances that are assumed to be grounded. In addition, it is accepted that considering all spatial correlations are computationally very expensive.

In this chapter, we propose a timing dependent dynamic power[1] estimation framework for combinational circuits that considers the impact of coupling and glitches. Our contributions are summarized as follows.

[1]In this chapter, we denote the switching power consumption as the dynamic power, and do not explicitly consider short-circuit power. Note that this is not a limitation; traditional approaches to short-circuit power estimation can still be employed.

- We signify the timing dependence of coupling power and show that dynamic power estimation should not be de-coupled from detailed timing analysis. We present a timing dependent framework for dynamic power estimation that considers the impact of coupling and glitches.
- A representation to approximate the switching probability distribution on any wire (denoted as switching-window distribution) is developed. In addition, we present an approach to efficiently propagating these switching-window distributions through a given circuit without incurring exponential run-time.
- We propose an efficient single-pass approach to propagating switching-window distributions while considering the impact of coupling induced delay variations.
- We present an approach to coupling and total power estimation based on the obtained switching-window distributions, and additionally describe how the proposed framework is amenable to incorporating the impact of glitches on power.

We develop the proposed framework and compare results obtained when (i) coupling power is ignored; (ii) coupling capacitances are assumed to be grounded during power estimation; and (iii) the timing dependence of coupling power is considered (our approach). Experimental results for the ISCAS'85 benchmarks demonstrate accuracy improvements of up to 59% in coupling power estimation (up to 25% in total power estimation). Since ignorance of the timing dependence of coupling power is found to result in both underestimation or overestimation of power, using a *guard-band* during power estimation while ignoring timing is meaningless. We therefore conclude that it is critical to consider the timing dependence of coupling power in total power estimation.

The rest of this chapter is organized as follows. We present a motivational example in Section 7.1 to illustrate the significance of the timing dependence of power consumption

in coupling capacitances. The proposed approach to timing dependent power-consumption estimation is described in Section 7.2. Experimental results are reported in Section 7.3, and we draw conclusions in Section 7.4.

7.1. A motivational example

We perform HSPICE simulations for two coupled nets with typical local interconnect dimensions [**3**], and having driving and loading gates implemented in $90nm$ technology. The energy consumption per switching in the circuit is evaluated for the cases of single net switching (SNS), simultaneous similar switching (SSS), simultaneous opposite switching (SOS), similar switching with large relative delay (SSWD), and opposite switching with large relative delay (OSWD). Results obtained are presented in Table 7.1. For each of the above cases, we present the energy consumption when (i) the coupling capacitance is ignored (No C_c); (ii) the coupling capacitance is considered as ground capacitances on the nets (Grounded C_c); and (iii) the coupling capacitance is considered between the coupled nets (Exact C_c).

It is observed that ignoring coupling leads to large underestimation of power in most cases (up to 53% in case of simultaneous opposite switching). In addition, considering coupling as capacitance to ground also leads to large errors; ranging from underestimating power by 26% (for the case of simultaneous opposite switching) to overestimating power by 56% (in case of simultaneous similar switching). Furthermore, we observe that relative delays between switchings cause significant differences in power consumption. It is thus evident that coupling power should be estimated accurately considering both timing and relative switching activities.

Table 7.1. Simulated energy consumption (nJ) per switching

Switching case	No C_c	Grounded C_c	Exact C_c
SNS	1.442	2.249	2.249
SSS	2.884	4.498	2.884
SOS	2.888	4.498	6.090
SSWD	2.884	4.498	4.498
OSWD	2.888	4.498	4.498

7.2. Timing dependent power estimation

In this section, we describe our approach to dynamic power estimation. It is established that coupling power is dependent on the relative switching times of coupled interconnects. However, accounting for all possible switchings in large circuits is impractical and necessitates assumptions of some switching probability distribution on each interconnect. We first describe our approach to efficiently approximating the distribution of a switching on any wire, which we denote as a switching-window distribution. We next present an approach to window propagation while considering coupling induced delay variations. Finally, we describe our power estimation technique based on the obtained switching-window distributions.

7.2.1. Switching-window distribution representation

Considering complete functional information during timing analysis of a given circuit involves enumeration of all possible input vectors. This approach is exponential in complexity, and therefore, computationally prohibitive. On the other hand, ignorance of functional information introduces uncertainty that makes timing information on each wire of the circuit a set of possible signal switchings. Each set is represented as a switching-window and a set of slew rates such that any signal switching falling in the window and having a slew in the range is in the set. Symbolically, we denote the switching-windows for the *rise* and *fall* transitions on any wire $\mathbf{x}$ as x^r and x^f, respectively.

Each switching-window x (x^r or x^f) on a wire is often defined as an interval $x \triangleq [l_x, h_x]$, such that the time t of any possible signal switching on the wire lies in this interval. An assumption of a uniform probability density for a switching in this interval is unrealistic. We formally denote this probability density function (PDF) of a switching on a given window x as $\phi_x(t)$. The following is immediate.

$$\phi_x(t) \;=\; 0, \qquad \forall t < l_x \;\vee\; \forall t > h_x \tag{7.1}$$

Efficient representation and propagation of an arbitrary window distribution $\phi_x(t)$ is challenging and can be computationally very expensive. We propose to represent a given switching-window x using a set of M sub-switching-windows x_i ($i = 1, 2, \ldots, M$), each having a constant probability density ϕ_{x_i} in their respective intervals $[l_{x_i}, h_{x_i}]$. Such an approach captures the PDF of x more accurately than the assumption of a uniform density in the interval $[l_x, h_x]$ of x.

We now present our approach to representing a switching-window x with a given PDF $\phi_x(t)$ as a set of M uniform density sub-switching-windows. For simplicity, we initially segment the interval $[l_x, h_x]$ into M equal length segments. The interval of a sub-switching-window $x_i \triangleq [l_{x_i}, h_{x_i}]$ is therefore given by the following.

$$l_{x_i} \;=\; l_x + \frac{(i-1)(h_x - l_x)}{M} \tag{7.2}$$

$$h_{x_i} \;=\; l_x + \frac{i(h_x - l_x)}{M} \tag{7.3}$$

To evaluate the constant probability density ϕ_{x_i} for any sub-switching-window x_i, we match the probability of switchings in x_i's interval to that of x in the same interval. Formally,

ϕ_{x_i} is computed as the following.

$$\phi_{x_i} = \frac{1}{(h_{x_i} - l_{x_i})} \int_{l_{x_i}}^{h_{x_i}} \phi_x(t)dt \tag{7.4}$$

Lemma 7.2.1.

$$\sum_{i=1}^{M} (h_{x_i} - l_{x_i})\phi_{x_i} = \int_{l_x}^{h_x} \phi_x(t)dt$$

It is immediate that our approach preserves the probability of switching in a window x. This procedure is employed separately to represent the rise and fall switching-windows on a given wire, each into M sub-switching-windows. As a result, the probability that the signal on a wire switches from low-to-high (rise transition) and from high-to-low (fall transition) in a single clock cycle is given by the following.

$$Pr_x[rise] = \sum_{i=1}^{M} (h_{x_i^r} - l_{x_i^r})\phi_{x_i^r} \tag{7.5}$$

$$Pr_x[fall] = \sum_{i=1}^{M} (h_{x_i^f} - l_{x_i^f})\phi_{x_i^f} \tag{7.6}$$

7.2.2. Switching-window distribution propagation

We next describe our approach to propagating the set of sub-switching-windows through logic blocks in a circuit. We discuss the window propagation for a two-input *AND* gate for illustration. The extension to other common logic gates, including those with greater than two inputs is similar.

We consider a logic *AND* gate with inputs **a** and **b**, and output **c** as shown in Figure 7.2. The switching-window on input **a** for the rise (fall) transition is denoted by M uniform density sub-switching-windows a_i^r (a_i^f), $(i = 1, 2, \ldots, M)$. The interval of any sub-switching-window

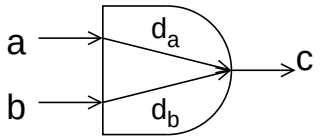

Figure 7.2. A two-input *AND* gate

a_i^r is denoted by $[l_{a_i^r}, h_{a_i^r}]$, and the density within this interval is denoted by a constant $\phi_{a_i^r}$. The probabilities of the possible transitions in input **a**, namely low (steady at logic 0), rise, fall, and high (steady at logic 1) for a clock cycle are denoted by $Pr_a[l]$, $Pr_a[r]$, $Pr_a[f]$, and $Pr_a[h]$, respectively. We denote the delay of the timing arc from input **a** to output **c** for a rise transition as d_{a^r}. A similar representation is used for the input **b**. Given these information, our goal is to approximate the rise and fall switching-windows at **c** each with M sub-switching-windows, for further propagation. Note that for $M = 1$, our approach falls back to the traditional single switching-window approach with $Pr_a[r]$ as the traditional switching activity on **a**.

If we ignore logical correlation between the inputs **a** and **b**, the probability that **c** has a fall transition is given by:

$$Pr_c[f] \;=\; Pr_a[h]Pr_b[f] + Pr_a[f]Pr_b[f] + Pr_a[f]Pr_b[h].$$

The PDF $\phi_{c^f}(t)$ of the fall transition switching-window c_f at any time t is thus given by the following.

$$\begin{aligned}\phi_{c^f}(t) = \\ & Pr_a[h]\phi_{b^f}(t - d_b^f) + \; \phi_{a^f}(t - d_a^f) \int_{-\infty}^{t} \phi_{b^f}(x - d_b^f)dx + \\ & \phi_{b^f}(t - d_b^f) \int_{-\infty}^{t} \phi_{a^f}(x - d_a^f)dx + \; Pr_b[h]\phi_{a^f}(t - d_a^f)\end{aligned} \tag{7.7}$$

Given that the input switching-windows are represented as a set of sub-switching-windows, $\phi_{c^f}(t)$ is expressed as:

$$\begin{aligned}
\phi_{c^f}(t) &= Pr_a[h] \sum_{i=1}^{M} \phi_{\mathbf{b_i^f}}(t - d_b^f) \\
&+ \sum_{i=1}^{M} \sum_{j=1}^{M} \phi_{\mathbf{a_i^f}}(t - d_a^f) \int_{-\infty}^{t} \phi_{\mathbf{b_i^f}}(x - d_b^f) dx \\
&+ \sum_{i=1}^{M} \sum_{j=1}^{M} \phi_{\mathbf{b_i^f}}(t - d_b^f) \int_{-\infty}^{t} \phi_{\mathbf{a_i^f}}(x - d_a^f) dx \\
&+ Pr_b[h] \sum_{i=1}^{M} \phi_{\mathbf{a_i^f}}(t - d_a^f)
\end{aligned} \tag{7.8}$$

where, (expressions for $\phi_{\mathbf{a_i^r}}(t)$, $\phi_{\mathbf{a_i^f}}(t)$, $\phi_{\mathbf{b_i^r}}(t)$ are similar)

$$\phi_{\mathbf{b_i^f}}(t) \triangleq \begin{cases} 0 & \forall t < l_{b_i^f} \\ \phi_{b_i^f} & \forall t \in [l_{b_i^f}, h_{b_i^f}] \\ 0 & \forall t > h_{b_i^f} \end{cases} \tag{7.9}$$

Each inner integral in (7.8) denotes the area of that sub-switching-window lying on the left of t, that is, the probability of the sub-switching-window having a transition before time t. The earliest possible switching time for a fall transition at $\mathbf{c}$ is given by the minimum of $(l_a^f + d_a^f)$ and $(l_b^f + d_b^f)$. Similarly, the latest possible switching time for a fall transition at $\mathbf{c}$ is given by the maximum of $(h_a^f + d_a^f)$ and $(h_b^f + d_b^f)$. We partition the interval formed with these earliest and latest switching times into M equal segments. The probability of a fall transition within any segment is given by the area under $\phi_{c^f}(t)$ in that segment. The uniform density of this segment $\phi_{c_i^f}$ is computed by dividing the obtained probability by the segment width (similar to (7.4)). In practice, this computation is done faster by evaluating the probability of

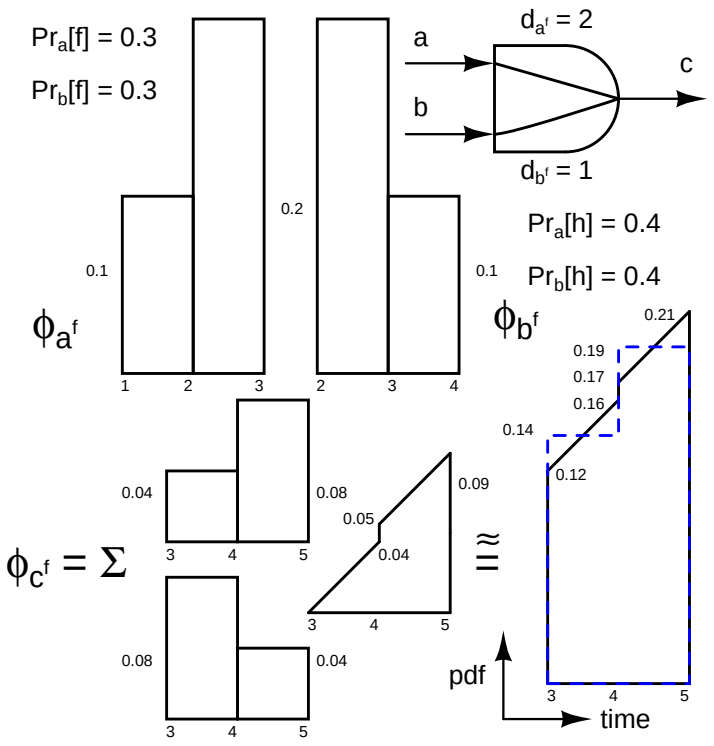

Figure 7.3. Fall switching-window propagation example

switching in any segment as the difference of the cumulative distribution function $\Phi_{c^f}(t)$ of a fall transition on **c**, evaluated at the segment's upper and lower bounds, respectively. $\Phi_{c^f}(t)$ can be evaluated analytically without numerical integration. This procedure is employed to compute $\phi_{c_i^f}$ for $i = 1, 2, \ldots, M$.

Figure 7.3 illustrates the proposed approach with an example. Here, $M = 2$, and switching probabilities and times are as shown in the figure. $\phi_{c_i^f}$ is obtained as a sum of the three PDFs, the third corresponding to the sum of the two middle terms in (7.8), while the other two correspond to the first and last term in (7.8). The PDF obtained using our proposed approximation approach is shown using a dashed line in the figure. It is observed that the probability of a switching in a sub-interval is preserved. Increasing the number of the sub-intervals (M) improves the accuracy of our approach.

The procedure to compute the sub-switching-windows for a rise transition is similar. The proposed approach is extensible to other logic blocks, and is not limited to two-input gates. In the simplest case, a multi-input gate can be expressed as a logic function of multiple two-input gates, and the proposed approach is directly applicable to each of them. The sub-switching-window computation for a *NAND* gate is equivalent to the computation for an *AND* gate, except that, in this case, the computed rise and fall windows should be swapped. Non-inverting buffers and wires time-shift the sub-switching-windows by their delay values, and do not affect their probability densities.

7.2.3. Considering coupling induced delay variations

Simultaneous switchings on coupled wires induce delay variations (often termed as delay pushouts) in each of them. The relative switching times and direction (rise or fall) of the switchings impact these delay variations. For timing estimation, a coupling induced delay pushout model is considered that evaluates the possible delay variation as a function of the overlap between the switching-windows on the coupled wires. A coupling between two wires lying in the fanin and fanout cone of a logic gate, respectively, causes timing dependencies that necessitate iterations during timing analysis. Starting with some assumption of the delay pushout, multiple iterations of timing analysis are performed. Zhou [**30**] formally proved that iterations starting with an assumption of zero delay pushouts on all wires would converge and yield timing analysis results with minimal pessimism. Simplistic delay pushout models only consider the worse case overlap length between the switching-windows on coupled nets. Such models result in a monotonically increasing sequence [**30, 100**] of timing information on all wires, and thus, guarantee convergence. However, ignorance of correlations between

subsequent iterations when using accurate delay pushout models that consider the probabilities of switchings may not guarantee convergence. On the other hand, considering all possible correlations is computationally too expensive. Consequently, a majority of timing analysis tools that consider coupling start iterations with an initial assumption of worst case delay pushout for all coupled nets. A finite number of iterations are performed to reduce pessimism using accurate pushout models, but not till convergence is achieved.

In this work, we adopt the former approach for simplicity. Timing analysis of a given circuit is initially performed with two switching-windows on each wire that denote the rise and fall switching-window, respectively. Using an overlap based delay pushout model [**77**], the converged switching-windows for all wires in the circuit are obtained. We term these switching-windows as the *simple switching-windows*.

Next, we propagate our sub-switching-windows through the circuit as explained earlier. To evaluate the delay variation for any sub-switching-window on a coupled wire, we consider a delay pushout model that yields the variation as a function of that sub-switching-window's PDF and the intervals of the *simple switching-windows* on its coupled neighbors. Each of the sub-switching-windows may now broaden, and may have regions of overlap with other sub-switching-windows. The impact of coupling does not affect the total probability of switching in any wire's window; it may only affect the PDF in some intervals. We therefore scale the uniform density in any sub-switching-window such that the original probability in its interval is retained. We do not impose a restriction that all sub-switching-windows on a wire for a given transition should have mutually exclusive intervals. Since the *simple switching-windows* are pre-determined and do not change, this approach does not require multiple iterations. Our approach to propagating the switching-window distributions is therefore a single pass procedure and is illustrated diagrammatically in Figure 7.4.

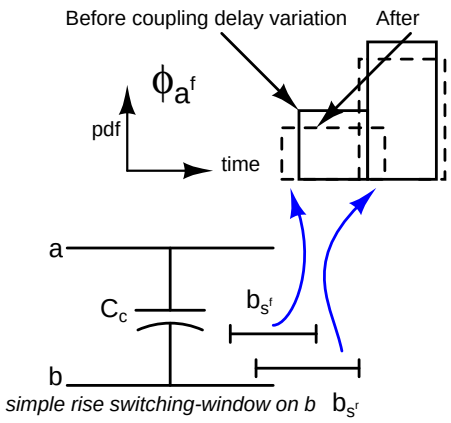

Figure 7.4. Estimation of coupling induced delay pushouts

Alternately, we could start with assumptions of worse case delay pushout values, and perform iterations with the sub-switching-window distributions. We choose the former approach as it is faster, and does not involve a heuristic for picking a suitable time to stop further iterations, when the pushout model does not guarantee convergence.

7.2.4. Power estimation with timing dependent coupling

Once switching-window distributions and transition probabilities have been evaluated on the fanout nets of all gates in a circuit, the dynamic power consumption is computed by a summation of the switching power corresponding to each gate in the circuit. For each gate, the average dynamic power P_d consumed per clock cycle is given by

$$\begin{aligned} P_d &= P_g + P_c \\ &= 0.5V_{dd}^2 C_g(Pr[r] + Pr[f]) + \sum_{i=1}^{k} P_c^i \end{aligned} \tag{7.10}$$

where, P_g denotes the power consumption corresponding to the gate's and its fanout net's total ground capacitance C_g, and $Pr[r]$ ($Pr[f]$) denotes the probability of a rise (fall) transition at the gate's output. P_c denotes the power consumption due to coupling on its fanout net, and is given by the sum of coupling power P_c^i consumed due to each coupled neighbor $i = 1, 2, \ldots, k$ of the gate's fanout net. For a single coupled neighbor i with coupling capacitance C_c,

$$P_c = P_{quiet} + P_{opp} + P_{sim} \tag{7.11}$$

$$P_{quiet} = 0.5V_{dd}^2 C_c(Pr[r] + Pr[f])(Pr_i[l] + Pr_i[h]) \tag{7.12}$$

where, P_{quiet}, P_{opp} and P_{sim} denote the coupling power when the coupled neighbor is not switching, switching in the opposite direction and switching in the same direction, respectively. $(Pr_i[l] + Pr_i[h])$ denotes the probability that net i does not switch. P_{opp} and P_{sim} are timing dependent, that is, they depend on the relative switching times on the two nets. Analytically, we express

$$P_{opp} = 0.5V_{dd}^2 C_c \int_{-\infty}^{\infty} \psi_{opp}(x) p_{opp}(x) dx \tag{7.13}$$

where, $p_{opp}(x)$ denotes the probability density that the nets switch in the opposite direction with a time skew of x, and $\psi_{opp}(x)$ denotes some function that gives an effective power factor as a function of the time skew x. In general, $\psi_{opp}(x)$ depends on the slews of the switching signals and is symmetric in nature. We illustrate a typical linear and an exponential $\psi_{opp}(x)$ model in Figure 7.5. It is intuitive that when $x \approx 0$, the coupled nets switch simultaneously in the opposite direction and C_c charges (or discharges) with $\approx 2V_{dd}$ across itself. Attributing half of the power consumed to each driving-gate of the coupled nets is equivalent to having

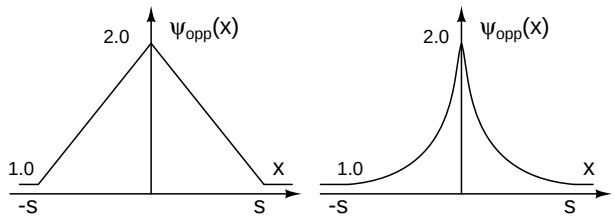

Figure 7.5. A typical linear and exponential model for $\psi_{opp}(x)$

$\psi_{opp}(x) \approx 2$. Similarly, when the nets switch with a large skew ($\geq s$, in Figure 7.5), power consumption attributed to each driving gate is exactly same as the case of a switching with the coupled net quiet, and hence $\psi_{opp}(x) \approx 1$. In the figure, s is dependent on the slews of the signals on the coupled wires. For two coupled wires **a** and **b**, $p_{opp}(x)$ is analytically evaluated as the following.

$$p_{opp}(x) = \int_{-\infty}^{\infty} \phi_{a^r}(t)\phi_{b^f}(t+x)dt + \int_{-\infty}^{\infty} \phi_{b^r}(t)\phi_{a^f}(t+x)dt \tag{7.14}$$

Given switching-window distributions for the rise and fall transitions on **a** and **b** each as a set of M sub-switching-windows, the computation translates to the following.

$$p_{opp}(x) = \sum_{i=1}^{M}\sum_{j=1}^{M} \int_{-\infty}^{\infty} \phi_{\mathbf{a_i^r}}(\mathbf{t})\phi_{\mathbf{b_j^f}}(\mathbf{t}+\mathbf{x})dt + \sum_{i=1}^{M}\sum_{j=1}^{M} \int_{-\infty}^{\infty} \phi_{\mathbf{b_i^r}}(\mathbf{t})\phi_{\mathbf{a_j^f}}(\mathbf{t}+\mathbf{x})dt \tag{7.15}$$

We illustrate the shape of a PDF obtained from one inner integral of (7.15) in Figure 7.6. Given an analytical expression for $\psi_{opp}(x)$, the product of $p_{opp}(x)$ and $\psi_{opp}(x)$ are expressed analytically. P_{opp} can now be computed from analytical expressions obtained from the integration of the above product in appropriate intervals. Numerical integration is not required. A similar approach is employed for the computation of P_{sim}. Figure 7.7 presents a typical linear and an exponential model for $\psi_{sim}(x)$.

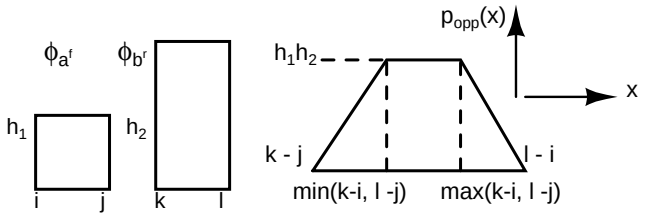

Figure 7.6. Sub-switching-windows switching with skew x PDF

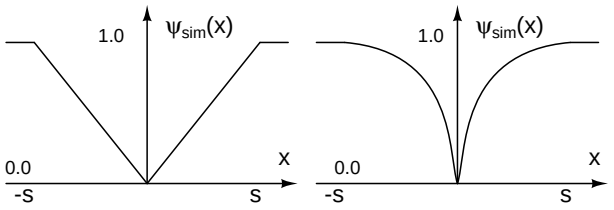

Figure 7.7. A typical linear and exponential model for $\psi_{sim}(x)$

7.2.5. Power estimation with timing dependent glitches

In this section, we describe how the proposed framework is amenable to incorporating power dissipation in glitches, often termed as *toggle power*. We illustrate our approach with an example of a two-input *AND* gate as shown in Figure 7.2. In this case, a glitch may be formed at output **c** only when both input signals switch with some time skew, in the opposite direction. The power consumption due to a glitch is some function $\varphi(x)$ of the difference x in their switching times, given their slews. Note that $\varphi(x)$ is not symmetric. If x denotes the time by which the rising signal switches before the falling signal, we have the following.

$$\varphi(x) \quad \geq \quad \varphi(-x) \tag{7.16}$$

From (7.15), we can compute the probability density of the function $p_{opp}(x)$ that denotes that the inputs switch in the opposite direction with time skew x. The toggle power

consumption per cycle is estimated as the following.

$$P_{toggle} = 0.5V_{dd}^2 C_g \int_{-\infty}^{\infty} \varphi(x) p_{opp}(x) dx \quad (7.17)$$

The computation of $p_{opp}(x)$ should also consider the difference in the timing arcs delays d_a and d_b for the corresponding cases. The proposed approach can be extended to estimate toggle power consumption in other logic gates similarly. Here we assume that glitches do not propagate through multiple logic levels, although theoretically our framework does not restrict that either. However, it may cause an exponential complexity of glitch power estimation if we do not keep an upper bound on the maximum number of logic levels that a glitch could propagate through.

7.3. Experimental results

In this section, we present obtained power consumption results for the ISCAS'85 benchmarks [**36**]. Realistic parameters for the benchmarks are generated from a 90nm technology library. We use a simple delay model [**55**] and a linear $\psi(x)$ model in our experiments. $Pr[l]$, $Pr[r]$, $Pr[f]$, and $Pr[h]$ on all primary inputs are set to 0.25 each.

We denote the total power obtained from our approach (considering timing dependent coupling power) with M sub-switching-windows, for any benchmark b, as P_M^b. Simulation based (or Monte Carlo) approaches to accurate power estimation require a very large number of input vectors (exponential in the number of inputs), and are thus, prohibitive. HSPICE simulations for small test cases show less than 1% error in P_{1000}^b. Although the accuracy of power estimation improves for larger M, the run time increase is not commensurate with the accuracy gain. As an example, for the benchmark *C5315*, the run-time to obtain P_{1000}^{C5315} is $\approx$ 28 hours. We therefore choose P_{1000}^b to be the true power value and use it as reference

for error estimation. The absolute error in the result P_M^b obtained using our approach (with M sub-switching-windows), for a benchmark b, is defined as:

$$\text{Absolute error}(b, M) \triangleq \frac{|P_M^b - P_{1000}^b|}{P_{1000}^b} \times 100.0. \tag{7.18}$$

We plot absolute error values as a function of M for the benchmark *C5315* in Figure 7.8. In the same graph, we also show the run time ratio of our approach to the one where all coupling capacitors are assumed to be grounded. Similar plots are observed for other benchmarks. From these plots, we choose $M = 6$ as the default for all benchmarks. In this case, our estimation error is $\leq 1.6\%$, and average run time increase is by a factor of $\leq 5X$. For comparison, we build the following power estimation engines.

(1) *NC* (No Coupling) denotes an implementation of a power estimation engine that completely ignores the presence of coupling capacitances.

(2) *FC* (Fixed Coupling) denotes an implementation of a power estimation engine that assumes all coupling capacitances are grounded. Consequently, it ignores the timing dependence of coupling power.

(3) *TDC* (Timing Dependent Coupling) denotes our approach to power estimation considering timing dependent coupling power with $M = 6$.

For any approach A (could be NC, FC or TDC), we define the error in coupling power estimation ($\%\Xi_C^A$) and total power estimation ($\%\Xi_T^A$) of a circuit as the following.

$$\%\Xi_C^A \triangleq \frac{P_{coupling}^A - P_{coupling}^{TDC}}{P_{coupling}^{TDC}} \times 100.0 \tag{7.19}$$

$$\%\Xi_T^A \triangleq \frac{P_{total}^A - P_{total}^{TDC}}{P_{total}^{TDC}} \times 100.0 \tag{7.20}$$

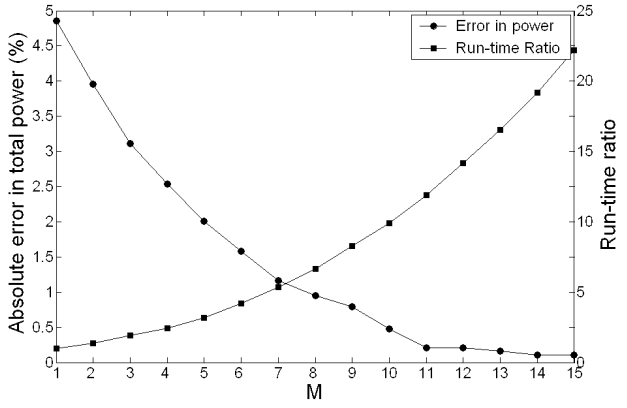

Figure 7.8. % Error and run time ratios with varying M

Table 7.2. % Errors in coupling and total power estimation

Circuit	Nodes	$\%\Xi_C$		$\%\Xi_T$		t^{TDC}/t^{FC}
		NC	FC	NC	FC	
C432	198	-100	59	-42	25	3.5
C499	245	-100	51	-37	19	5.0
C880	445	-100	37	-40	15	4.6
C1355	589	-100	-19	-56	-11	3.6
C1908	915	-100	-21	-58	-12	4.1
C2670	1428	-100	35	-46	16	4.6
C3540	1721	-100	-6	-53	-3	3.8
C5315	2487	-100	28	-43	12	4.2
C6288	2450	-100	5	-49	3	3.0
C7552	3721	-100	-17	-57	-10	4.4
|Average|		100	28	48	13	4.1

We present error values and run-time overheads (t^{TDC}/t^{FC}) for the benchmarks in Table 7.2. Since *NC* ignores coupling power, it is immediate that $\%\Xi_C^{NC} = -100\%$. Furthermore, *NC* underestimates total power on the average by 48%. We observe that considering coupling capacitances as grounded capacitances result in both overestimation and underestimation of coupling power by 59% (for benchmark *C432*) and 21% (for benchmark *C1908*), respectively. These numbers translate to overestimation by 25% and underestimation by 12%, respectively, of the total power consumption. Taking the arithmetic mean of absolute

error values, we observe that coupling and total power values predicted by *FC* are off the true values by 28% and 13%, respectively. For $M = 6$, power estimation using our approach (TDC) for each benchmark takes between 0.03 to 1.95 secs. Experiments are performed on a Pentium 2.4GHz machine, with 1Gb RAM, running RedHat Linux 9.0.

7.4. Conclusions

In this chapter, we present a timing dependent power estimation framework which is amenable to considering the effects of glitches and crosstalk-noise power [**101**] as well. We ignore the effects of incomplete voltage swings since it has been shown that their effects are negligible for designs with typical activity factors [**101**]. The proposed approaches to switching-window distribution representation do not impose any restriction of how the switching-window interval must be segmented; alternative approaches can readily be accommodated. In addition, time slots [**102**] may be used in estimation of the *simple switching-windows* for pessimism reduction during timing analysis.

Since the impact of the timing dependence of coupling power may lead to both underestimation or overestimation of power, using a *guard-band* during power estimation while ignoring timing, is meaningless. This is true even if multiplication factors are employed when considering coupling capacitances as grounded capacitances. The best factor for the same would be 1 under the reasonable assumption that signals on coupled wires switch in similar and opposite directions with equal probability, on the average. Given that our experiments reveal both underestimation and overestimation of power using such an approach (FC), it is evident that we cannot obtain bounds on the power consumption by using other multiplication factors as well, while ignoring timing. We therefore conclude that it is critical to consider the timing dependence of coupling power in total power estimation.

CHAPTER 8

Impact of Modern Process Technologies on the Electrical Parameters of Interconnects

With CMOS feature sizes scaling down to the deep sub-micron regime, there has been a tremendous growth in the number of transistors and the complexity of modern ICs. The integration of numerous active elements within an IC necessitates that it features nine or more layers of high density metal interconnect. The electrical parameters of these interconnects (primarily the resistance and capacitance) critically determine the timing characteristics of the circuit. Modern processing techniques are employed at the back end-of-line (BEOL) manufacturing process to improve manufacturability of these interconnects.

Starting from the 180nm generation, the semiconductor industry has transitioned from using aluminum interconnect metal to copper interconnect metal. This has been motivated by the lower resistivity and higher reliability of copper in comparison to aluminum. In contrast to an aluminum interconnect metal process, the oxide between the metal layers is patterned instead of the metal for a copper interconnect process, as copper is more difficult to etch than aluminum.

Chemical-mechanical polishing or planarization (CMP) is performed to achieve uniformity of conductor and dielectric thickness in the manufacturing process. For copper interconnect, the underlying metal is polished during CMP. This is also known as the *damascene CMP* process. Although CMP provides good local planarization, it is unable to guarantee

global uniformity due to multiple factors including planarization length, underlying non uniform pattern density and feature perimeter sum [**103**] [**104**]. A density range design rule is therefore employed to equilibrate density and limit metal thickness variability due to CMP. This is achieved by insertion of *dummy fill* metal structures in the empty regions of each metal layer [**105**]. Although dummy fill insertion improves the uniformity of metal feature density and enhances the planarization that can be obtained by CMP, it contributes to increased coupling capacitances, and thereby increased total capacitances of the interconnects.

Modern BEOL manufacturing processes additionally employ multiple layers of low-k dielectrics between metal layers, instead of the traditional uniform Silicon Dioxide dielectric. Most low-k porous dielectrics are hydrophobic and fragile in character, and it is critical that the surface hard mask, located on the top of the Inter Level Dielectric (ILD) stack, shield them during CMP. These multiple, thin dielectrics affect the parasitic capacitances between the metal lines. Furthermore, due to the presence of these thin surface hard masks of different dielectric permittivity, the overall dielectric stack has more ILDs, thereby stressing capacitance extractors.

The etching process during manufacturing can also cause significant variations in interconnect patterns, not only along the x-y plane but also on the side-walls. Thereby, the interconnect cross-sections are increasingly trapezoidal in nature [**106**]. With the reduction in spacing between interconnects and the interconnects themselves becoming narrower and longer, the effects of etching and trapezoidal shapes on the electrical parameters can no longer be neglected.

With an increasing significance of variability-driven considerations in the design of interconnects [**107**] for modern circuits, it is important to determine the effect of the above factors on the electrical parameters of interconnects. In this chapter, we employ accurate industrial

parasitic extractors and simulators to quantify variations on the interconnect resistance and capacitance in a set of industrial benchmarks. Technology and process parameters are obtained from two foundries, and experiments are performed on a set of industrial designs.

The rest of this chapter is organized as follows. Section 8.1 presents a brief description of multiple modern process technologies on interconnects. We present the impact of each of these technologies and factors using experimental results in Section 8.2, and derive conclusions in Section 8.3.

8.1. Modern process technologies

8.1.1. Dummy fill insertion

Dummy fills are floating (or grounded) metal objects inserted in each metal layer of the design to satisfy given design density rules for the BEOL manufacturing process and to enhance the planarization obtained by CMP [**108**]. However, they cause additional coupling capacitances as shown in Figure 8.1. In this figure, the interconnects are shown as white rectilinear boxes, while the shaded objects represent the fills. These additional capacitances depend on factors like fill patterns, minimum inter-fill spacing and minimum conductor-to-fill spacing values.

8.1.2. Chemical mechanical polishing

Systematic variations in the metal thickness following a damascene CMP process due to factors like planarization length, layout density and perimeter sum cause *dishing* and *erosion* of interconnects (Figure 8.2). There is a high amount of dishing for wide lines, and erosion increases with increasing metal pattern density. Erosion generally dominates dishing for fine pitched lines, specially at high density [**109**]. Dishing and erosion may cause a defocus issue in

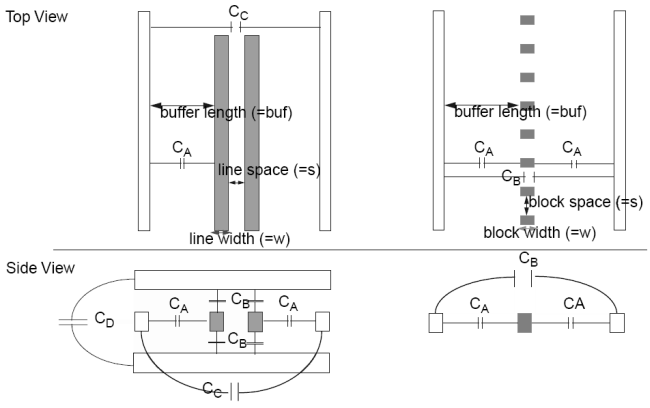

Figure 8.1. Increased interconnect capacitances due to fills

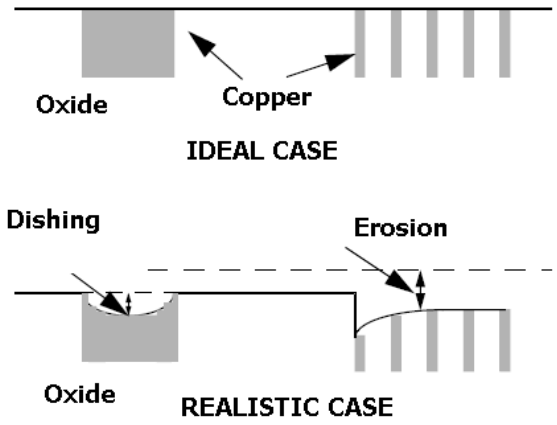

Figure 8.2. Dishing and erosion caused by CMP

the lithography process following CMP and therefore is a concern from the manufacturability perspective. They can also cause interconnect resistance and capacitance variations due to changes in interconnect cross-section.

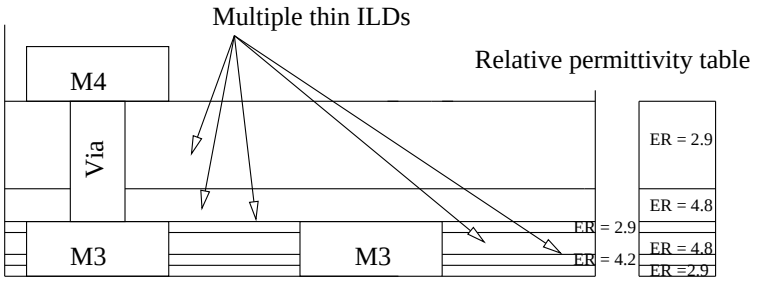

Figure 8.3. Vertical profile for a given process

8.1.3. Multiple thin dielectrics

Modern manufacturing process employ multiple ILDs between vertically adjacent metal layers. The use of low-k dielectrics help in reducing the parasitic capacitances for the interconnects. However, these soft dielectrics are shielded using surface hard masks (typically large-k dielectrics) to aid planarization obtained by CMP.

For example, Figure 8.3 shows a vertical profile between two metal layers featuring six ILDs. Parasitic extractors often assume a single homogeneous ILD since extraction with multiple ILDs is time consuming. However, this introduces inaccuracies in the extracted interconnect capacitances. In our experiments, we replace the dielectric stack between every two adjacent metal regions, as shown in Figure 8.3, with a corresponding homogeneous dielectric with permittivity equal to the weighted average of permittivities of individual ILD layers between the metal regions. The weighting factor is taken as the thickness of the ILD layer. In other words, if $\epsilon_1, \epsilon_2, \ldots, \epsilon_k$ are the permittivities of ILDs with $d_1, d_2, \ldots, d_k$ being their thickness, the permittivity of homogeneous dielectric is taken as $\epsilon = \frac{\sum_{i=1}^{k} \epsilon_i . d_i}{\sum_{i=1}^{k} d_i}$.

8.1.4. Trapezoidal conductor cross-sections

Interconnect cross-sections for modern day ICs are better approximated as trapezoids than rectilinear geometries due to under-etching and electrolytic growth. Such variations in the

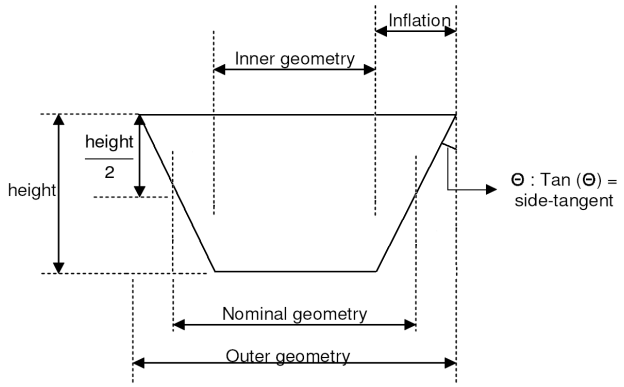

Figure 8.4. Trapezoidal interconnect cross-section specification

interconnect cross-section directly impacts its parasitic capacitance and the resistance. However, capacitance extraction tools use rectilinear geometries for efficiency purposes. To retain the efficiencies, trapezoidal conductors are often specified by their *nominal geometry*, which is rectilinear, and a *side-tangent*, which captures the slope. Figure 8.4 shows the specification for a trapezoidal conductor cross-section. Often trapezoidal conductor cross-sections are approximated to a staircase of rectilinear geometries, but at the expense of increased memory usage and run time.

8.2. Impact of modern process technologies

In this section, we present our experimental flow and the impact of each of the mentioned process technologies on a set of designs implemented in 90nm technology. Some of the designs are testcases obtained from Opencores [**110**]. In addition, we use a few real industrial testcases (*custom*1, *custom*2) in our flow.

We use a commercial, fast extraction tool to extract parasitics from connected databases that represent integrated circuit layout designs. For each design, the extracted parasitics, the synthesized library information and the design gate-level netlist is submitted to a commercial

Table 8.1. Design characteristics

Design	# total nets	# critical nets	Area ($\mu m \times \mu m$)
add16	63	19	22×22
risc16	1226	61	135×135
cordic	2526	121	199×199
usb	8082	101	248×248
fpu	9163	1034	260×260
custom1	28332	177	504×504
des	48737	161	734×734
custom2	35382	349	743×750

timing analysis tool. Timing analysis of the design with given timing constraints produces a set of (*50* for our experiments) most critical paths. We denote the nets lying on these critical paths as the critical nets for that design. Extremely accurate parasitic extractors using field solvers are employed to extract the electrical parameters for these critical nets, since these tools are relatively slow and it is impractical to employ them for parasitic extraction for the entire design.

For each design, we run the above flow for a base case, which does not contain dummy fills and considers process specified nominal metal and ILD thickness. The nominal thickness for the metal layers do not represent the pre-CMP thickness, but a mean value assuming CMP achieves uniform global planarization. The true thickness for a layer at any point could therefore be either more or less than the nominal value. Vertically adjacent metal layers are assumed to be separated with a single ILD, with a uniform dielectric constant. Finally, the base case considers rectilinear conductors (nominal geometries in Figure 8.4). For each design, we denote the timing critical nets obtained from its base case as the critical nets for that design. Table 8.1 presents the set of designs used in our experiments. For each design, we report the total number of nets, number of critical nets and its size.

Table 8.2. Impact of dense fills (all nets)

Design	Total cap variation (%)			
	μ	σ	max	min
add16	48.1	18.9	110.4	15.4
risc16	45.2	30.0	232.4	-7.3
cordic	54.9	32.8	222.7	-4.8
usb	32.7	23.6	182.0	-19.7
fpu	32.2	25.6	266.8	-21.9
custom1	31.1	22.5	191.8	-23.0
des	42.0	30.1	266.8	-21.5
custom2	15.0	12.4	144.0	-24.0

Table 8.3. Impact of sparse fills (all nets)

Design	Total cap variation (%)			
	μ	σ	max	min
add16	21.4	9.6	48.5	4.0
risc16	14.4	14.0	136.6	-12.0
cordic	18.9	17.0	117.2	-12.1
usb	7.8	9.7	109.4	-20.5
fpu	9.1	12.3	111.4	-29.1
custom1	5.6	7.1	75.9	-22.9
des	12.3	13.2	122.9	-32.5
custom2	3.6	5.2	72.3	-25.3

8.2.1. Impact of fills

We present the impact of dummy fill insertion on parasitic capacitances in this section. For each design, we generate dummy fills using a commercial Placement and Routing tool and set the minimal fill-to-metal spacing for a given metal layer to the same as the inter metal spacing for that layer. We denote this design with fills as the design with *dense* fills. In addition, we generate dummy fills in the base design with minimal fill-to-metal spacing for a given metal layer as twice the minimal inter metal spacing for that layer and term this design as the design with *sparse* fills. The experimental flow is run for the designs with *dense* as well as *sparse* fills. Since fill insertion does not directly impact the parasitic resistance of an interconnect, we present results obtained for resistance variations due to CMP based on underlying fill patters in the next section.

The extracted lumped capacitance of a net is termed as its *Wire cap* and includes its self and coupling capacitances. The sum of all capacitances on each of the pin the net connects to (that is, the driver and loading gate capacitances) is termed as the net's *Pin cap*. The sum of the net's *Wire cap* and *Pin cap* is termed as the *Total cap* of that net.

Mathematically, the wire and total capacitance variation due to dummy fills is computed as the following.

$$\%C\_var_{fills}^{Wire} \triangleq \frac{C_{fills}^{Wire} - C_{base}^{Wire}}{C_{base}^{Wire}} \times 100.0 \tag{8.1}$$

$$\%C\_var_{fills}^{Total} \triangleq \frac{C_{fills}^{Total} - C_{base}^{Total}}{C_{base}^{Total}} \times 100.0 \tag{8.2}$$

Note that process technologies do not affect interconnect *Pin caps*.

Tables 8.2 and 8.3 present the *Total cap* variations for all nets due to *dense* and *sparse* fills, respectively obtained using the fast parasitic extractor. For each design, we present the mean variation (μ), the standard deviation of the variation (σ) and the *max* and *min* variation. From these tables, we infer that *dense* and *sparse* fills cause an increase in the total capacitance of an interconnect by 37% and 11% respectively, on the average over all designs. We also observe that fills can cause as high as a 2.6X increase on the total capacitance of an interconnect (design *des* due to *dense fills*).

Tables 8.4 and 8.5 present accurate *Wire cap* and *Total cap* variations obtained from the field solver for the timing critical nets of each design. Results from accurate capacitance extraction show that although maximal wire capacitance variations can exceed 200%, the total capacitance variation is constrained because of the unaffected pin capacitances to less than 150%. We evaluate from these tables that on the average, the mean *Wire cap* and the

Table 8.4. Impact of dense fills (critical nets)

Design	Wire cap variation (%)				Total cap variation (%)			
	μ	σ	max	min	μ	σ	max	min
add16	104.0	35.8	195.0	39.3	49.9	24.7	114.6	11.3
risc16	80.5	28.1	160.0	5.1	52.6	29.0	133.4	0.1
cordic	84.9	31.7	208.7	9.2	48.3	28.0	142.6	0.8
usb	46.1	19.9	108.4	9.1	34.9	20.7	102.0	1.5
fpu	72.0	29.9	217.8	2.2	32.2	18.1	141.8	0.3
custom1	50.8	29.5	219.4	-1.3	31.3	26.5	146.5	-0.3
des	76.2	31.3	154.5	5.1	53.1	32.3	126.9	1.0
custom2	33.1	15.2	97.4	2.1	15.4	13.6	67.5	0.0

Table 8.5. Impact of sparse fills (critical nets)

Design	Wire cap variation (%)				Total cap variation (%)			
	μ	σ	max	min	μ	σ	max	min
add16	31.0	11.5	52.9	8.6	14.9	7.9	36.0	2.4
risc16	20.4	9.7	48.0	3.4	13.3	8.7	44.2	0.1
cordic	21.6	13.0	77.0	2.1	12.3	9.5	51.0	0.0
usb	7.7	5.8	25.4	0.3	5.9	4.8	19.6	0.1
fpu	16.0	10.0	88.0	-2.1	7.1	5.6	62.3	-0.7
custom1	7.7	8.0	65.4	-3.0	4.7	5.8	43.7	-0.7
des	18.6	11.2	58.4	-3.0	12.8	9.3	45.0	-0.9
custom2	7.5	6.0	35.5	-5.7	3.6	4.2	27.7	-0.8

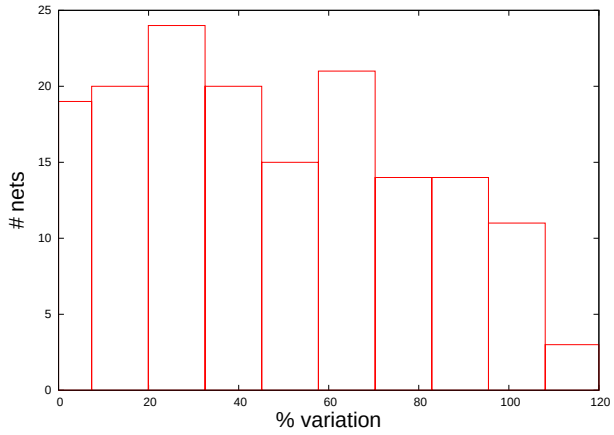

Figure 8.5. Accurate impact of dense fills on total capacitance for timing critical nets (design *des*)

Total cap variations for the designs with *dense* fills are 68% and 40%, respectively. These numbers for the design with *sparse* fills are evaluated to be 16% and 9%, respectively.

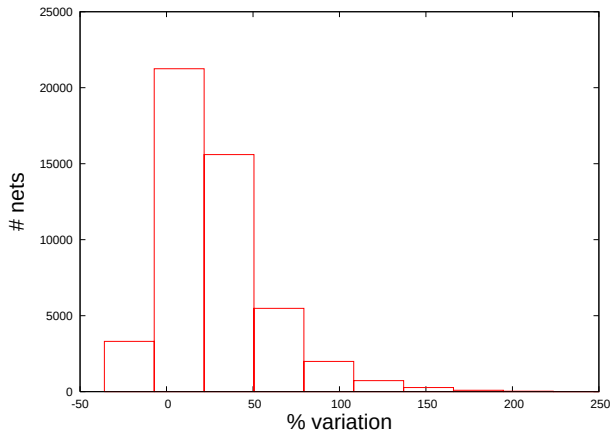

Figure 8.6. Impact of dense fills on total capacitance for all nets (design *des*)

Figure 8.5 shows the distribution of the % *Total cap* variation for the timing critical nets of the design *des* due to *dense fills*. The X-axis shows eleven uniformly spaced % *Total cap* variation buckets over the entire range of variation for the design. We plot the number of nets lying in a particular bucket (range of % *Total cap* variation) in the Y axis. We observe that the distribution is positively skewed with a mean variation of 31%. Similar plots showing the distribution of the % *Total cap* variation for all nets in the design *des* due to *dense fills* is presented in Figure 8.6.

8.2.2. Impact of CMP

We use a prototype industrial CMP simulator to obtain metal thickness variations for all layers in a design having dummy fills. The topographies obtained from the CMP simulator are fed to the field solver to obtain the impact of CMP on the resistance and capacitance of the timing critical nets. In the CMP simulation, we do not consider the effects of multi-layer accumulative topography variation on the metal thickness variations. The variations of dielectric thickness and relative height of metal lines due to the multi-layer accumulative topography variations are not considered either. Note that the base case here has dummy

Table 8.6. Impact of CMP with dense fills (critical nets)

Design	Wire cap variation (%)				Total cap variation (%)			
	μ	σ	max	min	μ	σ	max	min
add16	0.2	0.9	2.0	-1.3	0.1	0.6	1.4	-0.8
risc16	0.1	0.8	2.4	-1.7	0.0	0.5	1.2	-1.6
cordic	-0.1	1.4	3.3	-3.5	-0.1	0.9	1.8	-2.4
usb	0.3	1.0	4.6	-2.5	0.2	0.6	2.2	-1.2
fpu	0.0	1.4	5.7	-4.3	0.0	0.8	2.9	-2.7
custom1	0.1	1.2	5.5	-3.2	0.1	0.6	1.9	-3.2
des	0.1	1.2	4.1	-5.4	0.1	0.7	2.2	-2.2
custom2	0.0	1.2	3.9	-5.9	0.0	0.5	1.3	-4.5

Table 8.7. Impact of CMP with sparse fills (critical nets)

Design	Wire cap variation (%)				Total cap variation (%)			
	μ	σ	max	min	μ	σ	max	min
add16	0.4	1.3	2.8	-1.8	0.1	0.9	1.7	-1.8
risc16	0.1	1.2	3.5	-2.9	0.1	0.8	3.5	-1.5
cordic	-0.1	1.5	3.1	-4.7	0.0	0.9	2.3	-2.8
usb	0.0	1.0	2.8	-4.9	-0.1	0.7	1.5	-2.7
fpu	0.0	1.7	6.1	-6.0	0.0	0.8	3.0	-4.0
custom1	0.0	1.4	4.5	-5.9	0.0	0.6	1.7	-1.7
des	0.2	1.3	4.5	-4.2	0.2	0.9	3.5	-2.6
custom2	0.0	1.2	3.7	-4.5	0.0	0.5	2.1	-3.0

fills and assumes uniform metal thickness. We use this base case for comparison to a design having the same fill pattern but considering thickness variations due to CMP.

Tables 8.6 and 8.7 present the *Wire cap* and *Total cap* variations due to CMP for designs having *dense* and *sparse* fills, respectively. We observe that the variations in the wire and total capacitances due to CMP are negligible on the average, having a standard deviation of about 0.7%. We observe that the thickness variations caused due to CMP are small $(5\% - 10\%)$. Similar thickness variations found for all metal layers explain the negligible impact of CMP observed. We present the distribution of the % *Total cap* variation on the critical nets with underlying *dense* fills for the design *des* in Figures 8.7 and 8.8.

It is seen that the distributions are approximately symmetric around a 0% mean. The absolute maximal *Total cap* variation is found to be 4.5% (design *custom2*). However, note

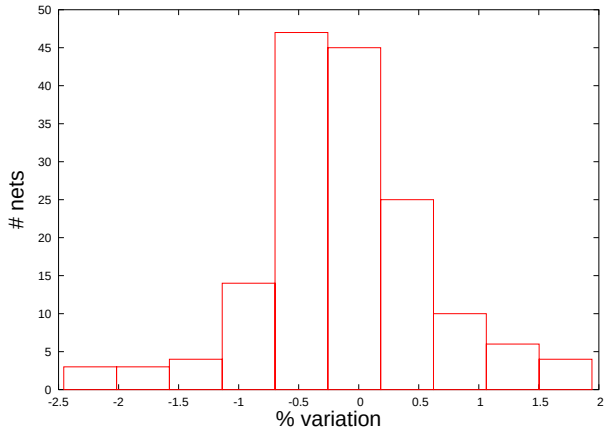

Figure 8.7. Accurate impact of CMP (dense fills) on total capacitance for timing critical nets (design *des*)

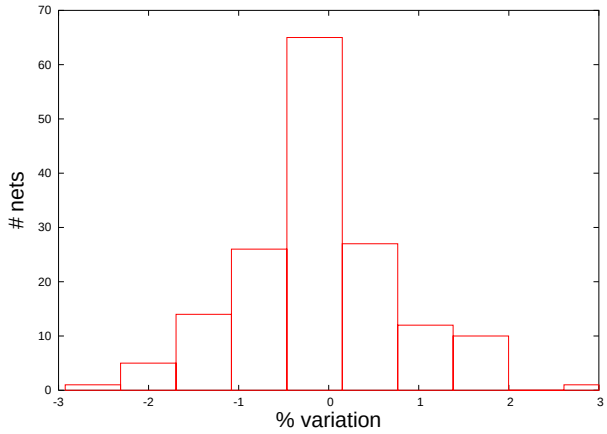

Figure 8.8. Accurate impact of CMP (sparse fills) on total capacitance for timing critical nets (design *des*)

that the effects of dielectric thickness variations and relative height variations of metal lines on capacitance are not considered in our study.

The resistance between two given pins of a net is termed as its *Port* resistance. The resistance of the gate timing-arc that drives a given net is termed as the *Drive* resistance. The sum of the port and drive resistance is termed as the net's *Total* resistance. Tables 8.8 and 8.9 present the variation in the port and total resistance for the critical nets in each design due to CMP with underlying *dense* and *sparse* fills, respectively. We observe that the

Table 8.8. Impact of CMP with dense fills (critical nets)

Design	Wire res variation (%)				Total res variation (%)			
	μ	σ	max	min	μ	σ	max	min
add16	9.1	3.0	12.4	-0.6	1.0	3.0	10.7	-0.6
risc16	8.4	0.9	10.8	5.3	0.0	0.0	0.1	0.0
cordic	6.3	1.0	8.7	4.4	0.0	0.0	0.1	0.0
usb	4.5	0.9	5.9	0.9	0.5	1.4	5.9	0.0
fpu	3.6	0.7	10.2	0.3	0.0	0.0	0.1	0.0
custom1	2.7	0.9	7.4	0.26	0.2	0.5	4.3	0.0
des	1.6	2.0	4.5	-20.7	0.0	0.6	2.1	-8.1
custom2	3.2	3.0	14.7	0.4	0.1	0.2	0.6	0.0

Table 8.9. Impact of CMP with sparse fills (critical nets)

Design	Wire res variation (%)				Total res variation (%)			
	μ	σ	max	min	μ	σ	max	min
add16	17.1	5.0	23.8	3.9	1.9	5.0	19.1	0.0
risc16	4.1	1.0	7.3	2.2	0.0	0.0	0.0	0.0
cordic	2.1	1.5	6.4	0.2	0.0	0.0	0.0	0.0
usb	-0.7	0.7	1.3	-2.2	0.0	0.2	1.3	-0.3
fpu	-0.2	0.7	9.5	-1.8	0.0	0.0	0.0	0.0
custom1	-1.6	1.0	7.1	-2.8	0.0	0.3	3.0	-2.5
des	-0.2	0.8	2.7	-1.8	0.0	0.1	0.0	-1.8
custom2	0.1	4.8	38.3	-7.2	0.0	0.1	0.1	-0.6

total resistance variations due to CMP are not significant on the average. This is because the drive resistance is usually much larger than the port resistance. However, the absolute maximal variations observed in the total resistance are large ($2.5\% - 19.1\%$) for several designs (designs *add16*, *custom1* and *usb*). The variation in total resistances for the design *des* is presented in Figures 8.9 and 8.10.

A detailed study of these designs shows that the drive resistance is in the same order as the port resistance on the net with the maximal total resistance variation. This variation may be large enough to cause a re-ordering of critical paths.

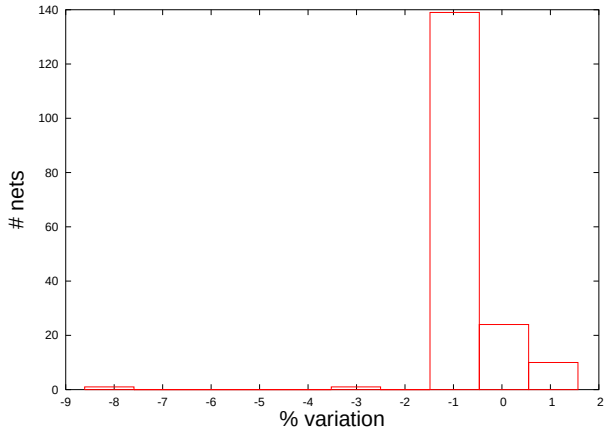

Figure 8.9. Accurate impact of CMP (dense fills) on total resistance for timing critical nets (design *des*)

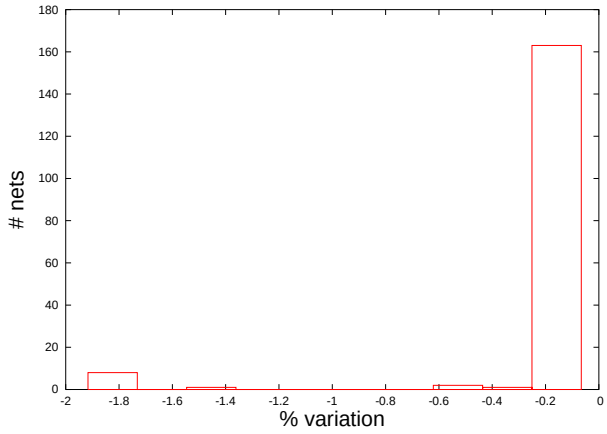

Figure 8.10. Accurate impact of CMP (space fills) on total resistance for timing critical nets (design *des*)

8.2.3. Impact of multiple thin dielectrics

We next consider variations in interconnect capacitances for a design having multiple ILDs (specifications obtained from the respective foundries) with respect to a base case which considers average single ILDs. The dielectric constant of the single ILD in a region is determined by a thickness weighted average of the dielectric constants of the specified multiple ILDs in that region. Single ILDs are computed for each region comprising of a routing layer

Table 8.10. Impact of multiple thin dielectrics (all nets)

Design	Total cap variation (%)			
	μ	σ	max	min
add16	0.0	0.9	1.6	-2.4
risc16	0.6	0.7	7.5	-3.5
cordic	0.6	0.9	7.4	-6.9
usb	0.4	0.7	6.1	-6.1
fpu	0.4	0.6	10.8	-4.2
custom1	0.4	0.7	4.8	-11.7
des	0.4	0.6	8.6	-6.4
custom2	0.0	0.9	7.5	-11.2

Table 8.11. Impact of multiple thin dielectrics (critical nets)

Design	Wire cap variation (%)				Total cap variation (%)			
	μ	σ	max	min	μ	σ	max	min
add16	-1.8	1.3	1.2	-4.2	-0.7	0.6	0.6	-1.7
risc16	-2.3	1.2	1.8	-4.8	-1.4	0.9	0.5	-3.2
cordic	-2.6	1.8	5.4	-8.0	-1.3	1.2	2.7	-6.8
usb	-2.8	1.6	0.2	-10.1	-2.0	1.3	0.2	-5.8
fpu	-2.0	1.1	2.0	-6.6	-0.8	0.6	0.5	-5.5
custom1	-2.6	1.3	0.6	-7.0	-1.4	1.1	0.5	-6.0
des	-2.2	1.9	5.7	-9.2	-1.5	1.5	1.5	-8.6
custom2	-1.8	2.2	5.8	-8.5	-0.9	1.3	3.9	-8.2

(metal/poly) and the via layer above it. For the vertical dielectric profile shown in Figure 8.3, a single ILD would be computed for the region starting from the bottom level of metal layer 'M3' to the bottom level of the metal layer 'M4'. If there are n layers in a region, each of which has a dielectric constant of ϵ_i and an extent of w_i, $i = 1..n$, then the effective uniform dielectric constant ϵ_{eff} in that regions is the weighted volume average of all the individual dielectric constants. Since the area is the same, ϵ_{eff} is computed as follows.

$$\epsilon_{eff} = \frac{\sum_{i=1}^{n} \epsilon_i \times w_i}{\sum_{i=1}^{n} w_i} \tag{8.3}$$

We observe negligible impact of the assumption of a single ILD for faster extraction over all nets in Table 8.10 (obtained from the fast parasitic extractor) on the average. Table 8.11 presents very accurate wire and total capacitance variations on the critical nets of each

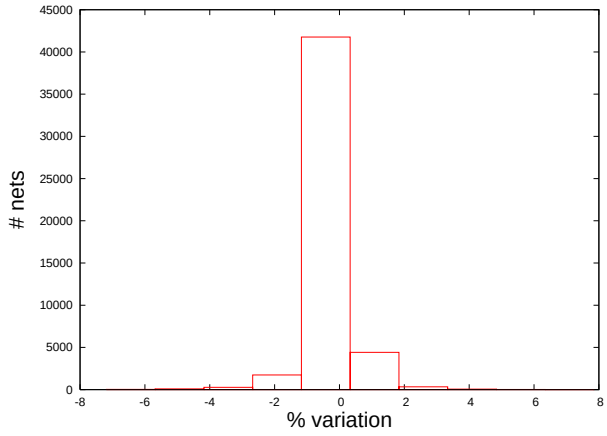

Figure 8.11. Impact of thin dielectrics on total capacitance for all nets (design *des*)

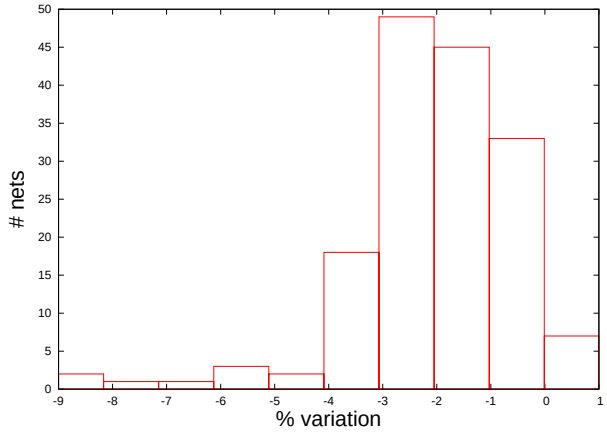

Figure 8.12. Accurate impact of thin dielectrics on total capacitance for timing critical nets (design *des*)

design (obtained from the field solver). From Table 8.11, we infer that the assumption of averaged ILDs between metal layers overestimates the total net capacitances by 1.4% on the average. Figures 8.11 and 8.12 present the distribution of the *Total cap* variations on all nets and critical nets, respectively for design *des*. The distributions are found to be negatively skewed.

Table 8.12. Impact of trapezoidal conductors (all nets)

Design	Total cap variation (%)			
	μ	σ	max	min
add16	2.3	1.9	6.8	-1.6
risc16	5.9	3.0	16.6	-5.3
cordic	4.7	3.0	19.2	-4.7
usb	7.1	3.8	21.6	-6.2
fpu	5.3	3.2	19.1	-6.2
custom1	7.1	4.1	23.8	-3.0
des	6.2	3.2	22.0	-5.4
custom2	2.5	2.5	18.4	-5.6

Table 8.13. Impact of trapezoidal conductors (critical nets)

Design	Wire cap variation (%)				Total cap variation (%)			
	μ	σ	max	min	μ	σ	max	min
add16	16.1	3.9	23.6	9.6	7.4	4.1	20.2	2.2
risc16	18.6	3.6	30.9	10.1	11.1	5.4	20.8	0.6
cordic	18.1	4.2	31.4	9.1	9.8	5.1	26.2	0.3
usb	22.6	4.2	33.2	9.6	16.3	7.4	29.8	0.9
fpu	18.9	3.8	34.6	6.5	7.6	3.0	23.9	0.2
custom1	21.4	4.1	31.7	11.2	11.2	6.7	26.6	0.7
des	18.3	4.1	29.9	6.5	11.8	5.8	28.0	0.3
custom2	17.9	3.4	26.6	6.8	7.6	5.7	25.4	0.1

8.2.4. Impact of trapezoidal conductor cross-sections

We consider the impact of trapezoidal cross-sections and etching of conductors on interconnect parasitics in this section. We extract parasitics of a design considering the specifications for the outer and nominal geometries and the side tangent (Figure 8.4) obtained as functions of metal thickness and density from respective foundries. The extracted parasitics are compared to those obtained from the base case that considers only nominal geometries and does not consider effects caused due to etching and trapezoidal conductor cross-sections.

We present the total capacitance variation observed for all nets in Table 8.12 (obtained from the fast parasitic extractor). Table 8.13 presents very accurate wire and total capacitance variations on the critical nets of each design (obtained from the field solver). Figures 8.13 and 8.14 present the distribution of the *Total cap* variations for all nets and timing

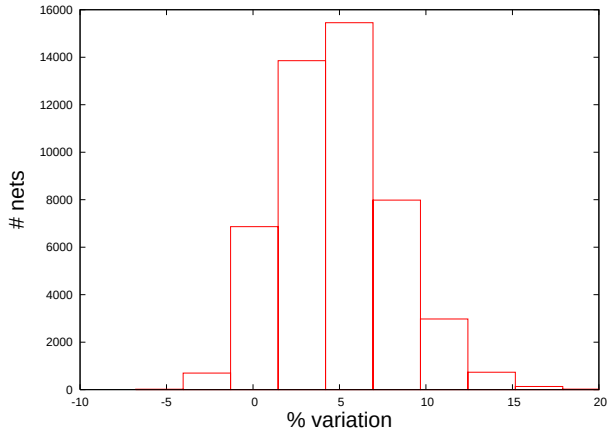

Figure 8.13. Impact of trapezoidal conductor cross-sections on total capacitance for all nets (design *des*)

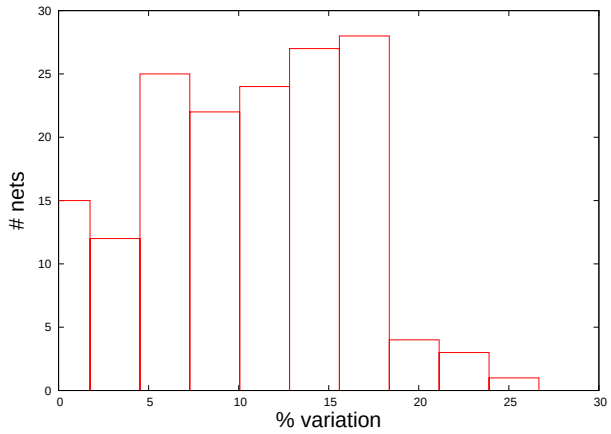

Figure 8.14. Accurate impact of trapezoidal conductor cross-sections on total capacitance for timing critical nets (design *des*)

critical nets, respectively of the design *des*. We evaluate from Table 8.13 that the presence of trapezoidal conductor cross-sections and etching increase the total net capacitances of timing critical nets by 10% on the average.

8.3. Conclusions

Based on the obtained experimental results, we conclude that fills can contribute significantly to the total capacitance of a net. In addition, we observe that varying fill patterns

cause significantly varying impact on a net's total capacitance. It is thus important that fills be added carefully.

Results obtained on our set of designs show negligible impact of CMP on the average. However, sufficiently large variations of the total resistance is observed on some nets. This variation may cause a re-ordering of the critical paths or make non-critical path critical.

It should be noted that the current CMP simulation does not consider the dielectric thickness and relative wire height variations induced by multi-layer accumulative effects. While larger across-chip CMP thickness variations are expected for a design with a larger area, the area of designs used in our CMP simulations is relatively small. The impact of CMP thickness variations on resistance and capacitance may be more significant by using a larger design. In addition, with the reduction of the nominal metal thickness, a higher impact of CMP on the resistance and capacitance variation is expected at 65nm and 45nm nodes.

We observe that the assumption of averaged ILDs between metal layers overestimates total capacitance of a net on the average by 1.4%. However, extraction with averaged ILDs is much faster than the extraction with actual ILDs. Finally, we conclude that trapezoidal conductor cross-sections and etching can significantly impact the total capacitance of a net. For sub 90nm designs, it is expected that these variations would be more significant and must be considered in circuit design and optimization.

APPENDIX A

A.1. Variance matching validation

Using notations defined in Chapter 4, we prove that the variance matching method in (4.10) never involves the computation of the root of a negative quantity. Formally, we prove that

$$[\xi \triangleq \sigma^2_{max} - \sum_{i=1}^{n} c_i^2] \geq 0.$$

Proof.

$$\begin{aligned}
\mu_{max} &= a_0\Phi(\alpha) + b_0\Phi(-\alpha) + \theta\phi(\alpha) \\
\sigma^2_{max} &= (\sigma_A^2 + a_0^2)\Phi(\alpha) + (\sigma_B^2 + b_0^2)\Phi(-\alpha) + (a_0 + b_0)\theta\phi(\alpha) - \ \mu^2_{max} \\
\sum_{i=1}^{n} c_i^2 &= \sum_{i=1}^{n} \left[a_i\Phi(\alpha) + b_i\Phi(-\alpha)\right]^2
\end{aligned}$$

From the above, we have the following.

$$
\begin{aligned}
\xi &= \sigma^2_{max} - \sum_{i=1}^{n} c_i^2 \\
&= (\sigma_A^2 + a_0^2)\Phi(\alpha) + (\sigma_B^2 + b_0^2)\Phi(-\alpha) + (a_0 + b_0)\theta\phi(\alpha) - \ a_0^2\Phi(\alpha)^2 - b_0^2\Phi(-\alpha)^2 \\
&\quad - \theta^2\phi(\alpha)^2 - 2a_0\Phi(\alpha)\theta\phi(\alpha) - \ 2b_0\Phi(-\alpha)\theta\phi(\alpha) - 2a_0b_0\Phi(\alpha)\Phi(-\alpha) \\
&\quad - \Phi(\alpha)^2\sum_{i=1}^{n} a_i^2 - \Phi(-\alpha)^2\sum_{i=1}^{n} b_i^2 - 2\Phi(\alpha)\Phi(-\alpha)\sum_{i=1}^{n} a_ib_i \\
&= \sigma_A^2\Phi(\alpha) + a_0^2\Phi(\alpha) + \sigma_B^2\Phi(-\alpha) + b_0^2\Phi(-\alpha) \\
&\quad + (a_0 + b_0)\theta\phi(\alpha) - a_0^2\Phi(\alpha)^2 - b_0^2\Phi(-\alpha)^2 \\
&\quad - 2a_0\Phi(\alpha)\theta\phi(\alpha) - 2b_0\Phi(-\alpha)\theta\phi(\alpha) - \theta^2\phi(\alpha)^2 \\
&\quad - 2a_0b_0\Phi(\alpha)\Phi(-\alpha) - \sigma_A^2\Phi(\alpha)^2 + a_{n+1}^2\Phi(\alpha)^2 \\
&\quad -\sigma_B^2\Phi(-\alpha)^2 + b_{n+1}^2\Phi(-\alpha)^2 - \ 2\Phi(\alpha)\Phi(-\alpha)\sum_{i=1}^{n} a_ib_i \\
&= (\sigma_A^2 + \sigma_B^2 + a_0^2 + b_0^2)\Phi(\alpha)\Phi(-\alpha) + (a_0 + b_0)\theta\phi(\alpha) \\
&\quad - \theta^2\phi(\alpha)^2 - 2a_0\Phi(\alpha)\theta\phi(\alpha) - 2b_0\Phi(-\alpha)\theta\phi(\alpha) \\
&\quad - 2a_0b_0\Phi(\alpha)\Phi(-\alpha) + a_{n+1}^2\Phi(\alpha)^2 + b_{n+1}^2\Phi(-\alpha)^2 \\
&\quad - 2\Phi(\alpha)\Phi(-\alpha)\sum_{i=1}^{n} a_ib_i \\
&= \Phi(\alpha)\Phi(-\alpha)\big[(\sigma_A^2 + \sigma_B^2 - 2\sum_{i=1}^{n} a_ib_i) + (a_0 - b_0)^2\big] \\
&\quad + \theta\phi(\alpha)\big[a_0(1 - 2\Phi(\alpha)) + b_0(1 - 2 + 2\Phi(\alpha))\big] \\
&\quad - \theta^2\phi(\alpha)^2 + a_{n+1}^2\Phi(\alpha)^2 + b_{n+1}^2\Phi(-\alpha)^2
\end{aligned}
$$

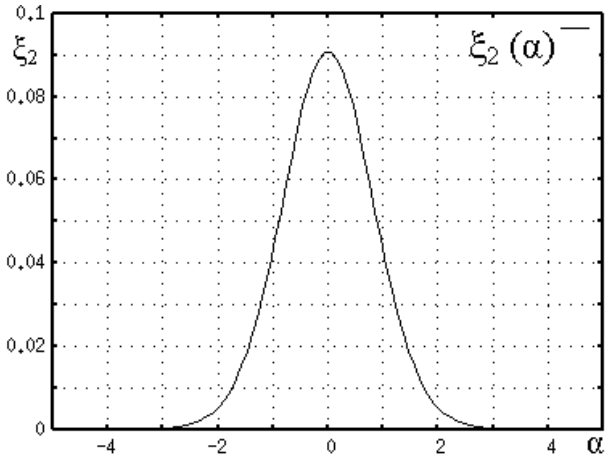

Figure A.1. Plot of ξ_2 against α

To show $\xi \geq 0$, it is sufficient to show that

$$\xi_1 \triangleq \xi - a_{n+1}^2 \Phi(\alpha)^2 - b_{n+1}^2 \Phi(-\alpha)^2 \geq 0.$$

$$\begin{aligned}
\xi_1 &= \Phi(\alpha)\Phi(-\alpha)\big[(\sigma_A^2 + \sigma_B^2 - 2\sum_{i=1}^{n} a_i b_i) + (a_0 - b_0)^2\big] \\
&\quad + \theta\phi(\alpha)\big[a_0(1 - 2\Phi(\alpha)) + b_0(-1 + 2\Phi(\alpha))\big] - \theta^2\phi(\alpha)^2 \\
&= \Phi(\alpha)\Phi(-\alpha)\big[\theta^2 + (a_0 - b_0)^2\big] - \theta^2\phi(\alpha)^2 \\
&\quad + \theta(a_0 - b_0)(1 - 2\Phi(\alpha))\phi(\alpha) \\
&= \theta^2\big[\Phi(\alpha)\Phi(-\alpha) - \phi(\alpha)^2\big] + \theta(a_0 - b_0)(1 - 2\Phi(\alpha))\phi(\alpha) \\
&\quad + \Phi(\alpha)\Phi(-\alpha)(a_0 - b_0)^2
\end{aligned}$$

If $\theta = 0$, $\xi_1 = \Phi(\alpha)\Phi(-\alpha)(a_0 - b_0)^2 \geq 0$. For positive θ (since $\theta \geq 0$), it is sufficient to show that

$$\xi_2 \triangleq \xi_1/\theta^2 \geq 0.$$

$$\begin{aligned}\xi_2 &= \Phi(\alpha)\Phi(-\alpha) - \phi(\alpha)^2 + \frac{a_0 - b_0}{\theta}(1 - 2\Phi(\alpha))\phi(\alpha) + \Phi(\alpha)\Phi(-\alpha)\frac{(a_0 - b_0)^2}{\theta^2} \\ &= \Phi(\alpha)\Phi(-\alpha) - \phi(\alpha)^2 + \alpha(1 - 2\Phi(\alpha))\phi(\alpha) + \Phi(\alpha)\Phi(-\alpha)\alpha^2\end{aligned}$$

$\xi_2(\alpha)$ is symmetric and is found to be non-negative for all real values of α. For values of $|\alpha| \geq 3$, ξ_2 approaches 0 with both $\phi(\alpha)$ and $\Phi(\alpha)$ tending to 0. Figure A.1 shows the plot of ξ_2 as a function of α. □

A.2. Error equivalence validation

We consider approximating $Z \triangleq max(X, Y)$ with a Gaussian $Z_G \sim N(\mu_Z, \sigma_Z^2)$, and approximating $Z' \triangleq max(X', Y')$ with a Gaussian $Z'_G \sim N(\mu_{Z'}, \sigma_{Z'}^2)$. $\varphi_Z(t)$ and $\varphi_{Z_G}(t)$ denote PDFs as defined earlier and $\varphi_{Z'}(t)$ and $\varphi_{Z'_G}(t)$ denote the PDFs of Z' and Z'_G, respectively. From (5.9) and (5.12), we have the following.

$$\varphi_{Z'_G}(t) \triangleq \frac{1}{\sqrt{2\pi}\sigma_{Z'}} e^{-\frac{(t-\mu_{Z'})^2}{2\sigma_{Z'}{}^2}} \tag{A.1}$$

$$\varphi_{Z'}(t) \triangleq \frac{1}{\sigma_{Y'}}\phi(\frac{t-\mu_{Y'}}{\sigma_{Y'}})\Phi\big[\frac{t-\rho(\frac{t-\mu_{Y'}}{\sigma_{Y'}})}{(1-\rho^2)^{1/2}}\big] + \phi(t)\Phi\big[\frac{(\frac{t-\mu_{Y'}}{\sigma_{Y'}})-\rho t}{(1-\rho^2)^{1/2}}\big] \tag{A.2}$$

Lemma A.2.1. *$\mu_Z = \mu_X + \sigma_X\mu_{Z'}$ and $\sigma_Z = \sigma_X\sigma_{Z'}$*

Proof. The results are trivially derived from (5.5)–(5.6). □

Lemma A.2.2. $\varphi_{Z_G}(t) = \frac{1}{\sigma_X}\varphi_{Z'_G}(\frac{t-\mu_X}{\sigma_X})$

Proof. The result is immediate from a simple substitution in (A.1) and from Lemma A.2.1. □

Lemma A.2.3. $\varphi_Z(t) = \frac{1}{\sigma_X}\varphi_{Z'}(\frac{t-\mu_X}{\sigma_X})$

Proof. The result is immediate from a simple substitution in (A.2) which yields (5.12). □

We next prove that the error in approximating Z with Z_G is the same as the error in approximating Z with Z'_G. From our error definition in (5.8), and from the results obtained

above, we have the following.

$$
\begin{aligned}
&\Xi_{(Z')(Z'_G)} \\
&= \int_{-\infty}^{\infty} |\varphi_{Z'}(t) - \varphi_{Z'_G}(t)| \; dt \\
&= \int_{-\infty}^{\infty} |\varphi_{Z'}(\frac{t-\mu_X}{\sigma_X}) - \varphi_{Z'_G}(\frac{t-\mu_X}{\sigma_X})| \; d(\frac{t-\mu_X}{\sigma_X}) \\
&= \sigma_X \int_{-\infty}^{\infty} |\varphi_Z(t) - \varphi_{Z_G}(t)| \; d(\frac{t-\mu_X}{\sigma_X}) \\
&= \sigma_X \int_{-\infty}^{\infty} |\varphi_Z(t) - \varphi_{Z_G}(t)| \; dt \; \frac{1}{\sigma_X} \\
&= \Xi_{(Z)(Z_G)}
\end{aligned}
$$

A.3. Expression for $P_n(\alpha)$

We prove the following using mathematical induction.

$$\begin{aligned} \text{Let} \quad P_n(\alpha) &\triangleq (-1)^n\, n! \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor} \frac{(-1)^i\, \alpha^{n-2i}}{2^i\, i!\, (n-2i)!} \\ \text{then} \quad \Phi^{(n+1)}(\alpha) &= \phi^{(n)}(\alpha) = \phi(\alpha) P_n(\alpha). \end{aligned}$$

Proof. The above can be trivially verified for $n = 0$. We assume that the given expression is true for a given $n = m$ and attain to compute the coefficient of α^{m+1-2i} in $\frac{\phi^{(m+1)}(\alpha)}{\phi(\alpha)}$.

$$\begin{aligned} \phi^{(m)}(\alpha) &= (-1)^m\, \phi(\alpha)\, m! \sum_{i=0}^{\lfloor \frac{m}{2} \rfloor} \frac{(-1)^i\, \alpha^{m-2i}}{2^i\, i!\, (m-2i)!} \\ &= (-1)^m\, \phi(\alpha)\, m! \big[\frac{\alpha^n}{n!} + \ldots + \\ &\quad \frac{(-1)^{i-1}\, \alpha^{m-2i+2}}{2^{i-1}\, (i-1)!\, (m-2i+2)!} + \frac{(-1)^i\, \alpha^{m-2i}}{2^i\, i!\, (m-2i)!} \ldots \big] \\ \phi^{(m+1)}(\alpha) &= \frac{d}{d\alpha} \phi^{(m)}(\alpha) \\ &= (-1)^m\, \phi(\alpha)\, m! \big[\frac{\alpha^{n-1}}{(n-1)!} + \ldots + \\ &\quad \frac{(-1)^{i-1}\, \alpha^{m-2i+1}}{2^{i-1}\, (i-1)!\, (m-2i+1)!} + \frac{(-1)^i\, \alpha^{m-2i-1}}{2^i\, i!\, (m-2i-1)!} \ldots \big] \\ &\quad + (-1)^{m+1}\, \phi(\alpha)\, m! \big[\frac{\alpha^{n+1}}{n!} + \ldots + \\ &\quad \frac{(-1)^{i-1}\, \alpha^{m-2i+3}}{2^{i-1}\, (i-1)!\, (m-2i+2)!} + \frac{(-1)^i\, \alpha^{m-2i+1}}{2^i\, i!\, (m-2i)!} \ldots \big] \end{aligned}$$

The coefficient of α^{m+1-2i} in $\frac{\phi^{(m+1)}(\alpha)}{\phi(\alpha)}$ is found to be the following.

$$\begin{aligned} & (-1)^m\, m!\big[\frac{(-1)^{i-1}}{2^{i-1}\,(i-1)!\,(m-2i+1)!}\big] + (-1)^{m+1}\, m!\big[\frac{(-1)^i}{2^i\, i!\,(m-2i)!}\big] \\ = \;& (-1)^{m+1}\,(m+1)!\;\frac{(-1)^i}{2^i\, i!\,(m-2i+1)!} \end{aligned}$$

This is identical to the coefficient of α^{m+1-2i} in $P_{m+1}(\alpha)$. □

A.4. Discussion on Taylor series expansion of $\Phi(\alpha)$

Statistical timing analysis is often used to guide timing optimization [**15, 55, 56, 57, 53**]. Guthaus *et al.* [**58**] propose a gate-sizing algorithm to optimize circuit area while satisfying a given timing yield target. They employ a sensitivity metric based on slack distributions to select gates for resizing. Their sensitivity metric is computed from the node and edge criticalities [**48**] of the circuit. An approach to computing node and edge sensitivities considering correlations is proposed by Li *et al.* in [**111**]. Criticality (sensitivity) computations in both the approaches involve evaluating tightness probabilities or $\Phi(\alpha)$. We therefore realize that accurate computations of tightness probabilities are critical to statistical timing analysis, and subsequently to statistical timing optimization. At the same time, it is important that these computations be efficient. Ideally, it is desirable to have an option of dynamically choosing a runtime-accuracy trade-off point during any such computation in a timing optimization tool.

A well defined closed form of $\phi^{(n)}(\alpha)$ from (5.19) facilitates the computation of $\Phi(\alpha)$ and $\phi(\alpha)$ using the Taylor series expansion about any point. For practical evaluation, the infinite series is truncated to a finite one. The sum of the terms lost is called the *Lagrange remainder* or the *residual*. When the first $n+1$ terms are retained in (5.18), (*i.e.*, terms $\frac{\Phi^{(n+1)}(k)}{(n+1)!}(\alpha-k)^{n+1}+\ldots$ are truncated), the Lagrange remainder R_n is expressed as the following (from [**64**]).

$$\begin{aligned} R_n &\triangleq \underbrace{\int_k^\alpha \ldots \int_k^\alpha}_{n+1} \Phi^{(n+1)}(\alpha)(d\alpha)^{n+1} \\ &= \int_k^\alpha \Phi^{(n+1)}(t)\frac{(\alpha-t)^n}{n!}dt = \frac{(\alpha-k)^{n+1}}{(n+1)!}\Phi^{(n+1)}(\alpha^*) \end{aligned}$$

for some $\alpha^* \in [k, \alpha]$ (using the *mean-value* theorem). R_n is used as an accuracy estimate in the computation of $\Phi(\alpha)$ (and $\phi(\alpha)$). We next present the following bound.

Lemma A.4.1.

$$|\Phi^{(2n+1)}(\alpha)| \ \leq \ \frac{(2n)!}{\sqrt{2\pi}\, 2^n} \qquad \forall \alpha \in (-\infty, \infty)$$

The derivative of $\Phi^{(2n+1)}(\alpha)$ is a multiple of α for any non-negative n. It is observed that the local extremal at $\alpha = 0$ corresponds to the global maximal of $|\Phi^{(2n+1)}(\alpha)|$. Figure A.4 plots $\Phi^{(2n+1)}(\alpha)$ against α for $n = \{0, 1, 2, 3\}$. The above bound is obtained by setting $\alpha = 0$ in (A.3). From this result and (A.3), we have the following.

$$|R_{2n}| \ \leq \ \frac{(|\alpha - k|)^{2n+1}}{\sqrt{2\pi}\, 2^n\, (2n+1)} \tag{A.3}$$

The given bound is a good bound for any α in the neighborhood of 0. We observe that the bound can be severely tightened in the given region by choosing a k very close to α, such that

$$| \ \alpha - k \ | \ << \ 1.$$

This makes the residual decrease exponentially with n. For practical purposes, we make the following assumptions.

$$\begin{aligned} \Phi(\alpha) \ &\approx \ 0 \quad \forall \alpha \ \leq -7 \\ \Phi(\alpha) \ &\approx \ 1 \quad \forall \alpha \ \geq \ 7 \\ \phi(\alpha) \ &\approx \ 0 \quad \forall |\alpha| \leq \ 7 \end{aligned}$$

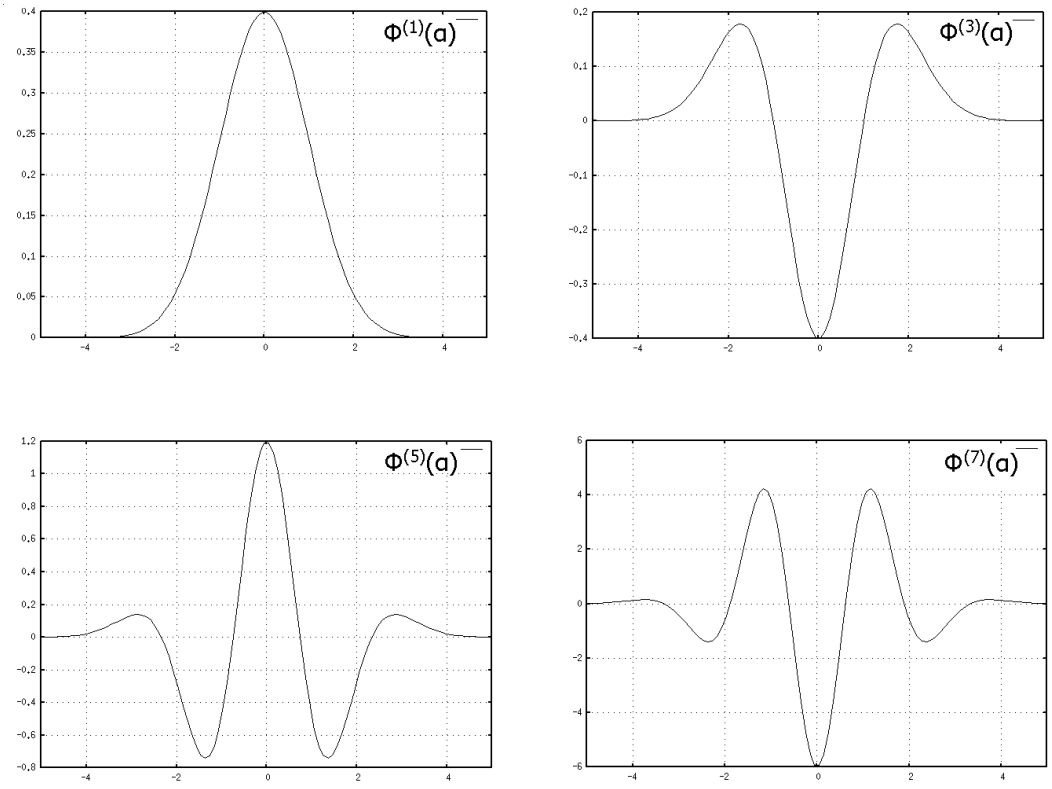

Figure A.2. Plots of $\Phi^{(2n+1)}(\alpha)$ against α for different values of n, showing global extremal at $\alpha = 0$

It is known that $(0.5 - |0.5 - \Phi(\alpha)|)$ and $\phi(\alpha)$ are monotonically decreasing functions of α. Consequently, the approximation errors in the above assumptions for any $|\alpha| > 7$ must be less than the approximation errors at $\alpha = 7$, which is less than 10^{-11}. Based on the desired level of accuracy, the boundary (here 7) can be increased or decreased.

For a uniformly sampled LUT with step-size p, an upper bound on the loss of accuracy in the computation of $\Phi(\alpha)$ with $2n + 1$ terms in the Taylor series expansion is given by the following.

$$|R_{2n}| \quad \leq \quad \frac{\left(\frac{p}{2}\right)^{2n+1}}{\sqrt{2\pi}\, 2^n \,(2n+1)} \tag{A.4}$$

Table A.1. Max residuals R_n in $\Phi(\alpha)$ for various LUT step-sizes ($p = \frac{7.0}{\#Entries}$)

# Entries	p	Max Residual				
		n = 0	n = 2	n = 4	n = 6	n = 8
35	0.2000	3.98×10^{-02}	6.65×10^{-05}	1.99×10^{-07}	7.12×10^{-10}	2.77×10^{-12}
70	0.1000	1.99×10^{-02}	8.31×10^{-06}	6.23×10^{-09}	5.56×10^{-12}	5.41×10^{-15}
350	0.0200	3.98×10^{-03}	6.65×10^{-08}	1.99×10^{-12}	7.12×10^{-17}	2.77×10^{-21}
700	0.0100	1.99×10^{-03}	8.31×10^{-09}	6.23×10^{-14}	5.56×10^{-19}	5.41×10^{-24}
1050	0.0067	1.33×10^{-03}	2.46×10^{-09}	8.21×10^{-15}	3.26×10^{-20}	1.41×10^{-25}
1400	0.0050	9.97×10^{-04}	1.04×10^{-09}	1.95×10^{-15}	4.34×10^{-21}	1.06×10^{-26}
2100	0.0033	6.65×10^{-04}	3.08×10^{-10}	2.56×10^{-16}	2.54×10^{-22}	2.75×10^{-28}
2800	0.0025	4.99×10^{-04}	1.30×10^{-10}	6.09×10^{-17}	3.40×10^{-23}	2.06×10^{-29}
3500	0.0020	3.98×10^{-04}	6.65×10^{-11}	1.99×10^{-17}	7.12×10^{-24}	2.77×10^{-30}
7000	0.0010	1.99×10^{-04}	8.31×10^{-12}	6.23×10^{-19}	5.56×10^{-26}	5.41×10^{-33}

Table A.1 presents an upper bound on the maximal residual for various number of LUT entries and number of finite Taylor series expansion terms $(n + 1)$. The choice of n in the Taylor expansion can be made dynamically based on the desired accuracy for a given LUT step-size. The table indicates that a very high level of accuracy can be reached with expanding extremely few terms in the Taylor expansion. In addition, it shows that obtaining a given accuracy result using naive table lookup ($n = 0$) without Taylor series expansion requires a much larger table size. For example, it takes $\approx 1.7 \times 10^9$ entries to compute $\Phi(\alpha)$ with the same accuracy as with a table having 700 entries and a Taylor series expansion with only 3 terms.

Though we propose to perform Taylor expansion for the interval $|\alpha| < 7$, the expansion can theoretically be performed for any α outside the interval to obtain results with higher precision. However, for practical purposes the approximation error at the boundary points is used as an accuracy estimate and the table size is determined accordingly. This method of computing $\Phi(\alpha)$ and $\phi(\alpha)$ facilitates dynamic runtime-accuracy trade-off options.

We consider our approach to computing $\Phi(\alpha)$ and $\phi(\alpha)$ while evaluating the *max* of two Gaussians for study. Two independent global parameters of variation are considered in

addition to a local independent component (similar to [**48**]) in the experiments. Cumulative parametric variations are constrained within 10% of their mean values.

We implement the proposed techniques for estimating $\Phi(\alpha)$ and $\phi(\alpha)$, and compute average run time for over 10^6 random *max* operations. For fair comparison, we compute the average run time for the same set of *max* operations in a different implementation which uses the *erf()* and *exp()* functions from the standard unix math library to compute the definite integral $\Phi(\alpha)$ and the exponential $\phi(\alpha)$, respectively. We do not consider the time to create the LUTs in our comparison because they need to be computed only once. Run time comparisons demonstrate that our proposed method is faster by 8.2 times on the average in evaluating the mean and the variance of the *max* of two Gaussians. To ensure the same level of accuracy, the implementation is performed for 3500 entries in the LUTs and Taylor expansion of 3 terms (accuracy $\geq 6.65 \times 10^{-11}$). A naive LUT would require $\approx 2 \times 10^{10}$ entries for the same accuracy.

References

[1] S. C. Wong, G. Y. Lee, and D. J. Ma, "Modeling of interconnect capacitance, delay and crosstalk in VLSI," in *IEEE Transactions on Semiconductor Manufacturing, Vol 13*, 2000, pp. 108–111.

[2] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit design," in *Proc. Custom Integrated and Circuits Conference*, 2000, pp. 201–204.

[3] Nanoscale Integration and Modeling Group at ASU, *Predictive technology models*. http://www.eas.asu.edu/˜ ptm/.

[4] S. I. Association, "National technology roadmap for semiconductors," 1999.

[5] R. Levy, D. Blaauw, G. Braca, A. Dasupta, A. Grinshpon, C. Oh, B. Orshav, S. Sirichotiyakul, and V. Zoloto, "Clarinet: A noise analysis tool for deep submicron design," in *Proc. of the Design Automation Conf.*, 2000, pp. –.

[6] K. L. Shepard and V. Narayanan, "Noise in deep submicron digital design," in *Proc. of the Design Automation Conf.*, 1996, pp. 524–531.

[7] T. Xiao and M. Marek-Sadowska, "Gate sizing to eliminate crosstalk induced timing violation," in *Proc. Intl. Conf. on Computer Design*, 2001, pp. 186–191.

[8] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitectures," in *Proc. of the Design Automation Conf.*, 2003, pp. 338–342.

[9] S. Nassif, "Modeling and analysis of manufacturing variations," in *Proc. Custom Integrated Circuits Conf.*, 2001.

[10] C. Visweswariah, "Death, taxes and failing chips," in *Proc. of the Design Automation Conf.*, 2003, pp. 343–347.

[11] D. Sinha, H. Zhou, and C. Chu, "Optimal gate sizing for coupling noise reduction," in *International Symposium on Physical Design*, 2004, pp. 176–181.

[12] D. Sinha and H. Zhou, "Gate sizing for crosstalk reduction under timing constraints by Lagrangian Relaxation," in *Proc. Intl. Conf. on Computer-Aided Design*, 2004, pp. 14–19.

[13] ——, "Gate-size optimization under timing constraints for coupling-noise reduction," in *IEEE Transactions on Computer-Aided Design, 24(6) June 2006.*

[14] ——, "Yield driven gate sizing for coupling-noise reduction under uncertainty," in *Proc. Asian and South Pacific Design Automation Conference*, 2005, pp. 192–197.

[15] D. Sinha, N. V. Shenoy, and H. Zhou, "Statistical gate sizing for timing yield optimization," in *Proc. Intl. Conf. on Computer-Aided Design*, 2005, pp. 1037–1041.

[16] ——, "Statistical timing yield optimization by gate sizing," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, accepted for publication.*

[17] D. Sinha and H. Zhou, "A unified framework for statistical timing analysis with coupling and multiple input switching," in *Proc. Intl. Conf. on Computer-Aided Design*, 2005, pp. 837–843.

[18] ——, "Statistical timing analysis with coupling," in *IEEE Transactions on Computer-Aided Design, accepted for publication.*

[19] D. Sinha, H. Zhou, and N. V. Shenoy, "Advances in computation of the maximum of a set of random variables," in *Proc. Intl. Symposium on Quality Electronic Design*, 2006, pp. 306–311.

[20] ——, "Computing tightness probabilities for statistical timing with dynamic runtime-accuracy trade-off options," in *Proc. Custom Integrated Circuits Conference, under review*, 2006.

[21] ——, "Advances in computation of the maximum of a set of Gaussian random variables," in *IEEE Transactions on Computer-Aided Design, under revision.*

[22] D. Sinha, D. Khalil, Y. Ismail, and H. Zhou, "A timing dependent power estimation framework considering coupling," in *Proc. Intl. Conf. on Computer-Aided Design, under review*, 2006.

[23] D. Sinha, J. Luo, S. Rajagopalan, S. Batterywala, N. V. Shenoy, and H. Zhou, "Impact of modern process technologies on the electrical parameters of interconnects," in *Proc. Intl. Conf. on Computer-Aided Design, under review*, 2006.

[24] H. Zhou and D. F. Wong, "Global routing with crosstalk constraints," in *Proc. of the Design Automation Conf.*, 1998, pp. 374–377.

[25] P. Saxena and C. L. Liu, "Crosstalk minimization using wire perturbations," in *Proc. of the Design Automation Conf.*, 1999, pp. 100–103.

[26] C. Alpert, A. Devgan, and S. T. Quay, "Buffer insertion for noise and delay optimization," in *Proc. of the Design Automation Conf.*, 1998, pp. 362–367.

[27] T. Xiao and M. Marek-Sadowska, "Crosstalk reduction by transistor sizing," in *Proc. Asian and South Pacific Design Automation Conference*, 1999, pp. 137–140.

[28] M. Hashimoto, M. Takahashi, and H. Onodera, "Crosstalk noise optimization by post-layout transistor sizing," in *International Symposium on Physical Design*, 2002, pp. 126–130.

[29] M. R. Becer, D. Blauuw, I. Algor, R. Panda, C. Oh, V. Zolotov, and I. N. Hajj, "Post-route gate sizing for crosstalk noise reduction," in *Proc. of the Design Automation Conf.*, 2003, pp. 954–957.

[30] H. Zhou, "Timing analysis with crosstalk is a fixpoint on a complete lattice," in *IEEE Transactions on Computer-Aided Design, September 2003*, pp. 1261–1269.

[31] C. Chen, C. Chu, and D. F. Wong, "Fast and exact simultaneous gate and wire sizing by Lagrangian Relaxation," in *IEEE Transactions on Computer-Aided Design, July 1999*.

[32] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows: Theory, algorithms, and application*. Prentice Hall, 1993.

[33] M. L. Fisher, "An applications oriented guide to Lagrangian Relaxation," in *Interfaces, vol 15, March/April 1985*, pp. 10–21.

[34] D. G. Luenberger, *Linear and nonlinear programming*. Addison Wesley Publishing Company, 1984.

[35] I. H. Jiang, Y. Chang, and J. Jou, "Crosstalk-driven interconnect optimization by simultaneous gate and wire sizing," in *IEEE Transactions on Computer-Aided Design, September 2000*, 2000, pp. 999–1010.

[36] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinatorial benchmark circuits," in *Proc. Intl. Symposium on Circuits and Systems*, 1985, pp. 695–698.

[37] B. A. Davey and H. A. Priestley, *Introduction to lattices and order*. Cambridge, 1990.

[38] P. Cousot and R. Cousot, "Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints," in *ACM Symposium on Principles of Programming Languages*, Los Angeles, CA, Jan. 1977, pp. 238–252.

[39] J. Cong, D. Z. Pang, and P. V. Srinivas, "Improved crosstalk modeling for noise constrained interconnect optimization," in *ACM Intl. Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, 2000.

[40] I. Lemke and G. Sander, "Visualization of compiler graphs," in *Design report, USAAR-1025-visual, ESPRIT Project 5399 Compare, Universität Saarlandes, FB 14 Informatik*, 1993, pp. D 3.12.1–1.

[41] D. Gries and F. B. Schneider, *A logical approach to discrete math.* Springer-Verlag New York, Inc., 1994.

[42] A. Devgan and C. Kashyap, "Block-based static timing analysis with uncertainty," in *Proc. Intl. Conf. on Computer-Aided Design*, 2003, pp. 607–614.

[43] A. Agarwal, D. Blaauw, and V. Zolotov, "Statistical timing analysis for intra-die process variations with spatial correlations," in *Proc. Intl. Conf. on Computer-Aided Design*, 2003, pp. 900–907.

[44] S. Bhardwaj, S. Vrudhula, and D. Blaauw, "TAU: Timing analysis under uncertainty," in *Proc. Intl. Conf. on Computer-Aided Design*, 2003, pp. 615–620.

[45] V. Khandelwal, A. Davoodi, and A. Srivastava, "Efficient statistical timing analysis through error budgeting," in *Proc. Intl. Conf. on Computer-Aided Design*, 2004, pp. 473–477.

[46] H. Chang and S. S. Sapatnekar, "Statistical timing analysis under spatial correlations," in *IEEE Transactions on Computer-Aided Design, accepted for publication*, 2004.

[47] J. Le, X. Li, and L. Pileggi, "STAC: Statistical timing with correlation," in *Proc. of the Design Automation Conf.*, 2004, pp. 343–348.

[48] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan, "First-order incremental block-based statistical timing analysis," in *Proc. of the Design Automation Conf.*, 2004, pp. 331–336.

[49] M. Chao, L. Wang, K. Cheng, and S. Kundu, "Static statistical timing analysis for latch-based pipeline designs," in *Proc. Intl. Conf. on Computer-Aided Design*, 2004, pp. 468–472.

[50] E. T. A. F. Jacobs and M. R. C. M. Berkelaar, "Gate sizing using a statistical delay model," in *Proc. DATE: Design Automation and Test in Europe*, 2000, pp. 283–291.

[51] M. Hashimoto and H. Onodera, "A performance optimization method by gate sizing using statistical static timing analysis," in *International Symposium on Physical Design*, 2000, pp. 111–116.

[52] S. Raj, S. B. K. Vrudhula, and J. Wang, "A methodology to improve timing yield in the presence of process variations," in *Proc. of the Design Automation Conf.*, 2004, pp. 448–453.

[53] M. Mani and M. Orshansky, "A new statistical optimization algorithm for gate sizing," in *Proc. Intl. Conf. on Computer Design*, 2004, pp. 272–277.

[54] S. H. Choi, B. C. Paul, and K. Roy, "Novel sizing algorithm for yield improvement under process variation in nanometer technology," in *Proc. of the Design Automation Conf.*, 2004, pp. 454–459.

[55] A. Agarwal, K. Chopra, and D. Blaauw, "Statistical timing based optimization using gate sizing," in *Proc. DATE: Design Automation and Test in Europe*, 2005, pp. 400–405.

[56] A. Agarwal, K. Chopra, D. Blaauw, and V. Zolotov, "Circuit optimization using statistical static timing analysis," in *Proc. of the Design Automation Conf.*, 2005, pp. 321–324.

[57] J. Singh, V. Nookala, Z. Luo, and S. Sapatnekar, "Robust gate sizing by geometric programming," in *Proc. of the Design Automation Conf.*, 2005, pp. 315–320.

[58] M. Guthaus, N. Venkateswaran, C. Visweswariah, and V. Zolotov, "Gate sizing using incremental parameterized statistical timing analysis," in *Proc. Intl. Conf. on Computer-Aided Design*, 2005, pp. 1029–1036.

[59] C. E. Clark, "The greatest of a finite set of random variables," in *Operations Research, Vol. 9, No. 2 (Mar - Apr)*, 1961, pp. 145–162.

[60] O. Coudert, "Gate sizing : A general purpose optimization approach," in *Proc. of the European Design and Test Conference*, 1996, pp. 214–218.

[61] O. Coudert, R. Haddad, and S. Manne, "New algorithms for gate sizing: A comparative study," in *Proc. of the Design Automation Conf.*, 1996, pp. 734–739.

[62] A. Srivastava, S. Shah, K. Agarwal, D. Sylvester, D. Blaauw, and S. Director, "Accurate and efficient gate-level parametric yield estimation considering correlated variations in leakage power and performance," in *Proc. of the Design Automation Conf.*, 2005, pp. 535–540.

[63] H. Chang, V. Zolotov, S. Narayan, and C. Visweswariah, "Parameterized block-based statistical timing analysis with non-Gaussian parameters, non-linear delay functions," in *Proc. of the Design Automation Conf.*, 2005, pp. 71–76.

[64] E. W. Weisstein, *Taylor series - Wolfram web resource.* http://mathworld.wolfram.com/TaylorSeries.html.

[65] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions with formulas, graphs, and mathematical tables.* Dover, New York, 1972.

[66] E. W. Weisstein, *Hermite polynomial - Wolfram web resource.* http://mathworld.wolfram.com/HermitePolynomial.html.

[67] D. E. Knuth, *Fundamental algorithms*, 2nd ed., ser. The art of computer programming. Addison-Wesley, 1973, vol. 1.

[68] X. Li, J. Le, P. Gopalakrishnan, and L. Pileggi, "Asymptotic probability extraction for non-normal distributions of circuit performance," in *Proc. Intl. Conf. on Computer-Aided Design*, 2004, pp. 2–9.

[69] R. Arunachalam, K. Rajagopal, and L. T. Pileggi, "Taco: Timing analysis with coupling," in *Proc. of the Design Automation Conf.*, 2000, pp. 266–269.

[70] A. Agarwal, F. Dartu, and D. Blaauw, "Statistical gate delay model considering multiple input switching," in *Proc. of the Design Automation Conf.*, 2004, pp. 658–663.

[71] H. Chang and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single PERT-like traversal," in *Proc. Intl. Conf. on Computer-Aided Design*, 2003, pp. 621–625.

[72] A. B. Kahng, S. Muddu, and E. Sarto, "On switch factor based analysis of coupled RC interconnects," in *Proc. of the Design Automation Conf.*, 2000, pp. 79–84.

[73] P. Chen, D. A. Kirkpatrick, and K. Keutzer, "Miller factor for gate-level coupling delay calculation," in *Proc. Intl. Conf. on Computer-Aided Design*, 2000, pp. 68–74.

[74] ——, "Switching window computation for static timing analysis in presence of crosstalk noise," in *Proc. Intl. Conf. on Computer-Aided Design*, 2000, pp. 331–337.

[75] P. D. Gross, R. Arunachalam, K. Rajagopal, and L. T. Pileggi, "Determination of worst-case aggressor alignment for delay calculation," in *Proc. Intl. Conf. on Computer-Aided Design*, 1998, pp. 212–219.

[76] F. Dartu and L. T. Pileggi, "Calculating worst-case gate delays due to dominant capacitance coupling," in *Proc. of the Design Automation Conf.*, 1997, pp. 46–51.

[77] S. S. Sapatnekar, "A timing model incorporating the effect of crosstalk on delay and its application to optimal channel routing," *IEEE Transactions on Computer Aided Design*, 2000.

[78] S. Hassoun, C. Cromer, and E. Calvillo-Gámez, "Static timing analysis for level-clocked circuits in the presence of crosstalk," in *IEEE Transactions on Computer Aided Design*, 2003, pp. 1270–1277.

[79] V. Chandramouli and K. A. Sakallah, "Modeling the effects of input transition on gate propagation delay and transition time," in *Proc. of the Design Automation Conf.*, 1996, pp. 617–621.

[80] F. Dartu, K. Killpack, C. Amin, and N. Menezes, "Evaluating the factors influencing timing accuracy," in *ACM Intl. Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, 2005.

[81] A. Agarwal, D. Blaauw, V. Zolotov, and S. Vrudhula, "Statistical timing analysis using bounds and selective enumeration," in *ACM Intl. Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, 2002, pp. 29–36.

[82] A. Agarwal, K. Chopra, D. Blaauw, and V. Zolotov, "Statistical timing based optimization using gate sizing," in *ACM Intl. Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, 2005.

[83] B. Wu, J. Zhu, and F. N. Najm, "Dynamic range estimation for nonlinear systems," in *Proc. Intl. Conf. on Computer-Aided Design*, 2004, pp. 660–667.

[84] G. Grimmett and D. Stirzaker, *Probability and random processes*. Oxford University Press, 2002.

[85] C. F. Fang, R. A. Rutenbar, and T. Chen, "Fast, accurate static analysis for fixed-point finite-precision effects in DSP designs," in *Proc. Intl. Conf. on Computer-Aided Design*, 2003, pp. 275–282.

[86] D. W. Dobberpuhl *et al.*, "A 200Mhz, 64bit dual-issue CMOS microprocessor," in *IEEE Journal of Solid State Circuits, Vol 27(11)*, 1992, pp. 1555–1564.

[87] D. Liu and C. Svensson, "Power consumption estimation in CMOS VLSI chips," in *IEEE Journal of Solid State Circuits, Vol 29*, 1994, pp. 663–670.

[88] R. Mehra, L. M. Guerra, and J. M. Rabaey, "A partitioning scheme for optimizing interconnect power," in *IEEE Journal of Solid State Circuits, Vol 32*, 1997, pp. 433–443.

[89] E. D. Man and M. Schobinger, "Power dissipation in the clock system of highly pipelined ULSI CMOS circuits," in *Proc. International Workshop on Low Power Design*, 1994, pp. 133–138.

[90] J. Pangjun and S. Sapatnekar, "Low power clock distribution using multiple voltages and reduced swings," in *IEEE Transactions on Very Large Scale Integration Systems, Vol 10*, 2002, pp. 309–318.

[91] M. Ghoneima and Y. Ismail, "Effect of relative delay on the dissipated energy in coupled interconnects," in *Proc. Intl. Symposium on Circuits and Systems*, 2004, pp. 525–528.

[92] S. M. Kang, "Accurate simulation of power dissipation in VLSI circuits," in *IEEE Journal of Solid State Circuits, Vol 21(5)*, 1986, pp. 889–891.

[93] M. A. Cirit, "Estimating dynamic power consumption of CMOS circuits," in *Proc. Intl. Conf. on Computer-Aided Design*, 1987, pp. 534–537.

[94] F. Najm, R. Burch, P. Yang, and I. Hajj, "Probabilistic simulation for reliability analysis of CMOS VLSI circuits," in *IEEE Transactions on Computer Aided Design*, 1990, pp. 439–450.

[95] G. I. Stamoulis and I. N. Hajj, "Improved techniques for probabilistic simulation including signal correlation effects," in *Proc. of the Design Automation Conf.*, 1993, pp. 379–383.

[96] C. Y. Tsui, M. Pedram, and A. M. Despain, "Efficient estimation of dynamic power consumption under a real delay model," in *Proc. Intl. Conf. on Computer-Aided Design*, 1993, pp. 224–228.

[97] T. Uchino and J. Cong, "An interconnect energy model considering coupling effects," in *Proc. of the Design Automation Conf.*, 2001, pp. 555–558.

[98] A. Ghosh, S. Devdas, K. Keutzer, and J. White, "Estimation of average switching activity in combinational and sequential circuits," in *Proc. of the Design Automation Conf.*, 1992, pp. 253–259.

[99] M. Xakellis and F. Najm, "Statistical estimation of the switching activity in digital circuits," in *Proc. of the Design Automation Conf.*, 1994, pp. 728–733.

[100] P. Chen, Y. Kukimoto, C. C. Teng, and K. Keutzer, "On convergence of switching window computation in presence of crosstalk noise," in *International Symposium on Physical Design*, 2002, pp. 84–89.

[101] P. Gupta and A. B. Kahng, "Quantifying error in dynamic power estimation of CMOS circuits," in *Proc. Intl. Symposium on Quality Electronic Design*, 2003, pp. 273–278.

[102] P. Chen, Y. Kukimoto, and K. Keutzer, "Refining switching window by time slots for crosstalk noise calculation," in *Proc. Intl. Conf. on Computer-Aided Design*, 2002, pp. 583–586.

[103] J. Luo, Q. Su, C. Chiang, and J. Kawa, "A layout dependent full-chip copper electroplating topography model," in *Proc. Intl. Conf. on Computer-Aided Design*, 2005, pp. 133–140.

[104] J. Luo and D. A. Dornfeld, *Integrated modeling of chemical mechanical planarization for sub-micron fabrication: from particle scale to feature, die and wafer scales.* Springer-Verlag, Berlin, Germany, New York, USA, 2004.

[105] R. Tian, D. F. Wong, and R. Boone, "Model-based dummy feature placement for oxide chemical mechanical polishing manufacturability," in *IEEE Transactions on Computer-Aided Design, vol. 20*, 2001, pp. 902–910.

[106] P. Gupta, A. B. Khang, O. S. Nakagawa, and K. Samadi, "Closing the loop in interconnect analyses and optimization: CMP, fill, lithography and timing," in *IEEE VLSI/UVLSI Multilevel Interconnection Conference*, 2005.

[107] L. He, A. B. Khang, K. H. Tam, and J. Xiong, "Variability-driven considerations in the design of integrated-circuit global interconnects," in *IEEE VLSI Multilevel Interconnection Conference*, 2004, pp. 214–221.

[108] R. Tian, X. Tang, and M. D. F. Wong, "Dummy feature placement for chemical-mechanical polishing uniformity in a shallow-trench isolation process," in *IEEE Transactions on Computer-Aided Design, vol. 21*, 2002, pp. 63–71.

[109] V. Mehrotra, "Modeling the effects of systematic process variation on circuit performance," in *PhD thesis, EECS, MIT*, 2001.

[110] *Opencores.* http://www.opencores.com.

[111] X. Li, J. Le, M. Celik, and L. Pileggi, "Defining statistical sensitivity for timing optimization of logic circuits with large-scale process and environmental variations," in *Proc. Intl. Conf. on Computer-Aided Design*, 2005, pp. 844–851.

Vita

Debjit Sinha has been a graduate student in the Department of Electrical Engineering and Computer Science at Northwestern University, USA since 2002. He received the Bachelor of Technology (Honors) degree in Electrical Engineering from the Indian Institute of Technology at Kharagpur, in 2001.

Between 2001 and 2002, he worked on a research and development project titled *Development tools for Compact RISC processors* for National Semiconductor Inc., USA. During his doctoral research, he spent the summers of 2004 and 2005 as an intern at the Advanced Technology Group of Synopsys Inc., USA. At Synopsys, he worked on projects titled *Statistical timing yield optimization* and *Impact of modern process technologies on interconnect parasitics*.

Debjit is the author of more than 10 papers in leading IEEE journals and refereed ACM/IEEE conference proceedings. His accolades include the prestigious Walter P. Murphy fellowship from Northwestern University, an honorable Teaching assistant of the year award, and a department nomination for the Intel foundation fellowship. Following his doctoral, he will be joining IBM Microelectronics, USA for research and development of statistical timing methodologies. His research interests include computer aided design for VLSI circuits and algorithm design.

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1]The macros used in formatting this dissertation are based on those written by Miguel A. Lerma, Mathematics Department of Northwestern University, which have been further modified by the author to accommodate electronic dissertation formatting guidelines.