ABSTRACT

Title of dissertation:	Optimization Schemes for Variability-Driven VLSI Design Automation Azadeh Davoodi, Doctor of Philosophy, 2006
Dissertation directed by:	Professor Ankur Srivastava Department of Electrical and Computer Engineering

Today's IC design is facing several challenges due to increasing circuit complexity and decreasing feature size, as it pushes to extend Moore's law into nano-scale dimensions. Apart from the challenges that arise directly as a result of feature scaling (e.g., increasing leakage power, reliability issues), imperfections in the manufacturing process have recently turned into a major design hurdle, due to the variations they cause in the device and interconnect parameters from their target values. From an IC design automation perspective, a shift in paradigm, from deterministic to probabilistic, is needed to handle the unpredictable nature of these fabrication variations.

In such a probabilistic paradigm, the varying circuit parameters such as leakage power or delay should be accurately modeled, and their correlations due to common sources of variations or physical location on the chip should be well captured. Moreover, variability-driven (probabilistic) design automation needs to efficiently generate a high quality solution.

A particular challenge in variability-driven design automation is to define optimality measures among candidate solutions, which allow for inferior solutions to be removed from the solution space thus reducing the run-time complexity. In this dissertation, the superiority probability is introduced as such an optimality measure, and two methods are proposed to compute this probability: an accurate Conditional Monte Carlo simulation method, and an efficient moment-matching approximation method. The effectiveness of using the superiority probability is shown in the context of two important design automation applications: 1) the buffer insertion problem, 2) the dual-V_{th} leakage optimization problem.

Another important task in variability-driven design automation is to develop optimization techniques that can provably generate the optimal solution in an efficient way. In this dissertation, the application of the gate sizing problem is explored to optimally reduce the loss due to fabrication variations in the presence of a timing constraint. The presented formulation, in contrast with the existing variability-driven approaches which are all based on heuristics, is provably optimal. Moreover, unlike existing approaches, it is independent of any assumption on the source and nature of variations.

Optimization Schemes for Variability-Driven VLSI Design Automation

by

Azadeh Davoodi

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2006

Advisory Commmittee:

Professor Ankur Srivastava, Chair/Advisor
Professor Joseph F. JaJa
Professor Bruce Jacob
Professor Andre L. Tits
Professor Robert W. Newcomb
Professor Jian-Guo Liu

© Copyright by

Azadeh Davoodi

2006

DEDICATION

to my family: my parents, my brother, and my husband,

and to the memory of my grandmother

who will always be my role model.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor Professor Ankur Srivastava for his continuous guidance and support. Working with him has been an invaluable experience. He has been a continuous source of motivation for me, and I want to sincerely thank him for all I have achieved.

I would also like to thank the members of my dissertation committee, Professors Joseph JaJa, Bruce Jacob, Andre Tits, Robert Newcomb, and Jian-Guo Liu, for their time and feedback.

I am much indebted to Professor Joseph JaJa for believing in me; his strong support has been crucial in achieving my goals.

I am sincerely grateful to Professor Andre Tits for his feedback, advice, and continuous encouragement.

I would also like to thank Professor Newcomb who has always been supportive in my decisions during the past six years. I will forever cherish his warm welcome in the airport when I first landed in this country.

I am also thankful to the Graduate School at University of Maryland for honoring me with the dissertation fellowship, as well as the members of the Electrical and Computer Engineering Department for nominating me for this award.

Finally, I would like to thank my family: my husband, my parents and my brother for standing by me and for being a constant source of energy, and morale.

Publications:

Journal:

1. Davoodi A., Khandelwal V., Srivastava A., "Probabilistic evaluation of solutions in variability-driven optimization", To Appear in IEEE Transactions on CAD.

2. Davoodi A., Srivastava A., "Effective techniques for the generalized low power binding problem", Proceedings of ACM Transactions on Design Automation of Electronic Systems, Vol. 11, No. 1, pp. 52-69, January 2006.

3. Davoodi A., Srivastava A., "Power-driven simultaneous resource binding and floorplanning: a probabilistic approach", Proceedings of IEEE Transactions on VLSI Systems, Vol. 13, No. 8, pp. 934-942, August 2005.

4. Davoodi A., Srivastava A., "Voltage scheduling under unpredictabilities: a risk management paradigm", Proceedings of ACM Transactions on Design Automation of Electronic Systems, Vol. 10, No. 2, pp.354-368, April 2005.

5. Davoodi A., Khandelwal V., Srivastava A., "Empirical models for net-length probability distribution and applications", Proceedings of IEEE Transactions on VLSI Systems, Vol. 12, No. 10, pp. 1066-1075, October 2004.

6. Wong J., Davoodi A., Khandelwal V., Srivastava A., Potkonjak M., "A statistical methodology for wire-length prediction", To Appear in IEEE Transactions on CAD, 2006.

7. Khandelwal V. and Davoodi A., Srivastava A., "Simultaneous Vt selection and assignment for leakage optimization", Proceedings of IEEE Transactions on VLSI Systems, Vol. 13, No. 6, pp. 762-765, June 2005.

8. Wang L., French M., Davoodi A., Agarwal D., "FPGA dynamic power minimization through placement and routing constraints", Proceedings of Eurasip Journal on Embedded Systems, April 2006.

Conference and Workshop:

1. Davoodi A., Srivastava A., "Variability-driven gate sizing for binning yield optimization", To appear in Design Automation Conference, July 2006.

2. Davoodi A., Srivastava A., "Probabilistic evaluation of solutions in variability-driven optimization", Proceedings of International Symposium on Physical Design, April 2006.

3. Davoodi A., Srivastava A., "Variability-driven buffer insertion considering correlations", Proceedings of International Conference on Computer Design, October 2005.

4. Davoodi A., Srivastava A., "Probabilistic dual-Vth leakage optimization under variability", Proceedings of International Symposium on Low Power Electronics and Design, August 2005.

5. Davoodi A., Srivastava A., “Wake-up protocols for controlling current surges in MTCMOS-based technology”, Proceedings of Asia South Pacific Design Automation Conference, January 2005.

6. Davoodi A., Srivastava A., “Simultaneous floorplanning and binding: a probabilistic approach”, Proceedings of Asia South Pacific Design Automation Conference, January 2005.

7. Davoodi A., Khandelwal V., Srivastava A., “Variability inspired implementation selection problem”, Proceedings of International Conference on Computer Aided Design, November 2004.

8. Wong J., Davoodi A., Khandelwal V., Srivastava A., Potkonjak M., “Wirelength prediction using statistical techniques”, Proceedings of International Conference on Computer Aided Design, November 2004.

9. Khandelwal V., Davoodi A., Srivastava A., “Efficient statistical timing analysis through error budgeting”, Proceedings of International Conference on Computer Aided Design, November 2004.

10. Davoodi A., Khandelwal V., Srivastava A., “High level techniques for power-grid noise immunity”, Proceedings of Great Lakes Symposium on VLSI, April 2004.

11. Khandelwal V., Davoodi A., Nanavati A., Srivastava A., “A Probabilistic Approach to Buffer Insertion”, Proceedings of International Conference on Computer Aided Design, November 2003.

12. Davoodi A., Srivastava A., “Voltage scheduling under unpredictabilities: a risk management paradigm”, Proceedings of International Symposium on Low Power Electronics and Design, August 2003.

13. Davoodi A., Srivastava A., “Effective graph theoretic techniques for the generalized low power binding problem”, Proceedings of International Symposium on Low Power Electronics and Design, August 2003.

14. Davoodi A., Srivastava A., “Variability driven gate sizing for binning yield optimization”, Proceedings of International Workshop on Logic and Synthesis, June 2006.

15. Davoodi A., Srivastava A., “Variability driven buffer insertion considering correlations”, Proceedings of International Workshop on Logic and Synthesis, June 2005.

16. Davoodi A., Srivastava A., “Efficient stochastic pruning for variability-driven dual-Vth leakage optimization”, Proceedings of International Workshop on Logic and Synthesis, June 2005.

17. Davoodi A., Srivastava A., “Simultaneous floorplanning and binding: a probabilistic approach”, Proceedings of International Workshop on Logic and Synthesis, June 2004.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Motivation and Background

1.1 Existing Issues in Deep Sub Micron Chip Design

Today's IC design is driven by the demand for having more functionalities on a single chip. This includes having multiple processor cores, caches, multimedia engines, etc. embedded all on the same chip.

To realize this demand, the existing thrust has been to follow up with Moore's Law to double the number of transistors on a chip every generation (typically every 18 months). This is accompanied by the increasing fabrication cost, currently in the range of $1B, and the small timing window to have the product to market.

Following Moore's Law translates into decreasing the minimum (fabricated) feature size that is currently around 90nm dimensions. This decreasing or scaling into Deep Sub Micron (DSM) feature sizes introduces many design challenges that didn't exist before, or amplifies many of the effects that used to be ignored. As an example interconnect delay used to be ignored in many stages of the design over the gate delay, while now it is the dominant component.

Some of these DSM issues are as follows:

- Exponential increase in leakage power beyond the 250nm feature size, and its dominance over dynamic power as scaling continues [6].

- Thermal issues and hot spots on the chip due to increasing transistor density, which cause challenges in cooling and packaging of ICs [7].

- Deterioration in reliability due to increase in various types of noise (e.g., cross-coupling noise, power-grid noise), soft errors due to cosmic radiation and continuous scaling of supply voltage, and fabrication defects [53].

- Dominance of the interconnect delay over gate delay beyond the 250nm feature size [31]. The interconnects on the chip decrease in size as scaling continuous. This shrinking in size increases their resistance (R). In addition because interconnects are placed closer to each other on the chip, their cross-coupling capacitance (C) also increases. Therefore the interconnect delay which is proportional to both R and C increases [6].

1.2 Process Variations: The New Challenge in Sub-90nm Design

The outlined DSM issues appeared from 250nm CMOS technology nodes. However as scaling continues beyond 90nm dimensions, *process variations* appear as a new, and yet very significant challenge.

Process variations refer to those variations caused due to the imperfections in different steps of the IC manufacturing process [48]: These could be due to the limited resolution of the photolithographic stage within the fabrication process which results in variations in the width and length of transistors on the chip. It could also be from non-uniform conditions during the diffusion stage in which impurities are introduced.

The stated manufacturing imperfections cause variations in the electrical properties of the transistors and interconnects on the chip from their designed values. These could be caused by variations in the geometries of the transistors on the chip (e.g., effective channel length, oxide thickness), or due to random dopant fluctuations (affecting the threshold voltage of the transistors).

Some of the variations in the device and interconnect parameters are uncorrelated to each other. For example, variations in the channel length of MOSFET transistors are independent of variations in their threshold voltage because they are caused by different stages of the fabrication process [48]. Variations at transistor-level, although are not very significant but result in significant chip-level variations. The measurable effect of the process variations may be a substantial deviation of the circuit behavior from its nominal response, which could be either positive or negative. It has been shown that for 1000 samples representing the same design in 0.13μ technology, up to 30% variation in frequency and 20X variation in leakage power exist [7].

In high performance applications such as microprocessor design, the goal is to meet a target frequency. The significant degree of variation in frequency makes the decision making very difficult for the designer. A common approach has been based on a worst-case estimate of all parameters in the presence of variations. Even though this approach is safe but it is too pessimistic, and underestimates the true potentials of a design [16, 17].

1.2.1 Components

Process variations could be identified at different levels depending on various stages of the manufacturing process: wafer-to-wafer, die-to-die, and within-die [46]. As an example radial variation seen at each wafer is due to the spin stage of the fabrication process. Within-die variations on the other hand is position dependent: potential location of a component on various parts on the chip determines the degree of variations in its parameters at a within-die level.

Process variations have an overall unpredictable nature because of not having enough control over different steps of fabrication process for sub-90nm technology nodes. These imperfections in the fabrication process translate into uncertain behavior of transistors and interconnects on the chip. From a higher level of abstraction different performance metrics of a design (such as frequency and power) will be uncertain and effectively become random variables [7].

1.2.2 Handling Process Variations

Process variations could be handled at different stages. One could reduce the *source* of variations by focusing on perfecting the fabrication process, or by introducing new transistor structures that could more accurately be fabricated.

As an example FinFET is a new transistor structure that could be realized with a gate-length of about 10 nanometer [29]. Another example is the trigate transistor from Intel which has also been successfully manufactured in nanometer scale featuring higher speed and lower power [22].

To handle process variations, one could also reduce the *effects* of variation, either pre- or post- fabrication:

The pre-fabrication techniques are those "design-time" techniques that could potentially reduce variations. These design-time techniques could be at different stages of the hierarchical design framework based on their levels of abstraction.

These include techniques at micro-architectural level [24], circuit-level [61], or techniques that use different design styles such as asynchronous (clock-less) designs which are inherently more tolerant to variations [58]. These would be integrated as new techniques within the design-aid framework and tools. On the other hand the existing design-aid framework could be modified to consider process variations [16, 17]. This could be by including process variations during different stages of design analysis and optimization.

The effect of process variations could also be reduced post-fabrication, using tuning techniques such as adaptive body biasing [62] and adaptive supply voltage [63]. In these techniques controlling of the body bias / supply voltage of different transistors could be done after fabrication which affects the variation in parameters of transistors. For example changing the voltage of the body (substrate) of a transistor impacts its threshold voltage, which results in speed / power tradeoffs which directly relate to sensitivity to process variations in that transistor [62].

Tunable elements could also exist for post-fabrication tuning. As an example [60] proposes a tunable buffer that could be programmed in order to change its effective capacitive loading, hence its delay. Another example is [69] which proposes an automatically tunable delay element for domino logic.

Given the broad levels of handling process variations, in this work I have focused on reducing the effects of variations, pre-fabrication, by considering variations within the existing design-automation tools. This means the existing computer aided design framework will be modified to consider process variations. This could be in terms of adding bounds on the maximum variation as a design constraint, or trying to minimize the effects of variations as a design objective, as will be explained in detail in the following chapters. Such a design-automation framework will hence be called a variability-driven framework through-out this dissertation.

Please note that process variations are considered to be static. Dynamic variations also exist, which refer to fluctuations in supply voltage or temperature as a function of the input vector [7, 46], which are not considered in this dissertation.

1.3 Contributions

The contributions of this work are listed as follows:

- Proposing the “superiority probability” as a metric for comparison of potential solutions in the presence of process variations.
- Proposing and evaluating two methods to compute the superiority probability: 1) an accurate Conditional Monte Carlo simulation method, 2) an efficient moment matching method.
- Evaluating the effectiveness of using the superiority probability in two Design Automation applications in the presence of process variations: 1) buffer insertion, 2) dual-V_{th} assignment for leakage optimization.

- Formulating the gate sizing problem in the presence of process variations as a mathematical program with the following properties:
  - The formulation is provably convex, which means it can efficiently be solved to obtain the optimal solution.
  - The formulation is not restricted by the sources and the models of process variations.
  - To solve the convex formulation, “any” statistical timing analysis can be used.
  - The optimization objective is to minimize the loss associated with violating a frequency constraint due to process variations, which is applicable in speed-binning of microprocessors.

Chapter 2

Variability-Driven Design Framework: A Futuristic Perspective

In this chapter I will describe a design-automation framework that could account for the effects of process variations. First the existing hierarchical design-automation flow is explained. Then modification of the existing flow is discussed to consider process variations in a "deterministic" fashion. Finally my focus would be on a "probabilistic" extension of the design-automation framework in which all the varying circuit parameters are modeled as random variables.

2.1 Hierarchical Design-Automation Flow

The existing design-automation flow is composed of different optimization algorithms that automate the process of transferring an abstract specification of a design, at Register Transfer Level (RTL), to a low-level layout description that could be sent to fabrication. These steps are shown in Figure 2.1 and are as follows:

Initially the design is described in a Hardware-Description Language (HDL). The technology library is the database containing the data that models the pre-designed cells in the underlying process technology for the logic synthesis and physical design tools. User constraints convey limitations regarding the speed, area and power of the design. Logic synthesis transforms the HDL description into a graph called a netlist. Netlist is a graph in which each vertex represents a cell in the

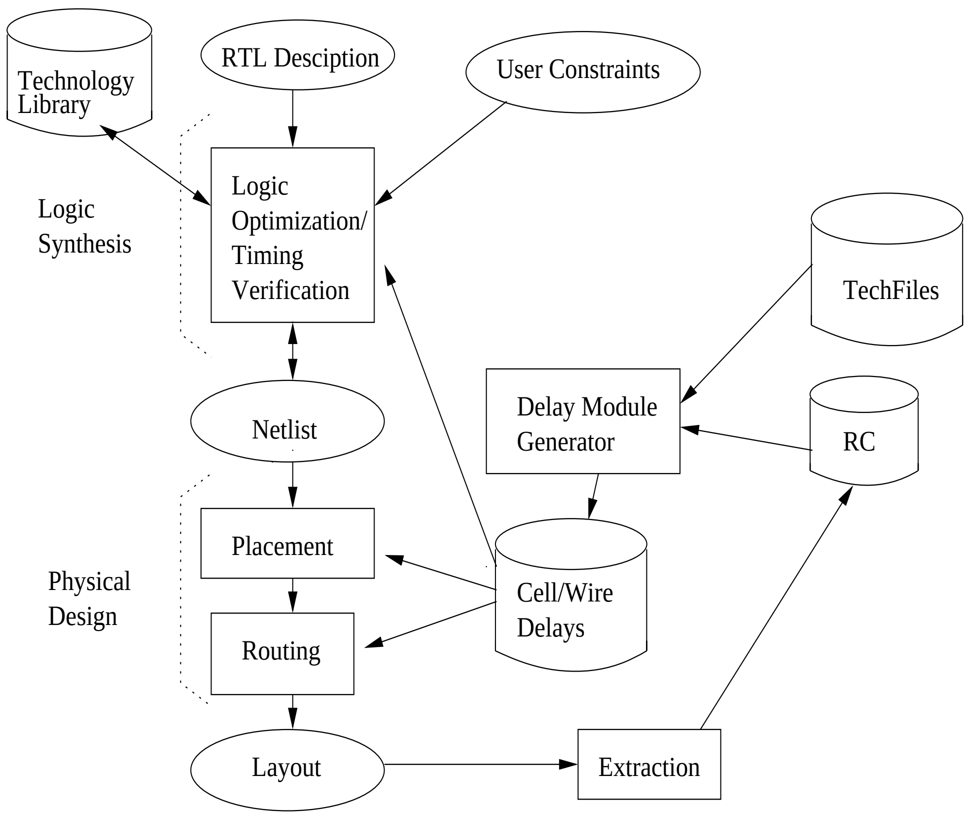

Figure 2.1: The hierarchical computer-aided design flow.

technology library and each edge represents a wired connection between the cells.

Logic synthesis optimizes the circuit according to user constraints and ensures that design rules are met. Typical tasks in this step include logic minimization, structuring, mapping, gate sizing and buffering [20].

Physical design is the process by which the synthesized netlist is transformed into a layout, which is used to fabricate the integrated circuit. Information contained in technology library and user constraints ensure that the output of physical design could be fabricated in the designated semiconductor process. User constraints restrict the location of pads and signals, the area resources available for implementation and the timing behavior. Typical steps in this category are cell placement, global and detailed routing, sizing and clock/power distribution [54].

Both logic synthesis and physical design tremendously suffer from insufficient parameter estimations. In a typical design flow, one popular method of dealing with estimation uncertainty is that after the layout generation if the constraints are not satisfied, the synthesized netlist is back-annotated with more accurate values of parameters through parasitic extraction. This is illustrated in Figure 2.1. These could get incorporated to generate more accurate delay models which will be fed back to logic synthesis and physical design steps resulting in another round of optimization. For designs with performance priorities several iterations between synthesis and physical design are required to converge to a desired solution.

A *variability-driven design framework* considers process variations and their effects within different stages of the flow explained above. This is done by modifying or extending the optimization algorithms involved in these stages to become variability-driven. This could be done in either two ways: 1) modifying the existing algorithms to consider process variations deterministically, 2) considering process variations as random variables within the algorithms, which changes their inherent deterministic nature. Next I will elaborate on these two alternatives to consider process variations, and in particular focus on the probabilistic approach.

2.2 Modification of Existing Deterministic Approach

To consider process variations within the existing design-automation framework the first natural choice would be to extend the existing design-automation algorithms to consider variations, without really changing the algorithms. These algorithms are deterministic, meaning they are based on fixed estimates of parameters.

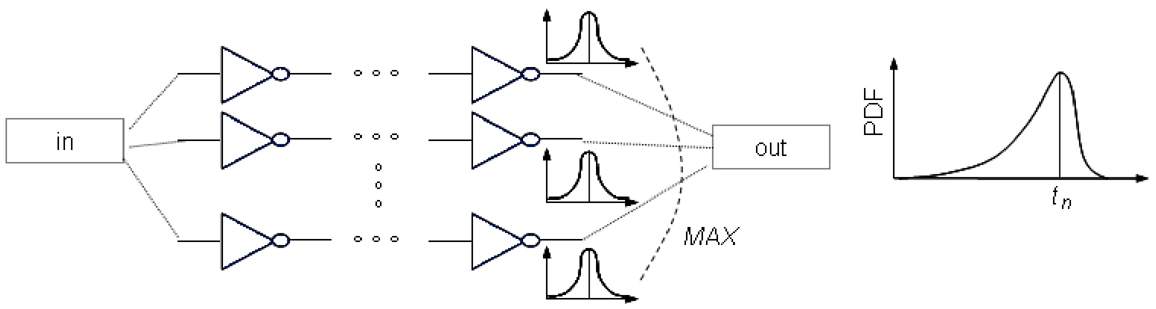

Figure 2.2: Deterministic approach can not accurately compute operations such as "max" in the presence of variations.

To consider process variations, these deterministic estimates could be replaced by new deterministic estimates that account for variations. This could be a worst-case or an average-case estimate [16, 17], or an empirical one. Given these new estimates the same algorithm will be used in a variability-driven deterministic approach.

In these deterministic approaches it would be difficult to model possible correlations that exist among the varying circuit parameters. In addition, a good deterministic analysis requires accurate computation of different operations that might be involved, while considering variations.

As an example to find the circuit delay, one needs to compute the "maximum" of the delays of all the paths in the circuit. In the presence of process variations, the delay of each path in the circuit is a random variable that has a corresponding Probability Density Function (*pdf*) as shown in Figure 2.2. When applying the max operation to these *pdf*s, the resulting *pdf* representing the outcome of the max will be stretched towards higher delay values because of the way max operates on random variables. To estimate the circuit delay deterministically, the best estimate should be the highest probability delay value in the output *pdf* (t_n in Figure 2.2).

However making that estimate in the deterministic approach is not easy. This is because a good deterministic estimate would need the information embedded in the *pdf*s of individual path delays which is ignored in a deterministic approach.

This variability-driven deterministic extension could become too optimistic (e.g., in an average-case estimate), or too pessimistic (e.g., in a worst-case estimate) [16, 17]. The most effective of deterministic approaches would be the one in which variations are incorporated based on an empirical factor [44]. However it is usually very difficult to build empirical models to capture sub-90nm process variations because of inaccessibility to accurate statistics. Next an alternative approach is proposed in which all the varying parameters are represented as random variables.

2.3 Probabilistic Design Framework

In a probabilistic design framework, all the varying circuit parameters are modeled as random variables. These random variables may or may not be correlated to each other. At the lowest level, variations happen in the geometries of interconnects (e.g., length, width) and of transistors (e.g., effective channel length, oxide thickness), transistor doping (as well as other possibilities). These low-level factors are modeled as random variables that in general have a Joint-Probability Density Function (*jpdf*) [18]. Given the *jpdf* of these low-level factors, one could model the effects of variations at higher levels by finding the *pdf* of performance metrics of a design such as its frequency and power, as these performance metrics are ultimately a function of the low-level factors [11, 37].

The modeling of *jpdf* should be done to capture different ways that varying circuit parameters might be correlated to each other. These are the followings:

1) *Global correlations* exist because of having a common manufacturing process that would affect all the components on a chip similarly. Example is [65] that models the circuit delay assuming common (global) variables representing the delays of different components in the circuit.

2) *Spatial correlations* are due to the locations of different components on the chip. Those components that are physically closer are more likely to have similar variations in their parameters, as they are more likely to be affected in a similar way by the imperfections in the fabrication process at a within-die level. One example is [11] that models variations in leakage power considering spatial correlations.

3) *Path-based correlations* are due to the topology of the design. If we consider a design as a circuit with different paths, some of these paths might be partially overlapping with each other. The parameters of these paths such as their individual delays will be correlated because of these overlaps. One example of path-based correlations is in circuit topologies that contain many "reconvergent-fanout" gates. These are gates with multiple outputs, in which different output-paths converge to each other later on in the circuit [21].

In a probabilistic design framework, variation-aware models are random variables that should be expressed in a way to capture the above-mentioned sources of correlations. The conventional algorithms should then be changed to incorporate these models in a probabilistic framework. Next I will discuss the details of such a probabilistic framework, its properties and challenges.

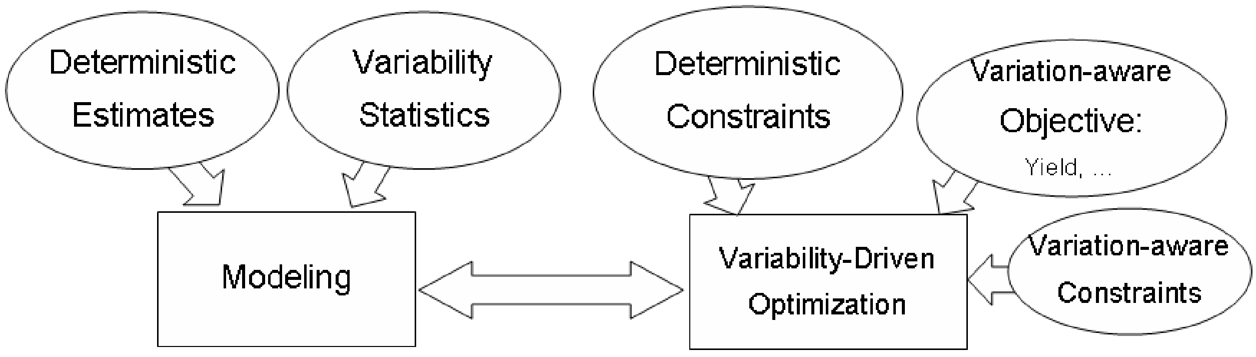
Deterministic Estimates
Variability Statistics
Deterministic Constraints
Variation-aware Objective: Yield, ...
Modeling
Variability-Driven Optimization
Variation-aware Constraints

Figure 2.3: Components of a probabilistic framework.

2.3.1 Components: Modeling and Optimization

A probabilistic design framework has the following two components:

1) Modeling

Modeling is one essential component of any design framework. Models could be used to evaluate potential solutions in a design. Accuracy of the models determine the effectiveness of the design techniques to find the highest-quality solution under constraints for performance, power, area, etc. [11, 36, 37].

In the presence of process variations, the existing models should be modified to capture variations. As Figure 2.3 shows statistics on the effects of process variations could be combined with existing models to build variation-aware models.

One way of building such models is to describe different parameters of interest as functions of random variables that represent variations in low-level design parameters. As an example the circuit delay could be modeled in terms of the gate delays which could further be modeled in terms of the geometries of the transistors inside each gate [36, 37]. Under process variations the circuit delay could be modeled as a random variable described as a function of those random variables representing the varying transistor geometries. These variation-aware models will then be used within an optimization framework in the context of different design techniques.

2) Optimization

The second component in variability-driven design is optimization. There are different ways to capture variations in an optimization framework:

New metrics could be defined to represent the degree of variation in a design, and could be alternatively used as the *objective* in variability-driven optimization. One such metric is the timing yield (or in general parametric yield) which is defined as the probability of a design to meet its timing requirement (or other requirements) in the presence of variations. Analysis on the effects of variations could be done for any potential realization of a design, to evaluate its corresponding timing yield. This could be an indication of the quality of that potential realization in the presence of process variations. Optimizing in order to maximize the timing-yield has been a popular objective under variations [13, 28, 55, 57]. This metric could be used for parameters other than timing, such as power, as illustrated in Figure 2.4. In the generic case that the timing and power of a potential realization of a design might be of concern, a joint-yield could be defined in terms of meeting a timing constraint, and a power constraint under process variations.

The *constraints* in variability-driven optimization could also incorporate vari-

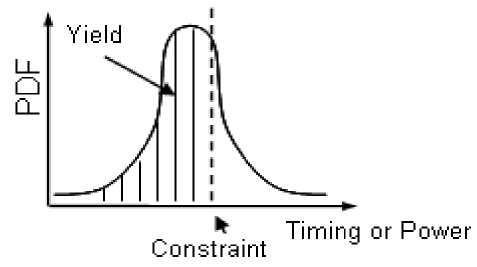

Figure 2.4: Parametric yield: a variability-driven objective.

ations [43]. A variability-driven constraint could be in the form of bounding the probability of violating certain design criteria to be at most a designer's maximum allowed risk under process variations. For the example of the circuit delay, one could add a constraint to ensure the timing-yield is at least a certain desired level given as input by the designer. This will be helpful in the conventional hierarchial design-automation framework, for which different levels of risk exist based on the level of abstraction for different optimization algorithms.

2.3.2 Properties and Challenges

Given the two components of modeling and optimization, a variability-driven design framework should have the following properties:

On one hand considering modeling, accurate models are required that not only capture variations but also capture different types of correlations that might exist between different design parameters. These models also need to be evaluated fast to be applicable in an optimization framework.

On the other hand considering optimization, efficiency is an important desired property in the presence of variations. Particularly a probabilistic framework should not be significantly slower than the corresponding deterministic one. Quality of solution is another important property. Identification of a good quality solution in the presence of variation is another challenge. Metrics such as parametric yield are just one way of evaluating potential solutions. Overall, a variability-driven framework needs to generate a good quality solution within a reasonable run-time.

Such a framework on the other hand should be flexible to incorporate possible variation-aware models. These models could be presented using different expressions, and the optimization framework should not be constrained on representation of the variation-aware models in a particular format.

A variability-driven framework should also be compatible with different optimization formulations that represent design-automation techniques such as dynamic programming formulations (e.g., buffer insertion problem [16, 17, 67]), mathematical programming formulations (e.g., gate sizing problem [13, 19, 55, 57]), etc.

In this thesis, variability-driven optimization algorithms are studied for different design-automation techniques. Important challenges in a variability-driven framework will be explained in the upcoming chapters.

Chapter 3

Probabilistic Comparison of Solutions in Variability-Driven Optimization

In this section an important challenge in a variability-driven optimization (in which characteristics of a potential solution could be random variables) will be studied. Initially, the generic characteristics of solutions in design automation are discussed, and then the superiority probability is introduced as a metric that can be used to compare potential solutions in the presence of process variations. Application of this metric is then illustrated in the context of the buffer insertion problem, in which variations are modeled using linear expressions. The case in which variations are modeled as polynomial expressions will be studied afterwards, and will be illustrated in the context of the dual-V_{th} leakage optimization problem.

3.1 Deterministic Optimization in Design Automation

Typical optimizations in design automation minimize a cost function such as power or area, while satisfying a timing requirement. Several strategies such as gate sizing, buffer insertion, leakage and dynamic power optimization fall under this generic optimization scenario.

Depending on the structure of the problem and application, several techniques for optimizing design cost under timing constraint have been proposed. These include dynamic programming (popularly used in buffer insertion [17, 35, 67], and iterative techniques [55, 57]). Most of these optimization methods compare potential solutions to the problem to determine the one with better quality. Any potential solution S_i, is characterized by two fields; an associated timing t_i and cost c_i. While comparing two potential solutions S_i and S_j, the superior solution is the one that has better timing and cost:

$$S_i \;\; superior \;\; S_j \Leftrightarrow t_i \leq t_j, c_i \leq c_j \tag{3.1}$$

In this work, it is assumed that a smaller timing and cost to be desirable. Other possibilities can be converted to the form above. For two potential solutions and their associated timings and costs, this superiority evaluation is decided in constant time, as fixed deterministic values are compared.

3.2 Superiority Probability: Metric for Comparison of Solutions

The manufacturing process of the Deep Sub Micron (DSM) technology causes significant variations on design parameters. These variations cause fluctuations in the device properties such as the channel length or oxide thickness, which directly affect the device characteristics such as its delay and power. It is crucial to account for these variations in the optimization framework. A popular method to consider process variations, is by representing the varying design parameters as random variables [12, 15, 56, 70]. This corresponds to the timing and cost of each potential solution to be random variables as well. The random variables for the timing and cost of a solution S_i, are denoted by T_i and C_i respectively.

Having the timing and cost of a solution S_i as random variables, S_i is superior to S_j if with an approximate probability of 1 it has better timing and cost:

$$S_i \;\; superior \;\; S_j \Leftrightarrow P(T_i \leq T_j, C_i \leq C_j) \approx 1 \tag{3.2}$$

For two solutions S_i and S_j, the probability P_{ij} is the value of the probability above and is called the *superiority probability* in this thesis. For two solutions S_i and S_j, the superiority probability can alternatively be written as:

$$P_{ij} = P(T \geq 0, C \geq 0) = \int_0^\infty \int_0^\infty f_{T,C}(t,c)dtdc \tag{3.3}$$

where T=$T_j - T_i$ and C=$C_j - C_i$ are newly defined random variables.

These two random variables are correlated to each other, due to correlated timings and costs of the solutions. This correlation occurs because both timing and cost are affected by common sources of variations. Therefore a *joint* probability density function (*jpdf*) is defined for T and C ($f_{T,C}(t,c)$ in the above equation). The superiority probability is defined as the computation of the double integral of the above equation.

3.2.1 Challenges

As described earlier, comparison of solution pairs is a very important step, that is explicitly or implicitly performed in most design optimization algorithms. When the cost and timing are deterministic values, the evaluation is performed in constant time (equation 3.1). Unfortunately in presence of variability-induced randomness, the superiority evaluation (computing equation 3.3) is very challenging.

The superiority probability defined in equation 3.3 denotes the probability that one solution is better than the other, by generating a value ranging between 0 and 1. The superiority probability can be used to effectively prune out solutions that are *probabilistically* sub-optimal, as will be explained in detail later. In an iterative optimization framework, it can be used to drive the direction of optimization towards probabilistically better solutions. Hence computing the superiority probability is highly imperative in variability-driven optimizations. To compute the superiority probability the following challenges exist:

1. Accuracy: Computation of equation 3.3 requires accurate characterization of the Joint Probability Density Function (*jpdf*) of random variables T anc C. This includes accurately capturing their existing correlation. In addition, integration of the *jpdf* also needs to be accurately computed.

2. Speed: The computation in equation 3.3, which corresponds to evaluation of one solution pair, must be performed in an efficient way to be applicable in the optimization framework. Any optimization framework would involve comparison of many solution pairs to determine the highest quality solution.

In this chapter three methods are proposed / evaluated to address the above challenges to compute the superiority probability. The first method is regular Monte Carlo simulation, used as the basis of comparison. The second method approximates the *jpdf* by well-known *jpdfs* (such as bi-variate Normal) and uses closed-form expressions to compute the integral in equation 3.3. The last method referred as Conditional Monte Carlo uses analytical bounds to quickly compute the integral for certain integration regions, combined with regular Monte Carlo for the rest.

Within the probabilistic comparison of solutions the T and C random variables could be approximated as linear expressions and therefore would have a Normal density function individually and bi-variate Normal density function jointly. However in general, the assumption is that the joint density function of T and C could be of any type, and then effective methods are proposed to compute the superiority probability. Initially the importance of the superiority probability in the context of the buffer insertion problem is illustrated. Then the theory behind the three proposed methods are presented and the details of the proposed methods are elaborated in variability-inspired optimization problems, and in particular when potential solutions are characterized to be in general polynomial expressions. Another important design automation application, the dual-V_{th} leakage optimization, is then studied as an example in which characteristics of solutions are modeled as polynomial expressions.

In the simulations conducted on these problems, it is shown that regular Monte Carlo simulation is very slow, therefore infeasible to get incorporated in an optimization framework. The *jpdf* approximation method is very fast but generates solution of lower quality, when compared to Conditional Monte Carlo method, due to lack of accuracy in computation of the superiority probability. The Conditional Monte Carlo method is on average 25 times faster than regular Monte Carlo method, but it is slower than *jpdf* approximation technique. It generates solutions of better quality compared to *jpdf* approximation technique, because of better accuracy.

In the next section, I will show the application of the pruning probability in the context of the buffer insertion problem.

3.3 Application: Variability-Driven Buffer Insertion

In this section the effects of process variations in the buffer insertion problem is studied in a variability-driven (probabilistic) framework in order to illustrate the application of the superiority probability for more effective comparison of solutions.

3.3.1 Introduction to Buffer Insertion

Buffer insertion is a critical step in design automation and is one of the most successful techniques for timing optimization. In Deep Sub Micron (DSM) era, the long interconnect delay has particularly turned into a serious obstacle [31]. Each interconnect typically has a source representing an originating gate and a set of sinks representing the fanouts or loads of the source. In practice there is a required arrival time for a signal to travel from the source of an interconnect and reach any of its sinks. A good example is the clock network in which the clock signal should arrive the fanouts of the clock network (in this case flip-flops) at certain required arrival times. These required arrival times at the sinks in general might be different from each other.

Figure 3.1 illustrates a typical fanout interconnect tree with one source connected to a set of sinks. Each node in the interconnect tree illustrates a bifurcation comprising of two or more children. Each interconnect segment in the tree is represented as a lumped RC network. The resistance (R) and capacitance (C) values for any segment depends on the segment length and the parasitics.

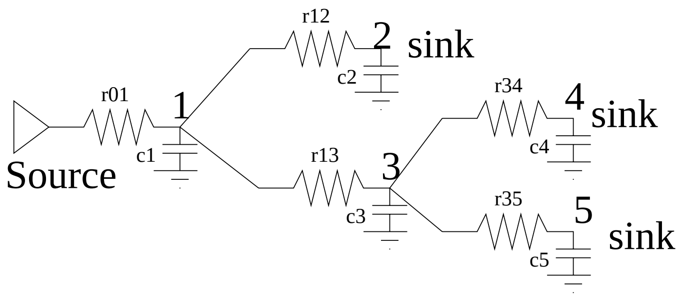

Figure 3.1: Example of an RC tree network.

In order to make sure the required arrival times are satisfied at the sinks, buffers are inserted in the interconnect tree. This tree has potential buffer locations at some of the nodes to improve the delay. For a given interconnect tree, the delay of each segment depends on the resistances and capacitances of all the other segments that are located in its downstream path to the sink. The delay of each interconnect path from the source to any of its sink is the summation of the delays of the individual segments on the path.

If the delay from the source to any sink in this tree is larger then what is required, buffer(s) could be placed along the path to decrease the delay. For example in Figure 3.1 if the path from the source to node 5 has a large delay, a buffer can be inserted in node 3. Inserting buffer at node 3 reduces the capacitance that is seen downstream to the sink, because effectively only the buffer will be seen in the downstream path, and the capacitance of the buffer in practice is much smaller than the remaining interconnect segments (capacitances of nodes 4 and 5 in this case).

In the buffer insertion problem, given an interconnect tree, a set of nodes of this tree is also provided which represents the potential buffer locations. The goal of the buffer insertion problem is to assign buffers to these potential locations in order to meet the required arrival times at all the sink nodes on the tree.

The van Ginneken algorithm is a dynamic programming formulation that can efficiently find an optimal solution to this problem [26]. This is assuming the delay of each interconnect segment is modeled using a first-order expression known as the Elmore delay model [23]. This delay model in general is shown to be extensively overestimating the actual delay. More accurate delay models have also been applied to solve this problem [4]. They result in a higher quality solution.

These approaches assume that the interconnect and buffer parasitics are fixed values. However under process variations, these parameters become random variables, resulting in the arrival times of the signals to become random variables too.

Probabilistic buffer insertion has been studied recently. The approach in [35] is one such example assuming uncertainty in the interconnect-length. This approach does not take into account correlations among the lengths of different interconnect segments, and is applicable only when interconnect delay are modeled using a first-order approximation (Elmore delay model [23]). The approach in [67] does probabilistic buffer insertion considering correlated interconnect and buffer variations, however the presented method is formulated for the Elmore delay model only, and the comparison of solutions is not very effective because it does not consider the correlation among solutions. The paper [39] investigates the impact of process variation in effective channel length of a device and Chemical Mechanical Polishing to build a model for RC parasitics, and uses it to solve for the simultaneous buffer insertion, interconnect-sizing and fill insertion problem. However this technique does not consider correlations among variables while estimating the characteristics of potential solutions.

In this study the interconnect and buffer parasitics are random variables that in general might be correlated to each other due to getting affected by common sources of variation. It has been shown in [10, 65] that ignoring correlations results in tremendous over-estimation of signal path delays in a circuit. Because of considering correlations, the proposed buffer insertion approach has a better estimate of the delays and costs of a solution. In addition different buffering solutions are also correlated to each other. Ignoring this type of correlation results in misleading comparison of potential solutions during optimization, resulting in inaccurately identifying a sub-optimal solution as optimal. As will be illustrated comparison of potential solutions are effectively done using the metric: superiority probability.

In summary the proposed comprehensive buffer insertion approach has the following contributions:

- Variations are assumed to affect both the interconnect and buffer parameters.
- Correlations among design parameters due to common sources of variation are considered.
- Higher order delay models for interconnects in addition to the Elmore delay model can be used.
- The superiority probability effectively compares potential solutions while considering their correlations.

In the upcoming subsections initially the methods to compute the signal propagation in a buffered interconnect tree using different interconnect delay models are reviewed. Then these models will be modified to consider process variations. The probabilistic algorithm to buffer insertion that is an extension of the van Ginneken algorithm will then be presented. Finally at the end of this section supporting simulation results will be presented.

Simulation results indicate that the solutions from deterministic approaches which met the timing requirement deterministically, under process variations on average only had 0.19 probability of meeting the timing constraint. However, our proposed probabilistic buffer insertion, generated solutions that met the timing constraint with an average probability of 0.63.

Next I will summarize the conventional buffer insertion problem which includes the formal definition of this problem, and different delay models of a buffered interconnect tree, and then extend these to consider process variations.

3.3.2 Preliminaries

Definition of the Buffer Insertion Problem

Given the fanout interconnect tree with parasitic resistances and capacitances, interconnect-lengths, potential buffer locations, sink required arrival times, sink capacitive loads, the buffer insertion problem is the problem of placing buffers into the tree such that required arrival time at input of the driving gate is maximum.

Buffer Delay Model

Each buffer has two intrinsic parameters r_{buf} and c_{buf} where r_{buf} is the driving resistance and c_{buf} is the input pin capacitive loading. Given a buffer placed at a certain node n_i in the tree, the delay of the buffer d_{buf} is $r_{buf} \times C_{T_i}$, where C_{T_i} is the capacitance of the interconnect tree rooted at n_i.

The computation of C_{T_i} is explained in the following example. Assume buffers exist at n_1 and n_3 in figure 3.1. The delay in the buffer at n_1 is $d_{buf_1} = r_{buf_1} \times C_{T_1}$, where $C_{T_1} = C_1 + C_2 + c_{buf_3}$. Here c_{buf_3} is the capacitance of the buffer placed at n_3. If there is no buffer at n_3 then $C_{T_1} = C_1 + C_2 + C_3 + C_4 + C_5$. Therefore having a buffer shields off the downstream capacitive loading.

Interconnect Delay Model: First-Order Approximation

Several delay models have been proposed that estimate the arrival time at the source of a buffered tree. The simplest one is the Elmore delay model [23], which is a first-order approximation of delay. Using the Elmore delay model, the delay of an interconnect segment (i, j from n_i to n_j) is:

$$d_{i,j} = r_{i,j} \times C_{T_j} \tag{3.4}$$

Here $r_{i,j}$ is the resistance of the segment connecting the two nodes, where n_j is the child of n_i (in the fanout tree). C_{T_j} is calculated as explained in section 3.3.2. The delay of a path P from n_i to a sink n_t in the tree is:

$$d_{i,t} = \sum_{\forall net-segment:k,j \in P} d_{k,j} + \sum_{\forall buf \in P} d_{buf} \tag{3.5}$$

For a tree with appropriate placement of buffer locations, the required arrival time at the source is computed in the following procedure using the Elmore delay model. The tree is traversed topologically from the sinks to the source. At each node n_i, two important parameters denoted by R_i and C_{T_i} are computed, where R_i is the required arrival time at n_i, and C_{T_i} is the capacitive loading seen at n_i. These parameters are computed bottom-up using the following equations:

$$R_i = Min_{\forall j \in child(i)}(R_j - d_{i,j}) \tag{3.6}$$

$$C_{T_i} = C_i + \sum_{\forall j \in child(i)} C_{T_j} \tag{3.7}$$

where $d_{i,j}$ is the Elmore delay between n_i and its child n_j.

If a buffer exists at n_i then R_i and C_{T_i} are adjusted as: $R_i = R_i - d_{buf}$ and $C_{T_i} = c_{buf}$, where d_{buf} is computed as explained in section 3.3.2. The required arrival time at the source of the driving gate is R_{source} which is computed using the above equations in topological interconnect tree traversal.

Interconnect Delay Model: Second-Order Approximation

A second-order delay model from a node n_i to a sink n_t requires computation of the first and second moments between these two nodes. Note sink n_t can either be a buffer or an actual sink of the interconnect tree rooted at n_i. The second moment from a node n_i to sink n_t denoted by $m2_{i,t}$ can be computed recursively in terms of a node n_j in the path between i to t as proposed in [42]:

$$m2_{i,t} = m2_{i,j} + m2_{j,t} + P_{i,t} \tag{3.8}$$

where

$$m2_{i,j} = r_{i,j} Z_{i,j}$$

$$Z_{i,j} = m1_{i,j} C_{T_j} + \sum_{k \in child(j)} Z_{j,k}$$

$$P_{i,t} = m1_{i,j} \times m1_{j,t}$$

Here $m1_{i,j}$ is the first moment (Elmore delay model). The variables $P_{i,t}$ and $Z_{j,k}$ can be thought of as auxiliary variables. The variable C_{T_j} refers to the total lumped capacitance seen from n_j, where n_j is the child of n_i that contains sink n_t in its fanout tree. Using these first and second moments, the second-order delay model, denoted by D2M, from n_i to sink n_t denoted by $d_{i,t}$ is computed as follows [3]:

$$d_{i,t} = \frac{(m1_{i,t})^2}{\sqrt{m2_{i,t}}} ln2 \tag{3.9}$$

This is an empirical model that is always upper bounded by the Elmore delay model ($m1$). To obtain this empirical expression, it was observed in [3] that in general for any path, such as the path from n_i to n_t, the ratio of $\frac{(m1_{i,t})^2}{m2_{i,t}}$ is much smaller than 1 at n_i, and slighter lager than 1 at the n_t. Consequently, this "scaled Elmore delay" was adjusted, resulting in the above D2M expression.

Compared to other delay models, D2M has the following advantages [3]:

- It is simpler than higher order delay models of [34, 41] and therefore better suited in optimization.

- It is more accurate than the Elmore model with a significant decrease in the error of approximating the delay.

Given a buffered interconnect tree, the required arrival time at the source is calculated as follows: As the tree is traversed topologically from POs to the source, for each node n_i in the tree, the required arrival time and capacitive loadings (R_i and C_{T_i}) are computed. C_{T_i} is found using equation 3.7. Required time R_i is:

$$R_i = Min_{\forall sinks:t}(R_t - d_{i,t}) \tag{3.10}$$

where $d_{i,t}$ is the D2M delay model found by equation 3.9 using first and second moments that are already computed. In equation 3.10 the sinks of n_i are buffers or tree sinks that are reachable directly (without any intermediate buffers) from n_i. If a buffer exists at n_i then R_i and C_{T_i} are adjusted as: $R_i = R_i - d_{buf}$ and $C_{T_i} = c_{buf}$. The required arrival time at the source of the driving gate is R_{source} which is computed bottom-up using the above equation in topological tree traversal.

3.3.3 Variation-Aware Delay of A Buffered Interconnect Tree

In the previous section, fixed delay calculation from any node n_i to any sink n_t in a buffered fanout tree of a driving gate was explained. This delay was caused due to interconnect and buffer parasitics. One method to consider process variations, is by representing the varying device and interconnect parameters as random variables similar to [10, 35, 39, 67], for which probability density functions (*pdf*s) are modeled. Given the *pdf* for each of the interconnect and buffer parasitics, computing the *pdf* of the arrival time in the source of a buffered interconnect tree is explained in this section, while considering the correlations among random variables due to common sources of process variation.

Modeling Variability in Parasitics and Their Correlations

For each segment i, j in the interconnect tree, the resistance $r_{i,j}$ and capacitance C_j are random variables. These random variables are expressed in a linear form, as in [1, 10, 65]:

$$V_j = \mu_{vj} + \sum_{i=1}^{n} a_{ij}^{(v)} X_i + b_j^{(v)} Y_j \qquad (3.11)$$

Here V_j represents the random variable for the resistance or capacitance of an interconnect segment or buffer parasitics. In the above equation X_is are independent random variables representing variable chip parameters such as effective channel length or oxide thickness. In practice if the X_i random variables are correlated to each other, using a principal component analysis [33], one can always transform the set of correlated random variables into a set of independent random variables. These principal components are assumed to have a standard Normal distribution (N∼(0,1)) [37, 40, 65]. Also in the above equation the variable Y_j has a standard Normal random distribution and represents uncorrelated variation in the interconnect segment (or buffer) and is assumed to be independent of the other X_i variables. The constant μ_{vj} is the expected value of V_j. All parasitics share the common principal components and therefore illustrate correlated behavior. The buffer parasitics r_{buf}, c_{buf} are also approximated using this linear expression.

Linear representation provides a very elegant way of capturing global correlations, due to the common X_i random variables. Similar to the approaches in [1, 10, 65], all delay and capacitance random variables at each node of a buffered interconnect tree are presented as linear expressions, which is explained next.

First-Order Approximation

In computing the arrival time of a buffered tree using Elmore delay model, the total capacitance in any subtree rooted at node n_i is the sum of parasitic capacitance terms (equation 3.7). Each parasitic capacitance is described in a linear expression using the common X_i principal components and a Y_j variable indicating uncorrelated behavior in the system. Addition of capacitances, each expressed in a linear form is represented in a linear form by summation of corresponding coefficients of X_i and Y_j variables:

$$C_{T_i} = \sum C_j = \sum \mu_{cj} + \sum_{\forall k} (\sum a_{kj}^{(c)}) X_k + \sum b_j^{(c)} Y_j \tag{3.12}$$

In this representation, the total number of variables are the sum of principal components and random terms Y_j.

Delay of an interconnect segment i, j from node n_i to its child n_j denoted by $d_{i,j}$ is computed using Elmore delay, where $r_{i,j} = \mu_{r_{ij}} + \sum a_{ij}^{(r)} X_i + b_j^{(r)} Y_j$ and $C_j = \mu_{c_j} + \sum a_{ij}^{(c)} X_i + b_j^{(c)} Y_j$, each represented in a linear form, are multiplied, resulting in the following expression:

$$d_{i,j} = r_{ij} \times C_j = \mu_{r_{i,j}} \mu_{c_j} + \sum (a_{i,j}^{(r)} + a_{i,j}^{(c)}) X_i + \sum (b_{i,j}^{(r)} + b_{i,j}^{(c)}) Y_i + ... \tag{3.13}$$

The multiplication result is approximated in a linear form by ignoring the second order terms. Once the delay of an interconnect segment is represented in a linear form, the required arrival time at node n_i is computed bottom-up. Recall this was done by topologically traversing the nodes from the sink nodes to the source. Given the required arrival time R_j for any child n_j of node n_i expressed in a linear form, the required arrival time of n_i, denoted as R_i, is computed using equation 3.6 ($Min_{\forall j \in child(i)}(R_j - d_{i,j})$). The required arrival time of n_i is approximated in a linear form as follows:

- Initially the linear expressions of R_j and $d_{i,j}$ are subtracted:

  $R_j - d_{i,j} = \mu_{R_j} - \mu_{d_{i,j}} + \sum(a_{i,j}^{(R)} - a_{i,j}^{(d)})X_i + \sum(b_{i,j}^{(R)} - b_{i,j}^{(d)})Y_i$

- Then a Min operation (equation 3.6) is done on required arrival times.

The Min of two linear expressions is approximated as a linear expression, as will be explained. If node n_i has more than two children, Min is iteratively done. Let $R3 = Min(R1, R2)$. We have:

$$\begin{gathered} R1 = a_{10} + \textstyle\sum_{i=1}^{n} a_{1i}X_i + \sum_{i=1}^{m_1} b_{1i}Y_i \\ R2 = a_{20} + \textstyle\sum_{i=1}^{n} a_{2i}X_i + \sum_{i=1}^{m_2} b_{2i}Y_i \\ R3 = Min(R1, R2) = a_{30} + k(\textstyle\sum_{i=1}^{n} a_{3i}X_i + \sum_{i=1}^{m_1+m_2} b_{3i}Y_i) \end{gathered} \tag{3.14}$$

The coefficients a_{30}, a_{3i} and b_{3i} are computed using [14]:

$$\begin{gathered} a_{30} = a_{10}\phi(\alpha) + a_{20}\phi(-\alpha) + \theta\varphi(\alpha) \\ a_{3i} = a_{1i}\phi(\alpha) + a_{2i}\phi(-\alpha); \quad b_{3i} = b_{1i}\phi(\alpha) + b_{2i}\phi(-\alpha) \\ \theta^2 = a_{10}^2 + a_{20}^2 - 2a_{10}a_{20}\rho; \quad \alpha = (a_{20} - a_{10})/\theta \end{gathered} \tag{3.15}$$

where $\varphi(\alpha)$ and $\phi(\alpha)$ are the *pdf* and cumulative distribution function (*cdf*) for a standard normal (N$\sim$(0,1)) random variable. The parameter ρ is the correlation coefficient between $R1$ and $R2$. Constant k=$\sigma_{actual}/\sigma_{approx}$ is defined such that the variance of the actual and approximate distributions match:

$$\begin{aligned}\sigma^2_{actual} &= (a^2_{10} + \sigma^2_{R1})\phi(\alpha) + (a^2_{20} + \sigma^2_{R2})\phi(-\alpha) \\ &\quad +(a_{10} + a_{20})\theta\varphi(\alpha) - a^2_{30} \\ \sigma^2_{approx} &= \textstyle\sum_{i=1}^{n} a^2_{3i} + \sum_{i=1}^{m_1+m_2} b^2_{3i}\end{aligned} \tag{3.16}$$

Therefore using this technique the output of the Min operation is approximated back in a linear form. Note if the node has a buffer location, then $R_i = R_i - d_{buf}$ and $C_{T_i} = c_{buf}$. Here d_{buf} and c_{buf} are represented in a linear form and the result is a linear expression once again. Finally the required arrival time at R_{source} is computed bottom-up using equation 3.6.

Second-Order Approximation

In order to compute the second order delay $d_{i,k}$ from node n_i to any node n_k, the first and second moments between these two nodes are calculated. The first moment is calculated using equation 3.5, where the delays of buffers d_{buf} and of interconnect segments $d_{k,j}$ are all expressed in a linear form and the results of their additions are in a linear form. The second moment is computed using equation 3.8, by adding and multiplying linear forms, also expressed in linear form.

The D2M delay model is computed using the first and second moments using equation 3.9. Assuming the first and second moments are represented in a linear form, the D2M delay model is approximated back in the linear form similar to the approach in [1]. If

$$m1_{i,t} = m_{10} + \textstyle\sum_{i=1}^{n} a_{1i}X_i + \sum_{i=1}^{m_1} b_{1i}Y_i$$

$$m2_{i,t} = m_{20} + \textstyle\sum_{i=1}^{n} a_{2i}X_i + \sum_{i=1}^{m_2} b_{2i}Y_i$$

then according to [1]:

$$D2M_{i,t} = ln2 \times \frac{m_{10}^2}{\sqrt{m_{20}}}(1 + \sum_{i=1}^{n} a_{3i}X_i + \sum_{i=1}^{m_1+m_2} b_{3i}Y_i) \tag{3.17}$$

where

$$a_{3i} = \frac{2a_{1i}}{m_{10}} - \frac{a_{2i}}{2m_{20}} \quad b_{3i} = \frac{2b_{1i}}{m_{10}} - \frac{b_{2i}}{2m_{20}} \tag{3.18}$$

Once the delay is computed using the D2M model, the required arrival time at each node is computed using equation 3.10 during the topological traversal of the tree from POs to the source. This involves subtraction and Min of linear expressions, which are all presented back in linear form. Finally the required arrival time at R_{source} is found as a linear expression during the topological traversal of tree.

3.3.4 Variability-Driven Buffer Insertion: The Algorithm

A probabilistic approach to design optimization should be able to handle the variability in DSM fabrication technology. Such an approach takes the probabilistic estimates of critical design metrics such as timing and perform design optimization such that the likelihood of satisfying all design constraints simultaneously are maximized. In this work such an approach to the buffer insertion problem is presented. The primary objective is to decide the locations of buffers in a tree such that the overall likelihood of satisfying a required timing constraint at the source is maximized:

$$Objective = Maximize \int_{D}^{\infty} f(t)dt \tag{3.19}$$

where D is the arrival time constraint at the source of the tree and $f(t)$ is the *pdf* of the arrival time of a solution at the source. Through linear representation, the proposed approach can effectively consider correlations while performing probabilistic optimization. The variability-driven algorithm is generic enough in capturing all kinds of parametric variability (as compared to only interconnect-length variability in [35]) including buffer parasitic variability. Finally, the formulation uses the second order delay model and is therefore significantly more accurate and practical.

Algorithm

The overall structure of the algorithm is similar to [4] and is generalizable both to the Elmore delay model and the second order D2M model. The algorithm traverses the tree topologically from the sinks to the source.

At each node a set of potential buffering solutions for the fanout subtree rooted at the node is determined. This is followed by pruning of the sub-optimal solutions. This procedure is repeated at each node until the source where the solution with the best probability of meeting the timing constraint is chosen.

Any solution $S(n)$ at a node n is characterized by four parameters: $S(n).req$, $S(n).ceff$, $S(n).bufs$ and $S(n).sinks$. Here $S(n).req$ and $S(n).ceff$ are the required arrival time and the effective capacitive loading respectively for the buffering solution of the subtree rooted at n. Both these parameters are random variables expressed in a linear form. $S(n).buff$ remembers the buffer locations and $S(n).sinks$ contains the list of all nodes in the buffer sub-tree rooted at n directly within reach from n (without any intermediate buffers). The first and second moments to these sinks from n are also stored.

Algorithm I illustrates how to calculate the set of solutions for a node n. As the tree is traversed topologically from the sinks to the source. For each node n, the following steps are done:

- If node n is a sink, one solution is formed for which the required arrival time, $S.req$, and the effective capacitance, $C.eff$, is set to the required arrival time and effective capacitance of that sink, provided as input. Since the assumption is that the sink itself is not a potential buffer location then $S.bufs$ will be the empty set. $S.sink$ which stores the sinks that are seen so far, will be the node itself.

- If node n has one child, the solution set formed for the child will be used to generate the solution set of the node. For each solution of the child ($S_{child}(i)$), one solution is initially generated for the node ($S(i)$). Here $S(i).ceff$ is found by the addition of the linear expressions of $S_{child}(i).ceff$ with C_n of node n. The stored sink nodes and buffers will remain the same ($S(i).sinks = S_{child}(i).sinks$, $S(i).bufs = S_{child}(i).bufs$).

  To find the required arrival time $S(i).req$, the delay from node n should be computed to any of its sink k that is in $S(i).sinks$. This delay is computed as a linear expression using the Elmore delay model or the second order delay model as previously explained. The required arrival time $S(i).req$ of the node for solution i is the minimum of these required arrival times which is found using equation 3.6.

- Finally if node n has more than one child, initially a new solution set is generated by merging all the solutions of all the children.

  Algorithm II shows how to merge two such solutions from the node's children. To merge solutions S_i and S_j from two children of node n, the union of $S_i.sinks$ and S_jsinks will be the new set of sinks. Also the union of $S_i.bufs$ and S_jbufs will be the new set of stored buffers. The required arrival time is found similarly by finding the minimum of the required arrival times of solution S_i and S_j. The effective capacitance is found by adding the linear expressions of $S_i.ceff$ and $S_j.ceff$. For more details please refer to algorithm II.

  This merging is done for all combinations of all solutions of the children of the node to generate the new solution set for n.

- Once the solution set of node n is generated, possibility of adding a buffer is investigated. If n is a potential buffer location, each solution $S(i)$ of n is replaced by two solutions, to store both possibilities of adding / or not adding a buffer. If a buffer is not added, the attributes of the solution will stay the same. If a buffer is added, node n will be added to the set of stored buffers for that solution. The effective capacitance of the buffered solution will be set to the effective capacitance of the buffer. The sinks of the buffered solution will be only node n itself, and finally the required arrival time of the buffered solution will be the required arrival time of the unbuffered solution minus the buffer delay. These details are shown in Algorithm I.

Algorithm 1 *Probabilistic Buffer Insertion*

```
INPUTS: Fanout tree of node $n$, Required Arrival times at the sinks
OUTPUT: Set of co-optimal solutions for fanout tree of $n$
S = ø
if($n$ is a sink)
    $S.req = R_n$ /*$R_n$ = required arrival time at sink $n$*/
    $S.c_{eff} = 0$    $S.bufs = ø$    $S.sinks = n$
else if($n$ has one child)
    for($i$=1;$i \leq |S|$; $i$++)
        $S(i).sinks = S_{child}(i).sinks$
        /* Compute the first and second moments to each sink */
        $S(i).c_{eff} = ProbSum(S_{child}(i).c_{eff}, C_n)$
        $S(i).bufs = S_{child}(i).bufs$
        for each sink $k$ in $S.sinks$
            $Delay = ComputeDelay(n, k)$ /*Using D2M or Elmore Delay*/
            $ReqTimes(k) = k.req - Delay$
            /* k.req = required arrival time at sink k*/
        $S(i).req = ProbMin(ReqTimes)$
else
    for($i$=1; $i \leq |S_{child1}|$; $i$++)
        for($j$=1; $j \leq |S_{child2}|$; $j$++)
            $S_i = S_{child1}(i)$
            $S_j = S_{child1}(j)$
            $S = Union(S, Merge(S_i, S_j))$
if($n$ has a feasible buffer location)
    $BufferSols = ø$
    for($i$=1; $i \leq |S|$; $i$++)
        $BufferSols(i).req = S(i).req - BufDelay$
        /*Note Subtraction is on canonical expressions*/
        $BufferSols(i).c_{eff} = buffer.c_{eff}$
        $BufferSols(i).bufs = Union(S.bufs, n)$
        $BufferSols.sinks = \{n\}$
    $S = Union(S, BufferedSols)$
$Prune(S)$
return($S$)
```

Algorithm 2 *Merge Solutions*

```
INPUTS: Solution $S_i$ of child1 and Solution $S_j$ of child2 of child $n$
OUTPUT: Combined solution $S$
$S.sinks = Union(S_i.sinks, S_j.sinks)$
$ReqTimes = \emptyset$
for each sink $l$ in $S.sinks$
        $Delay = ComputeDelay(n, k)$
        /*Using D2M or Elmore Delay*/
        $ReqTimes[k] = k.req - Delay$
$S.req = ProbMin(ReqTimes)$
$S.c_{eff} = ProbSum(S_i.c_{eff}, S_j.c_{eff})$
$S.bufs = Union(S_i.bufs, S_j.bufs)$
```

Algorithm I is called for all the nodes in topological order from the sinks to the source. Once the solution set in the source node is generated, among all the solutions, the one that has the maximum probability of meeting the required arrival time constraint (computed from equation 3.19) is selected, and its corresponding buffer locations will be the result of the algorithm.

As can be seen when a node has more than one child, the size of the new solution set after combining the solution of the node's children will exponentially grow, because all the possibilities are stored. More over if the node is a potential buffer location, the size of the solution set is doubled. Unfortunately in practice it is not possible to store and evaluate all the possibilities, and it is also important to efficiently obtain a high quality solution.

To do this at each node pruning is done using the concept of the superiority probability introduced in this chapter. As Algorithm I shows, once the solution set of the node is generated, pruning is done to identify and store only a limited number of “co-optimal” solutions (or high-quality solutions). The number of stored co-optimal solutions will be summation of the number of solutions of the node’s children. Storing only a “linear” number of solutions ensures practical run-time for this algorithm. Next the use of the superiority is explained to effectively prune sub-optimal solutions in this problem.

Probabilistic Pruning of Solutions

Given a set of buffer insertion solutions at a node, the goal of the pruning criterion is to select a set of co-optimal solutions of these. In a deterministic case, two solutions S_i and S_j are co-optimal if $(C_i \leq C_j, R_i \geq R_j)$ or $(C_j \leq C_i, R_j \geq R_i)$. Given a set of solutions van Ginneken [26] proposes a criteria to identify these co-optimal solutions (assuming the Elmore delay model is used). This is done as follows: For a given solution set, among all the solutions that have the same required arrival time, the one with minimum capacitive loading can be stored and the rest of them can be pruned (removed). Therefore for each required arrival time only one solution is stored. This criteria ensures that only a linear number of solutions are stored. To understand this, assume two solutions sets of size m and n are combined. The combination of these two sets results in $m \times n$ number of solutions. However these $m \times n$ solutions could have at most $m + n$ values for their corresponding required arrival times. This is because the required arrival time is always the minimum of the required arrival times. The stored $m + n$ solutions are guaranteed to include the optimal solution as the interconnect tree is traversed topologically.

In the probabilistic case, it is extremely important how co-optimality among solutions are defined, when the required arrival time and capacitive loading of each solution are random variables. The superiority probability proposed in this section can be directly applied in this case as follows:

For any two potential solutions S_i and S_j, their effective capacitive loadings and required arrival times, denoted by C_i, R_i, C_j, R_j respectively, are random variables that are presented in a linear combination of the X_i and Y_i random variables. Solution S_i is said to be superior to S_j if with a probability of almost 1 it has better arrival time and capacitance:

$$S_i \ \ superior \ \ S_j \Leftrightarrow P(R_i \geq R_j, C_i \leq C_j) \approx 1$$

Recall that the superiority probability is the value of the probability above:

$$P_{ij} = P(R \leq 0, C \leq 0) = \int_{-\infty}^{0} \int_{-\infty}^{0} f_{R,C}(r,c) dr dc \tag{3.20}$$

where $R{=}R_j - R_i$ and $C{=}C_i - C_j$ are newly defined random variables, which are also presented as linear combination of X_i and Y_i variables. These two random variables are correlated to each other, due to correlated arrival times and capacitive loadings of the solutions. Therefore a *joint* probability density function (*jpdf*) is defined for R and C ($f_{R,C}(r,c)$ in the above equation).

The superiority probability P_{ij} of solution S_i over S_j expresses the extent to which S_i is better than S_j, by generating a value ranging between 0 and 1. It can be used to effectively prune out solutions that are *probabilistically* sub-optimal. This is due to considering correlations between the capacitive loading and arrival time of a solution together with the correlation between two solutions.

Other probabilistic pruning criteria use less effective comparison techniques in terms of the quality of final generated solution. As an example [67] uses the following pruning criteria:

Threshold values are defined for the arrival time and cost of each solution. In general any threshold α given by the designer specifies a corresponding value π_α for the time or cost of a solution given by:

$$\alpha = \int_{-\infty}^{\pi_\alpha} f(x)dx \tag{3.21}$$

where $f(x)$ is the *pdf* of the arrival time or cost of a solution. For example given the threshold $\alpha = 0.1$ might have a corresponding arrival time of $\pi_\alpha = 100nsec$, meaning that with probability of 0.1, the timing of the solution is at most 100nsec.

Four threshold values are provided by the designer corresponding to the minimum and maximum accepted thresholds for the arrival time ($\alpha_l^{(r)}$, $\alpha_u^{(r)}$) and capacitive loading ($\alpha_l^{(c)}$, $\alpha_u^{(c)}$) of each solution. In this pruning criteria [67], a solution S_i is better than S_j if the following two conditions hold: $\pi^{(i)}_{\alpha_l^{(r)}} > \pi^{(j)}_{\alpha_u^{(r)}}$ and $\pi^{(i)}_{\alpha_u^{(c)}} < \pi^{(j)}_{\alpha_l^{(c)}}$.

These two inequalities indicate that S_i is better than S_j if its corresponding worst-case arrival time ($\pi^{(i)}_{\alpha_l^{(r)}}$) is still better (larger) than the best-case arrival time of S_j ($\pi^{(j)}_{\alpha_l^{(r)}}$), and similarly for the capacitive loading, the worst-case capacitive loading of S_i ($\pi^{(i)}_{\alpha_u^{(c)}}$) is better(smaller) than the best-case capacitive loading of S_i ($\pi^{(j)}_{\alpha_l^{(c)}}$), where the definition of the "best-case" and "worst-case" depend on the threshold levels defined by the designer.

The effectiveness (quality of solution and efficiency) of this pruning criteria depends on how the four threshold are defined by the designer, and how fast the corresponding π_α given a threshold α can be determined. Our pruning criteria on the other hand uses the notion of superiority probability which can more effectively compare potential solutions.

Assuming the principal components (X_i random variables) and the Y_i random variables representing random noise are independent with standard Normal distribution (N$\sim$(0,1)), the following observation is made: All random variables represented as linear combination of the X_i and Y_i random variables have a Normal distribution individually and multivariate normal distribution jointly.

This observation follows from the definition of multivariate Normal variables: Two random variables are jointly Normal iff all their linear combinations are Normal. In our case any linear combination of the arrival time or cost is written as a linear expression in terms of X_i and Y_i random variables (that have standard Normal distribution), that are independent. Therefore any linear combination has a Normal distribution as well. Therefore the *jpdf* in equation 3.20 ($f_{R,C}(r,c)$) is bivariate Normal.

For computing the superiority probability, or equivalently the double integral, [27] has reviewed many approximation methods. Here the standard procedure using tetrachroric series is used:

$$\begin{aligned} P_{ij} = \phi_C(\tfrac{-\mu_C}{\sigma_C})\phi_R(\tfrac{-\mu_R}{\sigma_R}) + \varphi_C(\tfrac{-\mu_C}{\sigma_C})\varphi_R(\tfrac{-\mu_R}{\sigma_R}) \\ + \textstyle\sum_{k=0}^{\infty} \tfrac{1}{(k+1)!} H_{e_k}(\tfrac{-\mu_R}{\sigma_R}) H_{e_k}(-\tfrac{\mu_C}{\sigma_C})\rho^{k+1} \end{aligned}$$

where φ_C, φ_R are the pdfs and ϕ_C and ϕ_R are cdfs of the normally distributed R and C random variables. The parameter ρ is the correlation coefficient between R

and C. Finally $H_{e_k}(x)$ is the Hermite polynomials described as:

$$H_{e_k}(x) = \sum_{m=0}^{[\frac{k}{2}]} \frac{k!}{m!(k-2m)!}(-1)^m 2^{-m} x^{k-2m}$$

It has been shown that the above polynomial description, estimates the superiority probability P_{ij} with an accuracy of 1% if expanded until the 5^{th} order [64]. Therefore, the computation of the double integral is hugely simplified to computation of a 5^{th} order polynomial.

For each solution pair S_i and S_j, $P(S_i \ \ prunes \ \ S_j)$ and $P(S_j \ \ prunes \ \ S_i)$ are calculated. For each solution S_i, a co-optimality metric is defined which reflects the likelihood that it prunes the rest of the solutions:

$$co-optimality(S_i) = \sum_{\forall j \neq i} P(S_i \ \ prunes \ \ S_j)$$

Once the co-optimality metric for each solution is determined, a number of co-optimal solutions equal to the sum of the total number of solutions at the children of the node is chosen. For example if the node has 2 children with p and q number of solutions, $p+q$ solutions are chosen out of the $p \times q$ solution set.

The $p+q$ solutions with the largest value of their co-optimality metric are chosen. Note that selecting linear number of solutions enforces a linear growth of solution space which allows for reasonable run-time of the algorithm. This pruning criterion is applied at each node while traversing the tree in topological order.

3.3.5 Results

The goal of the simulations is to illustrate that probabilistic buffer insertion considering correlations under more accurate delay model is the most effective approach in presence of variability. The following approaches are compared:

- D2-PC: Probabilistic with D2M model and correlations
- D1-PC: Probabilistic, Elmore delay and correlations
- D2-D: Deterministic with D2M delay model (such as [4])
- D1-D: Deterministic with Elmore delay[26]

D2-PC and D1-PC are the proposed probabilistic methods with correlations for D2M and Elmore delay models. D2-D and D1-D are based on the D2M and Elmore delay model and are deterministic methods similar to the approach of [4] and van Ginneken algorithm [26].

To consider process variations, the linear expression for resistance and capacitance of each interconnect segment was constructed as follows:

Each interconnect tree has segments of length between 10λ to 100λ for 0.18u technology. This defined the mean value for the parasitic interconnect resistances and capacitances. In the simulations, variations in the interconnect-width and thickness, and the buffer channel length were assumed to exist. These were assumed to have Normal distribution with a standard deviation of 20% from their mean. Additionally a random variable was considered to reflect uncorrelated variation for each interconnect segment and one for each buffer. These variables were assumed to have Normal distribution with mean 0 and standard deviation of 1% from the mean. Then all these random variables were normalized to new random variables that have standard Normal distributions similar to [5]. For capturing correlations,

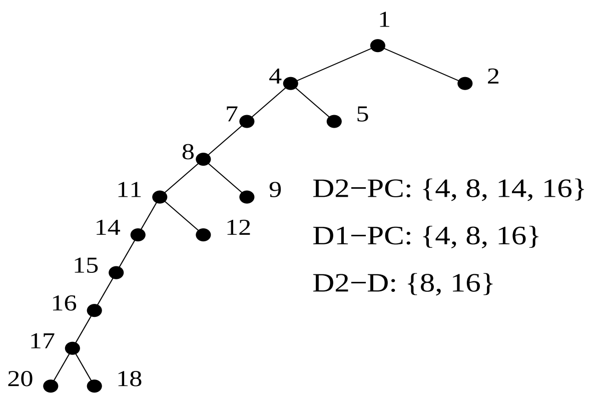

Figure 3.2: Example: Buffering solution of different techniques for net n1.

random correlation coefficients were assumed among these variables.

Figure 3.2 shows the topology of an example net with 6 sinks. For this net the 4 presented techniques were applied. The buffering solution of these techniques are also given in this Figure. Figure 3.3 shows the probability density function of the required arrival time at the source (node 1) of this tree.

The required arrival-time at the sink nodes is 3 nsec. The required arrival time at the source node is 2.6 nsec. For each technique, the probability of meeting this timing constraint is the area under their corresponding *pdf* from the required arrival time of 2.6 nsec to ∞, which are accepted signal arrival-times at the source. Among different techniques the D2M model considering correlations has the highest probability (of 0.84). After that D1-PC has the best solution (probability of 0.73). Deterministic approach with D2M model has a probability of 0.62. D1-D method did not generate a good solution, and hence was not plotted.

Table 3.1 reports the results of these methods for different nets. Columns 2 reports the relative required arrival time at the sinks (R_{cons}) of the tree (the actual arrival times at the sink and the source nodes are reported in Table 3.3). For each

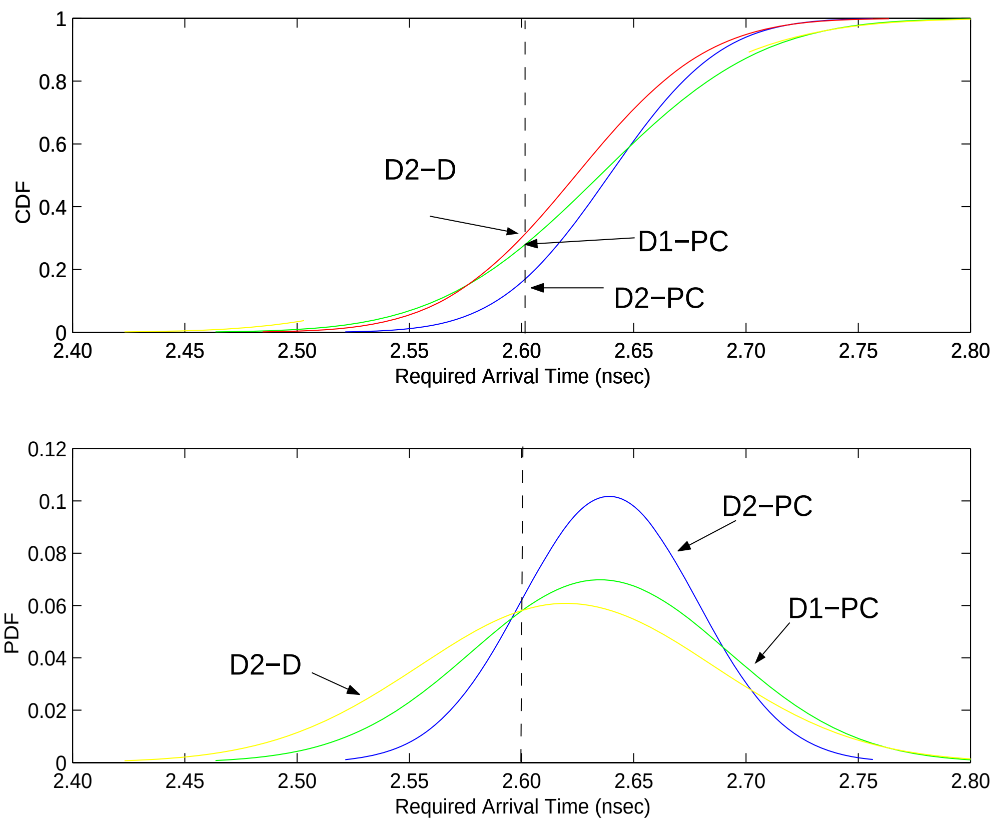

Figure 3.3: Comparison of different techniques for net n1.

technique the probability of meeting R_r and the mean μ and variance σ of the arrival-time of the buffered solution is reported here.

It can be seen that both D2-PC and D1-PC result in higher probability of satisfying timing constraint than other approaches. D2-PC is also superior than D1-PC on some nets since it incorporates a better delay model. Therefore considering correlations is imperative in probabilistic optimization.

Table 3.2 reports the number of buffers for each of these techniques. The number of sinks for each net is also reported in column 2 of Table 3.2. In deterministic D2-D and D1-D approaches, a solution is created only if it satisfies the timing con-

net	R_{cons}	D2-PC			D1-PC			D2-D			D1-D		
		P	μ	σ	P	μ	σ	P	μ	σ	P	μ	σ
n1	0.40	0.84	2.64	0.04	0.73	2.63	0.06	0.62	2.62	0.07	-	-	-
n2	0.60	0.54	9.42	0.19	0.54	9.42-	0.19	-	-	-	-	-	-
n3	0.74	0.58	9.27	0.05	0.54	9.27	0.11	0.58	9.27	0.05	-	-	-
n4	1.50	0.38	8.42	0.26	0.38	8.42	0.26	-	-	-	-	-	-
n5	1.70	0.68	7.86	0.25	0.68	7.86	0.25	-	-	-	-	-	-
n6	1.10	0.56	8.98	0.55	0.56	8.98	0.55	-	-	-	-	-	-
n7	1.85	0.52	8.16	0.75	0.48	8.14	0.88	-	-	-	-	-	-
n8	1.10	0.89	59.72	0.71	0.74	59.16	0.41	-	-	-	-	-	-
n9	1.50	0.43	58.33	0.36	0.43	58.33	0.36	-	-	-	-	-	-
n10	6.73	0.68	93.58	0.78	0.68	93.58	0.78	-	-	-	-	-	-
Av.		0.61			0.58			0.12			-	-	-

Table 3.1: Probability of meeting the required arrival time constraint at the source node.

straint deterministically. If a solution cannot be created the tree with no buffers is taken. These are indicated by dashed lines in Table 3.2. Deterministic approaches did not create a solution at many instance due to the stringent timing constraint.

The next experiment explores how the probability of meeting the timing constraint is affected by varying the degree of correlation among principal components. Since access to actual correlation data was not possible, I assumed 4 different cases where the correlation coefficients of the linear expressions of interconnect and buffer parasitics were varied. In all these cases the D2M delay model was used.

For the 4 techniques for each interconnect-segment and buffer, the variance of the linear expressions of the parasitics were equal.

	#sinks	D2-PC	D1-PC	D2-D	D1-D
n1	6	4	3	2	-
n2	8	2	2	-	-
n3	10	5	4	5	-
n4	16	3	3	-	-
n5	16	2	2	-	-
n6	32	4	5	-	-
n7	32	5	6	-	-
n8	64	3	2	-	-
n9	69	4	4	-	-
n10	73	3	3	-	-

Table 3.2: Number of buffers of different techniques.

net	R_s	R_r	P			
			RC	HC	FC	NC
n1	3	0.26	0.84	0.78	0.65	0.89
n2	10	9.40	0.54	0.49	0.45	0.61
n3	10	9.26	0.58	0.58	0.52	0.62
n4	10	8.50	0.38	0.38	0.34	0.43
n5	10	8.30	0.68	0.46	0.56	0.65
n6	10	8.90	0.56	0.53	0.23	0.64
n7	10	8.15	0.52	0.41	0.49	0.59
n8	60	58.90	0.89	0.86	0.84	0.88
n9	60	58.50	0.43	0.38	0.23	0.41
n10	100	93.23	0.68	0.71	0.55	0.83
Av.			0.61	0.56	0.49	0.65

Table 3.3: Probability of meeting the required arrival time for different correlation coefficients.

Table 3.3 reports the probability of meeting the timing constraint. The first method is the case in which the correlation coefficients among the interconnect-segment parasitics were randomly generated. This is indicated as RC (for Random Correlation) in column 4 of the Table. The second case was at the extreme, in which all correlation coefficients were 0. This was done by setting the coefficients of all principal components to be 0. This is given by NC (for No Correlation) in column 7. Third method reflects the Full Correlation case, in which the coefficients of the linear expressions of all principal components were set such that the interconnect-segment parasitics were fully correlated to each other. This is given by FC in column 5. A Half Correlation case was also explored, indicated by HC in column 6, in which the coefficients of the principal component corresponding to the interconnect-width was set to 0.

In all methods of Table I, correlation were assumed to exist between the underlying variables. However here in NC method, it is assumed that the underlying variables are not correlated at all.

When considering the average case, NC has the highest probability, ignoring correlations. On the other hand, FC has the lowest probability, considering highest correlation. The probability of HC is slightly smaller than RC but higher than FC. In general it was observed that the required arrival time corresponding to the NC case to have a much higher variance than the other cases, but it also had a higher mean, which resulted for it to have a better probability. On the other hand was the FC case, that had the smallest variance of the arrival time, but also had a smaller mean, resulting in the lowest probability. This could be happening due to different places in the formulation that the correlation coefficient plays a role:

First when approximating the min of two random variables as a linear expression, the correlation coefficient is seen in equation 17, which affects the mean, variance, and all the coefficients of the approximated variable. Second, when computing the pruning probability using Hermite Polynomials, the correlation coefficient plays a role in equation 29, which might affect pruning.

So far, it was assumed that all the random variables for the buffer and interconnect parasitics to have standard Normal distribution. This resulted in the arrival-time and capacitive loading random variables at each node to have a Normal distribution, as they are written as linear combination of these independent standard Normally-distributed variables.

To consider how the assumption of having Normal distribution for arrival-time and capacitive loadings affect the quality of solution, a uniform distribution for the principal components and random terms was assumed, and Monte Carlo simulation was used to compute the superiority probability among solutions using the D2M delay model. Interestingly it was observed that the buffering tree of the final solution is the same as D2-PC approach, where in the D2-PC approach the principal components and random terms had standard Normal distribution. This can be explained as follows. When comparing two potential buffering tree solutions in pruning, the superiority of a solution over another is a relative relationship. This superiority is correctly identified in the D2-PC approach when compared to Monte Carlo. Even though the actual values of the superiority probabilities in these two cases might be different, but the superiority among two solutions is maintained in

net	D2-MC	D2-PC	D1-PC	D2-D	D1-D
n1	58.2	11.3	10.6	0.1	0.1
n2	6.4	1.7	1.4	0.1	0.1
n3	745.1	133.6	123.7	25.1	23.5
n4	12.7	4.3	3.9	0.3	0.3
n5	92.3	18.6	17.4	1.2	1.1
n6	547.2	99.4	97.2	3.6	1.2
n7	105.0	19.5	18.3	8.7	7.4
n8	115.5	24.5	16.3	10.6	9.5
n9	136.3	26.4	25.6	15.4	13.1
n10	157.1	28.1	25.4	14.2	12.9

Table 3.4: Run-time of different techniques (sec).

most cases, such that the final buffering tree solution is the same.

The Monte Carlo technique was observed to have a much higher run-time. The run-time of Monte Carlo technique indicated by D2-MC and the rest of the techniques are compared in Table 3.4. In the rest of the techniques the principal components and random terms had a standard Normal distribution. The longest run-time of our proposed techniques is about 134 seconds for n3, which is about 0.18 of the Monte Carlo method. All simulations were run on a SunOS 5.8 with 650MHz CPU and 512MB memory.

These results indicate the following 1) D2-PC is superior to all other approaches since it considers variability, correlations and a superior delay model. 2) D1-PC is slightly inferior to D2-PC since it does not consider accurate delay modeling. 3) Deterministic approaches are often incapable of generating a solution.

This section was just an example showing the effectiveness of using the superiority probability as a metric for comparison of potential solutions in the presence of process variations. The buffer insertion example however assumed that the expressions of the timing and cost of each solution are linear. This might not be true in the general case. In the next section three generic ways are presented to compute the superiority probability.

3.4 Computing the Superiority Probability: The Theory

In this section three methods to compute the superiority probability are discussed. The first method is *Monte Carlo simulation*, which is accurate but impractical due to high run-time complexity, and will only be used as a reference to evaluate the accuracy of the two other methods. The second method *approximates the jpdf* in equation 3.3 with a density function for which the probability integral is computable. This technique is shown to be very fast, however it does not have high accuracy. The third method is based on *Conditional Monte Carlo sampling* which is very accurate and yet much faster than regular Monte Carlo.

3.4.1 Monte Carlo Simulation

Monte Carlo (MC) simulation can be used to directly compute the superiority probability without the need for an analytical expression for the *jpdf* in equation 3.3. In MC simulation, several random samples of the possible values of T and C are computed. The superiority probability equals the percentage of the times that the values for T and C are both positive, which is equivalent to determining the probability that S_i has better timing and cost than S_j in equation 3.3.

In MC simulation, the superiority probability is accurately computed with enough number of samples. However the downside of this approach is in its high run-time complexity, as T and C need to be computed and evaluated for each sample and in practice many samples are necessary to obtain an acceptable accuracy.

3.4.2 Using *jpdf* Approximation

This method *approximates* the *jpdf* ($f_{T,C}$ in equation 3.3) with a simplified *jpdf*, such that the probability integral of equation 3.3 can be computed. This means that the *jpdf* of T and C denoted by $f_{T,C}(t,c)$ is approximated with a simpler *jpdf* $f_{X,Y}(x,y)$. The bivariate density $f_{X,Y}$ should well approximate the actual density $f_{T,C}$. In addition computing the probability integral should analytically be possible for $f_{X,Y}$. This approximation is done by matching the characteristic functions of $f_{T,C}$ and $f_{X,Y}$. The characteristic function of any joint density $f_{X,Y}$ between random variables X and Y is defined as the Fourier transform of that joint density function:

$$\Phi(t_1,t_2) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{i(t_1x+t_2y)} \times f_{X,Y}(x,y)dxdy \tag{3.22}$$

Expanding the exponential term in the above equation gives a series representation of $\Phi(t_1,t_2)$ as:

$$\begin{aligned}\Phi(t_1,t_2) = 1 + it_1\int\int xf_{X,Y}(x,y)dxdy + it_2\int\int yf_{X,Y}(x,y)dxdy \\ - \frac{t_1^2}{2}\int\int x^2f_{X,Y}(x,y)dxdy - \frac{t_2^2}{2}\int\int y^2f_{X,Y}(x,y)dxdy \\ - t_1t_2\int\int xyf_{X,Y}(x,y)dxdy + ...\end{aligned} \tag{3.23}$$

In equation 3.23, the coefficients of t_1 and t_2 are defined as the moments of $f_{X,Y}$, which are formalized as:

$$m_{ij} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^i y^j f_{X,Y}(x,y)dxdy = E[X^iY^j] \tag{3.24}$$

Therefore the characteristic function of $f_{X,Y}$ can be represented as an infinite series in terms of the moments:

$$\Phi(t_1,t_2) = 1 + it_1m_{10} + it_2m_{01} - \frac{t_1^2}{2}m_{20} - \frac{t_2^2}{2}m_{02} - t_1t_2m_{11} + .. \tag{3.25}$$

To approximate $f_{T,C}$ with $f_{X,Y}$, the moments of the two distributions are matched, or equivalently the corresponding terms of the two characteristic functions are matched. In general if the approximate density function, $f_{X,Y}$, has K underlying parameters to be completely specified, the first $K+1$ terms of the two characteristic functions are matched, which corresponds to matching the 1^{st} to the K^{th} moments. For $f_{T,C}$, the moments m_{ij} can be expressed as:

$$\begin{gathered} m_{10} = E[T], \quad m_{01} = E[C] \\ m_{20} = E[T^2],\ m_{02} = E[C^2], \quad m_{11} = E[TC], \ ... \end{gathered} \tag{3.26}$$

where $E[.]$ is the expectation operator. The first K moments of $f_{T,C}$ is computed using the above equations. The approximate density function is usually a standard *jpdf*, for which closed form expressions of the moments, in terms of the underlying parameters are often provided [50]. The presented method demands addressing the following challenges:

- The expression of the first K moments should be computable for $f_{(T,C)}$.
- Once equating the moments, solving for the K unknowns should be possible.

- The approximate density function should accurately model the actual one.

- Probability integral of the approximate density function should be computable.

3.4.3 Using Conditional Monte Carlo Simulation

Motivation The third method for finding the pruning probability is a bound-based technique based on Conditional Monte Carlo (CMC) simulation. The idea is that although it is difficult to analytically compute the double integral of equation 3.3, it maybe possible to evaluate part of the integral analytically and use simulation to obtain the other part. This is done by generating bounds on T and C, and then using the bounds to predict the integral value for certain ranges of T and C. Monte Carlo simulation is done on the remaining ranges.

The idea is that if the bounds are accurate, it will not be necessary to evaluate the expressions of T and C for each MC sample, and only evaluating the bounds is sufficient. Moreover if the bounds are simple and easily computable, speedup will be gained. To better illustrate the idea behind CMC, introduce another representation of the superiority probability will be introduced using an indicator function, which is inspired by [30]. Let us assume the indicator functions Φ_T and Φ_C are defined as:

$$\Phi_T = \begin{cases} 1; T \geq 0 \\ 0; T < 0 \end{cases} \qquad \Phi_C = \begin{cases} 1; C \geq 0 \\ 0; C < 0 \end{cases} \tag{3.27}$$

The function $\Phi = \Phi_T \times \Phi_C$ is then equal to:

$$\Phi = \begin{cases} 1;\ T \geq 0, C \geq 0 \\ 0;\ other\ \ wise \end{cases} \tag{3.28}$$

The superiority probability in equation 3.3 is the same as the expected value of indicator Φ $(P(T \geq 0, C \geq 0) = E[\Phi_T \times \Phi_C] = E[\Phi])$.

Assume lower and upper bounds for T and C are denoted as: $T^L \leq T \leq T^U$ and $C^L \leq C \leq C^U$. Then the following inequalities hold:

$$\Phi_{T^L} \leq \Phi_T \leq \Phi_{T^U}, \quad \Phi_{C^L} \leq \Phi_C \leq \Phi_{C^U} \tag{3.29}$$

where Φ_{T^L}, Φ_{T^U}, Φ_{C^L} and Φ_{C^U} are indicator functions that are 1 only if their arguments are positive, defined similar to the previous indicator functions (such as equations 3.27 and 3.28). Given the above defined bounds and functions, the following inequality holds: $\Phi_{T^L} \times \Phi_{C^L} \leq \Phi \leq \Phi_{T^U} \times \Phi_{C^U}$. This is the multiplication of the inequalities of equation 3.29, which is valid since the indicator functions are either 0 or 1. We denote $\Phi^L = \Phi_{T^L} \times \Phi_{C^L}$ and $\Phi^U = \Phi_{T^U} \times \Phi_{C^U}$.

Let us define the random variable $V = E[\Phi|\Phi^L, \Phi^U]$, where $E[.|.]$ operator denotes the conditional expectation. The desired superiority probability can then be expressed in terms of V as:

$$E[\Phi] = E[E[\Phi|\Phi^L, \Phi^U]] = E[V] \tag{3.30}$$

The above equation suggests that for finding the pruning probability, one can evaluate V instead of Φ and gain speedup, if V has a smaller variance. In fact it can be shown that since:

$Var(\Phi) = E[Var(\Phi|\Phi^L, \Phi^U)] + Var(E[\Phi|\Phi^L, \Phi^U]) = E[Var(\Phi|\Phi^L, \Phi^U)] + Var(V)$ and $E[Var(\Phi|\Phi^L, \Phi^U)] \geq 0$ (because $Var(.)$ is a positive quantity), the random variable V always has a smaller variance than Φ [30], therefore it is better to use MC simulation on V rather Φ to compute the double integral.

Conditional Monte Carlo Framework

The indicator Φ defined in equation 3.28 can be written by conditioning on the combination of the values of Φ^L and Φ^U:

$$\begin{aligned} E[\Phi] = E[\Phi|\Phi^L = 1, \Phi^U = 0]P(\Phi^L = 1, \Phi^U = 0) \\ + E[\Phi|\Phi^L = 0, \Phi^U = 0]P(\Phi^L = 0, \Phi^U = 0) \\ + E[\Phi|\Phi^L = 1, \Phi^U = 1]P(\Phi^L = 1, \Phi^U = 1) \\ + E[\Phi|\Phi^L = 0, \Phi^U = 1]P(\Phi^L = 0, \Phi^U = 1) \end{aligned} \quad (3.31)$$

In the above equation $E[\Phi|\Phi^L = 1, \Phi^U = 0] = 0$, because the lower bound Φ^L can never be larger than Φ^U. In addition $E[\Phi|\Phi^L = 0, \Phi^U = 0] = 0$ and $E[\Phi|\Phi^L = 1, \Phi^U = 1] = 1$ because both the lower and upper bounds of Φ have the same value. Therefore equation 3.31 is simplified to:

$$E[\Phi] = P(\Phi^L = 1, \Phi^U = 1) + E[\Phi|\Phi^L = 0, \Phi^U = 1]P(\Phi^L = 0, \Phi^U = 1) \quad (3.32)$$

The terms of the equation 3.32 are re-written as:

$$P(\Phi^L = 1, \Phi^U = 1) = P(\Phi^L = 1) = E[\Phi^L] \quad (3.33)$$

because Φ^L can only be 0 or 1. Also:

$$\begin{aligned} E[\Phi|\Phi^L = 0, \Phi^U = 1] = P(T \geq 0, C \geq 0|\Phi^L = 0, \Phi^U = 1) \\ = P(T \geq 0, C \geq 0|(T^L \leq 0, T^U \geq 0) or (C^L \leq 0, C^U \geq 0)) = P_{CMC} \end{aligned} \quad (3.34)$$

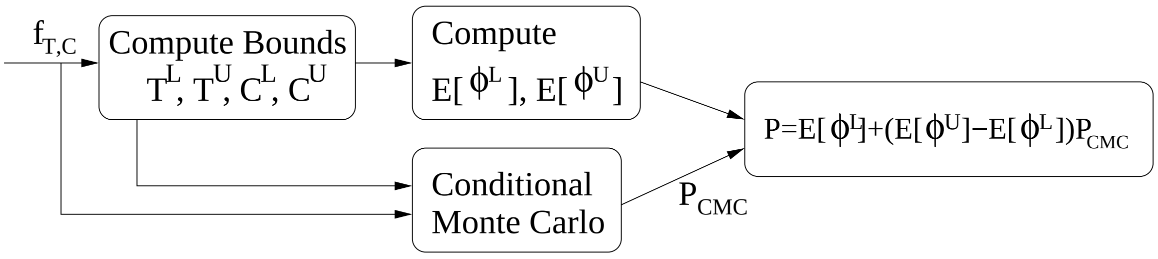

Figure 3.4: Conditional Monte Carlo framework.

where denote the conditional probability in the above equation as P_{CMC}. The third term in equation 3.32 is:

$$P(\Phi^L = 0, \Phi^U = 1) = P(\Phi^U = 1) - P(\Phi^L = 1) = E[\Phi^U] - E[\Phi^L] \tag{3.35}$$

Therefore equation 3.32 is simplified to:

$$E[\Phi] = E[\Phi^L] + (E[\Phi^U] - E[\Phi^L]) \times P_{CMC} \tag{3.36}$$

This is the final version of the equation used to find the pruning probability. In the above equation the values of $E[\Phi^L]$ and $E[\Phi^U]$ are determined separately from the bounds. The conditional probability, P_{CMC}, in the above equation, corresponds to the case when the value of the function cannot be predicted by its bounds, where the expressions for T and C should actually be evaluated. This corresponds to calculating the superiority probability of equation 3.3 in a smaller integration region where $\Phi^L = 0$ and $\Phi^U = 1$, which results in speedup when compared to pure Monte Carlo sampling.

The framework of the CMC method is shown in figure 3.4. Initially T^U, C^U, T^L and C^L are generated. Only if $\Phi^L = 0$ and $\Phi^U = 1$, the expressions for T and C are evaluated to compute P_{CMC}. The superiority probability is then found from equation 3.32.

Challenges

The speedup gained from the bound-based CMC technique depends on two important factors:

- The bounds Φ^L and Φ^U should be accurate enough to result in a small integration region.
- The bounds should be simple, so that their evaluation for each Monte Carlo sample becomes faster and simpler than evaluating the *jpdf* in equation 3.3.

In the next section, the approach (proposed by [32]) is discussed to compute such simple and accurate bounds, assuming the expressions for T and C are multivariate polynomials of arbitrary degree.

With the polynomial assumption for the T and C expressions, the simulation results indicate a 25 times speedup of this method compared to pure MC simulation, with a very high degree of accuracy.

3.5 Superiority Probability for Polynomials

In this section the superiority probability is computed for the three discussed methods, assuming the expressions for the T and C random variables are polynomials with arbitrary number of variables and of arbitrary degree.

3.5.1 Motivation: A Variability-Driven Perspective

For the generic optimization model, defined in section 3.1, the timing and cost of a solution are expressed as functions of device or interconnect parameters, etc., which are written as:

$$T_i = g_{T_i}(p_1, ..., p_n) \qquad C_i = g_{C_i}(p_1, ..., p_n) \tag{3.37}$$

In the above equations, the parameters can represent threshold voltage, interconnect length, etc. depending on the application. Under process variations, it has been shown in [12, 37, 40, 70] that the timing and cost of each solution can be expressed as functions of common (global) random variables:

$$T_i = h_{T_i}(X_1, ..., X_n) \qquad C_i = h_{C_i}(X_1, ..., X_n) \tag{3.38}$$

In the above equation X_is are independent random variables representing common chip parameters. In practice if the X_i random variables are correlated to each other, using a principal component analysis, one can always transform the set of correlated random variables into a set of independent random variables. Therefore the T and C random variables were assumed to be written as functions of independent common random variables.

Due to the common X_i random variables, the timing and cost of each solution are *correlated* random variables because they are affected by common sources of variation. Similarly correlation also exists between the timings and costs of any two solutions. For the above function, a polynomial representation can be obtained using a Taylor series expansion with respect to the X_i random variables in their local region of variation, that is written up to a desired degree of accuracy. Such polynomial representations are frequently used to express timings or costs in circuits as in [12, 37, 40]. In this section the functions for T_i and C_i random variables are assumed to be polynomials:

$$T_i = a_0^{(T_i)} + \sum_{k=1}^{n} b_k^{(T_i)} X_k + \sum_{k=1}^{n} \sum_{j=k+1}^{n} c_{kj}^{(T_i)} X_k X_j + \sum_{k=1}^{n} d_k^{(T_i)} X_k^2 + \dots$$
$$C_i = a_0^{(C_i)} + \sum_{k=1}^{n} b_k^{(C_i)} X_k + \sum_{k=1}^{n} \sum_{j=k+1}^{n} c_{kj}^{(C)} X_k X_j + \sum_{k=1}^{n} d_k^{(C)} X_k^2 + \dots \quad (3.39)$$

When comparing two solutions S_i and S_j in equation 3.3, the random variables $T = T_j - T_i$ and $C = C_j - C_i$ are similarly expressed as polynomials, because their corresponding timings and costs are expressed as polynomials.

3.5.2 Monte Carlo Simulation for Polynomials

In the Monte Carlo simulation, it is assumed that the distribution of the X_i random variables in equation 3.39 is known. Several random samples, following the underlying distribution of the X_i variables are generated and used to compute the values of T and C. The superiority probability equals the percentage of the times that T and C are simultaneously positive. The run-time complexity of MC simulation directly depends on the order of the polynomials in equation 3.39.

3.5.3 Approximating *jpdf* Using Bivariate Normal Density

Assume the polynomial expressions for the T and C random variables of arbitrary degree are as in equation 3.39. In this method, the goal is to compute the

Superiority probability of equation 3.3 with a distribution for which the probability integral is computable. One popular example that is shown in this section is the bivariate Normal distribution. Here, random variables T and C are approximated with new random variables X and Y that have a bivariate Normal probability density function:

$$\begin{aligned} f_{T,C}(t,c) \approx f_{X,Y}(x,y) = \tfrac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} exp(-\tfrac{z}{2(1-\rho^2)}) \\ z = (\tfrac{x-\mu_x}{\sigma_x})^2 - 2\rho\tfrac{(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y} + (\tfrac{y-\mu_y}{\sigma_y})^2 \end{aligned} \tag{3.40}$$

It can be seen that a bivariate Normal *jpdf* has 5 underlying parameters, $\mu_x, \mu_y, \sigma_x, \sigma_y$ and ρ to be completely specified. Moment matching is used to find these parameters. This is done as expressed in equation 3.23 and 3.25 by matching the terms 2 to 6 of the characteristic function of the bivariate Normal $jpdf$. This results in the following equations:

$$E[T] = \mu_x, \quad E[C] = \mu_y$$

$$E[T^2] = \sigma_x^2 + \mu_x^2, \quad E[C^2] = \sigma_y^2 + \mu_y^2,$$

$$E[TC] = \rho\sigma_x\sigma_y + \mu_x\mu_y$$

In the above equations the values of $E[T], E[C]$, $E[T^2]$, $E[C^2]$ and $E[TC]$ can be analytically determined, given the polynomial expressions of T and C, and the distribution of the X_i random variables.

Therefore by matching the first 5 moments, all the underlying parameters of the bivariate Normal distribution that approximates *jpdf* of T and C, are specified.

The next step is computing the superiority probability with bivariate Normal approximation of equation 3.23. For computing the bivariate Normal distribution,

[27] has reviewed many approximation methods. Here the standard procedure using tetrachroric series are used:

$$P(T \geq 0, C \geq 0) \approx \phi_{X^-}(\frac{-\mu_{X^-}}{\sigma_{X^-}})\phi_{Y^-}(\frac{-\mu_{Y^-}}{\sigma_{Y^-}}) + \varphi_{X^-}(\frac{-\mu_{X^-}}{\sigma_{X^-}})\varphi_{Y^-}(\frac{-\mu_{Y^-}}{\sigma_{Y^-}})\sum_{k=0}^{\infty}\frac{H_{e_k}(\frac{-\mu_{Y^-}}{\sigma_{Y^-}})H_{e_k}(\frac{\mu_{-X^-}}{\sigma_{X^-}})\rho^{k+1}}{(1+k)!} \quad (3.41)$$

where X^-=$-X$ and Y^-=$-Y$. Also φ_{X^-}, φ_{Y^-} are the probability density functions, and ϕ_{X^-} and ϕ_{Y^-} are cumulative distribution functions of $X^- \sim N(\mu_{X^-}, \sigma_{X^-})$ and $Y^- \sim N(\mu_{Y^-}, \sigma_{Y^-})$, that are assumed to be Normal. The parameter ρ is the correlation coefficient between X^- and Y^-. Finally $H_{e_k}(x)$ is the Hermite polynomial given by:

$$H_{e_k}(x) = \sum_{m=0}^{[\frac{k}{2}]}\frac{k!}{m!(k-2m)!}(-1)^m 2^{-m} x^{k-2m} \quad (3.42)$$

It is shown in [64] that expanding the above series until the 5^{th} degree is sufficient for a 99% accuracy in computing the bivariate Normal integral. This makes the proposed method extremely fast, since the series is expanded for a few terms.

To illustrate the accuracy of the bivariate Normal approximation, four-variate polynomial pairs were generated which were of degree 1, 2, 3, and 4. For each degree, 10000 number of polynomials of random coefficients were generated. The X_i variables in each polynomial (equation 3.39) were assumed to have a standard Normal distribution with mean 0 and variance of 1.

For each polynomial pair the superiority probability is computed by first approximating the *jpdf* to be bivariate Normal, and then using equation 3.41 to compute the superiority probability. This approximated value of the probability obtained using bivariate Normal is then compared against its correct value obtained

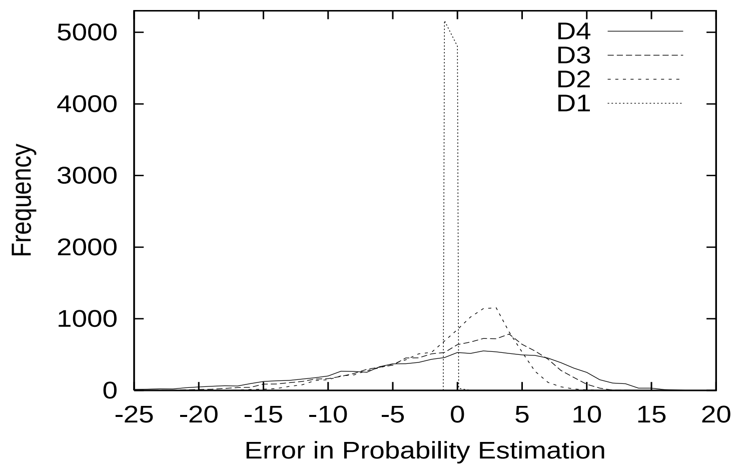

Figure 3.5: Frequency of error in bivariate normal approximation

using Monte Carlo Simulation. The error of the bivariate Normal approximation with respect to the reference Monte Carlo simulation was recorded.

Figure 3.5 illustrates the frequency distribution of the error associated with polynomials of degree 1, 2, 3, and 4. For degree 1, the accuracy is very high. This is because the bivariate distribution in this case is precisely bivariate Normal. However as the degree of the polynomial increases, the assumption of having a bivariate Normal *jpdf* becomes inaccurate and the error increases.

So far an example of approximating the joint density function with bivariate Normal was presented. However note that the presented method using moment matching is generic and approximation using other distributions can similarly be done.

3.5.4 Conditional Monte Carlo Simulation

Section 3.4.3 illustrated that by generating simple and accurate lower and upper bounds on random variables T and C, the superiority probability of equation 3.3 is computed much faster than Monte Carlo simulation.

This section describes how such bounds can be generated for a general multivariate polynomial of arbitrary degree. This is done based on the work in [32] that focuses on generating linear lower bounds for generic polynomials, which are computed by using Bernstein coefficients. Initially the Bernstein coefficients of a polynomial are explained and their properties are discussed. The Bernstein coefficients and their use to generate bounds are valid only when the X_i variables in equation 3.39 are between 0 and 1. A scaling scheme is presented to use the bounds when the variables of the polynomial vary in arbitrary range. Then computation of the bounds [32] will be explained.

Bernstein Form of A Polynomial

A general multivariate polynomial with n variables, x_1,x_2,..., x_n, with degree of each variable l_1,l_2...,l_n respectively, can be represented as:

$$p(x_1, .., x_n) = \sum_{i_1=0}^{l_1} .. \sum_{i_n=0}^{l_n} a_{i_1,..,i_n} x_1^{i_1} x_2^{i_2} .. x_n^{i_n} \tag{3.43}$$

where $a_{i_1,..,i_n}$ is the coefficient for the term $x_1^{i_1}...x_n^{i_n}$. The above polynomial is represented in its Bernstein form, over the unit hyper-box $I \in [0,1]^n$ (x_i between 0 and 1), as [9]:

$$p(x_1, .., x_n) = \sum_{i_1=0}^{l_1} .. \sum_{i_n=0}^{l_n} b_{i_1,..,i_n} B_{i_1,..,i_n}(x_1, .., x_n) \tag{3.44}$$

where $B_{i_1,..,i_n}(x_1, .., x_n)$ is given by [9]:

$$B_{i_1,..,i_n}(x_1, .., x_n) = \binom{l_1}{i_1} .. \binom{l_n}{i_n} x_1^{i_1} .. x_n^{i_n} (1 - x_1)^{i_1} .. (1 - x_n)^{i_n} \tag{3.45}$$

where $\binom{l_i}{i_j}$ denotes $C_{i_j}^{l_j}$. In the above equation $B_{i_1,..,i_n}(x_1, .., x_n)$ describes a new basis that can represent the polynomial, if each of the x_i variables vary in the unit hyper-box, i.e. between 0 and 1. For this generic polynomial the Bernstein coefficient $b_{i_1,..,i_n}$ is given by:

$$b_{i_1,..,i_n} = \sum_{j_1=0}^{i_1} .. \sum_{j_n=0}^{i_n} \frac{\binom{i_1}{j_1} .. \binom{i_n}{j_n}}{\binom{l_1}{j_1} .. \binom{l_n}{j_n}} a_{j_1,..,j_n} \tag{3.46}$$

In practice, if the variables do not vary between 0 and 1, by knowing the actual range of the variables x_1,...,x_n, a transformation is done to convert these variables into variables that range between 0 and 1. Assume in the polynomial $p(x_1, ..., x_n)$, for each variable x_i, we have $a_i \leq x_i \leq b_i$. Each variable x_i is converted to a new variable y_i as:

$$y_i = \frac{x_i - a_i}{b_i - a_i} \tag{3.47}$$

where y_i ranges between 0 and 1. The polynomial is then represented in terms of the y_i variables:

$$p(x_1, ..., x_n) = q(y_1, ..., y_n) \tag{3.48}$$

The new form of the polynomial in terms of $y_i = [0, 1]$, is used to obtain the Bernstein coefficients and the bounds.

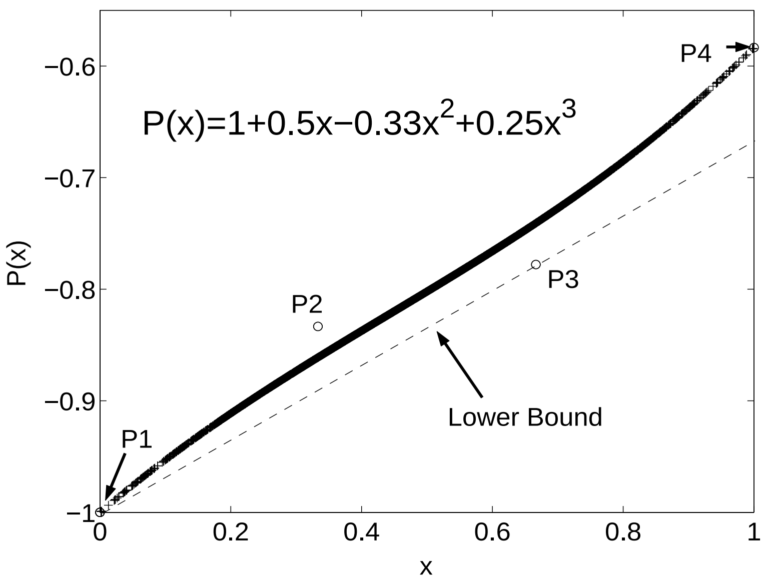

Figure 3.6: Bernstein Coefficients defining boundaries on polynomials.

Properties of Bernstein Coefficients

The Bernstein coefficients of a polynomial, specify *control points* that define a convex-hull that encloses the polynomial. A control point, $p_{i_1,..,i_n}$, is defined by Bernstein coefficient $b_{i_1,..,i_n}$ as:

$$\left(\frac{i_1}{l_1}; \;\; \frac{i_2}{l_2}; \;\; .. \;\; \frac{i_n}{l_n}; \;\; b_{i_1,..,i_n} \right)$$

This is a fundamental property of Bernstein coefficients that is used to find simple bounds around the polynomial. Figure 3.6 illustrates this property. Here a univariate polynomial is plotted with its control points in unit box. This function has 4 control points at P_1,...,P_4 at (0;-1), (0.33;0.83), (1;-0.58) and (0.67;-0.78), that encapsulate $p(x)$. These control points are used to define bounds on polynomials as explained next.

Obtaining Hyper-plane Lower Bounds

The Bernstein coefficients define control points that encapsulate the polynomial. These control points are used in [32] to define hyper-plane lower bound in the form of $L(x_1, .., x_n) = \sum_{i=0}^{n} c_i x_i$, for a generic n-variate polynomial of degree $(l_1, .., l_n)$ for the variables $(x_1, ..., x_n)$, over the unit hyper-box. If the x_i variables do not range between 0 and 1, the presented scaling scheme is initially applied to express the polynomial in terms of variables that range between 0 and 1. The hyper-plane bounds have a very simple form, which makes them suitable to use in the Conditional Monte Carlo framework. Next the algorithm proposed in [32] is explained to obtain hyper-plane lower bounds.

Algorithm III illustrated this [32]: it has n iterations for an n-variate polynomial, where at each iteration a linear system of equations with $n - 1$ unknowns is solved. The algorithm uses the Bernstein coefficients of the polynomials to find a lower bound. The obtained lower bound passes through the control point associated with the Bernstein coefficient that is minimum, as well as $n - 1$ other control points. In the end the obtained hyper-plane passes through a lower facet of the convex-hull spanned by the control points. Figure 3.6 shows the bound obtained using the above algorithm for the example polynomial.

Algorithm 3 *Algorithm to Compute Bounds (from [32])*

```
INPUTS: Given a polynomial $p(x_1, ..., x_n)$
OUTPUT: A hyper-plane lower bound for the polynomial
1) Iteration 1:
Define $u^1_{n\times 1} = (1;0;...;0)$
1.1) Let $b^0$ be the Bernstein coefficient with minimum value where $i^0_1, ..., i^0_n$ are its corresponding indices
1.2) Compute the slopes $g^1_{i_1,..,i_n}$: $g^1_{i_1,..,i_n} = \frac{b_{i_1,...,i_n} - b^0}{\frac{i_1}{l_1} - \frac{i^0_1}{l_1}}$ /*if denominator $\neq 0$ */
1.3) Find the indices $i^1_1, .., i^1_n$ that correspond to $g_{i^1_1,..,i^1_n}$ with the smallest absolute value.
1.4) Define $w^1_{n\times 1} = (\frac{i^1_1 - i^0_1}{l_1};...;\frac{i^1_n - i^0_n}{l_n})$
1.5) Compute the lower bound at iteration 1 as: $L_1(x_1, .., x_n) = b^0 + g_{i^1_1,..,i^1_n} u^1 \odot (x_1 - \frac{i^0_1}{l_1};...;x_n - \frac{i^0_n}{l_n})$
2) Iteration j (ranging 2 to n):
2.1) Find the $\widetilde{u}^j_{n\times 1} = \widetilde{u}^j_{n\times 1} = (\beta^j_1;...;\beta^j_{j-1};1;0;..;0)$ such that $\widetilde{u}^j \odot w^k = 0$ for $k = 1, ..., j-1$.
2.2) Normalize $u^{\sim}_j$: $u^j = \frac{\widetilde{u}^j}{||\widetilde{u}^j||}$
2.3) Compute the slope: $g^j_{i_1,..,i_f n} = \frac{b_{i_1,..,i_n} - L_{j-1}(\frac{i_1}{l_1},..,\frac{i_n}{l_n})}{(\frac{i_1 - i^0}{l_1};..;\frac{i_n - i^0}{l_n}) \odot u^j}$ /* if denominator $\neq 0$ */
2.4) Find the indices $i^j_1, .., i^j_n$ that correspond to $g_{i^j_1,..,i^j_n}$ with the smallest absolute value.
2.5) $w^j_{n\times 1} = (\frac{i^j_1 - i^0_1}{l_1};...;\frac{i^j_n - i^0_n}{l_n})$
2.6) Compute the lower bound at iteration j as: $L_j(x_i) = L_{j-1}(x_i) + g_{i^j_1,..,i^j_n} u^j \odot (x_1 - \frac{i^0_1}{l_1};...;x_n - \frac{i^0_n}{l_n})$
3) Output the hyper-plane from iteration n in the form $L_n(x_1, .., x_n) = \sum_{i=0}^{n} c_i x_i$.
```

Obtaining Upper Bounds

For a generic multivariate polynomial expressed in equation 3.43, an upper bound is found using the algorithm for finding the lower bound as follows:

- Let L be the hyper-plane that forms a lower bound for the multivariate polynomial p.
- Find the lower bound L^- for the polynomial p^-=$-p$.
- The upper bound of p, denoted by U will be: $U = -L^-$.

Computing the Superiority Probability

For computing the superiority probability using equation 3.36, recall that Conditional Monte Carlo sampling was used, as explained in section 3.4.3. Initially hyper-plane lower and upper bounds for the polynomially-expressed random variables T and C are computed using the presented technique above. The bounds are denoted by T^L,T^U,C^L,C^U. The superiority probability is found from equation 3.36, which is also illustrated in figure 3.4. Many samples were generated for the X_i random variables following their underlying density function. For each sample of X_i random variables, the corresponding values of T^L, T^U, C^L, C^U were evaluated to determine their sign. If $T^L < 0$ or $C^L < 0$ and $T^U \geq 0$ and $C^U \geq 0$, the sign of T and C for that sample is known from its bounds and we have:

$$\begin{aligned} T^U &= \textstyle\sum_{i=0}^{n} c_i^{(1)} X_i \geq 0 \text{ and} \quad & C^U &= \textstyle\sum_{i=0}^{n} c_i^{(2)} X_i \geq 0 \\ T^L &= \textstyle\sum_{i=0}^{n} c_i^{(3)} X_i < 0 \text{ or} \quad & C^L &= \textstyle\sum_{i=0}^{n} c_i^{(4)} X_i < 0 \end{aligned} \tag{3.49}$$

In the above inequalities, $c_i^{(1)}, c_i^{(2)}, c_i^{(3)}, c_i^{(4)}$ are the coefficients of the hyper-plane bounds. Only if the upper bounds are both positive (or zero) and at least one the lower bounds is negative, the polynomial expressions of T and C are computed, in order to compute the conditional probability of equation 3.34. Once the conditional probability is computed, the probability value is found using equation 3.36. The smaller integration region corresponds to smaller MC samples and results in speedup. In addition, in equation 3.36, $E[\Phi^L]$ and $E[\Phi^U]$ are:

$$E[\Phi^L] = E[\Phi_{T^L} \times \Phi_{C^L}] = P(T^L \geq 0, C^L \geq 0)$$

$$E[\Phi^U] = E[\Phi_{T^U} \times \Phi_{C^U}] = P(T^U \geq 0, C^U \geq 0)$$

where T^L, T^U, C^L, C^U are hyper-planes.

These hyper-planes can be expressed in terms of standard Normal random variables (with mean 0 and variance 1) using a transformation similar to equations 3.47, 3.48. With this assumption, T^L, T^U, C^L, C^U will each have a Normal distribution, where the pairs (T^L, C^L), (T^U, C^U) have bivariate Normal density functions. Therefore the probabilities in the above equation can get computed using equation 3.41. Note that this assumption is true since the X_i variables are assumed to have a Normal distribution.

The speedup of computing the superiority probability depends on how accurate the bounds predict the polynomial behavior. In general if a small change in the variables results in a large change in the value of the polynomial, the resulting bounds might not be very tight, which results in evaluating T and C expressions too many times, for a large integration region. To address this problem, hyper-plane bounds are computed for different regions of the X_i random variables. Generating the *piecewise* hyper-plane bounds is explained next.

Generating Piecewise Hyper-plane Bounds

Depending on the application, if the behavior of the polynomial is such that the generated hyper-plane bounds are not accurate, piecewise hyper-plane bounds are generated by dividing the polynomial into different regions. This is done by imposing separate ranges for some or all of the variables of the polynomial. As an example a variable x_i that is initially in the range [0,1], can be divided into two ranges (i.e. [0,0.5),[0.5,1]). The combination of these different ranges of the x_i variables, divides the polynomial into different regions. For each region, a lower and upper bound is found and stored.

Assume for any region R_i and for any variable x_i we have $a_i \leq x_i \leq b_i$. The polynomial is initially represented in terms of the y_i variables that range between 0 and 1 using equations 3.47 and 3.48. Then lower and upper bounds are determined for region R_i using the presented technique. At each region $E[\Phi]$ is estimated and in the end the superiority probability is determined by combining these estimates.

Imposing the piecewise scheme results in very accurate bounds. It was found that a few number of piecewise bounds, results in a very small integration region in equation 3.34, which results in huge speedups for our selected CAD application, explained next.

3.6 Application: Variability-Driven Dual-V_{th} Leakage Optimization

In this section the variability-driven dual-V_{th} leakage optimization problem is presented in which the models describing fabrication variation are polynomial expressions. The effectiveness of the use of the superiority probability is illustrated for this application.

Initially this optimization problem is formally defined, and the conventional formulation of this problem is presented (assuming variations are ignored). Then the variability-driven version of the problem will be presented.

3.6.1 Conventional Approach

Problem Definition

Given a gate-level netlist, the dual-V_{th} leakage optimization [38] technique decides the threshold voltage of each gate, out of two possible V_{th} choices. This allows to minimize leakage under a given timing constraint. For each gate, the decided threshold voltage, determines its sub-threshold leakage current and its delay.

The sub-threshold leakage current denoted by I_l, is expressed as a function of V_{th} by the following equation:

$$I_l = I_0 e^{\frac{V_{gs} - V_{th}}{nV_T}} \tag{3.50}$$

Here $I_0 = \mu_0 C_{ox}(W/L)V_T^2 e^{1.8}$, where C_{ox} is the gate oxide capacitance, (W/L) is the width to length ratio of the leaking MOS device, μ_0 is the zero bias mobility. In equation 3.52, V_{gs} is the gate to source voltage, V_T is the thermal voltage and n is the sub-threshold swing coefficient. From equation 3.52, it is evident that a higher threshold voltage results in lower sub-threshold leakage current.

The delay of a gate denoted by D, is expressed in terms of its V_{th} by the following equation[45] :

$$D \propto \frac{C_L V_{dd}}{(V_{dd} - V_{th})^{\alpha}} \tag{3.51}$$

Here C_L is the load capacitance at the gate output and α is the velocity saturation index which is about 1.3 for the 0.18 μm CMOS technology. Equations 3.50 and 3.51 show that a higher V_{th}, results in smaller leakage for a gate but higher delay.

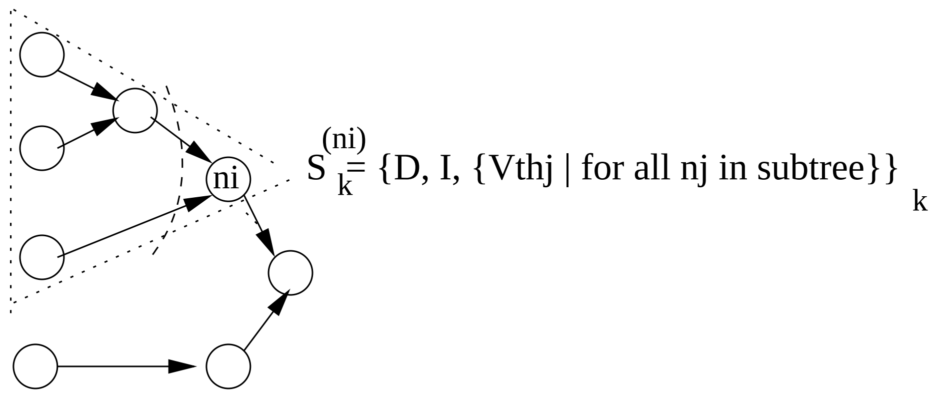

Figure 3.7: Solution at a node

Deterministic Approach

The given gate-level netlist is described as a Directed Acyclic Graph (DAG) where each gate is represented as a node and the nets are represented as directed edges in the graph that connect any source gate to its fanout(s). A virtual sink node is also added that has incoming edges from all the primary outputs.

A popular dynamic programming approach to solve the dual V_{th} assignment problem, traverses the nodes in topological order from the primary inputs to the primary outputs [59]. Each node n_i contains a set of pareto-optimal solutions as it is encountered in the topological traversal. The j-th pareto optimal solution, denoted by $S_j^{(n_i)}$, contains the V_{th} assignment to all the nodes that are located in the fanin subtree of n_i, including n_i itself. In addition, $S_j^{(n_i)}$ contains the arrival time at the node output denoted by $D_j^{n_i}$, and the estimated leakage of the node subtree which includes n_i, denoted by $I_j^{n_i}$. (Figure 3.7 illustrates this). For solution j of node n_i, the signal arrival time and the leakage of the node subtree are respectively denoted by $d_j^{(n_i)}$ and $c_j^{(n_i)}$. During the topological traversal, the co-optimal solution set of the current encountered node denoted by n_i, is determined in the following 3 steps: 1) Initially the solutions of the children (fanins) of the node are combined to generate a new solution set. 2) This solution set is then combined with the two possibilities of V_{th} choices of n_i, therefore it's size is doubled. 3) The resulting solutions are then compared and the sub-optimal solutions are removed.

In step 1, every solution combination of the children of n_i is considered. For any combination, the arrival time is computed as the maximum of the arrival times of the node's children. The leakage is the summation of the leakage of the node's children for that combination. The assigned V_{th}s, for the gates in the subtree of n_i, is the union of the V_{th} assignments of all the children's solutions. This resulting solution set is denoted by $S^{(n_i)}_{fanin}$. Note that the number of solutions in $S^{(n_i)}_{fanin}$ is the multiplication of the number of solutions of n_i's children.

The resulting solution set is combined with the two possibilities of the V_{th}s of n_i in step 2. For each solution $S_j \in S^{(n_i)}_{fanin}$, a new solution is generated. For this new solution, the arrival time is the summation of the arrival time of S_j and the delay of n_i for that V_{th} (determined by equation 3.53). The leakage is the summation of the leakage of S_j and n_i for that V_{th} (determined by equation 3.52). Finally the union of the V_{th}s of S_j and the V_{th} of n_i will be the V_{th} assignment for the nodes in the fanin subtree of n_i. The number of solutions at this point is twice the size of $S^{(n_i)}_{fanin}$.

The solution set generated in step 2 is then evaluated such that sub-optimal solutions are pruned out. This is a necessary step, otherwise the number of solutions will grow exponentially, resulting in impractical runtime. In the pruning step, the number of stored solutions is set to be proportional to the summation of the number of the solutions of the node's children plus 2 (for the two V_{th} choices of n_i). The stored pareto-optimal solutions are determined as follows. Among all solutions that have the same arrival time, the one with minimum leakage is a pareto-optimal solution and the rest are sub-optimal, thus are pruned out. Please note that this a purely heuristic approach.

In the end, in the virtual primary output node, among all solutions that have an arrival time smaller than the given timing constraint, the one with minimum leakage is selected.

3.6.2 Variability-Driven Implementation and Algorithm

The manufacturing process causes variations in different chip parameters such as the effective channel length or oxide thickness, which randomizes the timing and leakage of each solution. The leakage of each solution is presented in [70] as function of such random variables. The Taylor series of the leakage expression is [16]:

$$I(X_1, .., X_n) = I_{s0} \frac{W_{eff_i}}{a_{i0}} e^{\frac{V_{igs} - V_{ith0}}{nVT}} (1 + \frac{(\beta - 1)}{2a_{i0}} \sum_{j=1}^{n} a_{ij} X_j + ...) K \tag{3.52}$$

In the above equation, the X_i random variables are independent random variables that reflect variability in effective channel length and threshold voltage. They were obtained using a Principal Component Analysis on correlated global variables.

The subthreshold leakage current of each gate for each threshold voltage choice is written in the form above. The rest of the parameters are constant coefficients defined in [16, 70]. For leakage the Taylor series was expanded until the second degree. Similarly [12] represents delay as a linear expression. This linear expression is obtained from the Taylor series expansion of the delay expression:

$$D_{X_1,\ldots,X_n} = \frac{C_L V_{dd}}{(V_{dd} - V_{ith0})^{\alpha}}(1 + \frac{\alpha\beta}{a_{i0}(V_{dd} - V_{ith0})}\sum_{j=1}^{n} a_{ij}X_j) \tag{3.53}$$

With polynomial representation of leakage and delay, the variability-driven dual-V_{th} leakage optimization is defined as: Given a gate-level netlist, a constraint T_{cons} for the arrival time at the primary outputs, two choices of threshold voltages of the gates and a maximum allowed timing violation probability, denoted by P_{viol}, under variability decide one threshold voltage for each gate, such that the expected value of the subthreshold leakage current is minimized while the probability of violating the timing constraint is at most P_{viol}.

For each solution S_i, the corresponding arrival time T_i and leakage I_i are polynomially-expressed random variables. In the variability-driven dual-V_{th} assignment, the probability that the arrival time T_i, meets the given timing constraint T_{cons}, should be less than P_{viol}, for any selected solution. Moreover the superior solution is the one with minimum expected leakage.

The algorithm for the variability-driven problem is similar to the traditional method. At each node, the steps in the traditional method are applied to combine the solutions of the node's children. During these steps, the summation and max operations conducted on leakage and arrival times of the solutions, are performed statistically on random variables. The summation of two random variables expressed as polynomials, is a new polynomially-expressed random variable, obtained by adding the corresponding coefficients of the polynomials. The statistical max operation on two linear delay expressions is based on [12].

At each node, the timing and leakage random variables of the solutions are compared based on equation 3.2. *The superiority probability among any pair of solutions is computed, and any solution that is pruned out by another solution with a probability of more than 0.95 is removed.* In the end, at the primary output, among all solutions that violate the timing constraint with probability smaller than P_{viol}, the one with minimum expected leakage is chosen. Next, the results of using these techniques were presented for computing the superiority probability for this application.

3.6.3 Results

The proposed methods to compute the superiority probability were applied and evaluated in the dual-V_{th} leakage optimization framework, using the SIS [52] programming environment and MCNC benchmarks. A variability of 15% was assumed, with respect to the mean value, in the effective channel length of a gate, and consequently in its threshold voltage [70]. Next polynomial expressions for delay and leakage of a gate were obtained using equations 3.52 and 3.53. Four principal components (variables in equation 3.52 and 3.53) were assumed to represent these variations, each with a Standard Normal distribution (mean=0 and variance=1).

Initially the accuracy and speedup in computation of the superiority probability were investigated using different techniques. In the simulations 2600 solutions pairs from the dual-V_{th} framework were evaluated for the MCNC benchmarks. The superiority probability was computed for each solution pair using the bivariate nor-

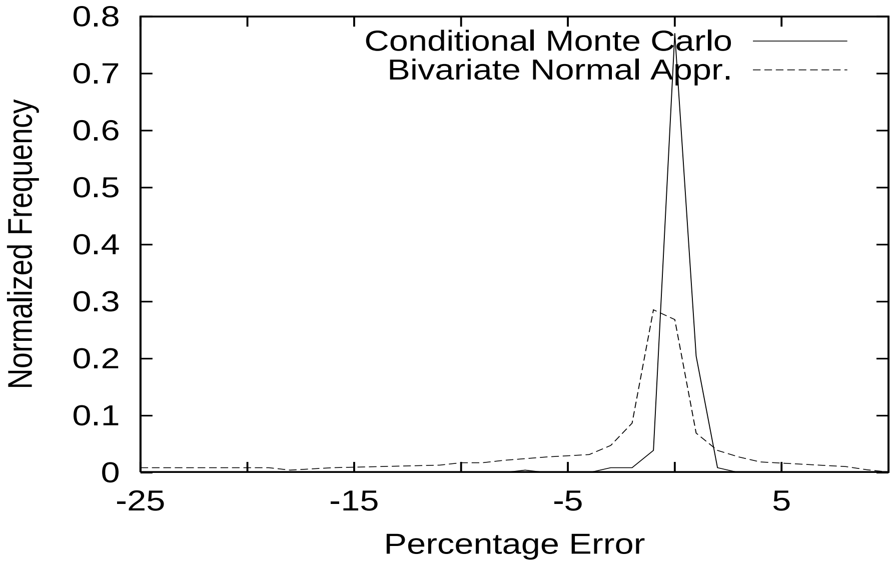

Figure 3.8: Normalized frequency of %error in computation of the superiority probability.

mal approximation of $jpdf$, and the Conditional MC technique, and compared the accuracy and speedup with MC simulation.

Figure 3.8 shows the normalized frequency of percentage error in the superiority probability, for the two applied methods. The figure shows the high accuracy of the CMC method. In fact the small error associated with CMC is due to the round-off error in computation of the bounds. The bivariate normal approximation has higher associated error when compared to MC simulation.

Figure 3.9 shows the normalized frequency of the speedup of the two techniques when compared to MC simulation. The speedup of the bivariate normal approximation is extremely high, because the probability integral expressed as a series in equation 3.41 is computed for only 5 terms. However the speedup of the CMC method is also significant (on average 25 times), when compared to the MC

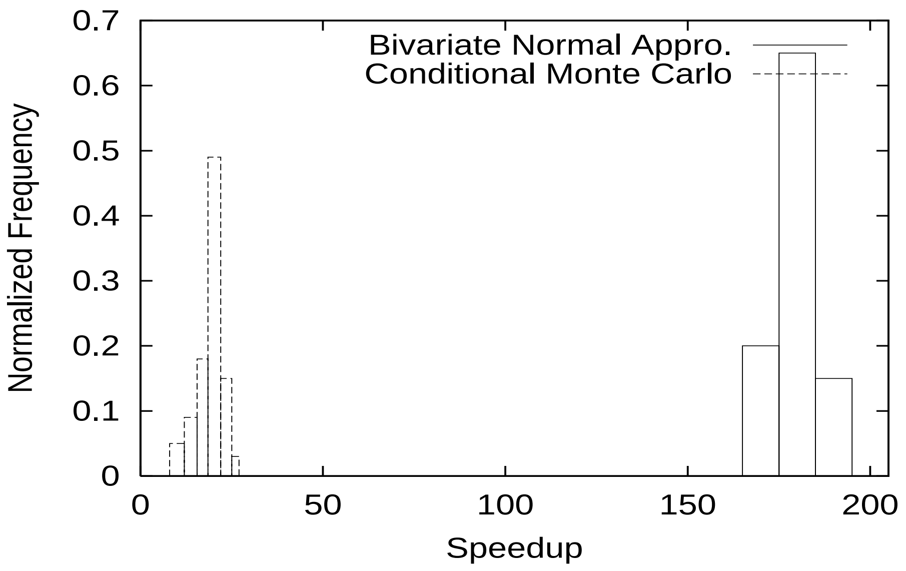

Figure 3.9: Normalized frequency of speedup in computation of the superiority probability.

simulation. This is because, the lower and upper bounds defined in different regions were predicting the behavior of the polynomials extremely well. On average in 92.7% of the cases, the T and C expressions were not needed for evaluation.

Next the final solutions in the dual-V_{th} framework were compared when different proposed methods were used to compute the superiority probability. A maximum allowed risk of timing constraint violation (P_{viol}) of 0.3 was assumed. Also worst-case deterministic approach was evaluated, with the proposed techniques.

In the deterministic approach, the delay and leakage of each gate was approximated with its worst-case value (expected value + 3-sigma) and performed deterministic optimization as explained in section VA. Then, a statistical analysis was done based on the assumed variability for the Vth assignment generated by the worst-case deterministic approach.

	T_{cons}	Worst-Case Deterministic			*jpdf* Appr.				Conditional MC			
	(nsec)	$E[I]$	$P_v(T)$	t	$E[I]$	%imp	$P_v(T)$	t	$E[I]$	%imp	$P_v(T)$	t
C432	33.0	10634	0.11	10	6910	35	0.27	13	5701	46	0.25	552
C499	17.5	14285	0.14	29	12186	15	0.17	51	11082	22	0.14	1582
C880	32.0	16650	0.11	12	10092	39	0.29	14	8294	49	0.30	610
C1355	18.0	17182	0.08	40	12187	29	0.11	50	11082	36	0.09	1572
C1908	29.0	13768	0.13	37	9179	33	0.18	40	8417	39	0.16	1025
C3540	42.0	38561	0.18	123	29662	23	0.23	181	22427	42	0.22	23582
C5315	31.0	42032	0.12	160	40031	5	0.13	173	26723	36	0.16	21449
C6288	110.0	45343	0.19	1131	44454	2	0.19	1699	44403	2	0.19	10539
alu2	8.0	13340	0.03	13	10262	23	0.04	20	8682	35	0.03	753
alu4	12.0	23317	0.06	65	19540	16	0.07	70	13894	40	0.07	1525
dalu	27.0	35812	0.12	68	28462	21	0.15	104	21748	39	0.17	1419
Ave.			0.12			21.9	0.17			35.1	0.16	

Table 3.5: Comparison of quality of solution and runtime (sec).

Table 3.5 illustrates these results for the MCNC benchmarks. Here column 2 is the timing constraint (in nsec). For each method, the expected leakage ($E[I]$ in pA), probability of violating the timing constraint ($P_v(T)$) and the run time (in seconds) is reported in the table. For the *jpdf* and CMC methods, the %improvement in $E[I]$ was reported and compared to the worst-case deterministic approach. It can be seen that on average the *jpdf* and CMC methods result in 21.9% and 33.7% improvement in $E[I]$ respectively. In these simulations, because the maximum probability of timing violation is 0.3, there is no need to over-optimize the speed.

However the deterministic approach, over-estimates the timing and therefore over-optimizes the speed, which directly translates into more leakage. Therefore it results in a smaller probability of timing violation. However the timing violation probability of *jpdf* and CMC methods are both acceptable as they are within the maximum allowed risk of 0.3. The over-estimation in the worst-case deterministic approach is to the extent that the expected leakage values in this case are much higher than those generated by the other two methods.

The CMC method, results in solutions that have better cost ($E[I]$) and often smaller probability of violating the timing constraint when compared to *jpdf*. (on average about 17% additional improvement in $E[I]$ compared to *jpdf*). However the runtime of the *jpdf* approximation method is on average 29 times faster than CMC method. The MC method did not converge in reasonable time.

In this chapter comparison of solutions was shown to be done using the superiority probability as an evaluation metric in the presence of process variations. This was verified in the context of the buffer insertion problem where the timing and cost of each solution was modeled as a linear expression. Also the dual-V_{th} leakage optimization technique was studied, in which the timing and cost of each solution were modeled as polynomials. These applications although demonstrated the effectiveness of using the superiority probability to generate a good quality solution compared to a deterministic case, however they could not provably generate the optimal solution.

In fact another significant challenge of a variability-driven framework is guaranteeing that the final solution is indeed the optimal one. In the next chapter I will present an example of a popular design automation technique, namely the gate sizing problem, considering variations. This problem can provably generate the optimal solution in the presence of variations, and will be discussed in detail next.

Chapter 4

Variability-Driven Gate Sizing Formulation and Generalizations

In this chapter a very important design automation problem, namely the gate sizing problem is studied in the presence of process variations. Given a circuit described at gate-level (the types of each gate and their interconnections are known), the gate sizing problem decides a size-variable for each gate in order to minimize a cost function such as area of power, while meeting a performance requirement. The size-variable can be thought of as a scaling factor for the sizes of the transistors inside each gate. This problem has been modeled as a convex program many years ago when process variations were ignored [25, 51]. This chapter studies the effects of process variations on this conventional formulation in the context of the problem of the speed-binning which is defined as follows:

In high performance systems, fabrication variability results in a considerable spread in the frequency of the chips (about 30% according to [7]). In some cases, the chips that violate the timing constraint are simply discarded and in other cases they are sold at a loss. In the latter case, those chips that fail to meet the nominal frequency after fabrication are binned based on their speed. Some work such as [49] design hardware to do speed binning in microprocessor design. For each speed bin a loss value exists for selling the chips in that bin for a reduced price. Therefore, depending on the spread in the circuit delay, there exists a binning yield-loss.

In this chapter minimization of the binning yield-loss is studied in the context of the gate sizing problem. Many researchers have investigated the gate sizing problem from a fabrication-variability perspective [2, 5, 13, 28, 43, 55, 57]. These approaches could be grouped into worst case approaches [55], sensitivity-based approaches [2, 28, 13, 57], and the ones based on a mathematical programming framework [5, 43]. These approaches try to addresses different objectives under variability. For example, [55] minimizes area while considering the worst case uncertainty ellipsoid of parameter variations in a convex formulation. Others minimize the yield-loss [2, 57] or leakage power [5, 43] or combination of both [13].

Firstly none of these approaches consider the binning yield-loss and focus on more traditional definitions of yield. Secondly these approaches they do not guarantee convergence to the optimal solution in a general case, or at least not from a yield perspective. Some of these approaches may converge to the optimal for their own problem specification but that may not lead to the optimal solution from a yield perspective. For example, the worst case approaches like [55], although look promising do not guarantee optimality of the yield function.

The sensitivity-based approaches optimize the cost function in a neighborhood and do not guarantee convergence to the optimal. The mathematical programming approaches do consider optimality but make constraining assumptions like the Gaussian nature of uncertainty [43] or work with specific models of fabrication variability [5]. Also the approach of [5] approximates the yield percentiles by their upper bounds, and thereby it is not provably optimal. In this chapter, a gate sizing approach is presented to optimize the binning yield.

The specific contributions of this work are enumerated below:

1. We optimize the binning yield and propose an optimal algorithm to minimize the same using gate sizing. Our algorithm can be trivially extended to minimize the binning yield under area/power constraints as well. The core of our algorithm is based on the proof of convexity of the binning yield function w.r.t. gate sizes, which allows usage of various convex optimization schemes.

2. The proof of convexity and consequently the optimality of the algorithm is not constrained by any assumptions on the underlying nature of the fabrication variability and/or the model of correlation used.

3. We use Kelley's Cutting Plane algorithm [8] to optimize the binning yield function. Usage of this scheme allows the integration of our approach with any of the existing statistical timing analysis (STA) methods (Gaussian [65] or Non-Gaussian [12], [37]).

4. In case the objective is optimizing the traditional yield, our binning yield-based approach could be used as a heuristic to optimize this objective. We prove that *if there exists a solution in which the traditional yield-loss is 0, our binning yield-loss approach will find such solution. Also, if the optimal value of the binning yield-loss is non zero, then there does not exist a solution to the traditional yield problem in which the yield loss is 0.* This is an important result if we are interested in generating solutions that have a yield-loss of 0.

In this chapter, initially the traditional gate sizing formulation is defined, and then the impact of process variations on the formulation is shown. Then a variation-aware objective is defined that represents the binning yield-loss, and then a convex formulation is presented that describes minimization of the binning yield-loss using gate sizing. One possible method to solve this formulation is discussed in this chapter, while exploring other methods could be pursued as one future direction.

In the simulations comparison was made to a sensitivity-based alternative to solve this problem, and an improvement of 72% in the binning yield-loss was obtained with a small area overhead of on average 6%, while achieving a 2.69 times speedup.

4.1 Preliminaries

4.1.1 Conventional Gate Sizing Formulation

Let s_i denote the size variable of gate i. The variable s_i is proportional to the channel width of the gates' transistors as the channel lengths are usually kept uniform. Let t_i denote the arrival time at the output of gate i from the primary inputs, and d_i denote the delay of gate i. The gate sizing problem is formulated as:

$$Minimize \;\; \textstyle\sum_{\forall gate \; i} c_i \times s_i$$

$$Subject \;\; to: \begin{cases} t_j + d_i(\vec{s}) \leq t_i \;\; \forall j \in fanin(i); \forall i \\ t_i \leq T_{cons} \;\; \forall i \in PO \\ s_{min} \leq s_i \leq s_{max} \quad \forall gate \;\; i \end{cases} \tag{4.1}$$

These constraints ensure that the delay of any path in the circuit is at most T_{cons}. The objective is minimizing the area of the circuit given as summation of s_i variables with a c_i proportionality factor. The solution is the set of gate sizes given as $\vec{s} = \{s_1; s_2; ...; s_n\}$.

Minimizing area while meeting a timing constraint is a common gate sizing objective [55, 28]. Other works optimize the yield-loss [2, 57, 13], or power [43, 5] using gate sizing. The formulation could also be written so as to find the feasible solution for a given timing constraint. The delay of gate i depends on its size and of its fanouts sizes. In the above constraints, this dependence is shown as $d_i(\vec{s})$. Therefore the objective and the arrival times in the formulation also depend on $\vec{s}$.

Computing Delay of A Gate As A Posynomial

The delay of a gate can be written as a posynomial function of its transistors sizes using the Elmore delay model [25, 51]. Each transistor is represented using an equivalent on-resistor (r), and a capacitor (c) given as a function of its channel width (w) as [48]:

$$\begin{aligned} r &= k_r \tfrac{1}{w} & \qquad k_r &= f_r(v_{th}, l_{eff}, v_{dd}, t_{ox}) \\ c &= k_c w & \qquad k_c &= f_c(l_{eff}, t_{ox}) \end{aligned} \tag{4.2}$$

where k_r and k_c are *positive constants* that are expressed as functions of parameters such as threshold voltage, effective channel length, supply voltage or oxide thickness as expressed above. The delay of each gate is the time to charge/discharge the capacitors in the resistive path to vdd/ground.

Using the Elmore model, this delay is written as a posynomial function of the resistors and capacitors in the gate and of the capacitors of the gates' fanouts. Given that s_i is proportional to the channel widths of the gates' transistors, the delay of a gate i is expressed as [51]:

$$d_i(\vec{s}) = a_{0i} + a_{1i}\frac{\sum_{\forall j} s_j}{s_i} \quad j \in fanout(i) \tag{4.3}$$

In the above posynomial expression, a_{0i} and a_{1i} are *positive constants* that depend on k_r and k_c values of the transistors. The inequalities of 4.1 will therefore be a posynomial formulation.

4.1.2 Convex Representation

The presented posynomial formulation is translated into a convex one by the change of variable $s_i = e^{x_i}$ [8]. Therefore $\vec{s} = \{e^{x_1}; e^{x_2}; ...\}$. The formulation in inequalities of 4.1 is then transformed to:

$$Minimize \ \ \sum_{\forall gate \ i} c_i \times e^{x_i}$$

$$Subject \ \ to : \begin{cases} t_j + d_i(\vec{x}) \leq t_i \ \ \forall j \in fanin(i) \\ t_i \leq T_{cons} \ \ \forall i \in PO \\ s_{min} \leq e^{x_i} \leq s_{max} \quad \forall gate \ \ i \end{cases} \tag{4.4}$$

The above formulation has been shown to be convex with respect to $\vec{x}$ [51].

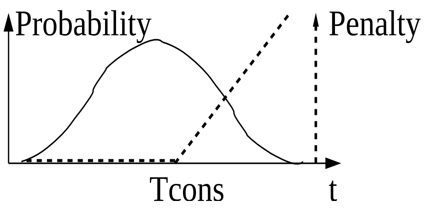

Figure 4.1: Binning yield-loss based on a linear penalty function.

4.2 Objective: Minimizing the Binning Yield-Loss (BYL)

In high performance systems, fabrication variability results in a considerable spread in the frequency of the chips (about 30% according to [7]). The chips that have a frequency lower than the nominal frequency can either be discarded, or be sold at a loss. For the latter case, the chips that violate the timing constraint are sorted (binned) according to their speed. [49] is a recent work which presents the hardware for doing this speed-binning. Depending on the degree of timing constraint violation for each bit, the chips are sold at a loss. This loss is defined by a penalty function; slower a chip, higher its penalty and loss. All the chips of at least the nominal speed will not have any penalty.

Let t denote the delay of a chip. Let us define a linear penalty function as follows:

$$penalty(t) = \begin{cases} t - T_{cons}; & t \geq T_{cons} \\ 0; & else \end{cases} \tag{4.5}$$

where T_{cons} is the timing constraint (nominal delay) that the chips are designed for. The chips that have a delay larger than T_{cons} have a penalty equal to their delay-offsets from T_{cons}. This linearity assumption will be relaxed later. Let $f_T(t)$ denote the probability density function (pdf) for the potential delay values of a design.

For the above penalty function, the overall *binning yield-loss* (BYL) is defined as follows:

$$BYL = \int_{-\infty}^{\infty} penalty(t) f_T(t)dt = \int_{T_{cons}}^{\infty} (t - T_{cons}) f_T(t)dt \tag{4.6}$$

In this chapter, the objective is to minimize BYL based on the penalty function of equation 4.5. An optimal and efficient algorithm to minimize this objective is presented. The optimality of the proposed approach holds even if the penalty function is convex (and not necessarily linear).

The delay of a design and consequently our objective can be expressed in terms of the gate sizes, among other parameters:

$$BYL(\vec{s}) = \int_{T_{cons}}^{\infty} (t(\vec{s}) - T_{cons}) f_T(t(\vec{s}))dt \tag{4.7}$$

where $\vec{s}$ is a vector of the gate sizes in the design. In this chapter optimization of $BYL(\vec{s})$ is done over $\vec{s}$ (using gate sizing).

Most of the exiting related work have focused on gate sizing to minimize the yield-loss (YL) under fabrication variability [57], [13], where the YL is given by:

$$YL = \int_{T_{cons}}^{\infty} f_T(t)dt \tag{4.8}$$

4.3 Gate Sizing for Minimizing the BYL

In this section, minimization of the BYL over the gate sizes is shown to be *optimally* achieved. Initially the effects of fabrication variability on the traditional formulation is discussed, and then the proposed method is discussed.

4.3.1 Effects of Variability on the Traditional Formulation

Fabrication variability randomizes different device parameters such as L_{eff} or T_{ox} etc.. The resistance and capacitance of a device expressed in equation 4.2 will therefore be a random variable (r.v.), as they are expressed in terms of such varying parameters.

Assume $\vec{\Psi}$ is a random vector which represents a set of varying parameters in equation 4.2 such as the effective channel length (L_{eff}) or oxide thickness (T_{ox}). Each sample vector $\vec{\psi} \in \vec{\Psi}$ represents a set of samples from the assumed field of uncertainty (which can have any associated density function and any correlation).

In equation 4.3, the coefficients of the delay expression of each gate become r.v.s, and are represented as $a_{0i}(\vec{\Psi})$ and $a_{1i}(\vec{\Psi})$. In equation 4.1 the delay of gate i also becomes a r.v.:

$$d_i(\vec{x}, \vec{\Psi}) = a_{0i}(\vec{\Psi}) + a_{1i}(\vec{\Psi}) \frac{\sum_{\forall j} e^{x_j}}{e^{x_i}}$$

4.3.2 Minimizing BYL: Mathematical Formulation

Under fabrication variability our objective to minimize the BYL can be formulated in terms of $\vec{x}$ (defined in section 4.1.2) as:

$$Minimize \;\; BYL(\vec{x})$$

$$Subject \;\; to: \begin{cases} t_j + d_i(\vec{x}, \vec{\psi_0}) \leq t_i \;\; \forall j \in fanin(i); \forall i \\ t_i(y_i) \leq T_{cons} \;\; \forall i \in PO \\ s_{min} \leq e^{x_i} \leq s_{max} \quad \forall gate \;\; i \end{cases} \tag{4.9}$$

In the above formulation $\vec{\psi_0}$ represents the nominal value of $\vec{\Psi}$ assuming no variations. The delay of each gate $(d_i(\vec{x}, \vec{\psi_0}))$ is also at its nominal value. The above formulation therefore ensures that T_{cons} is satisfied in the nominal case.

If the goal is to also have a small area, an upper bound on the overall area can be added as a new constraint: $\sum c_i e^{x_i} \leq A_{max}$.

4.3.3 A Two-Stage Stochastic Programming Formulation

In the above formulation BYL is a function of $\vec{x}$. To elaborate the consideration for variability in the objective function, let us define the following r.v.:

$$V(\vec{x}, \vec{\Psi}) = \begin{cases} T(\vec{x}, \vec{\Psi}) - T_{cons}; \quad T \geq T_{cons} \\ 0; \quad else \end{cases} \tag{4.10}$$

where $T(\vec{x}, \vec{\Psi})$ is a r.v. that represents the delay of the design. This r.v. depends on both the gate size vector $\vec{x}$ and also the random field $\vec{\Psi}$. The r.v. $V(\vec{x}, \vec{\Psi})$ represents the degree of violating T_{cons}. For a given value of $\vec{x}$, the pdf of V can be written in terms of the pdf of the delay of the circuit $f_T(t)$:

$$f_V(v) = \begin{cases} f_T(t); \quad v > 0 \\ \int_{-\infty}^{T_{cons}} f_T(t)dt; \quad v = 0 \end{cases} \tag{4.11}$$

Note that both $f_T(t)$ and $f_V(v)$ are functions of $\vec{x}$. Now the objective in equation 4.6 can be expressed in terms of V as:

$$BYL(\vec{x}) = \int_{T_{cons}}^{\infty} (t - T_{cons}) f_T(t)dt = \int_{-\infty}^{\infty} v f_V(v)dv = E[V] \tag{4.12}$$

Since both $f_T(t)$ and $f_V(v)$ are functions of $\vec{x}$, so will BYL be.

Also as illustrated, minimizing the BYL can be thought of minimizing the expected value of violating the timing constraint. Now let $v(\vec{x},\vec{\psi})$ be the value for V for a given $\vec{x}$ and a sample $\vec{\psi}$ from the field of uncertainty. Equation 4.12 can be written as:

$$BYL(\vec{x}) = \int_{-\infty}^{\infty} v(\vec{x},\vec{\psi}) f_{\vec{\Psi}}(\vec{\psi}) d\vec{\psi} \tag{4.13}$$

where $f_{\vec{\Psi}}(\vec{\psi})$ is the pdf of $\vec{\Psi}$. Note that this is just another way of understanding BYL. No approximation has been done and no assumption has been made on the nature of the variabilities and their correlations. Therefore equation 4.13 states that for a known $\vec{x}$ the corresponding $BYL(\vec{x})$ can be found by finding the $E[V(\vec{x},\vec{\Psi})]$ for all values $\vec{\psi}$ of $\vec{\Psi}$.

Conceptually $v(\vec{x},\vec{\psi})$ is the degree of violating the delay constraint for a given choice of $\vec{x}$ and a sample $\vec{\psi}$. This itself can be written as a convex program as follows:

$$v(\vec{x},\vec{\psi}) = Minimize \;\; q$$

$$Subject \;\; to: \begin{cases} t_j + d_i(\vec{x},\vec{\psi}) \le t_i \;\; \forall j \in fanin(i); \forall i \\ t_i \le T_{cons} + q \;\; \forall i \in PO \\ q \ge 0 \end{cases} \tag{4.14}$$

Solving this formulation results in the arrival times of the gates t_i with the gate delays $d_i(\vec{x},\vec{\psi})$. The optimal value of q denoted by q^* is the degree of delay violation for a fixed $\vec{x}$ and $\vec{\psi}$.

This falls within the classic formulation of *Two-Stage Stochastic Programming* [66]. The optimization problem given by 4.9 is called the first stage problem and the one given by equation 4.14 is called the second stage problem. The region of feasibility for the first stage problem is a convex set (since it simply comprises of a set of convex function constraints). The objective BYL is the expected value of a random variable V which depends on ($\vec{x}$ and $\vec{\psi}$) according to the optimization set of 4.14. In the next subsection we will prove that $E[V]$ is a convex function of $\vec{x}$. In doing so we will extend the classic Two-Stage Stochastic Programming theory to incorporate convex first and second stage problems. The traditional theory was valid only for linear programs [66].

Please note that our presented formulation does not make any specific assumptions about the distribution of $\vec{\Psi}$ and the correlation of components of $\vec{\Psi}$.

4.3.4 Proof of Convexity of the Optimization Set

In this section we will prove that the formulation of the inequalities of 9 is convex. To do this it is sufficient to show the optimization's objective ($BYL(\vec{x})$) is convex, as the constraints in equation 4.9 can be represented in an exponential form similar to section 4.1.2 and therefore will be a convex set [8].

Theorem: $BYL(\vec{x})$ is convex.

Proof: As shown in equation 4.13, $BYL(\vec{x}) = E[V(\vec{x}, \vec{\Psi})]$. The $E[.]$ can be thought of the weighted summation of all the samples $v(\vec{x}, \vec{\psi})$ of V. The weights are the probability values $f_{\vec{\Psi}}(\vec{\psi})$ that are always positive. Therefore we will show that any $v(\vec{x}, \vec{\psi})$ is individually a convex function of $\vec{x}$ to conclude that the $BYL(\vec{x})$ is convex, because the summation of positively weighted convex functions is convex.

To show $v(\vec{x}, \vec{\psi})$ is a convex function we need to show for $\vec{x_1}$ and $\vec{x_2}$, the following inequality holds (for $0 \leq \theta \leq 1$) [8]:

$$v(\theta\vec{x_1} + (1-\theta)\vec{x_2}, \vec{\psi}) \leq \theta v(x_1, \vec{\psi}) + (1-\theta)v(\vec{x_2}, \vec{\psi}) \tag{4.15}$$

where $v(x_1, \vec{\psi})$ and $v(x_2, \vec{\psi})$ are the optimal solutions of the optimization set expressed by the inequalities of 4.14. The constraints in the inequalities of 4.14 are written for $\vec{x_1}$ and $\vec{x_2}$ as:

$$\begin{cases} t_j^{(1)} + d_i(\vec{x_1}, \vec{\psi}) \leq t_i^{(1)} \\ t_i^{(1)} \leq T_{cons} + q^{(1)} \\ q^{(1)} \geq 0 \end{cases} \quad \begin{cases} t_j^{(2)} + d_i(\vec{x_2}, \vec{\psi}) \leq t_i^{(2)} \\ t_i^{(2)} \leq T_{cons} + q^{(2)} \\ q^{(2)} \geq 0 \end{cases} \tag{4.16}$$

Let $\{\vec{t}^{*(1)}, q^{*(1)}\}$ and $\{\vec{t}^{*(1)}, q^{*(2)}\}$ be the optimal solutions of the left and right inequalities respectively. Multiplying the left inequalities by θ and the right ones by $(1-\theta)$ (for $0 \leq \theta \leq 1$) and adding the corresponding inequalities we get:

$$\begin{cases} (\theta t_j^{*(1)} + (1-\theta)t_j^{*(2)}) + (\theta d_i(\vec{x_1}, \vec{\psi}) + (1-\theta)d_i(\vec{x_2}, \vec{\psi})) \\ \leq \theta t_i^{*(1)} + (1-\theta)t_i^{*(2)} \\ \theta t_i^{*(1)} + (1-\theta)t_i^{*(2)} \leq T_{cons} + (\theta q^{*(1)} + (1-\theta)q^{*(2)}) \\ \theta q^{*(1)} + (1-\theta)q^{*(2)} \geq 0 \end{cases} \tag{4.17}$$

Since $d_i(\vec{x}, \vec{\psi})$ is convex in $\vec{x}$, we have:

$$d_i(\theta\vec{x_1} + (1-\theta)\vec{x_2}, \vec{\psi}) \leq \theta d_i(\vec{x_1}, \vec{\psi}) + (1-\theta)d_i(\vec{x_2}, \vec{\psi}) \tag{4.18}$$

Therefore inequalities of 4.17 can be written as:

$$\begin{cases} (\theta t_j^{*(1)} + (1-\theta)t_j^{*(2)}) + d_i(\theta \vec{x_1} + (1-\theta)\vec{x_2}, \vec{\psi}) \\ \qquad \leq \theta t_i^{*(1)} + (1-\theta)t_i^{*(2)} \\ \theta t_i^{*(1)} + (1-\theta)t_i^{*(2)} \leq T_{cons} + (\theta q^{*(1)} + (1-\theta)q^{*(2)}) \\ \theta q^{*(1)} + (1-\theta)q^{*(2)} \geq 0 \end{cases} \tag{4.19}$$

Let us introduce $\vec{x_3} = \theta\vec{x_1} + (1-\theta)\vec{x_2}$ and $\{\vec{t}^{(3)} = \theta\vec{t}^{*(1)} + (1-\theta)\vec{t}^{*(2)}, q^{(3)} = \theta q^{*(1)} + (1-\theta)q^{*(2)}\}$. By replacing these definitions in the inequalities of 4.19 we will obtain:

$$\begin{cases} t_j^{(3)} + d_i(\vec{x_3}, \vec{\psi}) \leq t_i^{(3)} \\ t_i^{(3)} \leq T_{cons} + q^{(3)} \\ q^{(3)} \geq 0 \end{cases} \tag{4.20}$$

This implies that for $\vec{x} = \vec{x_3}$, the following set:
$\{\vec{t}^{(3)} = \theta\vec{t}^{*(1)} + (1-\theta)\vec{t}^{*(2)}, q^{(3)} = \theta q^{*(1)} + (1-\theta)q^{*(2)}\}$ is a feasible solution to the inequalities of 4.14. Therefore the optimal solution at $\vec{x} = \vec{x_3}$ must be smaller than (or equal to) $\theta q^{*(1)} + (1-\theta)q^{*(2)}$.

The optimal solution is nothing but $v(\theta\vec{x_1}+(1-\theta)\vec{x_2})$. Therefore, $v(\theta\vec{x_1}+(1-\theta)\vec{x_2}), \vec{\psi}) \leq \theta v(\vec{x_1}, \vec{\psi}) + (1-\theta)v(\vec{x_2}, \vec{\psi})$, and therefore v and consequently $E[V(\vec{x}, \vec{\Psi})]$ are convex in $\vec{x}$.

4.4 Some Generalizations

4.4.1 Generalized Penalty Function

The proof of convexity of our objective outlined in section 4.3.4, assumed that the penalty of violating the timing constraint is a linear function of the degree of violation (equation 4.5). If we redefine this penalty as follows:

$$penalty(t) = \begin{cases} p(T(\vec{x}) - T_{cons}); & t \geq T_{cons} \\ 0; & else \end{cases} \tag{4.21}$$

where p is any convex function (with respect to $\vec{x}$, then the convexity of the new BYL still holds and optimality can still be achieved. The reason is as follows:

- Given a convex penalty function in the above form, the BYL can be expressed as: $BYL = \int_{T_{cons}}^{\infty} penalty(\vec{x}, \vec{\psi}) f_{\vec{\Psi}}(\vec{\psi}) d\vec{\psi}$.
- Similar to the previous proof, because $f_{\vec{\Psi}}(\vec{\psi})$ is a non-negative quantity, convexity of the BYL with respect to $\vec{x}$ depends only on the convexity of $penalty(\vec{x}, \vec{\psi})$ with respect to $\vec{x}$, and therefore as long as a convex penalty function is used, the same conclusions can be drawn.

4.4.2 Relation with Minimizing the Yield-Loss

The previous few sections discussed optimal minimization of BYL. From our simulations a high degree of correlation between optimizing BYL and Yield-Loss (YL) was found. In fact the proposed approach could be used as a heuristic for optimizing YL. But there are some important results that can be proved about the optimality of YL as illustrated below:

Theorem: The optimal BYL will be 0 iff the optimal YL is 0.

Proof: Let us suppose we have a solution for which $BYL = 0$. Referring to equation 4.12, this can happen only if $f_V(v) = 0$ for all v greater than (not equal to) 0. This means that the pdf of the timing of the circuit (for the given gate sizes) lies entirely within the timing constraint. Thus $YL = 0$. Now let BYL be more than zero, therefore $f_V(v)$ must have a positive value for some v greater than 0. Therefore, some part of the timing pdf must be greater than T_{cons}. Thus YL cannot be zero.

This is an important result, since by optimizing BYL we can 1)achieve a solution for which $YL = 0$, 2) or by looking at the optimal value of BYL check if a solution with $YL = 0$ exists.

4.4.3 Generalizing the BYL Definition

Our definition of the BYL can be extended to consider not only the loss associated with violating a timing constraint, but also other constraints that could be of importance.

For example BYL could be defined with respect to meeting a power and a timing constraint. Given a timing constraint T_{cons} and a power constraint P_{cons}, the BYL in this case could be expressed as:

$$BYL = \int_{T_{cons}}^{\infty} \int_{P_{cons}}^{\infty} penalty(T(\vec{x}), P(\vec{x})) f_{T,P}(t,p) dtdp \tag{4.22}$$

where $f_{T,P}$ is the joint density function of power and timing, as in general these two quantities might be correlated to each other. The penalty function could be defined in terms of power and timing, which both are ultimately a function of $\vec{x}$. The optimality results can all be extended to this case as long as the penalty function is convex with respect to $\vec{x}$.

4.5 Solving the Convex Formulation

In the previous sections the convexity of the proposed formulation to minimize the BYL was proven. This means that our formulation is optimally solvable using the convex optimization techniques. Kelley's Cutting Plane technique is used [8] (among other possible methods) to solve this formulation, which is explained below.

4.5.1 Kelley's Cutting Plane Algorithm

Kelley's algorithm is an iterative approach. At each iteration a linear lower bound of the convex objective is generated. This lower bound together with the lower bounds of the previous iterations develop a piecewise linear lower bound on the objective function. As the number of iterations increase, the linear lower bounds of the previous iterations converge to the accurate objective. At any iteration k, the objective function represented by the piecewise linear lower bounds is optimized while satisfying the feasibility criteria of the constraints. This gives us a solution vector x_k. At this point a new linear lower bound is computed for the true objective function and the entire process is repeated. These steps can be summarized in Algorithm 1:

Initially at Step 1 a feasible solution ($\vec{x}_1$) is found for the inequalities of 4.9. Kelley's algorithm follows an iterative approach: In the k^{th} iteration, the lower bound at $BYL(\vec{x}_{k-1})$ found in the previous iteration is used to generate a new solution $\vec{x}_k$. This lower bound is generated as follows: We find the sub-gradient $\alpha_k + \vec{\beta}_k.\vec{x}$ of the BYL function such that at $\vec{x} = \vec{x}_{k-1}$, $BYL(\vec{x}_{k-1}) = \alpha_k + \vec{\beta}_k.\vec{x_{k-1}}$

Algorithm 4 Kelley's Cutting Plane Algorithm

```
Step 1: Initialize
        Let $\epsilon > 0$ and $\vec{x}_1$ be a feasible solution satisfying the constraints.
        Let $k \leftarrow 0$ and define $l_0(\vec{x}) = -\infty$, $u_0(\vec{x}) = \infty$.
Step 2: Set $k \leftarrow k + 1$
Step 3: Define the Lower Bound at $\vec{x}_k$
        Evaluate $\alpha_k$ and $\vec{\beta}_k$ such that $l_k \geq \alpha_k + < \vec{\beta}_k, \vec{x} >$:
            $\alpha_k = BYL(\vec{x}_k) - \vec{\beta}_k \vec{x}_k \quad \vec{\beta}_k = \frac{\partial BYL(\vec{x})}{\partial \vec{x}}|_{\vec{x}_{k-1}}$
Step 4: Update the Optimization Set
        Add the following to the existing set of constraints:
            $l_k \geq l_{k-1} \quad l_k \geq \alpha_k + < \vec{\beta}_k, \vec{x} >$
        Update the objective function to $Minimize\ l_k$.
Step 5: Solve the Optimization to get $\vec{x}_k$ and Update the Bounds
        Let upper bound $u_k = Min\{u_{k-1}, BYL(\vec{x}_k)\}$ and lower bound $l_k$.
Step 6: Stopping Rule
        Stop if $u_k - l_k \leq \epsilon$, otherwise go to Step 1.
```

where $\vec{\beta}_k$ is conceptually the slope of the BYL function at $\vec{x} = \vec{x}_{k-1}$. By definition this sub-gradient is the linear lower bound of the BYL function. A new $\vec{x}_k$ is now chosen as follows: A new variable l_k is incorporated in the optimization framework which is constrained to be larger than all the lower bounds found so far. The actual objective is then replaced by l_k which approximates the BYL (Step 4 of Algorithm 1). This gives us a new value for $\vec{x}_k$ and the entire process is repeated till the lower bound approximation and the upper bound are within a user specified range of tolerance (note that each $\vec{x}_k$ corresponds to an upper bound $BYL(\vec{x_k})$). This approach *provably* reaches the optimal solution in convex optimization [8].

Next the application of statistical timing analysis (STA) will be explained within the Kelley's algorithm.

Please note that in case that the optimization of area and/or power is necessary, new constraints can be added to our formulation that bound the overall area or power. These can be expressed as convex constraints which allows the use of Kelley's Cutting-Plane algorithm to solve the new optimization formulation.

Also please note that Kelleys Cutting-Plane algorithm is one of the possible methods to solve the convex formulation. Our main contribution is the convex formulation, and once there is a provably convex formulation, all possible algorithms targeted for this class of problems can potentially be used. Kelleys algorithm is chosen as it integrates very well with Statistical Timing Analysis techniques as will be explained next.

4.5.2 Integration with STA

Computing the BYL

Given a gate-level circuit, statistical timing analysis can be used to efficiently compute the BYL. In section 4.3.3 parametric computation of BYL was explained over all samples $\vec{\psi}$ in $\vec{\Psi}$ and for a particular set of gate sizes using equation 4.13. It can also be equivalently obtained using equation 4.12. This is equivalent to doing an STA (for a given choice of gate sizes) on the circuit and then evaluating the expected value of violating the timing constraint in order to find the BYL (equation 4.12). Assuming variability in $\vec{\Psi}$, STA provides the spread of delay at the primary outputs (essentially the pdf $f_T(t)$) for a given $\vec{x}$. This STA can be done based on any possible approach such as [65] and [37].

Computing the Lower Bound in Kelley's Algorithm

The linear lower bound on BYL is expressed as $\alpha_k + < \vec{\beta}_k, \vec{x} >$ in Algorithm 1. As expressed in step 3 of the algorithm, $\vec{\beta}_k$ is found by evaluating the slope of the $BYL(\vec{x})$ at $\vec{x}_{k-1}$. The coefficient α_k is found such that $BYL(\vec{x}_{k-1}) = \alpha_k + < \vec{\beta}_k, \vec{x}_{k-1} >$. Therefore in order to find the lower bound, it is sufficient to show the computation of $\vec{\beta}_k$.

Finding the sub-gradient of a non-differentiable function is an important research problem. Many techniques have been proposed that can approximate the sub-gradient. In this chapter the finite-difference method is used [8].

The vector $\vec{\beta}_k = \{\beta_1; \beta 2; ...; \beta_n\}$, where β_i is the projection of $\vec{\beta}_k$ with respect to component x_i (or $\beta_i = \frac{\partial BYL(\vec{x})}{\partial x_i}|_{\vec{x}_{k-1}}$). In other words β_i expresses the sensitivity of the objective function with respect to x_i. We approximate this sensitivity as:

$$\beta_i = \frac{BYL(\{x_1; ...x_i; ...; x_n\}) - BYL(\{x_1; ...x_i + \Delta x_i; ...; x_n\})}{\Delta x_i}|_{\vec{x}_{k-1}} \tag{4.23}$$

Given an $\vec{x_{k-1}}$ vector, the sensitivity β_i is found using equation 4.23. Computation of BYL in the above equation can be done using STA as explained in the previous subsection. Therefore computation of β_i in the above equation requires doing two STAs for each component $\vec{x_{k-1}}$ vector, assuming x_i for gate i is slightly changed. The paper [13] proposes ways that allows the sensitivity to be more efficiently computed. Once $\vec{\beta}_k$ is found, α_k and consequently the lower bound are determined.

Note that the STA at any of these stages can be done using any of the proposed techniques in the literature such as [65] or [37], and can assume any distribution for $\vec{\Psi}$ *and any correlation model for its components.*

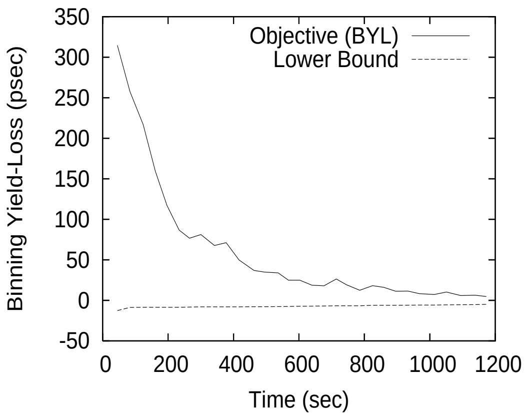

Figure 4.2: Convergence of BYL to its lower bound (C1908, T_{cons} =3500 psec).

4.6 Results

Our experiments were conducted on the ISCAS bench suite. Each benchmark was initially placed and correlation data between different gates were generated based on the model of [37]. Variability in the V_{th} was assumed for each device with a Normal distribution with a mean equal to the nominal value and a 12% standard deviation from the mean. A convex expression for the delay of each gate as a function of its size was determined assuming a 90nm technology (for which the information was obtained from [68]).

The proposed method was implemented in the SIS [52] framework and the MOSEK [47] convex optimization tool was also used. In the proposed method using the Kelley's algorithm, the STA method of [65] was used to compute BYL.

Figure 4.2 shows the values of our objective BYL and its lower bound as iterations progress. At each iteration the value of the objective corresponds to the upper bound of the optimal. Kelley's algorithm iteratively improves the lower bound till the lower and upper bounds converge. This algorithm guarantees optimality.

In order to make comparison with other methods, a sensitivity-based approach as well as a worst-case method were implemented. The sensitivity method had a framework as in [2] or [57]. In this method initially all the gates are set to their minimum size. The sensitivity-based method is a greedy iterative approach, in which at each iteration the most sensitive gate is determined and sized up. The most sensitive gate is the one that results in the maximum change in the objective due to a small change in its size. For comparison of this method, the objective of the sensitivity-based approach was set to be the BYL.

A worst-case deterministic approach was also implemented. The worst-case approach had a convex optimization framework similar to [51]. However the delay expression for each gate was computed assuming the value of V_{th} is fixed at its worst $(\mu + 3\sigma)$. In this approach the optimization objective was set to be minimization of the arrival time at the primary output nodes. A new constraint was also added to impose an upper bound on the maximum area of this approach. In order to make comparison with our proposed method, this maximum area of the worst-case approach was set to be the area of the optimal solution generated by the proposed approach.

Table 4.1 compares the BYL and area of these three methods for two different timing constrains for each benchmark. One of these timing constraints is more stringent than the other one. For the stringent timing constraint, the deterministic approach could not generate any solution as it was too pessimistic in approximating the delay of each gate and consequently of the timing constraint. For the more

bench	T_1					T_2						
	T_{cons}	Sensitivity		Kelley Convex		T_{cons}	Sensitivity		Worst-Case		Kelley Convex	
		BYL	$Area$	BYL	$Area$		BYL	$Area$	BYL	$Area$	BYL	$Area$
C17	210	21.60	353	6.83	369	300	0.00	365	0.00	321	0.00	342
C432	2500	252.47	10446	45.76	11504	3000	46.58	8908	1.73	8789	1.65	8789
C499	2300	32.73	15279	32.36	21869	2700	9.59	14408	1.46	13920	1.42	14684
C880	3150	226.05	13502	19.92	13336	3500	45.82	13935	1.80	13353	1.65	13485
C1355	2050	105.28	15821	17.79	21410	2300	32.73	15279	2.50	14977	1.59	14977
C1908	3000	327.94	18624	29.38	21812	3500	101.56	17139	1.95	18009	1.32	18009
C3540	4000	270.00	37547	76.67	37574	5500	8.66	36778	3.73	36728	3.72	36728
C5315	4000	105.92	50192	61.19	50138	5500	9.45	49661	8.59	49584	8.32	49596
C6288	15000	323.41	89201	181.65	88503	23000	8.77	87750	8.77	87750	8.77	87750
Ave.		185.04	27884	52.35	29613		29.24	27135	3.39	27047	3.16	27151

Table 4.1: Comparison of binning yield-loss (in psec) and area

relaxed timing, the worst-case approach however was able to generate solutions of good quality comparable to our method. Compared to sensitivity-based approach, an average of 72% improvement in the BYL was achieved with only a 6% area overhead given the stringent timing constraint. Also a better solution was generated when the timing constraint was relaxed.

Figure 4.3 shows the optimization of objective over time using our approach compared to the sensitivity-based method for C1908. It can be seen that our approach has clearly a faster convergence rate. In fact as the run-times are reported in Table 4.2, our method achieves an average of 2.69 speed up due to fewer number of iterations. Although each individual iteration takes longer in our method (as a convex optimization set needs to be solved at each iteration in our case), but due to the very few number of iterations, the overall run-time will be much smaller.

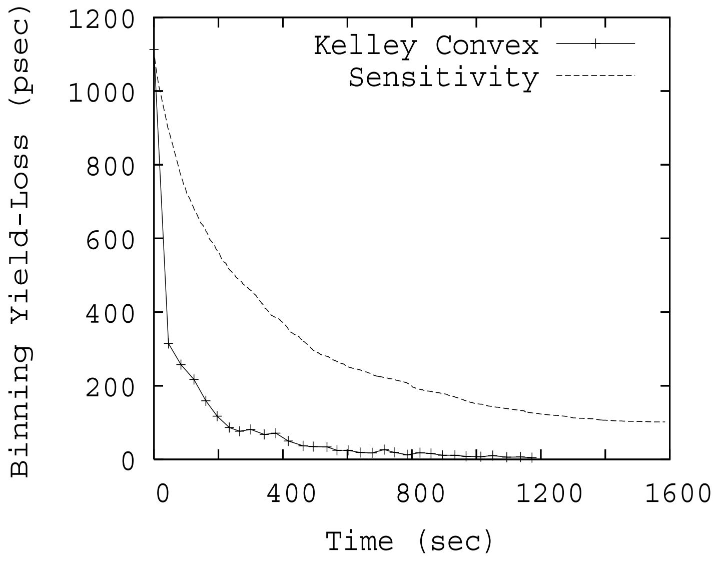

Figure 4.3: Binning yield-loss vs. time (C1908, T_{cons} =3500 psec).

The traditional Yield-Loss of the solution generated by the proposed approach was also compared to a sensitivity-based approach in which the most sensitive gate was defined as the one with maximum change in Yield-Loss due to the change in its size. Our method also improves the Yield-Loss on average by 61%. Table 4.3 shows in the information on yield-loss comparison.

Finally figure 4.4 shows the curve generated by our approach between the area and BYL. Each point corresponds to the solution of an iteration of Kelley's algorithm. It can be seen that as the iterations progress, increase in area results in a decrease in BYL.

bench	T_{cons}	Sensitivity		Kelley Convex	
		#itera.	time	#itera.	time
C17	210	73	0.06	4	1.33
C432	2500	1636	894.18	30	438.13
C499	2300	390	739.27	13	537.60
C880	3150	339	475.48	7	150.11
C1355	3000	390	772.82	7	216.99
C1908	3500	711	1585.47	31	1172.77
C3540	4000	120	2004.22	5	776.28
C5315	4000	164	5127.82	7	1870.41
C6288	15000	138	13616	4	1802.47

Table 4.2: Comparison of run-time (sec) and number of iterations

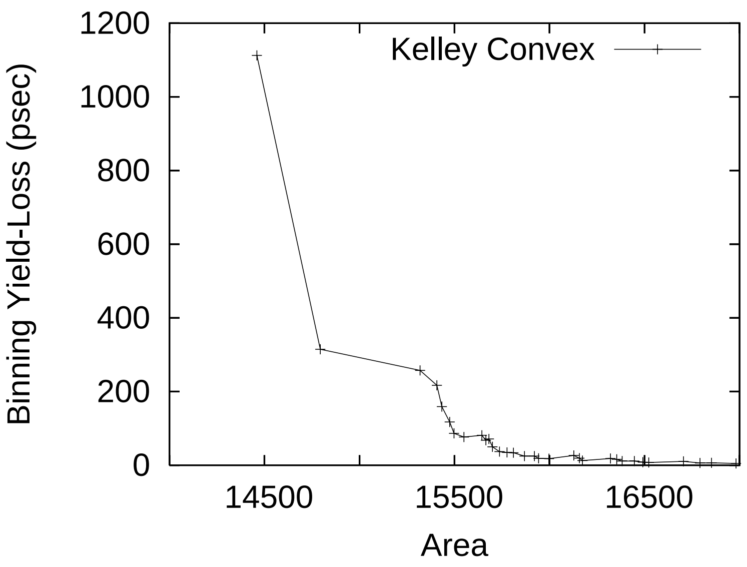

Figure 4.4: BYL vs. area generated at different iterations of Kelley's algorithm.

bench	T_{cons}	Sensitivity	Worst-Case	Kelley Convex
C17	210	0.75	N/A	0.40
C432	2500	0.76	N/A	0.28
C499	2300	0.23	N/A	0.23
C880	3150	0.72	N/A	0.15
C1355	2050	0.53	N/A	0.15
C1908	3000	0.82	N/A	0.18
C3540	4000	0.71	N/A	0.24
C5315	4000	0.33	N/A	0.18
C6288	15000	0.64	N/A	0.39

Table 4.3: Comparison of yield-loss.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this dissertation a variability-driven optimization framework is studied. In this framework the effects of process variations were considered by modeling the varying circuit parameters as random variables that in general might be correlated to each other.

A particular challenge in variability-driven design automation is to define optimality measures among candidate solutions, which allow for inferior solutions to be removed from the solution space thus reducing the run-time complexity. In this dissertation the superiority probability is introduced as such an optimality measure, and two methods are proposed to compute this probability: an accurate Conditional Monte Carlo simulation method, and an efficient moment-matching approximation method. The effectiveness of using the superiority probability is shown in the context of two important design automation applications: 1) the buffer insertion problem, 2) the dual-V_{th} leakage optimization problem.

Another important task in variability-driven design automation is to develop optimization techniques that provably converge to the optimal solution. One such optimization technique known as gate sizing was explored and its application to the domain of microprocessor speed-binning was illustrated.

The presented formulation, in contrast with the existing variability-driven approaches which are all based on heuristics, is provably optimal. Moreover, unlike existing approaches, it is independent of any assumption on the source and nature of variations.

5.2 Open Problems

5.2.1 Additional Speedup and Accuracy in Computing the Superiority Probability

In the proposed techniques to compute the Superiority Probability, the Conditional Monte Carlo method was on average 25 times faster than regular Monte Carlo simulation, but it is still considered to be slow when compared to the moment-matching technique. However the moment-matching technique lacks the desired accuracy. One interesting future direction is to explore new techniques that can compute the superiority probability faster than the Conditional Monte Carlo method, and yet more accurate than the moment-matching technique.

5.2.2 Extending the Definition of BYL to Consider the Joint-Timing and Power Loss

One of the interesting future directions in this problem is to investigate the case in which the BYL is defined as the joint-loss associated with a timing and a power constraint, as in practice those solutions that have a very fast timing have a very high power consumption, and vice versa. Therefore in the case that the goal is to find a medium point (reasonable timing with a reasonable power consumption), the stated joint BYL could be used.

5.2.3 Speedup in Solving the Convex Formulation

Another interesting direction is to explore different ways to speedup the algorithm used to solve the convex optimization. Currently Kelley's algorithm is used for this purpose.

Speeding Kelley's Algorithm

- Kelley's algorithm is an iterative method in which at each iteration a new optimization set is solved from scratch. However the two optimization problems associated with two consecutive iterations in Kelley's algorithm, only differ in one constraint. Using the result of the previous optimization round could potentially be very beneficial to speedup solving the next optimization problem. Therefore exploring the use of incremental techniques to solve Kelley's algorithm is one interesting future direction.

- In addition in each iteration of the Kelley's algorithm the sub-gradient of objective function is required. To find the sub-gradient, currently the sensitivity of the objective function is found with respect to the size of each individual gate. This means a separate Statistical Timing Analysis is done to evaluate the change of the objective function when the size of each gate is slightly changed.

The use of incremental timing analysis techniques could be very useful in this case, and might significantly speedup the computation of the sub-gradient.

Use of More Efficient Convex Solvers

Finally Kelley's algorithm does not have a good theoretical run-time complexity. The use of interior point methods that have a better run-time complexity, could be explored as another future direction.

BIBLIOGRAPHY

[1] Agarwal K., Sylvester D., Blaauw D. Variational delay metric for interconnect timing analysis. In *Proceedings of Design Automation Conference*, pages 381–384, June 2004.

[2] Agrawal A., Chopra K., Blaauw D., Zolotov V. Circuit optimization using statistical static timing analysis. In *Proceedings of Design Automation Conference*, pages 338–342, June 2005.

[3] Alpert C., Devgan A., Kashyap C. A two moment RC delay metric for performance optimization. In *Proceedings of Design Automation Conference*, pages 69–74, 2000.

[4] Alpert C., Devgan A., Quay S. Buffer insertion with accurate gate and interconnect delay computation. In *Proceedings of Design Automation Conference*, pages 479–484, 1999.

[5] Bhardwaj S., Vrdhula S. Leakage minimization of nano-scale circuits in the presence of systematic and random variations. In *Proceedings of International Conference on Computer-Aided Design*, pages 282–292, November 2005.

[6] Borkar S. Getting gigascale chips: challenges and opportunities in continuing Moore's Law. In *ACM Queue, Vol.1, No. 7*, pages 30–33, October 2003.

[7] Borkar S., Karnik T., Narendra S., Tschanz J., Keshavarzi A., De V. Parameter variations and impacts on circuits and microarchitecture. In *Proceedings of Design Automation Conference*, pages 338–342, June 2003.

[8] Boyd S., Vandenberghe L. Convex optimization. Cambridge 2004.

[9] Cargo G., Shisha O. The Bernstein form of a polynomial. In *J. Res. Nat. Bur. Standards*, pages 79–81, 70B 1966.

[10] Chang H., Sapatnekar S. Statistical timing analysis considering spatial correlations using a single pert-like traversal. In *Proceedings of International Conference on Computer-Aided Design*, pages 621–628, 2003.

[11] Chang H., Sapatnekar S. Full-chip analysis of leakage power under process variations, including spatial correlations. In *Proceedings of International Conference on Compound Semiconductor Manufacturing Technology*, pages 523–528, June 2005.

[12] Chang H., Zolotov V., Visweswariah C., Narayan S., Zhan Y. Parameterized block-based statistical timing analysis with non-Gaussian parameters and nonlinear delay functions. In *Proceedings of Design Automation Conference*, pages 71–76, 2005.

[13] Chopra K., Shah S., Srivastava A., Blaauw D., Sylvester D. Parameteric yield maximization using gate sizing based on efficient statistical power and delay gradient computation. In *Proceedings of International Conference on Computer-Aided Design*, November 2004.

[14] Clark C. The greatest of a finite set of random variables. In *Operations Research, Vol. 9*, pages 85–91, 1961.

[15] Davoodi A., Khandelwal V., Srivastava A. Variability inspired implementation selection problem. In *Proceedings of International Conference on Computer-Aided Design*, pages 423–427, 2004.

[16] Davoodi A., Srivastava A. Probabilistic dual-vth leakage optimization under variability. In *Proceedings of International Symposium on Low Power Electronics and Design*, pages 143–148, August 2005.

[17] Davoodi A., Srivastava A. Variability-driven buffer insertion considering correlations. In *International Conference on Computer Design*, pages 425–430, April 2005.

[18] Davoodi A., Srivastava A. Probabilistic comparison of solutions in variability-driven optimization. In *Proceedings of International Conference on Computer-Aided Design*, pages 17–24, April 2006.

[19] Davoodi A., Srivastava A. Variability-driven gate sizing for binning yield optimization. In *Proceedings of Design Automation Conference*, July 2006.

[20] Devadas S., Ghosh A,m Keutzer K. Logic synthesis. In *McGraw-Hill*, 1994.

[21] Devgan A., Kashyap C. Block based static timing analysis with uncertainty. In *Proceedings of International Conference on Computer-Aided Design*, page 607, June 2003.

[22] Doyle B., Datta S., Doczy M., Hareland S., Jin B., Kavalieros J., Linton T., Murthy A., Rios R., Chau R. High performance fully depleted tri-gate CMOS transistors. In *IEEE Electron Device Letters, Vol. 24, No. 4*, pages 263–265, April 2003.

[23] Elmore W. The transient response of damped linear networks with particular regard to wideband amplifiers. In *Journal of Applied Physics, vol. 19*, pages 5–63, January 1948.

[24] Ernst D., Das S., Lee S., Blaauw D., Austin T., Mudge T., Kim N., Flautner K. Razor: circuit-level correction of timing errors for low-power operation. In *IEEE Micro, Vol. 24, No.6*, pages 10–20, November 2004.

[25] Fishburn J., Dunlop A. TILOS: A posynomial programming approach to transistor sizing. In *Proceedings of International Conference on Computer-Aided Design*, pages 326–328, 1985.

[26] Ginneken L. Buffer placement in distributed RC-tree networks for minimal Elmore delay. In *Proceedings of International Symposium on Circuits and Systems*, pages 865–868, December 1990.

[27] Gupta S. Probability integrals of multivariate normal and multivariate t. In *Annals of Mathematical Statistics, Vol. 34*, pages 792–828, 1963.

[28] Guthaus M., Venkateswaran N., Visweswariah C., Zolotov V. Gate sizing using incremental parameterized statistical timing analysis. In *Proceedings of International Conference on Computer-Aided Design*, November 2005.

[29] Hisamoto D., Lee W.-C. Kedzierski J., Takeuchi H., Asano K., Kuo C., Anderson E., King T.-J., Bokor J., Hu C. FinFET-a self-aligned double-gate MOSFET scalable to 20 nm. In *IEEE Transactions on Electron Devices, Vol.47, No.12*, pages 2320–2325, December 2000.

[30] Huseby A., Naustdal M., Varli I. System Reliability Evluation Using Conditional Monte Carlo Methods. In *Statistical Research Report, No. 2*, 2004.

[31] International Technology Roadmap for Semiconductors. [Online]: http://www.itrs.net/Common/2005ITRS/Interconnect2005.pdf. page 9, 2005.

[32] J. Garloff, C. Jansson, A. Smith. An Improved Method for the Computation of Affine Lower Bound Functions for Polynomials. In *Frontiers in Global Optimization, Aegean Conferences Series*, 2003.

[33] Jolliffe I. Principal component analysis. In *Springer Series in Statistics, Second Edition.*

[34] Kay R., Pileggi L. PRIMO: Probability interpretation of moments for delay calculation. In *Proceedings of Design Automation Conference*, pages 463–468, June 1998.

[35] Khandelwal V., Davoodi A., Nanavati A., Srivastava A. A probabilistic approach to buffer insertion. In *Proceedings of International Conference on Computer-Aided Design*, pages 560–567, November 2003.

[36] Khandelwal V., Davoodi A., Srivastava A. Efficient statistical timing analysis through error budgeting. In *Proceedings of International Conference on Computer-Aided Design*, pages 473–477, November 2004.

[37] Khandelwal V., Srivastava A. A general framework for accurate satistical timing analysis considering correlations. In *Proceedings of Design Automation Conference*, pages 89–94, June 2005.

[38] Kuroda T. A 0.9 v 150 MHz 10MW $4mm^2$ 2-d discrete cosine transform core processor with variable threshold voltage scheme. In *Proceedings of IEEE Journal of Solid-State Circuits*, pages 1770–1779, Novemebr 1996.

[39] L. He, A. B. Kahng, K. Tam, J. Xiong. Simultaneous Buffer Insertion and Wire Sizing Considering Systematic CMP Variation and Random Leff Variation. In *Proceedings of International Symposium on Physical Design*, pages 78–85, April 2005.

[40] Li X., Le J., Gopalakrishnan P., Pileggi L. Asymptotic probability extraction of non-Normal distributions of circuit performance. In *Proceedings of International Conference on Computer-Aided Design*, pages 2–9, November 2004.

[41] Lin T., Acar E., Pileggi L. h-gamma: an RC delay metric based on Gamma distribution approximation to the homogenerous response. In *Proceedings of International Conference on Computer-Aided Design*, pages 19–25, November 1998.

[42] Liu F., Lillis J., Cheng C. Design and implementation of a global router based on a new layout-driven timing model with three poles. In *Proceedings of International Symposium on Circuits and Systems*, pages 1548–1551, 1997.

[43] Mani M., Devgana A., Orshansky M. An efficient algorithm for statistical minimization of total power under timing yield constraints. In *Proceedings of Design Automation Conference*, pages 309–314, July 2005.

[44] Miller D. Deterministic process control using a multivariate model. In *Proceedings of International Conference on Compound Semiconductor Manufacturing Technology*, April 2003.

[45] Mutah S., Douseki T., Matsuya Y., Aoki T., Shigematsu S., Yamada J. 1-V power supply high-speed digital circuit technology with multi threshold voltage CMOS. In *Proceedings of IEEE Journal of Solid-State Circuits*, pages 847–853, August 1995.

[46] Nassif S. Modeling and forecasting of manufacturing variations. In *Proceedings of Design Automation Conference*, pages 145–150, 2001.

[47] [Online]: http://www.mosek.com.

[48] Rabaey J., Chandrakasan A., Nikolic B. . In *Digital integrated circuits: a design perspective (2nd Edition).*

[49] Raychowdhury A., Ghosh S., Roy K. A novel on-chip delay measurement hardware for efficient speed-binning. In *Proceedings of International On-Line Testing Symposium, Vol. 0*, pages 287–292, July 2005.

[50] S. Kotz, N. Balakrishnan, N. L. Johnson. Continuous Multivariate Distributions, Models and Applications. In *Vol. 1*, 2000.

[51] Sapatnekar S., Rao V., Vaidya P., Kang S. An exact solution to the transistor sizing problem for CMOS circuits using convex optimization. In *IEEE Transactions on Computer Aided Design*, pages 1621–1634, November 1993.

[52] Sentovich E., Singh K., Lavagno L., Moon C., Murgai R., Saldanha A., Savoj H., Stephan P., Brayton R., Sangiovanni-Vincentelli A. SIS: a system for sequential circuit synthesis. Technical report, 1992.

[53] Shephard K., Narayan V. Noise in Deep Sub Micron digital design. In *Proceedings of International Conference on Computer Aided Design*, pages 524–531, January 1997.

[54] Sherwani N. Algorithms for VLSI physical design automation. In *Kluwer*, 1995.

[55] Singh J., Nookala V., Luo Z., Sapatnekar S. Robust gate sizing by geometric programming. In *Proceedings of Design Automation Conference*, pages 315–320, July 2005.

[56] Singh J., Nookala V., Luo Z., Sapatnekar S. Robust gate sizing by geometric programming. In *Proceedings of Design Automation Conference*, pages 315–320, June 2005.

[57] Sinha D., Shenoy N., Zhou H. Statistical gate sizing for timing yield optimization. In *Proceedings of International Conference on Computer-Aided Design*, November 2005.

[58] Spars J., Furber S. Principles of asynchronous circuit design - a systems perspective. In *Kluwer Academic Publishers*, 2001.

[59] Sundararajan V., Parhi K. Low power synthesis of dual threshold voltage CMOS VLSI circuits. In *Proceedings of International Symposium on Low Power Electronics and Design*, pages 139–144, August 1999.

[60] Tsai J., Zhang L., Chen C. Statistical timing analysis driven post-silicon-tunable clock-tree synthesis. In *Proceedings of International Conference on Computer Aided Design*, pages 575–581, November 2005.

[61] Tschanz J., Bowman K., De V. Variation-tolerant circuits: circuit solutions and techniques. In *Proceedings of Design Automation Conference*, pages 762–763, June 2005.

[62] Tschanz J., Kao J., Narendra S., Nair R., Antoniadis D., Chandrakasan A., De V. Adaptive body bias for reducing impacts of die-to-die and within-die Parameter variations on microprocessor frequency and leakage. In *IEEE Journal of Solid-State Circuits*, pages 1396–1402, November 2002.

[63] Tschanz J., Kao J., Narendra S., Nair S., V. De. Effectiveness of adaptive supply voltage and body bias for reducing impact of parameter variations in low power and high performance microprocessors. In *IEEE Journal of Solid-State Circuits*, pages 826–829, May 2003.

[64] Vasicek O. A series expansion for the bivariate normal integral. In *Journal of Computational Finance, Vol.1, No.4*, 1998.

[65] Visweswariah C., Ravindran K., Kalafala K., Walker S., Narayan S. First-order incremental block-based statistical timing analysis. In *Proceedings of Design Automation Conference*, pages 331–336, June 2004.

[66] Wets R. Stochastic programs with fixed recourse: the equivalent deterministic program. In *SIAM Review*, pages 309–339, July 1974.

[67] Xiong J., Tam K., He L. Buffer insertion considering process variation. In *Proceedings of the Conference on Design, Automation and Test in Europe, Vol. 2*, pages 970–975, March 2005.

[68] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu. New paradigm of predictive MOSFET and interconnect modeling for early circuit design. In *Proceedings of Custom Integrated Circuits Conference*, pages 201–204, 2000.

[69] Yee G., Sechen C. Post-fabrication automatically tunable programmable delay elements for clock-delayed domino logic. In *Proceedings of SRC TECHCON Conference*, September 2000.

[70] Zhang S., Wason V., Banerjee K. A probabilistic framework to estimate full-chip subthreshold leakage power distribution considering Within-Die and Die-to-Die P-T-V variations. In *Proceedings of International Symposium on Low Power Electronics and Design*, pages 156–161, August 2004.