This is to certify that I have examined this copy of a doctoral dissertation by

Jaskirat Singh Bindra

and have found that it is complete and satisfactory in all aspects,

and that any and all revisions required by the final

examining commitee have been made.

Professor Sachin S. Sapatnekar

Name of the Faculty Adviser

Signature of the Faculty Adviser

Date

GRADUATE SCHOOL

VARIATION-AWARE COMPUTER-AIDED DESIGN TECHNIQUES FOR VLSI DIGITAL CIRCUITS

A DISSERTATION

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

JASKIRAT SINGH BINDRA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Sachin S. Sapatnekar, Adviser

October 2006

ACKNOWLEDGEMENTS

The most important contributor for the successful completion of this thesis without any doubt is, my adviser, Prof. Sachin Sapatnekar. It is fair to say that without Sachin's suggestions, patience, and work ethics, I would have had a much more extended stay as a graduate student. My modest success as a researcher owes a lot to his indefatigable efforts to mentor me in my work and to teach me how to be meticulous in technical writing and presentation. In additional to being an excellent adviser, Sachin is a also a great person, which is bit of a rarity, as many of my graduate student friends, from other groups, would enviously admit.

I would like to express my gratitude to Prof. Paul Johnson's research, (from the Carlson School of Management, University of Minnesota) and to Semiconductor Research Corporation for providing financial support for my graduate education.

Thanks are also due to the former and current members of the *VEDA* lab and Prof. Kia Bazargan's lab, Shrirang Karandikar, Rupesh Shellar, Haifeng Qian, Anup Sultania, Brent Goplen, Tianpei Zhang, Yong Zhan, Vidyasagar Nookala, Sanjay Kumar, Cristinel Ababei and Pongstorn Maidee. I have had several useful discussions with them on technical and other topics. Special thanks to Hongliang Chang for providing the code of the *MinSSTA* software package and help with the benchmark circuits.

My five years in the Twin Cities were made most enjoyable by the wonderful group of friends I found here. Without the time spent with Aditya Saxena, Abhinav Das, Sourabh Dighe, Karthikeyan Bhasyam, Ranjit Eswaran, Manish Kapoor (Usman Khan), Nikhil Wale, Sagar Nookala, Salim Charaniya, Deepa Deepa and Anil Singh Bika, life would have been very uninspiring. These people were the ones who made Minnesota a home away from home for me.

The support of my big extended family has helped me overcome some of the difficult times I have had. Finally, the person I am and the life that i enjoy, is all due to the hard work and sacrifices of my parents. I am fortunate to have parents who have given me

complete independence to explore my interests and the courage to pursue my dreams. Without their struggles and efforts, my ability to even start this thesis would not have existed.

Dedicated to the wonderful family that i was lucky enough to have been born in.

ABSTRACT

One of the most significant challenges currently confronting the VLSI circuit design community is the problem of ever increasing variations in the manufacturing process and the operating environment of high performance digital circuits. The escalating impact of environmental and process variations on the performance of current and future technology VLSI circuits, necessitates the use of circuit design techniques that can account for these uncertainties.

From a circuit design perspective, the variations may be classified as controllable or uncontrollable in nature. Controllable variations are a class of variations that can be directly reduced or controlled by circuit design techniques that specifically target these types of variation. Some examples of such variations are temperature fluctuations, which can be controlled by modifying the temporal and spatial distribution of hot spots on chip, and voltage variation, which can be controlled by optimizing the power grid of a chip. The variations that are uncontrollable in nature are those for which a circuit design cannot exercise any direct influence. From the point of view of a circuit designer, most variations arising from the limitations of the manufacturing process are uncontrollable in nature. Although uncontrollable variations cannot be directly reduced, their impact on the circuit performance can be controlled by robust circuit design techniques. To enable such robust circuit design, it becomes essential for VLSI computer-aided design (CAD) tools to keep sufficient design margins by incorporating their effect. This thesis presents variation-aware design automation techniques, accounting for both controllable and uncontrollable types of variations, by focusing on three important issues in digital circuit design: power grid design, gate sizing, and timing analysis.

The first part of the thesis addresses the problem of mitigating the controllable variations in the operating environment of a digital circuit, manifested in the form of voltage drop on the power supply network of wires. To control the voltage variations, two

topology optimization heuristics for the design of power ground networks have been proposed. These power grid design techniques maintain the desirable property of regular structure of the supply network by proposing a locally regular, globally irregular power grid topology. The first power grid design scheme is based on a sensitivity based heuristic, which iteratively adds wire in the local regions of the power grid to obtain maximum reduction in the voltage drop on the grid wires, for a given increase in the wire area. A second power distribution network design algorithm is presented, based on a recursive bipartitioning approach. This algorithm runs considerably faster than the first one by utilizing the idea of *locality* of power grid, and employing abstractions of different parts of the power grid circuit. Our proposed piecewise-uniform grid topology has a better wire area utilization as compared to other commonly used grid topologies, and our power grid design algorithm runs considerably faster compared to some previous approaches.

In the next part of the thesis, we focus on the problem of improving the timing yield of a digital circuit by performing gate sizing in the presence of uncontrollable manufacturing process variations. Our method formulates this robust gate sizing problem as a geometric program by employing posynomial delay models and a bounded *uncertainty ellipsoid* variation model for the random process parameters. Through our formulation, we provide a novel worst-casing solution that reduces the pessimism involved in worst-casing by incorporating the effects of spatial correlations of circuit parameters in the optimization procedure. We use a graph pruning technique to reduce the number of constraints and intermediate variables for the optimization set up. This uncertainty-aware gate sizing problem is then solved efficiently using convex optimization tools. Experimental results show that for the same circuit area, our robust gate sizing solution has a better timing yield than the conventional, deterministically based worst-case gate sizing solution.

The last part of the thesis explores the problem of performing circuit timing analysis in the face of randomly varying process parameters. A statistical static timing analysis

(SSTA) technique, which incorporates correlated parameters, both Gaussian and non-Gaussian, is developed to predict the probability distribution of the circuit delay. Prior to our work, most SSTA techniques could handle only a few correlated non-Gaussian variables. The proposed technique is the first scalable SSTA method, which can include correlated non-Gaussian parameters of variation in the statistical timing analysis framework. The SSTA procedure employs *independent component analysis (ICA)* as a preprocessing step, which enables the procedure to efficiently handle the correlated non-Gaussian parameters. Our algorithm has a linear complexity ($O(n * N_G)$) in the number of grids (n), and the number of gates (N_G) in the circuit. We demonstrate the accuracy of our SSTA procedure by verifying it with the Monte Carlo analysis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

"*Much to learn you still have . . . my old padawan. . . . This is just the beginning!*"

-Master Yoda in Attack of the Clones

Chapter 1

Introduction

The self-fulfilling prophecy of Gordon Moore's law [Moo65], predicting that the number of transistors on a chip would approximately double every eighteen months, has led to aggressive technology scaling and shrinking of the feature size. As a direct consequence of technology scaling, from only a few transistors in 1965 [Moo65], hundreds of millions of transistors are being integrated on a chip today. Table 1.1, compiled from the latest version of International Technology Roadmap for Semiconductors (ITRS) [SIA05], lists some of the trends in technology scaling, and indicates an estimate of more than three-quarters of a billion transistors on a single chip in 30nm technology in 2010.

Year	Technology Node (nm)	Number of Transistors	Number of Wire Levels	f (GHz)	V_{DD} (V)	Power (W)
2005	54	193M	11	5.2	1.1	167
2006	48	193M	11	6.8	1.1	180
2007	42	386M	12	9.3	1.1	189
2008	38	386M	12	11.0	1.0	198
2009	34	386M	12	12.4	1.0	198
2010	30	773M	12	15.0	1.0	198
2011	27	773M	12	17.7	0.9	198

Table 1.1: Trends in IC technology parameters [SIA05].

This astronomical number of on-chip transistors chip, makes it increasingly difficult to control the operating conditions of the chip. The variations in the operating environment, such as the temperature and voltage changes lead to the problem of signal integrity and variable delay of a circuit. The limitations of the deep-submicron fabrication process

technology and new physical phenomena that express themselves as small geometries, make it practically impossible to control the dimensions of the critical device parameters. The manufacturing process-driven uncertainties in the device parameters results in causing a spread in circuit performance measures such as the delay and power. While the signal integrity issues arising from the environmental variations may cause a chip to fail in the worst case, the variations in the circuit parameters affect the timing and power yield of the chip.

The increasing impact of environmental and process variations on the performance of current and future technology VLSI circuits necessitates the use of circuit design techniques that can account for these uncertainties. Given the complexities of these variations, it is essential for the VLSI computer-aided design (CAD) tools to incorporate their effect, in order to enable the design of robust circuits that are insensitive to the variations as much as possible. This thesis focuses on such VLSI CAD techniques for variation-aware design of digital circuits. We address the problems arising from environmental and process-driven variations and provide robust design automation solutions to these problems.

In this chapter, we will first discuss some trends in and sources of environmental and process variations, and then list the contributions of our research and explain the organization of this thesis.

1.1 Variations: Trends and Sources

Although variations have been a long standing problem, trends in current and future technologies have made their impact a much more serious problem than it has ever been.

Table 1.2, compiled from [Nas00], shows the trends in the mean (μ), and the standard deviation (σ), of some important circuit parameters namely, the effective channel length (L_e), gate oxide thickness (T_{ox}), on-chip supply voltage (V_{DD}), n-mos transistor

threshold voltage (V_{th}), interconnect width (W_{int}), thickness (T_{int}) and resistivity (ρ).

Parameters	1997		1999		2002		2005		2006	
	μ	3σ	μ	3σ	μ	3σ	μ	3σ	μ	3σ
L_e (nm)	250	80	180	60	130	45	100	40	70	33
T_{ox} (nm)	5	0.4	4.5	0.36	4	0.39	3.5	0.42	3	0.48
V_{DD} (V)	2.5	0.25	1.8	0.18	1.5	0.15	1.2	0.12	0.9	0.09
V_{th} (V)	0.5	0.05	0.45	0.045	0.4	0.04	0.35	0.04	0.3	0.04
W_{int} (μm)	0.8	0.2	0.65	0.17	0.5	0.14	0.4	0.12	0.3	0.1
T_{int} (μm)	1.2	0.3	1	0.3	0.9	0.27	0.8	0.27	0.7	0.25
$\rho(\Omega m)$	45	10	50	12	55	19	60	19	75	25

Table 1.2: Trends in parameters variations [Nas00].

It is clear from this data, that the σ/μ ratio increases significantly as technology scales from 250 nm to 70 nm. Moreover, the number of parameter variations affecting the performance of the circuit is rapidly increasing. Some examples of the sources of these environmental and process variations are listed below.

1. *Supply Voltage Variations:* Shrinking of device sizes results in exponential increase in the chip densities. These extremely large number of devices draw large amounts of currents from the power/ground (P/G) distribution network of wires, which connect to all the transistors on the chip. Together with the increase in the amounts of current drawn, the resistances of the interconnects have also increased due to the decrease in wire widths. Moreover, the amount of currents drawn from the power grid wires in a given region of the chip varies, depending on the switching activities of the gates in the underlying functional block. For instance, an arithmetic-logic unit (ALU) block is likely to exhibit a much higher switching activity, on an average, than a cache unit. This variable switching behavior of the functional blocks results in different current density regions of the

chip, which leads to the problem of a variable voltage drop in the supply wires and fluctuations in the supply voltage distributed to the on-chip transistors.

2. *Subwavelength Lithography:* In nanometer technologies, the minimum feature sizes are much smaller than the wavelength of light used in the photolithography process. For example, 193 nm lasers are currently used to fabricate the devices of dimensions 90nm or less [GKSY03, SIA05]. Thus, the ability to precisely control the critical dimensions of devices in the nanometer regime becomes increasingly difficult.

   In the current technology nodes, not only the critical dimensions of the minimum feature size line, but also the quality of line and line edges is gaining importance. The line edge roughness of photoresist lines and the corresponding polysilicon lines is becoming significant as gate linewidth control becomes comparable to the size of the photoresist polymer unit. The current state-of-the-art roughness for sub-100nm gate length technology has been reported to be of the order of 5-15 nm [KWW04], which may lead to significant device parameter fluctuations.

3. *Diffusion Process for Nanometer Devices:* As the devices become smaller, the number of dopant atoms per transistors fall in the range of 10 to 100 [BKD04]. At these levels, it becomes extremely difficult for the diffusion and the ion implantation process technology to exactly guarantee a uniform number of dopant atoms for every transistor. This random dopant density causes variations in the threshold voltage of transistors on chip [AK98].

4. *Chemical-Mechanical Planarization:* Variations in interconnect height and width can arise from the chemical-mechanical planarization (CMP) process, and results from the difference in hardness between the interconnect material and the dielectric. Ideally, after etches have been trenched into the dielectric below an interconnect layer and copper on the wafer, the CMP process should remove the unwanted

copper, leaving only the wires and vias. However, as the copper line is softer than the dielectric material, erosion due to CMP process causes uneven removal of copper and dielectric, resulting in variations in the interconnect dimensions.

The above list is not exhaustive as several other sources of variations affect the circuit performance.

From a design perspective, these variations may be broadly classified into two categories:

- **Controllable Variations:** These are a class of variations that can be controlled directly by a circuit designed specifically to target these types of variation. Some examples of such variations are temperature and voltage fluctuations. A designer can use some well-known circuit design techniques to reduce the voltage and temperature variations. For instance, a method to appropriately place thermal vias in the chip area is likely to help in controlling the temperature gradient across the chip. Similarly, a scheme to ably size the wires of the power supply network or place decoupling capacitors (decaps) at the appropriate locations could aid in reducing the voltage variations.

- **Uncontrollable Variations:** The variations that are uncontrollable in nature are those for which a circuit design cannot cause any direct reduction. From the perspective of a designer, the process-driven uncertainties in the channel length, transistor width, via resistance, oxide thickness, etc., are uncontrollable in nature. However, the desired circuit performance must be achieved in spite of these uncontrollable variations. Although it is not possible to directly control or reduce these types of variations, it is still possible to account for their impact on the circuit performance. A circuit designer usually relies on some type of guard-banding approach to control the effect of these uncontrollable variations. To achieve the desired circuit performance, in the presence of these types of uncertainties, extra resources such as larger transistors, wider wires, redundant logic are typically

used.

1.2 Research Problems and Contributions

In this thesis, we propose CAD solutions for problems related to both controllable and uncontrollable type of variations. Specifically, we propose solutions to the following problems:

1. **Power distribution network design:** The P/G network of wires electrically connect the external V_{DD} and ground nodes to all the on-chip transistors. The problem of voltage drop [1] on these P/G wires, is a case of variations in the operating environment of a chip. These variations can be regarded as controllable variations, as circuit design techniques can directly control or reduce them. We present two topology optimization techniques for the design a high performance power supply network, subject to various reliability constraints.

    - In the first power grid design technique, we propose a sensitivity based greedy heuristic, which analytically estimates the reduction in voltage drop with an increase in the wire area, in different regions of the chip area, and then iteratively selects the most sensitive region for wire additions. We extend this approach to include a congestion cost in our objective function so that the construction of grids by our method does not aggravate the congestion problem. Compared to other commonly used grid structures, the power grids designed by our procedure show considerable savings in the wire area. However, the runtimes achieved by our algorithm are not very fast.
    - To overcome the efficiency issues in our first power grid design scheme, we present a second algorithm based on the idea of hierarchical grid design and the notion of locality (to be explained in Section 2.7) in power grid design.

[1] Also referred to as the IR drop problem.

This grid design procedure employs a recursive bipartitioning approach and uses abstractions of parts of the power grid system.

Our methods propose and optimize for a novel piecewise-uniform power grid topology, that is suggested in our work. Such a topology, employing a locally regular, globally irregular grid structure, has the advantages of judicious use of wire area, combined with the relative ease of grid design. Experimental results show that our algorithm is considerably fast: we can design large power grids consisting of thousands of wires, and more than a million nodes in about 6 to 13 minutes of runtime. Our proposed designs achieve significant savings in wire area compared to other grid topologies (about 12% to 24% reduction), and the power grids designed by a multigrid-based previous work [WMS05] (about 6% to 12% reduction). The abovementioned power grid design schemes were published in [SS05] and [SS06b].

2. **Robust gate sizing:** The variations arising from the process limitations result in the circuit parameters such as the transistor channel length, width, oxide thickness and dopant density to deviate from their nominal values. This results in change of the circuit delay from the original nominal value that it was designed for. If the deviation of the delay from its nominal value is a positive shift, the new delay could exceed the original target delay, and cause a decrease in the timing yield of the chip, which is defined as the fraction of total number of manufactured chips that meet the original delay specifications. Thus, the presence of random variations in the circuit parameters could lead to reducing the profitability of the manufactured chips. From a gate sizing point of view, the presence of these random perturbations can be seen as uncontrollable type of variations. These uncertainties, arising from the fabrication process limitations, cannot be directly controlled or reduced by a gate sizing scheme. However, their effect on the circuit performance can be controlled. By accounting for the worst-case impact of these variations on the

circuit timing, the desired timing yield can still be achieved.

Traditionally, the robust gate sizing problem has been solved by guard-banding approaches. These methods are based on padding the timing constraints by a margin that safeguards against the effect of variations. However, most of these methods employ an arbitrary amount of guard-banding, e.g., by setting the original delay specification much tighter than the required target delay. Such ad hoc methods fail to capture important statistical attributes of the circuit such as the path correlations or the spatial correlations between varying parameters, and may result in overly pessimistic designs, spending more resources than necessary to achieve a specified timing yield.

In this thesis, we propose a novel and an efficient worst-casing methodology for the robust gate sizing problem. Our scheme reduces the pessimism involved in traditional worst-casing methods by incorporating the effect of spatial correlations in the optimization procedure. We employ a bounded model for the parameter variations, in the form of an uncertainty ellipsoid, which captures the spatial correlation information between the physical parameters such as channel lengths and transistor widths. The use of the uncertainty ellipsoid, along with the assumption that the random variables, corresponding to the varying parameters, follow a multivariate Gaussian distribution, enable us to size the circuits for a specified timing yield. The value of the desired timing yield can be chosen from the quantile function tables of the well-known *Chi-square* distribution [JW02]. In our formulation,we reduce the problem of overestimation of the variational components of the delay terms, by employing a circuit graph pruning technique [VC99], and using variable size ellipsoids at different topological levels of the circuit. Generating a first order Taylor series approximation of posynomial gate delay models, we formulate the resulting robust optimization problem as a geometric program [BV04]. The optimization problem is solved using highly efficient convex optimization methods

such as the interior point algorithm. Experimental results show that for the same transistor area, the circuits sized by of our robust optimization approach have, on an average, 12% fewer timing violations as compared to the gate sizing solutions obtained via the traditional, deterministically based guard-banding method. An early version of this work was published in [SNLS05].

3. **Statistical timing analysis incorporating correlated non-Gaussian parameters:** The presence of random variations in the physical parameters of the transistors, and the interconnects of a circuit make the results of deterministic static timing analysis (STA) mostly irrelevant. To provide meaningful information that the designers can use for the desired optimization tradeoffs in the presence of parameter uncertainties, the timing analysis methods must be variation-aware. Traditional variation-aware timing analysis techniques consist of performing a multi-corned-based analysis or a Monte Carlo analysis. Both these methods suffer from some serious weaknesses. The corner-based analysis involves enumerating all possible corners, i.e., all combinations of min/max values of each varying parameter, which can be exponential in the number of parameters. More importantly, a corner-based methodology can arrive at a worst-case corner which may actually have an extremely low probability of occurrence. Accounting for such a worst corner case results in an overly pessimistic design. The Monte Carlo analysis technique is based on sampling the random variables from a known probability distribution, and performing repeated timing analysis on the sampled points. For reasonably accurate prediction of the probability distributions of a circuit, the method requires a static timing analysis step for each of the hundreds of thousands of sample points. This renders the Monte Carlo technique very inefficient and impractical to use for large circuits consisting of tens of thousands of gates.

   In recent years, statistical static timing analysis (SSTA) has emerged as a promising and an efficient alternative to perform variation-aware timing analysis of a

circuit. The SSTA procedures can predict the timing yield of the circuit by extracting the probability distributions of the circuit delay. Most SSTA algorithms [CS05, VRK+04, AMK+05] achieve efficiency in their methods by assuming that the varying parameters can be accurately modeled by random variables following a Gaussian distribution. The normality assumption enables the use of closed form analytical expressions to evaluate the result of the basic statistical timing operations. However, not all parameters of variations can be accurately modeled as Gaussians. Moreover, the non-normal parameters exhibit statistical dependence arising from the spatial correlations in the circuit layout. Thus, in the presence of these correlated non-Gaussian parameters, the SSTA algorithms that assume normality can result in significant inaccuracies in estimating the probability distribution of a circuit. There have been some recent works, in the existing literature on SSTA, that extend the Gaussian SSTA algorithms to include non-Gaussian random variables [KS05, CZNV05]. However, these extensions are can only handle a few non-normal variables, and are not scalable to problems with large number of variables.

We present an efficient SSTA algorithm which is the first published work that can scalably handle a large number of correlated non-normal random variables in a reasonable runtime. We can efficiently handle the non-Gaussian parameters by employing an *independent component analysis* (ICA) technique [Bel, HO99, HO00, MP99], that enables us to achieve statistical independence between correlated non-Gaussian parameters. Using a moment-matching-based scheme we can extract the probability distribution function (PDF) and the cumulative distribution function (CDF) of all arrival time and delay random variables in an analytical closed-form expression. The time complexity of our SSTA procedure is $O(n * N_G)$, where n is the number of grids the chip layout is divided into, and N_G is the number of gates in the circuit, which is the same linear complexity as

of the Gaussian SSTA algorithms. Our SSTA method can process as many as 256 correlated non-normal parameters in about 5 mins of runtime. We demonstrate the accuracy of our SSTA procedure by verifying it against Monte Carlo simulations. The average of the absolute errors of the proposed SSTA procedure, compared to Monte Carlo analysis, is 0.99% for μ, 2.05 % for σ, 2.33% for the 95% point, and 2.36% for the 5% quantile point of the circuit delay. An early version of this work was published in [SS06a].

1.3 Organization of the Thesis

The three research areas of power grid design, robust gate sizing and statistical timing analysis are each addressed in a separate chapter of this thesis. Chapter 2 comprises the algorithms for power grid design. In this chapter, Sections 2.6 and 2.7 contain the two proposed topology optimization solution techniques to design a high performance and a reliable power grid. Chapter 3 addresses the robust gate sizing problem, and consists of two formulations, explained in Sections 3.4 and **??**, to perform uncertainty-aware gate sizing. The problem of SSTA with non-Gaussian parameters is contained in Chapter 4. Finally, Chapter 5 summarizes the findings of this research thesis.

Chapter 2

Power Grid Design Algorithms

2.1 Introduction to Power Grid Design

The network of interconnects, routed across multiple metal layers on the chip, that distribute the power supply and the ground to the logic gates on the chip is known as the power/ground (P/G) network or the supply network. Since, the P/G network electrically connects to all the devices on chip, its design has a big impact on the chip performance. Thus, the P/G distribution networks must be efficiently simulated, analyzed and optimized. The key constraints in the design of the P/G networks are those of:

IR drop: The P/G networks consist of metal wires carrying currents. These wires offer resistance to the current flow, and hence a voltage drop occurs across them. Since the supply network is required to distribute power to all of the gates on the chip, large currents flow through these networks, and thus the voltage drop can be significant. This large voltage drop, along with the fact that the relentless push for low power has driven the supply voltage requirement below the 1 volt region [SIA05], drastically reduces the noise margins to maintain correct logic levels. Even if the drop is not large enough to cause logic inversion of voltage levels, it may still affect the performance of the chip by increasing the delays of logic gates as the current drive of the gates is proportional to the supply voltage.

Electromigration: When current flows through the metal wires, the electrons collide with the metal atoms. These collisions result in a momentum transfer between conducting electrons and diffusing metal atoms to produce a force on the latter in the direction of electron flow. Over a period of time, the metal wires can become ruptured because of this collision force. This phenomenon of displacement of metal atoms due to the electron flux is known as electromigration (EM). The

problem of electromigration is a serious one in the P/G wires, since the currents always flow in one direction in these wires, i.e., from the V_{DD} nodes to the ground nodes. This problem is further aggravated due to the increase in current densities (defined as current per unit cross-sectional area of the wire) in the P/G wires because of the increase in the currents and the concomitant reduction in the wire widths due to technology scaling.

Ground bounce due to inductive effects: A sudden change of current flowing through a wire will induce abrupt voltage changes on that wire and its neighboring wires due to the inductance of the power grid. If these wires are a part of the on-chip P/G network, the induced voltage fluctuation is called the Ldi/dt noise. Due to this induced noise on the ground lines, today, we can no longer assume the existence of a universal ground node on the chip. The inductive noise or the ground bounce can cause severe signal integrity issues. Moreover the presence of inductance also leads to difficulties in the accurate analysis of P/G networks, e.g., the coupling capacitors between the power and ground lines now become floating capacitors and the Modified Nodal matrix [LC01] ceases to be diagonally dominant, which can lead to nonconvergence of iterative solvers.

As mentioned in Chapter 1, supply voltage fluctuations can be regarded as controllable type of variations. Employing some well-known power grid design techniques can directly control and reduce the voltage variations. To meet the constraints of IR drop and electromigration, the typical techniques available to the designers of supply networks are:

1. **Wire sizing:** By increasing the wire widths, the interconnect resistances are decreased and hence the IR drop is reduced. Increasing the widths also decreases the current density, and hence addresses the electromigration problem.

2. **Adding decoupling capacitors (decaps):** Decoupling capacitors are the on-chip capacitors that are deliberately added between the power grid and the chip substrate. The decaps act as charge reservoirs and maintain the required voltage levels within the clock cycle to prevent the dynamic or transient IR drop, also known as the voltage droop.

3. **Using appropriate topologies for the P/G network:** By optimizing the topology of the supply network, i.e., by "wisely" removing or adding wires in the supply grid or by optimal assignment of pads, it is possible to meet the voltage drop constraints.

In this chapter of the thesis, we present two topology optimization schemes to design a power grid which meets the static IR drop and the electromigration constraint. Using our power grid solution, a suitable decap placement scheme may be used, as a post-processing step, to safeguard against the transient voltage droop problem. Since we use a DC power grid analysis, our methods do not focus on the inductive noise in the P/G network. At an early stage of design, it is important to use simple models to efficiently optimize the power grid system. A more detailed transient simulation method may be employed later to analyze and reduce the inductive noise in the power grid wires.

2.2 Previous Work

Most of the previous works in the area of P/G network design perform the design optimization by wire sizing and adding decoupling capacitors.

The methods of [TSL03, TS01, WC02, DMS89, WHC+01, BVGY01] all provide wire sizing schemes to design the P/G network. In [TS01, TSL03], equivalent circuit models of many series resistors in the original network is constructed, and then wire widths are optimized by transforming the constrained nonlinear programming to a sequence of linear programs. The problem is formulated by assuming the currents in segments to be

fixed, and representing the branch voltages as variables. An optimization scheme to calculate both the lengths and widths of P/G networks, using a sequential network simplex method is proposed in [WC02]. The authors of [DMS89] size the wires of P/G networks by building an optimization engine that first finds an initial feasible solution, and then iteratively looks for the optimal solution along a feasible direction, using a sensitivity analysis method. In [WHC+01], a wire sizing algorithm is proposed based on the conjugate gradient method and circuit sensitivity analysis. In this problem formulation, only the wire conductances are used as the variables and the adjoint method [PRV95] is used to calculate the gradient. A heuristic based on minimizing total wire area is developed in [BVGY01] that determines the optimal wire widths and the network topology, i.e., a tree or a mesh structure by solving a nonlinear convex optimization problem.

Schemes for optimal assignment of decoupling capacitors are presented in [SGS00, CL97, WMS05, ZPS+06]. A combined technique to appropriately size the wire widths and add decaps through a heuristic based on the transient adjoint sensitivity analysis is proposed in [SGS00]. In [CL97], a decap optimization procedure involving an iterative process of circuit simulation and floor planning is proposed. A multigrid-based approach to reduce the power grid system size is presented in [WMS05]. This method use a sequential quadratic programming method to optimize a cost function that adds decaps and performs wire sizing. Recently, in [ZPS+06], a decap budgeting algorithm, based on macromodeling was proposed.

The authors of [MK92, Cai88, OP98] all provide techniques for topology optimization of a power grid. In [MK92], a P/G network optimization method is provided by removal of selected wires. The problem is solves by formulating it as a nonlinear combinatorial optimization problem and relaxing some of the constraints. The wires are represented as conductance links between the nodes, and a decision variable vector is used to determine the presence or absence of these conductance links. Some other works in topology optimization [Cai88, OP98] address the problem of optimal pad assignment to the power/ground grid structures.

The wire sizing and decap placement methods of [TSL03, WC02, TS01, DMS89, WHC+01, BVGY01, SGS00, CL97] all assume that the topologies of P/G networks are fixed, and only the widths of the wire segments, and the positions of decaps need to be determined. These techniques of power grid design by wire sizing and decap placement have a significant cost of over-utilization of the chip area. Furthermore, if the wire widths of the supply network vary throughout the chip, the routing of signal nets becomes much more difficult as a lot of book-keeping must be done to keep track of the locations and widths of P/G wires. In the works on topology optimization, the emphasis is on optimal assignment of the pins to the pads and placement of pads on the power grid. The fact that the topology has a significant influence on the final layout area is recognized, but the quest for a good topology design technique remains an open problem.

2.3 Proposed Power Grid Topology

In general, it is desirable to have as much regularity as possible in the power grid in order to permit the locations of power grid wires to be easily accounted for during signal routing. Furthermore, a regular grid structure can easily be analyzed [SG03] as it results in simpler circuit models. A highly irregular grid (for example, one that has been sized irregularly, or one in which wires have been selectively removed) may well provide an excellent solution if the power grid design problem is viewed in isolation. However, if we consider the *entire* design flow, a high degree of irregularity can be an impediment to the design methodology, as it may require a large amount of book-keeping to keep track of the precise locations of the power wires, and to determine which regions have excessive wiring congestion. Moreover, the number of optimizable parameters for such a problem can be very large, which may make the optimization highly computational. At the other extreme, a fully regular grid has few optimizable parameters and is ideal for the signal router. However, the constraint of full regularity can be overly limiting, since

the requirement of regularity may cause the design to use excessive wiring resources. For instance, if all wire widths were to be required to be identical, the wires could be over-sized in regions that have relatively low current densities.

Our work proposes structured regularity in the power grid, with a topology that is intermediate to fully regular grids and highly irregular grids. These grids can be thought of as being a piecewise-uniform grid that is *globally irregular* and *locally regular*. This structure combines the best of both worlds: it has the advantages for faster routing that is afforded by fully regular nets, while offering the flexibility in optimization and better resource utilization permitted by irregular topologies. It is possible, in some cases, that due to the regularity in the grid structure, the number of tracks available for signal net routes in high congestion regions are insufficient. However, this aggravation of congestion arising due to a regular grid design, can be checked and controlled if such a problem is anticipated and accounted for in the grid design procedure. In [SHSN02], a procedure is developed to simultaneously design the supply grid, under voltage droop constraints, and the signal net, under congestion considerations. Starting from an initial dense regular grid, the non-critical power grid wires in the high congestion regions are removed followed by a heuristic wire sizing step to overcome the effects of wire removal. The resulting grid is irregularly sized and thus, loses the advantages of structured regularity.

We use a toy example to illustrate the possible design choices for a power grid topology. For simplicity, let us assume that for the given example problem we can divide the chip into four rectangular regions or tiles having different current densities as shown in Figure 2.1(a).

The design in Figure 2.1(b) corresponds to the case where the voltage drop constraints are met by constructing a regularly structured grid with regularly sized elements with the same number of wires in the four tiles, i.e., four wires in each tile. This design uses more wire resources than required, since the wire sizes and the minimum number of wires in each tile are chosen according to the region with the worst-case voltage drop. The design in Figure 2.1(c) meets the design constraints by employing three wires in

each tile but the wires in the upper half are sized individually and irregularly to decrease the resistance and reduce the voltage drop. Such a design makes the task of the signal router more difficult, since it must keep track of the variable amount of space available in each region. The design in Figure 2.1(d) utilizes the lowest wiring area by using variable pitches and by sizing individual wires separately. However, besides the fact that this design would make the routing of signal nets very difficult, the optimization itself involves numerous design variables and is therefore computationally intensive. The design in Figure 2.1(e) is essentially the design we propose and optimize in this work. This design is piecewise-uniform as within a tile it employs a near-constant pitch and uses the same wire sizing throughout the chip. The wires are sized uniformly throughout the chip so as to maintain regularity and meet the IR drop needs. Such a design is more economical in utilization of wiring resources than designs in Figure 2.1(b) and (c), and has the desirable property of regularity that Figure 2.1(d) lacks, and does not complicate the routing problem for signal nets. Moreover, due to an inherent structure in the design, it is easy to optimize.

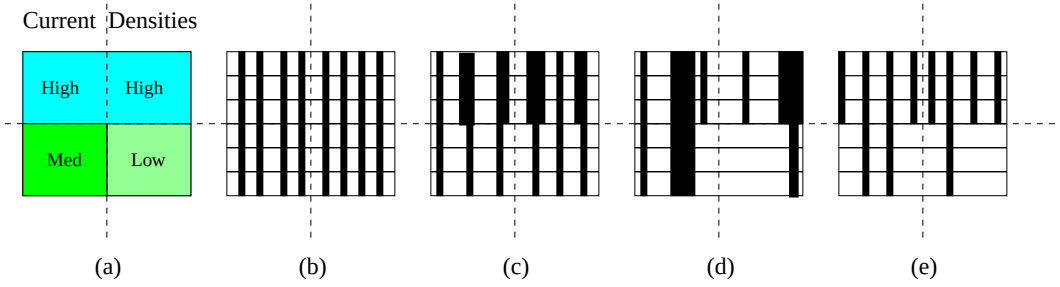


Figure 2.1: (a) Current densities for an example chip that illustrates the possible choices for a power grid topology. Possible Designs of P/G Network (b) Grid with 4 wires in each tile and regular wire sizing. (c) Grid with 3 wires in each tile and irregular wire sizing. (d) A non-uniform grid with variable pitches throughout and irregular sizing. (e) A piecewise-uniform grid with 4 wires in the upper half tiles, 2 wires in bottom-left tile and 1 wire in bottom-right tile and uniform sizing.

In the following sections of this chapter, after discussing the power grid design preliminaries and hierarchical analysis method , two techniques are presented to design a locally regular, globally irregular power grid that meets the reliability constraints. The procedure in the first technique is based on an iterative sensitivity based heuristic optimization. The second power grid design technique uses the principles of hierarchical design and the property of locality.

2.4 Preliminaries

2.4.1 Power Grid Circuit Model

A power grid comprises metal wires running in the orthogonal directions and spanning multiple layers (typically, 5 to 8 for current microprocessor designs). The wires in two consecutive layers of metal are electrically connected to each other by using vias. The wires in the top-most metal layers are electrically connected to the V_{DD} pads that are located either on the peripheral power ring, as in the case for a chip with a wire-bond package, or are distributed over the entire chip area, using C4 bumps, as in the case of a flip-chip package. This system of pad connections and network of metal wires carrying currents from the V_{DD} pads to the underlying gates in the functional blocks, can be modeled as an equivalent electrical circuit comprising possibly millions of nodes. Under DC conditions, as illustrated in Figure 2.2, the power grid can be modeled as a resistive mesh, with the pads replaced by voltage sources. As seen in the figure, the wires are replaced by their equivalent resistances and the worst-case switching activities of the gates in the underlying functional blocks determines the loading currents.

Using the circuit model as shown in Figure 2.2, the branch resistance r_i of branch *i* can be expressed as:

$$r_i = \rho_s \frac{p}{w_i} \tag{2.1}$$

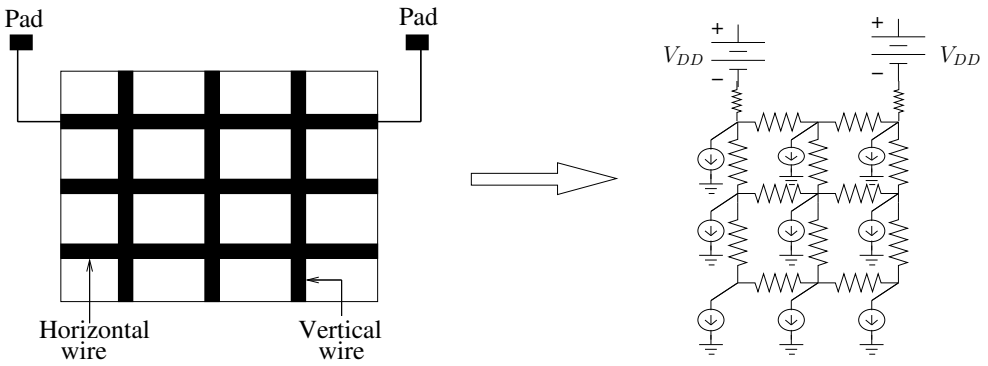


Figure 2.2: A power grid and its equivalent circuit model under DC conditions.

where ρ_s is the sheet resistivity, w_i is the width of the wire segment corresponding to the branch *i*, and p is the pitch of the power grid wires in the orthogonal direction, which is the same as the length of the wire segment. The pitch of the power grid wires could be different for different layers and in fact, may not even be constant for a given layer. After converting the voltage sources in the power grid model of Figure 2.2 to their Norton equivalents, the solution to the node voltages of the circuit is given by the following system of Modified Nodal Analysis (MNA) equations:

$$G \cdot \mathbf{V} = \mathbf{J}, \quad \mathbf{J}, \mathbf{V} \in R^n, \quad G \in R^{n \times n} \tag{2.2}$$

where n is the number of electrical nodes in the power grid circuit, G is the conductance matrix which contains stamps of all branch resistances, **V** is the vector of node voltages and **J** is a vector of load currents and the Norton currents of voltage sources.

For all nodes *j* and all branches *i* in the power grid circuit, the following constraints must be satisfied:

1. **The IR drop constraint**: $V_j > V_{spec}$
2. **The current density or EM constraints** : $|I_i| < \sigma w_i$

The voltage of node *j* is denoted as V_j, the branch current of branch *i* is denoted as I_i and σ is the specified current density for a fixed thickness (height) of the metal layer.

In the following sections of this chapter, we describe the design procedures for constructing a power grid that meets the two above constraints. The inputs to our power grid design problem are the number of power pads and their precise connection locations to the grid wires in the top layer, the placed functional blocks and an estimate of worst case currents drawn by the gates in the functional blocks. The amount of current drawn by each of the functional blocks can be determined by estimation techniques such as the ones proposed in [KNH95] and [WMR04]. Although, these methods extract time varying current waveforms, appropriate simplifications can be made to these techniques to estimate worst case steady state currents drawn by the functional blocks. We use the power grid circuit model as shown in Figure 2.2. Each node j in the power grid circuit is loaded by a constant current source c_j. Given the current estimate I_{f_k} drawn by a functional block k, and the physical coordinates of the placed functional block k, the values of constant current sources loading the power grid are chosen in such a way that the sum of all current sources at node locations lying over the functional block k add up to the functional block current I_{f_k}. The values of constant current sources and the functional block currents are expressed by the following relation:

$$\sum_{j \in \{\text{Nodes over block } k\}} c_j = I_{f_k} \tag{2.3}$$

2.4.2 Terminology

The following terminology would be adhered to in this chapter of the thesis. A *tile* or a *partition* is a rectangular region of the chip and the chip is divided into many tiles or partitions. A *skeleton grid* is an imaginary grid with wires running in orthogonal directions which is superimposed over the entire chip area. This skeleton grid which is a uniform and a continuous grid of constant pitch is a place-holder for adding the wires. The concept of the skeleton grid is explained in Section 2.6. An electrical node in a tile having links to other nodes in the same tile is called an *internal node*, the nodes at the edges of tiles that connect a tile to its neighboring tiles are called *port nodes* and the

nodes corresponding to the V_{DD} or ground pads are treated as *global nodes*.

2.5 Circuit Analysis by Macromodeling

To detect the nodes and branches in the power grid circuit which violate the IR drop and EM constraints, our iterative power grid optimization algorithms use an explicit power grid analysis step in each iteration. For the purposes of determining the most critical nodes and branches in the circuit, our work uses the hierarchical circuit analysis technique of [ZPS+00]. This section discusses the adaptation of the macromodeling method to our work.

As will be explained in Sections 2.6 and 2.7, both of our proposed power grid design schemes employ a method of dividing the large power grid area into smaller rectangular regions, referred to as partitions or tiles, and designing the subgrids locally in these smaller regions. In such a scenario of working with the partitions of a power grid system, the hierarchical modeling idea can be efficiently applied to our power grid systems.

Referring to the MNA Equation of (2.2), and splitting the system into internal nodes and port nodes, we can rewrite the system of equations as:

$$\begin{bmatrix} G_{aa} & G_{ab} \\ G_{ab}^T & G_{bb} \end{bmatrix} \begin{bmatrix} \mathbf{V_a} \\ \mathbf{V_b} \end{bmatrix} = \begin{bmatrix} \mathbf{J_a} \\ \mathbf{J_b} + \mathbf{I} \end{bmatrix} \tag{2.4}$$

where

- $\mathbf{V_a}$ and $\mathbf{V_b}$ are the vector of the voltages at the internal nodes and the port nodes, respectively.
- $\mathbf{J_a}$ and $\mathbf{J_b}$ are the vectors of current sources connected at the internal nodes and the ports, respectively.
- $\mathbf{I}$ is the vector of currents through the interfaces between the ports.
- G_{ab} is the conductance of links between the internal nodes and the ports.

- G_{aa} is the conductance matrix corresponding to the connections only between the internal nodes.

- G_{bb} is the conductance matrix corresponding to the connections only between the port nodes.

As we are employing a resistive mesh model and DC excitations, each rectangular tile can be abstracted as a multiport electrical element which has a linear current-voltage relationship. These multiport elements are referred to as *macromodels*. If m is the number of wires in the horizontal direction in a tile and n is the number of vertical wires in a tile, each macromodel can be regarded as a q-port linear element, where $q = 2(m + n)$, with its transfer characteristic given by the following equation:

$$\mathbf{I} = A\mathbf{V} + \mathbf{S} \tag{2.5}$$

where $\mathbf{I} \in \mathbf{R}^q, A \in \mathbf{R}^{q \times q}, \mathbf{V} \in \mathbf{R}^q, \mathbf{S} \in \mathbf{R}^q$, A is the port admittance matrix, $\mathbf{V}$ is the vector of voltages at the ports, corresponding to the voltages at the nodes on edges of the tiles, $\mathbf{I}$ is the current through the interface between the tiles, and $\mathbf{S}$ is a vector of current sources between each port and ground.

Referring to the matrix algebra, explained in detail in [ZPS+00], the macromodel elements, the admittance matrix A, and the current source vector $\mathbf{S}$, can be derived from the following relations:

$$\begin{aligned} A &= L_{bb} L_{bb}^T \\ \mathbf{S} &= L_{ba} L_{aa}^{-1} \mathbf{J_a} - \mathbf{J_b} \end{aligned} \tag{2.6}$$

where L_{aa}, L_{ba}, and L_{bb} represent the submatrices of the Cholesky factor matrix L, of the conductance matrix G, with indices a and b, corresponding to the internal nodes, and the ports, respectively.

Figure 2.3 shows the conversion of tiles of the power grid into macromodels. In this example, the power grid which is divided into 9 tiles, is reduced to a system of nine

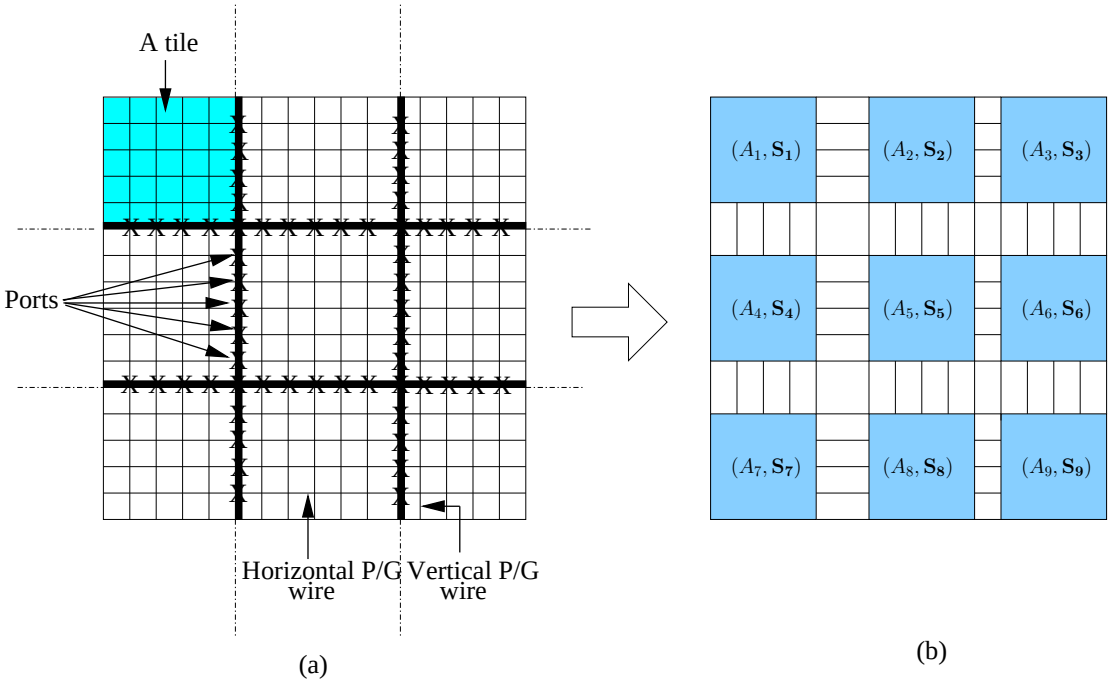


Figure 2.3: Converting the P/G network to a system of macromodels. (a) A P/G network with port nodes at the tile boundaries. (b) The P/G network changed to a system of macromodels connected to each other through the port nodes.

multiport elements given by the macromodel parameters (A,$\mathbf{S}$). The macromodels are connected to each other through port connections.

The macromodel parameters (A, $\mathbf{S}$) of each tile are stamped into the MNA equation in the global system given by

$$M\mathbf{X} = \mathbf{b} \tag{2.7}$$

where

- M is the matrix containing the conductance links between global nodes and the tiles, the conductance links between the tiles, and the stamps of A for each tile.
- $\mathbf{X}$ is the vector of voltages of global nodes and ports.

- **b** is the vector of current sources at global nodes and stamps of **S** for each tile.

The above Equation (2.7) is solved, either using a direct solver (as for the first power grid design scheme) or by using an iterative solver (as for the second power grid design procedure), to determine the global node and port voltages. If it is required to solve for the voltages of the internal nodes of any tile or partition, the elements of the interface current vector **I** are determined by Equation (2.5), and are fed back into Equation (2.4), which is then solved by a direct solver by reusing the Cholesky factors, and performing backward substitution of the already determined elements of the port voltage vector $\mathbf{V_b}$.

Since in our optimization procedures, while designing local power grids, only a few tiles or partitions are required to be processed in any iteration, the power grid system solution by hierarchical analysis proves very efficient. For the purposes of efficiency, we also use the step of sparsification of A matrix as proposed in [ZPS$^+$00]. This increases the sparsity of the global matrix M at the cost of reasonable simulation errors.

2.6 Solution Technique 1: A Sensitivity-based Heuristic

This section of the thesis presents a design method, based on the computation of the sensitivity of node voltages with respect to increase in wire area, for optimizing the P/G network by using locally regular, globally irregular grids. The procedure divides the power grid chip area into rectangular subgrids or tiles. Treating the entire power grid to be composed of many tiles connected to each other, enables the use of a hierarchical circuit analysis approach to identify the tiles containing the nodes having the greatest drops. Starting from an initial configuration with an equal number of wires in each of the rectangular tiles, wires are added in the tiles using an iterative sensitivity based optimizer. A novel and efficient table lookup scheme is employed to provide gradient information to the optimizer. Incorporating a congestion penalty term in the cost function ensures that regularity in the grid structure does not aggravate congestion.

The sequence of entire optimization procedure is summarized in the following steps:

1. The chip is divided into *k* rectangular tiles. An imaginary *skeleton grid* (to be defined in Section 2.6.1) is superimposed on the chip area. The actual supply grid is built on the skeleton grid, to maintain wire alignments across tile boundaries. Starting with an equal number of wires in all tiles in both horizontal and vertical directions, an initial sparse actual grid is formed on the skeleton grid.

2. Each tile is further divided into *b* bins or smaller rectangular regions. An initial congestion value is assigned to each bin in both horizontal and vertical directions. Such congestion values could be obtained from probabilistic congestion estimation techniques such as [LTKS02] and [WBG04].

3. The grid is analyzed using the macromodeling technique as described in Section 2.6.2, and the most critical node *x* in tile *i*, having the maximum voltage drop from V_{DD}, is determined.

4. The voltage sensitivity of the most critical node *x*, with respect to increase in wire area in tile *i*, due to the addition of *l* wires, is computed using the sensitivity calculation method, as described in Section 2.6.3.

5. The increase in congestion of bins of tile *i*, due to the addition of *l* wires is computed, as explained in Section 2.6.4. The cost function is calculated as the weighted sum of the voltage sensitivity term and the congestion term.

6. The number of horizontal or vertical wires in the tile having the minimum cost, is increased by *l*. The current sources to internal nodes of the tile are reassigned, so that the sum of the current sources at all internal nodes is the total current drawn by the P/G buses in that tile. The congestion values of the bins in the tile chosen for wire additions are updated.

7. Steps 4, 5, 6 and 7 are repeated until the voltage of the most critical node is greater than a specified value, and the EM constraints for all wires in the grid are met.

The following sections explain each of these steps in details.

2.6.1 Building the Power Grid

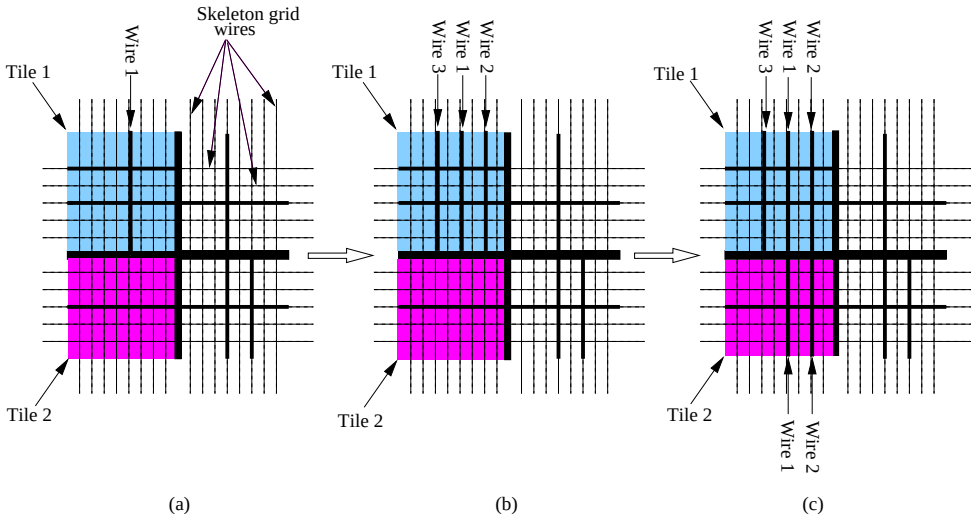


Figure 2.4: Illustration of the procedure to build the power grid on a skeleton grid. The P/G wires must lie on the skeleton grid, shown with light lines. The positions of actual P/G wires are shown with dark lines. (a) The initial structure of the grid. (b) The structure of the grid after two wires are added in tile 1. The wires are added to maintain a near-constant pitch within the tile. (c) The grid structure after addition of two wires in tile 2. The wires in tiles 1 and 2 are added at the same local positions so that they are aligned with each other.

Our optimization procedure builds a non-uniform, locally regular, globally irregular grid, with different densities of wires in different tiles. However, if each region uses a constant wire pitch within it, wires in the adjacent regions are likely to be misaligned which would result in the cost of extra vias to electrically connect these wires. We use

the concept of a *skeleton grid*, illustrated in Figure 2.4 (a), to avoid this scenario. The skeleton grid is an imaginary uniform grid superimposed over the layout area, and the wires of the non-uniform real grid, i.e., the actual grid to be designed, are placed on this grid. The pitches of the skeleton grid are chosen such that if the actual grid were to completely occupy the skeleton grid, it would have enough wires to meet the voltage drop constraints.

Figure 2.4 depicts how the actual non-uniform grid built on a uniform skeleton grid. In this example, the layout is divided into four tiles. The thick shaded lines are the power grid wires that demarcate the tile boundaries. The initial structure of a non-uniform power grid, built on a skeleton grid, is shown in Figure 2.4(a). One way to achieve perfect local regularity in the grid structure is to use a constant wire pitch inside a tile. However, if the wire densities of adjacent tiles are different, a constant pitch would result in the non-alignment of wires across tile boundaries. Hence, we place the wires inside a tile to achieve a near-constant pitch. As shown in Figure 2.4(b), the wires additions inside a tile are distributed to maintain a near-constant pitch, as per the idea of local regularity.

Our method maximizes wire alignment across tile boundaries by adding the wires on the skeleton grid in the same pre-determined order in all tiles in each iteration. For example, in Figure 2.4(b) and (c), the wires are added in the tiles in the following order: wire 1 first, and then wire 2 and wire 3. Since the wires in tiles are always added in the same order at identical local positions inside a tile, the wires in adjacent tiles are aligned with each other. As seen in Figure 2.4(c), wires 1 and 2 in adjacent tiles, tile 1 and tile 2 are aligned with each other. Such a structure aids the routing of signal nets as the only book-keeping that is required is related to the presence or absence of wires on the skeleton grid in the given tile.

2.6.2 Port Approximation Technique

After dividing the whole power grid area into smaller rectangular regions or tiles, we observe that the nodes on the edges of the tiles are on the same conducting wire. Therefore, it is reasonable to make an approximation that collapses some of the nodes[1] at the edge[2]. We use this observation to reduce the size of the macromodels, making an approximation to reduce the number of ports of a tile from $2(m_i + n_i)$ to $2(k + p)$ ports where $k \leq m_i$ and $p \leq n_i$. Each of the rectangular tiles is considered as a $2(k + p)$-port linear element by making an approximation that the voltage variation of some nodes on the edges of the rectangular tiles is small.

To further maintain the accuracy of the circuit solution, during the reduction of the edge nodes of the tiles, any V_{DD} and ground pad nodes that lie on the tile edges are always preserved and so are their immediate neighbors. Also for every removed node, its immediate neighbor is always preserved, thus approximating for only small voltage variations on the edges. The corner nodes of the tiles are never removed. Figure 2.5 illustrates this process where a 20 port tile is reduced to a 14 port tile.

To validate that this reduction from $2(m + n)$ ports to $2(k + p)$ ports is a reasonable approximation, we perform simulations of power grid circuits and empirically choose the value of *k* and *p* so that there is a reasonable upper bound on the error in the solution of the original and the approximated systems.

Tables 2.1 and 2.2 list the the orders of errors and runtime improvements for the approximations for the power delivery to a 2cm $\times$ 2cm chip with a V_{DD} value of 1.2 Volts, and pads distributed throughout the chip. For the experiments in Table 2.1, the chip is divided into 10×10 subgrids, i.e., 100 tiles with each tile having 10 horizontal and 10 vertical wires. The exact number of ports without any port approximations is 44

[1]This is a coarser and faster approximation than that used in multigrid-based methods.

[2]Even if the grid boundaries do not have these wires, the assumption is likely to be valid for reasonably dense grids.

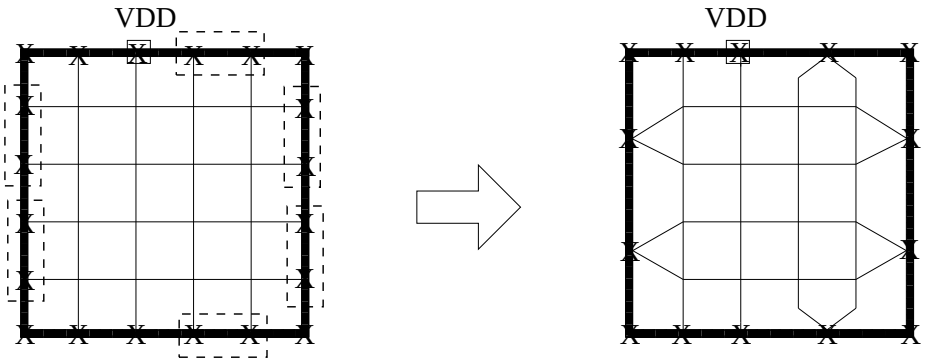


Figure 2.5: Reducing the number of ports of a tile of a power grid. (a) A tile of the original P/G network with 20 port nodes. (b) The port nodes of the tile reduced to 14 by combining some nodes.

and this corresponds to the simulation data on the last row. The range of voltages for the exact simulation without any port approximations is 0.88V - 1.20 V.

Table 2.2 represents the experiments for which the chip is divided into 12×12 power grid, i.e., 144 tiles with each tile having 12 horizontal and 12 vertical wires. The exact number of ports without any port approximations is 52 and this corresponds to the simulation data on the last row. The range of voltages for the exact simulation without any port approximations is 0.90V - 1.20 V.

Tables 2.1 and 2.2, show the orders of the average and maximum errors for simulations with the port approximations as compared to a simulation without removing any ports.

As seen in these tables, we gain significant runtime improvement from these approximations while ensuring that the errors remain within reasonable bounds. For a power grid design problem in which the bound on the worst case voltage drop is to be kept within 8%-10% of V_{DD}, the level of accuracy given by an average error of about 1%-3% is adequate. Since we are at the early design level, there is a fair amount of

# of Ports Kept	Avg Error %	Max Err %	Runtime (sec)
4	9.21%	13.24%	0.54
8	7.52%	10.02%	0.67
10	5.89%	9.50%	0.78
12	5.38%	8.08%	0.87
20	3.19%	6.58%	1.88
24	2.60%	5.08%	2.79
28	2.41%	5.06%	3.95
32	1.57%	3.51%	5.46
36	1.27%	2.71%	7.19
40	0.07%	1.02%	9.49
44	0.00%	0.00%	11.60

Table 2.1: Errors for port approximations for a 10×10 grid.

uncertainty involved in various design parameters like switching current waveforms and exact placement of the underlying functional blocks. Hence, it is advantageous to work with an efficient and reasonably accurate model of the power grid as opposed to a completely accurate but inefficient model.

2.6.3 Voltage Sensitivity Calculation

Sensitivity analysis is a standard technique employed for circuit optimization when it is desired to find out how the response of output changes with respect to changes in circuit element values. The advantage of using the sensitivity based method is that it eliminates the cost of doing an extra simulation after making changes in the circuit, and uses the factors of coefficient matrix of the original solution.

For our problem, the sensitivity calculations in matrix form provide the gradient

# of Ports Kept	Avg Error %	Max Err %	Runtime (sec)
4	10.41%	14.57%	0.71
8	8.78%	11.34%	1.23
12	5.87%	9.00%	1.90
16	4.88%	7.83%	2.81
20	3.78%	7.16%	4.14
24	3.33%	8.08%	6.09
32	2.58%	4.58%	11.27
36	2.35%	4.51%	15.07
40	1.52%	2.16%	19.26
48	0.05%	0.78%	30.58
52	0.00%	0.00%	36.55

Table 2.2: Errors for port approximations for a 12×12 grid.

information to guide the direction of optimization. The analysis, as explained in Sections 2.5 and 2.6.2, determines the most critical node that has the greatest voltage drop. The task of sensitivity computation is to determine the identity of the tile to which the addition of l horizontal or vertical wires would have the greatest impact, in terms of improving the worst case voltage drop. To explain the sensitivity calculation method, the following notation will be used:

k	:	Number of rectangular tiles into which the power grid is divided.
V_{xi}	:	Voltage of the most critical node *x* found in tile *i*.
p_i	:	Number of wires in tile *i*.
m_i	:	Number of wires in tile *i* in the horizontal direction.
n_i	:	Number of wires in tile *i* in the vertical direction.
W	:	The wire width for the power grid, assumed to be constant for the entire layout.
L_{hi}	:	The length of the horizontal wire in tile *i*.
L_{vi}	:	The length of the vertical wire in tile *i* .
Ar_i	:	Wiring area used in tile *i*.
V_i	:	The vector of port voltages of tile *i* which contains the most critical node *x* .
t	:	Numbers of ports kept for tiles after the port approximations.
l	:	The number of wires added in any tile in each iteration.

The following relations exist between the above defined terms.

$$Ar_i = W(m_i L_{hi} + n_i L_{vi}) \tag{2.8}$$

$$p_i = m_i + n_i \tag{2.9}$$

Using the chain rule, we can make the following calculations:

$$\frac{\delta \mathbf{V}_{xi}}{\delta Ar_i} = \frac{\delta \mathbf{V}_{xi}/\delta p_i}{\delta Ar_i/\delta p_i} \tag{2.10}$$

From Equations (2.8) and (2.9) it follows:

$$\frac{\delta Ar_i}{\delta p_i} = \frac{\delta Ar_i/\delta m_i}{\delta p_i/\delta m_i} + \frac{\delta Ar_i/\delta n_i}{\delta p_i/\delta n_i} \tag{2.11}$$

$$\frac{\delta Ar_i}{\delta p_i} = W(L_{hi} + L_{vi}) \tag{2.12}$$

To calculate the numerator of the right hand side (RHS) of Equation (2.10), i.e., $\delta \mathbf{V_{xi}}/\delta p_i$, we use the following procedure. Consider the global system, $M\mathbf{X} = \mathbf{b}$ as described by Equation (2.7). Let us assume that by the process of adding wires in tile *i*, M changes to $M + \delta M$, and **b** changes to **b** + δ**b**. It can be easily seen that the changes δM in the

global matrix M, are in the entries corresponding to the stamp of the first macromodel parameter A_i, and the changes $\delta\mathbf{b}$ in the vector $\mathbf{b}$, take place in the entries corresponding to stamps of second macromodel parameter $\mathbf{S_i}$. The sensitivities of particular response variables, i.e., the port voltages entries in the $\mathbf{X}$ vector in Equation (2.7), can be calculated by the following equations [PRV95]:

$$(M + \delta M)(\mathbf{X} + \delta\mathbf{X}) \;=\; \mathbf{b} + \delta\mathbf{b} \tag{2.13}$$

Simplifying Equation (2.13) by substituting from Equation (2.7), and neglecting the second-order variation, i.e., $\delta M \delta\mathbf{X}$, we obtain:

$$\begin{aligned} M\delta\mathbf{X} &= -\delta M\mathbf{X} + \delta\mathbf{b} \\ \delta\mathbf{X} &= M^{-1}(-\delta M\mathbf{X} + \delta\mathbf{b}) \end{aligned} \tag{2.14}$$

If we are concerned only with the j^{th} response variable, i.e., the voltage sensitivity of j^{th} port variable, Equation (2.14) can be written as:

$$\delta\mathbf{X_j} \;=\; [j^{th}\text{row of } M^{-1}](-\delta M\mathbf{X} + \delta\mathbf{b}) \tag{2.15}$$

The j^{th} row of M^{-1} can be calculated by solving the system

$$M^T\xi_j \;=\; e_j \tag{2.16}$$

where ξ_j is a column vector representing the j^{th} row of M^{-1}, and e_j is a column vector corresponding to the j^{th} column of the identity matrix. Equation (2.15) can be rewritten as:

$$\delta\mathbf{X_j} \;=\; \xi_j^T(-\delta M\mathbf{X} + \delta\mathbf{b}) \tag{2.17}$$

Referring to Equation (2.17), the following relations can be found:

$$\frac{\delta\mathbf{X_j}}{\delta\mathbf{b_k}} \;=\; \xi_{jk} \tag{2.18}$$

where ξ_{jk} is the k^{th} component of the ξ_j, and

$$\frac{\delta \mathbf{X_j}}{\delta M_{mn}} = -\xi_{jm}\mathbf{X_n} \tag{2.19}$$

where ξ_{jm} is the negative of m^{th} component of column vector ξ_j multiplied by n^{th} component of the original solution vector **X**. Using the Equations (2.18) and (2.19), and applying the chain rule, we can compute the sensitivities of voltages at the ports of the tile which contains the most critical node *x*, with respect to an addition of *l* wires in tile *i*, by the following equation:

$$\frac{\delta \mathbf{X_j}}{\delta p_i} = \sum_{m=1}^{t}\sum_{n=1}^{t}\frac{\delta \mathbf{X_j}}{\delta M_{mn}}\frac{\delta M_{mn}}{\delta p_i} + \sum_{k=1}^{t}\frac{\delta \mathbf{X_j}}{\delta \mathbf{b_k}}\frac{\delta \mathbf{b_k}}{\delta p_i} \tag{2.20}$$

where *t* is the number of kept ports for the tiles after the port approximations and the summation indices in the terms $\sum_{m=1}^{t}\sum_{n=1}^{t}$ and $\sum_{k=1}^{t}$ arise due to the change in the macromodel ($A_{t\times t}$,$\mathbf{S_{t\times 1}}$) stamps for tile *i*, where the extra wires are added. For the purposes of efficiency, the two unknown quantities in Equation (2.20), $\left(\frac{\delta M_{mn}}{\delta p_i}\right)$ and $\left(\frac{\delta \mathbf{b_k}}{\delta p_i}\right)$, are calculated using a table lookup scheme described in the following section.

Table Lookup

The number of wires in a tile in horizontal and vertical directions can assume a finite set of values starting from the initial number of wires, to a maximum number that corresponds to the number of wires on the skeleton grid inside a tile.

The macromodel matrix A depends only on the number of wires in a tile, and the vector $\mathbf{S}$ depends on the number of wires and the value of current sources in a tile. However, since our model assumes equal valued current sources placed at the internal nodes of the tile, the $\mathbf{S}$ vector can be calculated for a current source of unit value at the internal nodes, and then a new $\tilde{\mathbf{S}}$ vector can be computed as a scalar multiple of the $\mathbf{S}$ vector by the following relation:

$$\tilde{\mathbf{S}} = c\mathbf{S} \tag{2.21}$$

where $\tilde{\mathbf{S}}$ is the vector for the tile with a current source of magnitude *c* placed at the internal nodes, and $\mathbf{S}$ is the vector for the tile with a current source of unit magnitude placed at the internal nodes.

Index	# of Horizontal Wires	# of Vertical Wires	$A_{t\times t}$	$\mathbf{S_{t\times 1}}$
:	:	:	:	:
q	3	3	A_1	$\mathbf{S_1}$
:	:	:	:	:
p	3	7	$\hat{A_1}$	$\hat{\mathbf{S_1}}$
:	:	:	:	:
100	10	10	A_{100}	$\mathbf{S_{100}}$

Table 2.3: A Table lookup example to illustrate the simplification of voltage sensitivity computation.

Given that macromodel parameters $(A, \mathbf{S})$ depend on only the number of wires in a tile, we construct, in advance, a table that contains the corresponding macromodel parameters for a set of values of horizontal and vertical wires in a tile.

The structure of the table is shown in Table 2.3. We can now make the following computations from the table lookup:

$$\frac{\delta M_{mn}}{\delta p_i} \approx \frac{\Delta M_{mn}}{\Delta p_i} = (A_{uv})_p - (A_{uv})_q \tag{2.22}$$

$$\frac{\delta \mathbf{b_k}}{\delta p_i} \approx \frac{\Delta \mathbf{b_k}}{\Delta p_i} = c((\mathbf{S_r})_\mathbf{p} - (\mathbf{S_r})_\mathbf{q}) \tag{2.23}$$

where (*u*, *v*) are the rows and columns of $A_{t\times t}$ and (*m*, *n*) are the rows and columns of M corresponding to the stamp of A. Similarly *r* is the row of $\mathbf{S_{t\times 1}}$, and *k* is the row in $\mathbf{b}$ corresponding to the stamp of $\mathbf{S}$. The index *q* is the index in the table corresponding to the current numbers of horizontal and vertical wires in a tile, and *p* is the index in the

table corresponding to the number of horizontal and vertical wires after an addition of l wires in either of the directions. These equations hold because any change in the M matrix and the **b** vector, due to an addition of a wire in the tile, would only be in the stamps of A and $\mathbf{S}$.

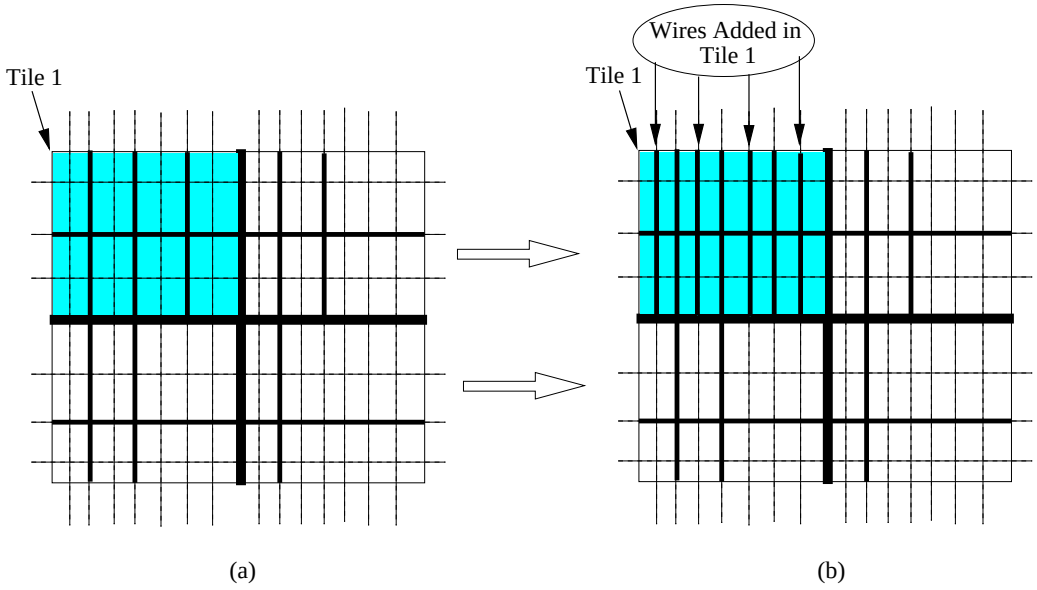


Figure 2.6: Change in the grid structure by addition of wires in tile 1. (a) Tile 1 with 3 wires initially. (b) Four more wires added in tile 1. The corresponding change in macromodel parameters $(A_1, \mathbf{S_1})$ can be calculated using Table 2.3 as a lookup table.

The table lookup procedure can be better understood with the help of a small example. Let us consider a chip area divided into four tiles. We wish to compute the terms in the left hand side (LHS) of Equations (2.22) and (2.23), with respect to addition of wires in tile 1. Figure 2.6 illustrates the change in the grid structure by wire additions in tile 1.

The global system of Equation (2.7), corresponding to the example of Figure 2.6,

where the chip is partitioned into four tiles, is described by:

$$M = \begin{bmatrix} G_{00} & G_{01} & G_{02} & G_{03} & G_{04} \\ G_{01}^T & A_1 & G_{12} & G_{13} & G_{14} \\ G_{02}^T & G_{12}^T & A_2 & G_{23} & G_{24} \\ G_{03}^T & G_{13}^T & G_{23}^T & A_3 & G_{34} \\ G_{04}^T & G_{14}^T & G_{24}^T & G_{34}^T & A_4 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} \mathbf{I_0} \\ -\mathbf{S_1} \\ -\mathbf{S_2} \\ -\mathbf{S_3} \\ -\mathbf{S_4} \end{bmatrix}$$

The entries corresponding to the index *0* are associated with the global nodes and the other indices correspond to the tile numbers.

As seen in Equation (2.24), after addition of two wires in tile 1, the only changes in M are in the entries corresponding to A_1, and the only changes in $\mathbf{b}$ are in the entries corresponding to $\mathbf{S_1}$. After the wire additions, the new macromodel parameters are ($\hat{A_1}$,$\hat{\mathbf{S_1}}$). Since we have a pre-constructed table of the form of as shown in Table 2.3, all that is required is to search in the table for index *q*, corresponding to the number of wires in tile 1, before the wire addition, and index *p*, corresponding to the number of wires in tile 1, after the wire addition. The entries stored in the table for index *q* are (A_1, $\mathbf{S_1}$), and the entries stored in the table for index *p* are ($\hat{A_1}$,$\hat{\mathbf{S_1}}$). Hence, we can now easily evaluate Equations (2.22) and (2.23).

All the terms in the RHS of Equation(2.20) are now known from the solutions of Equations (2.18), (2.19), (2.22) and (2.23). The evaluation of Equation (2.20), yields the sensitivities of the port voltages with respect to the wire addition in the tile.

We now require the sensitivity of the most critical (i.e., LHS of Equation (2.10)) node which could be an internal node of a tile. Since our model of the power grid as shown in Figure (2.2) is purely linear, we can relate the port voltages to the most critical node by the following equation:

$$\mathbf{V}_{xi} \;=\; \mathbf{C^T V} + D_x \tag{2.24}$$

where $\mathbf{V}$ is the vector of port voltages, $\mathbf{C^T}$ is a row vector and D_x is a constant. Referring to Equation (2.4), these terms are calculated as:

$\mathbf{C^T} = x^{th}$ row (for the internal node x) of matrix $-G_{aa}^{-1}G_{ab}$.

$D_x = x^{th}$ component of vector $G_{aa}^{-1}\mathbf{J_a}$.

If the most critical node is not in the same tile in which a wire is being added then the terms $\mathbf{C^T}$ and D_x are constant. Therefore, from Equation (2.24)

$$\frac{\delta \mathbf{V}_{xi}}{\delta p_j} = C^T \frac{\delta \mathbf{V}}{\delta p_j} \tag{2.25}$$

If the most critical node is indeed in the same tile in which a wire is being added,

$$\frac{\delta \mathbf{V}_{xi}}{\delta p_i} = \mathbf{C^T}\frac{\delta \mathbf{V}}{\delta p_i} + \mathbf{V}\frac{\delta \mathbf{C^T}}{\delta p_i} + \frac{\delta D_x}{\delta p_i} \tag{2.26}$$

Thus, from Equations (2.25) and (2.26) we get the LHS of Equation (2.10), which is simply the change in voltage of the most critical node by making an addition of *l* wires in tile *i*. This is the gradient information we require to guide the optimization process described in the Section 2.6.5. The table-lookup scheme is implemented as a file read, and therefore does not increase the memory requirements of the optimization procedure.

We also perform a ***fidelity test*** to check the sensitivity calculations. In this test, we first find the worst node in the circuit, and then compute the sensitivities of the worst node with respect to addition of *l* wires in various tiles, one tile at a time. The tiles are then sorted in decreasing order of sensitivity values. Next, we make the actual change of addition of *l* wires in the sorted order and observe the voltage change of the worst node. We find that the sorting of tiles according to the sensitivity calculations indeed match the order of the voltage change obtained by the actual addition of extra wires. In other words, the tile computed to have the greatest voltage sensitivity for the worst node obtains the maximum voltage improvement, or greatest voltage drop reduction, after actual addition of *l* wires in that tile. The tile calculated to have the second largest sensitivity has the second largest voltage improvement, and so on. The successful results of the fidelity test are critical, as they validate the greedy approach to add wires in the tile having the maximum sensitivity according to the sensitivity calculations.

We empirically choose to keep the values of *l* to be from 5-7 wires added in a tile in every iteration. For larger values of *l*, the orders of errors in the sensitivity computations

increase, as the assumptions made to neglect the second order variations, i.e., $\delta M \delta \mathbf{b}$ in the derivation of Equations (2.18) and (2.19) no longer hold.

2.6.4 Congestion-Aware Power Grid Design

The constraint of maintaining local regularity may come at the expense of worsening the congestion problem, where signal nets are unable to find sufficient routable resources to complete their routing, and are forced to take detours. If structured regularity enforces placement of power grid wires in regions where the demand for routing resources from signal nets is high, it would clearly aggravate congestion.

To overcome this potential problem of congestion arising due to local regularity in the grid structure, we follow a pre-emptive strategy without compromising the property of structured local regularity in the power grid design. We introduce a term in the cost function which penalizes addition of power grid wires in high congestion regions. Since the input to our power grid design problem is a floorplan or a placed net-list, probabilistic congestion estimation techniques such as [LTKS02] and [WBG04] can be used to assign congestion numbers to different regions of the chip. Dividing the chip into rectangular tiles for the purposes of building the power grid, produces tiles with fairly large areas from the perspective of signal net routing. Hence, we further tessellate each tile into smaller rectangular regions called *bins*. As shown in Figure 2.7(a) and (b), after the chip area is divided into rectangular subgrids or tiles, the tiles are further divided into smaller rectangular bins. Considering the bins in tile *i* arranged in y_i rows and z_i columns, probabilistic congestion estimation techniques can be use to generate two matrices U_{i_h} and U_{i_v}, each of size $y_i \times z_i$, whose entries correspond, respectively, to the associated horizontal and vertical usage of each bin due to the signal net routing. The horizontal and vertical capacities, c_{i_h} and c_{i_v}, respectively, of each bin in tile *i* is given by the

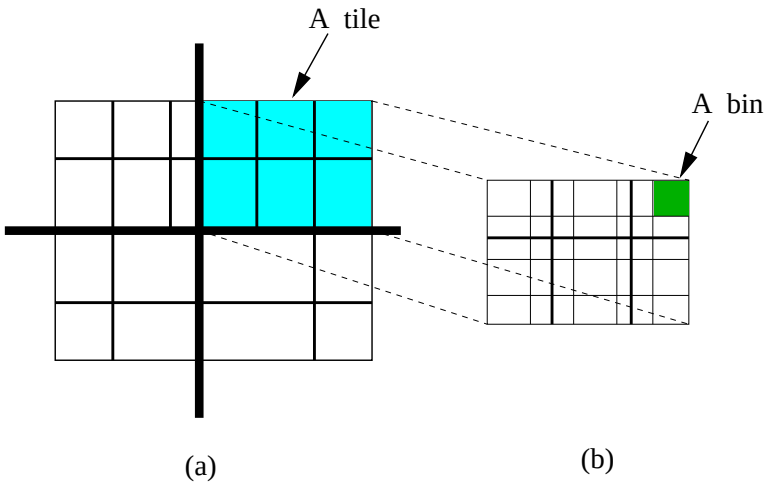


Figure 2.7: (a) An example of a chip divided into four tiles demarcated with thick dark lines representing P/G wires. The P/G wires inside a tile are shown with thin dark lines. (b) A tile is further divided into 16 bins. The dark lines are P/G wires inside the tile.

following relations:

$$c_{i_h} = \frac{h_i}{\min P_h} \tag{2.27}$$

$$c_{i_v} = \frac{w_i}{\min P_v} \tag{2.28}$$

where ($\min P_h$) and ($\min P_v$) are the minimum horizontal and vertical pitches, respectively, of the metal layer, and, h_i and w_i are the height and the width, respectively, of each bin in tile *i*. The horizontal and vertical congestions of each bin are calculated and stored in matrices C_{i_h} and C_{i_v}, respectively. For a bin indexed by row a and column b, the congestion value is given by the fraction of total bin capacity utilized by the following relation:

$$C_{i_h}[a,b] = \frac{U_{i_h}[a,b]}{c_{i_h}} \tag{2.29}$$

$$C_{i_v}[a,b] = \frac{U_{i_v}[a,b]}{c_{i_v}} \tag{2.30}$$

In each iteration of the optimization loop, addition of *l* wires in tile *i* change the values in congestion and usage matrices for exactly *l* columns or *l* rows, depending on whether

the wires added were in the vertical or the horizontal direction. For instance, addition of one vertical wire in tile *i*, at a position corresponding to all bins indexed by column b, changes the values of the usage matrix U_{i_v} in the following way:

$$U_{i_v}[j,b]_{new} = U_{i_v}[j,b]_{old} + \frac{W}{\min P_v}, \quad j = 1 \cdots y_i \tag{2.31}$$

where W is the wire width, assumed to be constant for the entire layout. Using Equations (2.29) and (2.30), the new congestion values, after wire additions, can be calculated.

We define $CongV_{avg_i}$ as the average vertical, and $CongV_{max_i}$ as the maximum vertical congestion of all bins indexed by column b as:

$$\begin{aligned} CongV_{avg_i}[b] &= \sum_{j=1}^{y_i} \frac{C_{i_v}[j,b]}{y_i} \\ CongV_{max_i}[b] &= \max C_{i_v}[j,b], \quad j = 1 \cdots y_i \end{aligned} \tag{2.32}$$

The average horizontal, $CongH_{avg_i}$, and the maximum horizontal, $CongH_{max_i}$, congestion of all bins indexed by row a is similarly defined as:

$$\begin{aligned} CongH_{avg_i}[a] &= \sum_{j=1}^{z_i} \frac{C_{i_v}[a,j]}{z_i} \\ CongH_{max_i}[a] &= \max C_{i_v}[a,j], \quad j = 1 \cdots z_i \end{aligned} \tag{2.33}$$

The average vertical and horizontal congestion for all of the bins in tile *i* is:

$$\begin{aligned} CongV_{avg_i} &= \sum_{j=1}^{z_i} \frac{CongV_{avg_i}[j]}{z_i} \\ CongH_{avg_i} &= \sum_{j=1}^{y_i} \frac{CongH_{avg_i}[j]}{y_i} \end{aligned} \tag{2.34}$$

The maximum vertical and horizontal congestion for all the bins in tile *i* is given by:

$$\begin{aligned} CongV_{max_i} &= \max(CongV_{max_i}[1], \cdots, CongV_{max_i}[z_i]) \\ CongH_{max_i} &= \max(CongH_{max_i}[1], \cdots, CongH_{max_i}[y_i]) \end{aligned} \tag{2.35}$$

The penalty term introduced for congestion is the following:

$$\begin{aligned} Cong_i = & \gamma(CongV_{avg_i} + CongH_{avg_i}) + \\ & (1-\gamma)(CongV_{max_i} + CongH_{max_i}) \end{aligned} \tag{2.36}$$

where, γ is a parameter $\in [0, 1]$ to appropriately penalize the average and the maximum congestion.

The cost function associated with adding wires in tile *i* is a weighted sum of the voltage sensitivity term, $\frac{\delta V_x}{\delta Ar_i}$, computed using the procedure described in Section 2.6.3, and the congestion penalty term, $Cong_i$.

$$Cost_i \;\; = \;\; \beta Cong_i - \alpha \frac{\delta V_x}{\delta Ar_i} \tag{2.37}$$

where, α and β are normalized weight parameters. The voltage sensitivity term represents the cost to benefit ratio of voltage drop reduction with increase in area, and the congestion term penalizes for aggravating congestion by wire additions in the congested regions. The cost function guides the optimization loop as the tile having the minimum value of the cost function is selected for wire additions.

2.6.5 Optimization Objective and Constraints

We use a greedy optimization heuristic based on the information obtained from the sensitivity and congestion computations. Using some of the definitions in the previous

sections, the optimization problem is formulated as follows:

$$\text{Minimize} \sum_{i=1}^{k} Cost_i \tag{2.38}$$

Subject to

$$Min(\mathbf{V_x}) \geq V_{spec} \quad \forall node_x$$

$$P_{vi} \approx c_1 \lambda_v, \forall i : 1 \text{ to } k$$

$$P_{hi} \approx c_2 \lambda_h, \forall i : 1 \text{ to } k$$

$$\frac{I_{wire_j}}{W} \leq \sigma \quad \forall wire_j$$

where *k* is the number of tiles in the P/G grid, P_{hi} and P_{vi} are the wire pitches in the horizontal and vertical direction in tile *i*, λ_h and λ_v are the pitches in the horizontal and vertical directions of the *skeleton grid*, c_1 and c_2 are some integer constants. These approximation constraints enforce the wires in a tile to be placed on the skeleton grid with a near-constant pitch within a tile *i*. As described in Section 2.6.1, these constraints aid in maximum alignment of wires in the adjacent tiles. The wire pitches are related to the line resistances by the following relations

$$R_{vi} = \frac{\rho_s P_{hi}}{W} \quad \text{and} \quad R_{hi} = \frac{\rho_s P_{vi}}{W} \tag{2.39}$$

where R_{vi} and R_{hi} are the line resistances of the vertical and horizontal wires in tile *i*, ρ_s is the sheet resistivity and *W* is the wire width which is assumed to be constant for the entire layout.

The last set of constraints are the EM constraints, where σ is the current density for a fixed thickness of the metal layer and I_{wire_j} is the current flowing through wire j.

The optimization procedure iteratively adds *l* wires in tile *i*, by selecting the tile index *i* based on $Cost_i$ as given by the cost function of Equation (2.37). The additions of wires on the skeleton grid ensures the approximation constraints to obtain a near-constant pitch within a tile. The IR drop and EM constraints, for nodes and branches in tile *i*, are checked by the hierarchical circuit analysis step described in Section 2.5. The

iterative method of wire additions in a tile continues until the power grid system satisfies the reliability constraints of IR drop and EM.

2.6.6 Speedup Techniques

In the optimization procedure, we can generate the following savings in the some of the steps to achieve significant speed up.

- Instead of calculating the voltage sensitivities and congestion values for all the tiles, we can define an *active sensitivity window* around the most critical node so that there are enough pads within the window, and then make the sensitivity and congestion calculations for only the tiles inside that window. Typically, when the pads are distributed over the entire chip, as in a flip-chip package, the sensitivity window would be the tile containing the worst node and its neighboring tiles.

- By adding a few wires in only one of the tiles, we need to compute the Cholesky factors of the G matrix for only the changed tile. The factors of other unchanged tiles are reused.

- In any analysis step after the initial one, we do not necessarily have to solve all the tiles to determine the most critical node. After solving for the tile *j*, that had the worst drop, all the tiles having greater minimum voltages in the previous iteration than the minimum of voltage of tile *j* in the current iteration, need not be solved.

2.6.7 Extension to Multiple Metal Layers

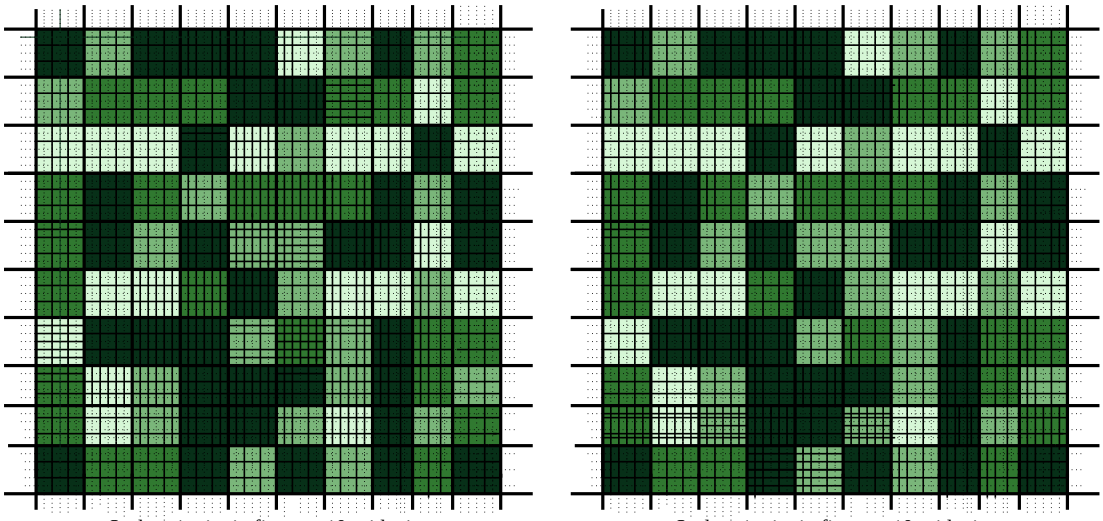
The proposed optimization technique can be easily extended to design a power grid for chips having multiple layers of metal. Typically, the upper metal layers use wider wire widths than the lower layers. Also, adding wires in upper layers would yield greater reduction in the IR drop, since the upper layer wires affect the voltages at larger number of sinks. The chip area can be divided into rectangular tiles in each of the layers. In

any iteration, to determine the layer and the tile for wire additions that would result in the greatest IR drop reduction, with minimum wire area increase, we require the voltage sensitivity, i.e., $\frac{\delta \mathbf{V_{xi}}}{\delta A r_i}$ for a few tiles in each layer. The penalty for congestion aggravation can also be set appropriately for different layers. Since the top metal layers are not much used for the signal net routing, they can be assigned a lesser congestion penalty term. Hence, the power grid is designed by iteratively adding wires in the tile and layer combination which yields the least cost.

2.6.8 Experimental Results

The proposed optimization scheme was implemented in C using a sparse matrix library [mes], and results on several power networks were tested [3]. Due to the unavailability of benchmark circuits for power networks, the circuits were randomly generated but with real circuit parameter values. The circuit parameter values, sheet resistivity (ρ_s), wire width (W), current density (σ), minimum wire pitches ($\min P_h$) and ($\min P_v$), and the ranges of worst case current sources were taken from [SIA01] and [Con00] for power delivery to a 2cm$\times$2cm chip in 130nm technology with V_{DD} = 1.2V. The voltage constraints for the power grids, i.e., V_{spec} was 1.08V, i.e., 90% of V_{DD}. Also, due to the unavailability of large block level benchmark circuits, we could not use the probabilistic technique to estimate congestion values for different regions of the chip. Instead, we randomly generated congestion numbers to model the initial horizontal and vertical usages of the bins due to the assumed signal net routing. The congestion values were generated such that the signal nets consumed between 30% and 60% of the bin capacities. The bin size was assumed to be $20\mu m \times 20\mu m$. The experiments were performed on Pentium-4 processor Linux machines, each with a clock speed of 2.4 GHz.

[3]The design of ground networks is fundamentally the same as that of a power network. The only difference in the design of ground networks is that the currents flow into the nodes of the ground network and all nodes are required to be less than a specified voltage.



(a) Optimized non-uniform power grid for a wire-bond package

(b) Optimized non-uniform power grid for a flip-chip package

Figure 2.8: The wire density pattern of power grids constructed by the proposed optimization for a 2cm$\times$2cm chip divided into 100 tiles, superimposed over the current density patterns. The regions with darker shades have higher current densities.

Figure 2.8 illustrates the non-uniform grids obtained at the end of the optimization heuristic for a 2cm$\times$2cm chip divided into 100 tiles. Figure 2.8(a) refers to the case when the V_{DD} pads are distributed at the periphery of the chip, as for a wire-bond package. Figure 2.8(b) shows the grid obtained when the pads are distributed throughout the chip, as for a flip-chip package. The grids are superimposed on the current density patterns, where darker colors represent higher current density regions. The light dashed lines represent the skeleton grid and the dark solid lines represent the actual grid. According to the scale chosen, one solid line is a substitute for ten wires in the actual grids. The highest current density regions, represented by the darkest shade in the figure, has a range of currents of about five to six times greater than the range of currents in the lowest current density regions, represented by the lightest shade in the figure. As seen

from the figure, it is not always true that the densest grids would lie in tiles with the highest current densities. The requirement to maintain sufficient voltage levels, in regions which draw high currents, can be met with fewer wires, provided there are enough pad connections to the P/G wires in that region. Hence, factors such as the pad locations also affect the density of the power grids in a given region.

Two sets of experiments were conducted that intended to compare our approach with:

1. The wire area utilized by a uniformly structured grid and constant wire width throughout the chip. Table 2.4 refers to this comparison.

2. The wire area utilized by a uniform grid for which wires are sized differently in different regions of the chip. Table 2.5 refers to this comparison. The number of wires for the uniform grids of Table 2.5 is less than that of the uniform grids of Table 2.4, but the wire widths are more for wires in some of the high current density tiles of the uniform grids of Table 2.5.

Ckt	# of Tiles	V_{init_x} (V)	V_{opt_x} (V)	# of Ports Per Tile	# of Wires Per Iter	Wire Area Regular Grid (cm^2)	Wire Area Proposed Design (cm^2)	% Reduction in Wire Area	CPU Time (mins)
1	80	0.852	1.081	52	5	0.0962	0.0739	23.12%	86.2
2	100	0.847	1.083	44	5	0.0840	0.0699	16.82%	67.4
3	144	0.852	1.087	36	6	0.0922	0.0764	17.14%	55.6
4	160	0.858	1.083	32	7	0.0984	0.0861	12.46%	48.3

Table 2.4: A comparison of the wire area used by the locally regular, globally irregular power grids, designed using the proposed method, and the grids employing a globally regular structure, and a constant wire size.

For the power grids of Table 2.4 and Table 2.5, the V_{DD} pads were distributed throughout the chip. The weights in the cost function of equation (2.37), were set to $\alpha = 1$ and $\beta = 0$, so that this power grid design for the comparison did not consider the

congestion problem. As listed in Table 2.4, the number of tiles into which the chip was divided for optimization are given in the second column. The third column (V_{init_x}) refers to the voltage of the most critical node before the optimization, when starting from an initial grid with equal number of wires in all tiles. Column four (V_{opt_x}) indicates the voltage at the end of the optimization process. This voltage is measured using an exact simulation of the power grid circuit, i.e., without the approximation of reducing the number of ports and sparsification of A matrix. The optimization is run for about two to five additional iterations even after meeting the voltage drop constraint, so that the errors introduced by the port approximations and sparsification can be accounted for by a little over-design. Also, if the grid meets the IR drop constraints but the EM constraints are not yet satisfied, the optimization is continued by iteratively adding wires in the tiles where EM constraints are violated, until all wires have a current density less than the specified current density. The number of ports retained for each tile after the process of *Port Approximation* is shown in the fifth column. Column six shows the number of wires that were added in every iteration in the tile having the least cost for wire addition. The seventh column shows the amount of wiring area that would be utilized by a P/G network topology having a constant pitch of wires throughout the chip and constant wire width. By enumeration, a minimum number of wires that satisfy the voltage drop constraint, was chosen to construct this regular grid. The eighth column (Wire Area Proposed Design) shows the amount of wire area used by the proposed design at the end of the optimization. The wire widths used by the proposed optimization is the same as the one used for the regular grid for a fair comparison. As seen from the table, there is a saving of about 12% to 23% in wire area by the proposed optimization scheme over the topology having a regular grid and constant sizing with same number of wires in all tiles.

Table 2.5 shows the comparison of wire area of the proposed P/G network structure with a design that employs wire width sizing, starting from an initial uniform grid with same number of wires in all tiles and uniform wire widths. It should be emphasized

Ckt	# of Tiles	# of Sized Tiles	Wire Area Sizing (cm^2)	Wire Area Proposed Design (cm^2)	% Reduction in Wire Area
1	80	32	0.0843	0.0739	14.07%
2	100	38	0.0883	0.0699	20.84%
3	144	48	0.0894	0.0764	14.54%
4	160	62	0.1096	0.0861	21.44%

Table 2.5: A comparison of the wire area used by the locally regular, globally irregular power grids, designed using the proposed method, and the grids employing a globally regular structure, with irregular wire sizing.

that the number of wires in the tiles for the regular grid of Table 2.5 is less than that (about 0.6-0.7$\times$) of the regular grid design that is listed in column seven of Table 2.4 and the minimum wire width of the regular grid design of Table 2.5 is the same as that of the proposed design. The third column in Table 2.5 refers to the number of tiles in which the wire widths were incrementally sized. The tiles with high current densities are identified and the wire widths in those tiles are incrementally sized until the voltage drop constraints are met. We compare the sizing solution against the four optimized power networks of Table 2.4. The fourth column (Wiring Area Sizing) lists the wire area used of various chips by the wire sizing solution. The fifth column (Wire Area Proposed Design) lists the wire area used by the proposed optimization. The last column indicates the percentage change in the wire area. There is a reduction of 14% to 21% in wire area by the proposed optimization scheme over the design with the wire width sizing technique.

To verify that the proposed design technique of producing locally regular and globally non-regular power grids is congestion-aware, we perform another series of exper-

Ckt	# of Tiles				$\gamma = 0.25$			$\gamma = 0.5$			$\gamma = 0.75$		
		$\alpha = 1, \beta = 0$			$\alpha = 1, \beta = 1$			$\alpha = 1, \beta = 1$			$\alpha = 1, \beta = 1$		
		Area (cm^2)	Congestion		Area (cm^2)	Congestion		Area (cm^2)	Congestion		Area (cm^2)	Congestion	
			Avg	Max		Avg	Max		Avg	Max		Avg	Max
1	120	0.753	0.615	2.039	0.788	0.565	1.312	0.795	0.545	1.395	0.784	0.512	1.476
2	150	0.712	0.593	1.895	0.734	0.572	1.214	0.742	0.561	1.314	0.734	0.534	1.371
3	180	0.789	0.658	1.952	0.809	0.591	1.118	0.816	0.572	1.232	0.812	0.567	1.291

Table 2.6: A comparison of non-congestion-aware and congestion-aware, piecewise-uniform power grid design.

iments. As shown in Table 2.6, we design the piecewise-uniform power grid proposed in this thesis, both with and without the congestion penalty term in the cost function. According to the cost function equation (2.37), the values in the table corresponding to the parameters, $\alpha = 1$ and $\beta = 0$, refer to the design of the proposed locally regular and globally regular supply grid with no penalty for congestion. The values in Table 2.6 corresponding to the parameters, $\alpha = 1$ and $\beta = 1$, refer to the congestion-aware design of the piecewise-uniform supply grid. In this design, the normalized weight parameters, α and β, are used in the cost function to balance the objectives of (i) reducing the IR drop with minimum increase in area and (ii) minimizing congestion aggravation. As given by equation (2.36), the value of the parameter γ is varied to relatively penalize the maximum and the average congestion of all bins. The value of $\gamma < 0.5$ penalizes the maximum congestion more than average congestion. As seen in Table 2.6, the congestion-aware design achieves substantial reduction in the maximum and the average congestion values over a design with no congestion penalty. However, the congestion-aware design utilizes more wiring area than the non-congestion-aware design. This is due to the fact that in a congestion aware design , in every iteration, the wires are placed in suboptimal locations from the point of view of IR drop reduction. Hence, more wires are needed to meet the IR drop constraint. The tradeoff between congestion reduction

and wire area increase can be explored by changing the relative weights, α and β, to penalize one objective more than the other in the cost function of equation (2.37).

The runtime of the proposed design of the power grid is a function of the number of wires added in each iteration step and the number of ports that are removed to make the macromodels. To better understand the runtime, accuracy and area-reduction tradeoffs, we perform a series of experiments in which we construct a grid using the proposed optimization scheme for power delivery to 2cm$\times$2cm chip divided into 100 tiles. The runtime, accuracy and wire area reduction tradeoff is explored by varying the number of ports kept for the tiles and the number of wires added per iteration.

Design	# of Ports	# Wires per Iteration	% Reduction in Wire Area	Average Error in Optimal Design	CPU time (mins)
1	44	5	19.41%	2.14%	63.4
2	44	6	16.23%	2.32%	60.6
3	44	7	13.34%	2.64%	58.9
4	40	5	19.09%	3.67%	56.2
5	40	6	16.50%	3.95%	54.5
6	40	7	14.41%	4.23%	53.4
7	36	5	19.52%	5.81%	41.6
8	36	6	17.12%	6.04%	40.4
9	36	7	13.21%	6.23%	39.2
10	28	5	18.98%	9.02%	19.7
11	28	6	16.47%	9.55%	18.6
12	28	7	13.59%	10.17%	17.1

Table 2.7: Various tradeoffs involving number of ports kept, number of wires added in each iteration, reduction in wire area, runtime, and accuracy in power grid design.

As shown by Table 2.7, on the one hand, the runtime significantly reduces by removing more ports while forming the macromodels, but on the other hand, the errors in measurement of voltages increase. If we add more wires per iteration for the same number of kept ports, the runtime slightly improves but at the cost of accuracy and savings in the wire area. By adding more wires per iteration, we are increasing the amount of over-design and hence reducing the improvement in area reduction over other topologies. Thus according to the level of accuracy and the order of runtime required, we can choose the appropriate parameters to guide the optimization.

2.7 Solution Technique 2: A Partition-based Approach using Locality

To improve the slow runtimes of our power grid design technique, described in Section 2.6, we present in this section of the thesis a highly efficient alternative algorithm for supply network design.

This work proposes an algorithm, which employs a successive partitioning and grid refinement scheme, for designing the power distribution network of a chip. In our iterative procedure, the chip area is recursively bipartitioned, and the wire pitches and the wire widths of the power grid in the partitions are repeatedly adjusted to meet the voltage drop and current density specifications. By using the macromodels of the power grid constructed in the previous levels of partitioning, the scheme ensures that a small global power grid system is simulated in each iteration. The idea is based on the notion that due to the locality properties of the power grid, the effects of distant nodes and sources can be modeled more coarsely than nearby elements, and includes practical methods that enhance the convergence of the iterative conjugate-gradient based solution engine that is used in each step. Finally, a post-processing step at the end of the optimization is employed to maximize the alignment of wires in adjacent partitions.

Schemes for automated power distribution network design can be divided into the following two categories, based on tradeoffs between the accuracy of the embedded power grid simulator and the level of sophistication of the optimizer:

(A) Heuristic iterative optimization methods, employing an explicit and exact circuit analysis step in the main optimization loop to determine constraint violations in the power grid [SS05], [SGS00].

(B) Mathematical optimization schemes, formulating the problem as a nonlinear program by approximating the circuit equations, and solved with the aid of nonlinear optimization techniques [TSL03, WHC+01, WC02, WMS05, MK92].

The desirable characteristic of the supply network design methods based on scheme (A) is the guaranteed accuracy of the final solution, ensured by the process of performing an explicit circuit simulation step in each iteration, to detect and fix the IR drop and EM violations. However, these methods typically have large runtimes as each simulation of a power network, comprising hundreds of thousands of electrical nodes, is extremely time consuming. The methods built on scheme (B), solve the supply net design problem by formulating it as an optimization problem of minimizing a nonlinear function subject to nonlinear constraints. In this scheme, typically, the circuit analysis is implicitly carried out by explicitly or implicitly listing the circuit constraints, i.e., the Kirchoff's current and voltage laws and the device equations, as a part of the constraints set. In the original form, the solution to such a nonlinear problem formulation is known to be computationally intensive which makes it prohibitive for problems involving millions of design variables. Hence, to achieve efficiency these methods typically either employ some constraint relaxations to transform a general nonlinear optimization problem to special forms of nonlinear programs such as the convex program, which can be efficiently solved, or introduce some approximations to reduce the problem size. Although, these methods provide a solution that is more efficient than those from scheme (A), the

final solution is inherently subject to inaccuracies due to the relaxations and approximations introduced in the original nonlinear problem formulation.

In this work, we propose a novel and fast, yet accurate, algorithm to design the power distribution network in the form of a non-uniform power grid. We use a hierarchical design approach, based on successive partitioning of the chip area, to design the supply network.

2.7.1 Locality in the Structure of Power Grid

Our procedure achieves efficiency by using the notion of *locality*, similar to that proposed in [Chi04]. [4] This concept is based on the observation that nearby elements have the greatest influence on the voltage at any node. Therefore, while constructing the power grid locally in a specific region of the chip, it suffices to use fine-grained and accurate models only in or near the that region. The regions of chip that are *far away* from the specific region are not likely to affect the local grid design in the specific region, and can be abstracted away using coarse models.

This concept of locality is illustrated in Figure 2.9, where a violating grid region, i.e., a region that violates the constraints, is shown by the shaded rectangle. Generally speaking, these violations can be fixed by adding more power grid wires locally in and around the violating region. Due to locality, as we make these local changes, it is reasonable for the details of parts of power grid in the regions far away from the violating region to be abstracted away, as shown in Figure 2.9(b). The use of these abstractions, or *macromodels*, for parts of the power grid circuit has two main advantages. Firstly, they improve the speed of the analysis, since they reduce the size of the system to be solved. Secondly, by focusing the grid design effort on the details of the local region only, the search space for choosing the optimal design parameters, e.g., the wire pitches

[4] Recently, the implications of property of locality for a transient power grid simulation was presented in [PC06]. Since our work deals with a static or a DC simulation of power grid, we do not incorporate these transient effects in our work.

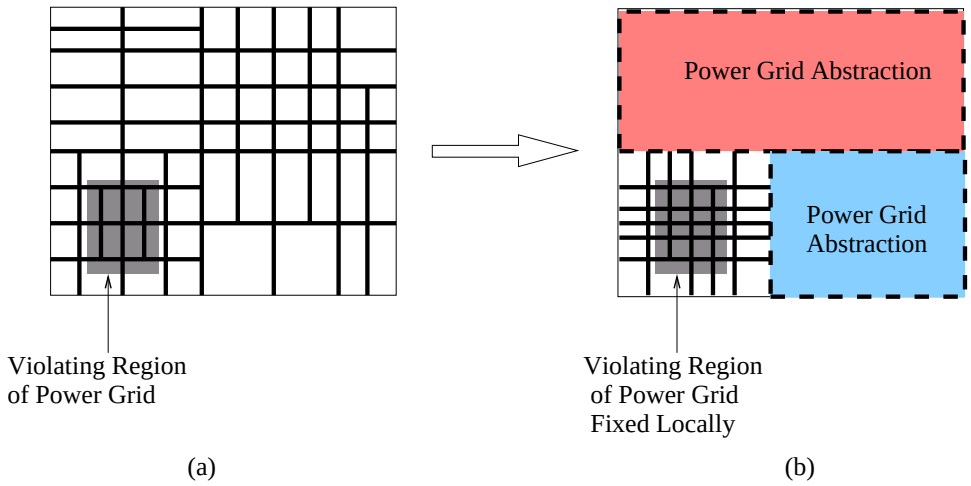


Figure 2.9: The concept of locality in power grid design. (a) A detailed power grid with a violating region shown with the shaded rectangle. (b) Details of regions of power grid far from the violating region abstracted away and the violations fixed locally.

and the wire widths, is significantly reduced.

The idea of working with detailed local models and abstractions of far away regions is especially useful in the case of flip-chip packages where the power pads are distributed throughout the chip area by using C4 bumps. For a flip-chip package, most violations of power grid in a specific regions can be fixed by locally modifying the power grid just in and around the violating regions. Due to the availability of a sufficient number of pads around the violating region, the power grid wires in the local region contain the path of least resistance for the current to flow from the nearest V_{DD} pads to the violating nodes. The same idea may not be true of chips with wire-bond packages, since the power pads are on the periphery of the chip, and the strategy of local modification of the grid may not work because of the concentration of power pads on the chip periphery. However, the hierarchical design method mitigates the effect of weak locality in grid design, as may be the case for wire-bond chips. By the process of successive bipartitioning, the hierarchical grid design procedure ensures the presence of low-resistivity conduction

paths in various regions of the chip. Due to the availability of these high-conductance paths, it is reasonable to expect that a violating region of the power grid, which may or may not exhibit a strong locality property, would be appropriately fixed by employing a top-down hierarchical partitioning scheme.

2.7.2 Outline of the Power Grid Design Procedure

Based on the hierarchical design approach and the notion of locality in power grid design, we propose an efficient and accurate grid design procedure. The method employs an iterative scheme of recursively partitioning the chip area and constructing the power grid locally in the partitions by adjusting the wire pitches and the widths. The power grids constructed by our design procedure have a locally regular, globally irregular structure, as described in Section 2.3. The grids within a partition are constructed to be uniform, i.e., they have the same wire width and wire pitch, but power grid wires in different partitions may have different widths and pitches, as determined by the current density requirements of the underlying functional blocks. Such a locally uniform, globally non-uniform power grid structure has the desirable properties of efficient wire area utilization, ease of power grid circuit modeling and optimization. The outline of our method is as follows:

- We present a heuristic algorithm to design a supply grid that meets the IR drop and the EM constraints. The optimization is carried out under DC conditions using an iterative refinement scheme. In our implementation, the power grid is designed for the top two metal layers of the chip, but the same procedure can be easily extended to design a power network spanning multiple layers of metal.

- We commence by dividing the chip area into two partitions. The power grid in the two partitions is then constructed by placing thick or very wide wires at an initial pitch. The pitches in the two partitions are then repeatedly reduced until the initial grid meets the constraints.

- In each of the subsequent iterations, a previous partition is further divided into two smaller partitions and a refined power grid is reconstructed *locally* in these smaller partitions. The grid refinement process comprises of decreasing the wire width by a factor of the width in the previous partitioning level, and then iteratively decreasing the wire pitches. The solution of the grid designed in the previous iterations is used to guide the design of the power grid in the current iteration.

- We use a previously proposed macromodeling technique [ZPS+00], described in Section 2.5, to construct abstractions of partition of the power grid. Employing a hierarchical circuit analysis method in each iteration, to determine the nodes and branches which violate the reliability constraints, ensures the accuracy of grid design. Since, in each iteration we construct the macromodels of only two partitions, and reuse the macromodels formed in previous iterations, the analysis step is very fast.

- We further speed up the circuit analysis step. In order to reduce the simulation time for the global matrix system, we use a preconditioned conjugate gradient based iterative linear solver, with an initial guess vector whose components are derived from the power grid solution of the previous iteration. Thus, reusing the grid solution of the previous iteration to drive the grid design in the current iteration aids in making the procedure faster.

- At the end of the optimization, a post-processing step is used to maximize the alignment of wires in adjacent partitions.

In the following sections we explain the power grid design scheme based on the recursive bipartitioning and grid refinement idea. We employ this grid design procedure to construct power grids for the top two metal layers. The extension of the grid design procedure for power grids spanning multiple metal layers in described in Section 2.7.7.

2.7.3 Grid Refinement

A power grid covering the entire chip area comprises thousands of wires, and its equivalent circuit, as shown in Figure 2.2, contains millions of electrical nodes, making it prohibitive to simulate. To achieve efficiency in the circuit analysis step, it thus becomes essential to work with a coarse initial power grid representation, which yields a power grid circuit of manageable size, and then iteratively refine the coarse model. In our power grid design procedure, we construct the coarse grid by using unrealistically thick power grid wires initially, and then subsequently improve the coarse grid model by a *grid refinement* operation, in which a power grid containing a smaller number of thick wires is replaced by a grid comprising of a larger number of thinner wires. In our power grid circuit model, we assume the via resistances to be proportional to the overlap area of horizontal and vertical wires. Hence, for thicker wires we have a few vias having higher resistances. Replacement of smaller number of thicker wires with larger number of thinner wires, results in more number of vias, but with lower resistances.

The advantage of using a grid with thick wires is that it greatly reduces the system size, as there are fewer electrical nodes in the equivalent circuit model of Figure 2.2. Moreover, since we employ a hierarchical approach for power grid design, the coarse grid representation is adequate for the abstractions of parts of power grid circuit. However, such a grid may result in over-utilization of the chip wiring resources. Hence in our optimization procedure, we begin with a grid with very thick power grid wires to gain speed up in the circuit analysis step, and then subsequently perform grid refinement in later iterations to achieve savings in the wire area.

The effective resistance of a given region of a power grid circuit depends on both the number of wires in the region, and the resistances of the individual wires. Having more wires reduces the effective resistance of the grid by increasing the number of paths that current can take from a V_{DD} pad to any node. Assuming a constant wire width for the given region, as required for a locally regular structure of the power grid in our work,

the number of wires is determined by the wire pitch of the region. On the other hand, the resistance of the individual wires depend on the widths of the wires.

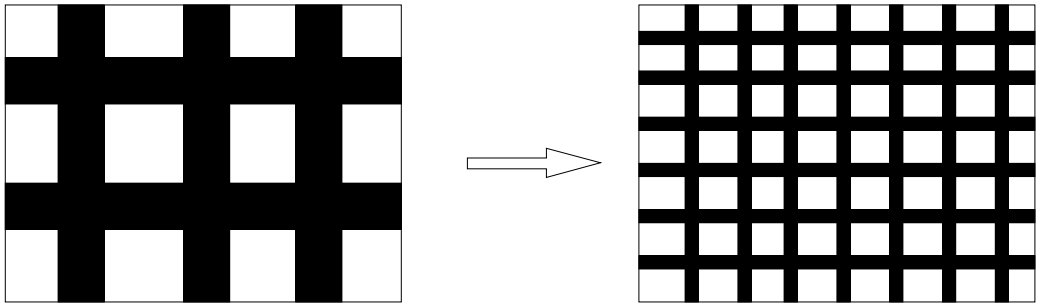


Figure 2.10: A power grid with thick wires and large pitch refined to a grid with thinner wires and smaller pitch.

Figure 2.10 depicts the grid refinement operation. As seen in the figure, by a grid refinement operation, a coarse grid with thick wires and a large pitch, is converted to a grid of thinner wires and smaller pitch. It is reasonable to expect that the impact of an increase in the resistances of the individual wire segments of the refined power grid, due to reducing the wire widths of the coarse grid, would be offset by having more wires in the refined grid. Moreover, since the magnitude of currents carried by more thinner wires of the refined grid reduces, the wires in the new and the old grid are expected to have similar current densities. This observation that the effective resistance of a power grid circuit depends both on the wire width, and the wire pitch, forms the basis of employing grid refinement in our optimization procedure.

The grid refinement idea is similar to the multigrid based method of coarsening the power grid system [WMS05]. However, the method in [WMS05] deals with a fixed topology of the supply network and optimizes the parameters of a pre-constructed power grid. As will be explained in the following sections, in our method, the optimization procedure decides the topology of the power grid by repeatedly adjusting the wire pitches and wire widths, while maintaining the locally regular, globally irregular structure of

the supply network.

2.7.4 Power Grid Design by Recursive Bipartitioning

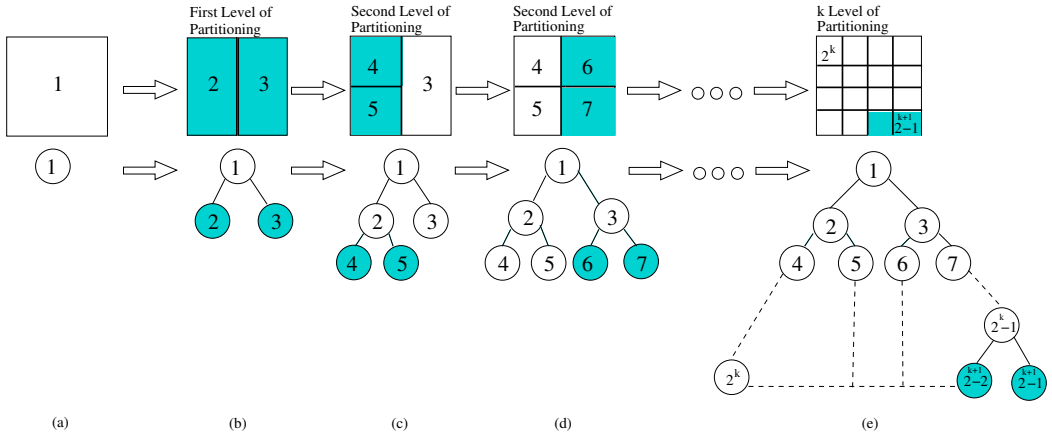


Figure 2.11: The recursive bipartitioning process to design the power grid. Each partition cut is equivalent to adding two elements in the binary *partition_tree*. The height of the tree represents the level of partitioning. The two shaded elements refer to the two partitions where the new power grid is designed in the current iteration.

We apply the *divide and conquer* approach to our power grid design procedure by successively dividing the chip area into smaller regions or partitions and iteratively constructing power grids in the smaller partitions, using the notion of locality, introduced in Section 2.7.1. The recursive bipartitioning idea employs the strategy of solving a small local power grid design problem in each step, and involves the selection of optimal pitches for two partitions, such that (i) the new grids constructed in the two partitions each meet all of their specifications, and (ii) the previously constructed grids in the other partitions maintain their correctness in terms of meeting the IR drop and EM constraints.

Figure 2.11 illustrates the recursive bipartitioning process. As shown in the figure, the process of partitioning the chip area to iteratively construct the power grid is equiv-

alent to growing an *almost complete* binary tree[5] referred to as the *partition tree*. Partitions that have already been processed are represented as the non-shaded tree elements, and the two partitions being processed in the current iteration, referred to as the *active partitions*, are shown as the shaded tree elements. Each element in the partition tree contains information about the wire widths and the pitches of the power grid constructed in the corresponding partition. The height of the partition tree determines the current level of partitioning. A new partition is constructed by making a vertical (horizontal) cut, by introducing a vertical (horizontal) *partition wire* halfway across the parent partition as a part of the power grid. Consecutive levels of partitioning alternate the cut directions between vertical cuts and horizontal cuts. The process of adding two children to a parent node in the tree is equivalent to the grid refinement operation described in Section 2.7.3. In other words, when two child nodes are added to a parent node in the tree, the coarse power grid in the partition corresponding to the parent node is replaced by finer grids of the child nodes.

In the following sections we explain the idea of recursive bipartitioning for power grid design in detail.

First Level of Partitioning

As seen in Figure 2.11(b), the first level of partitioning is equivalent to adding two child nodes to the root of the partition tree. In this step, we begin with the full chip area and divide it into two parts by adding a vertical power grid partition wire across the chip going through its middle. We define this as a *vertical cut* across the *first partitioning level*, referred to as $Part_1$.

Figure 2.12 depicts the division of the chip into two partitions. Next, we select an initial wire width and pitch for the partitions such that the worst case voltage drop meets

[5]As the smaller partitions are produced by the splitting the larger partition in sequence from left to right, the resulting partition tree has children added from left to right, which makes the tree an almost complete binary tree.

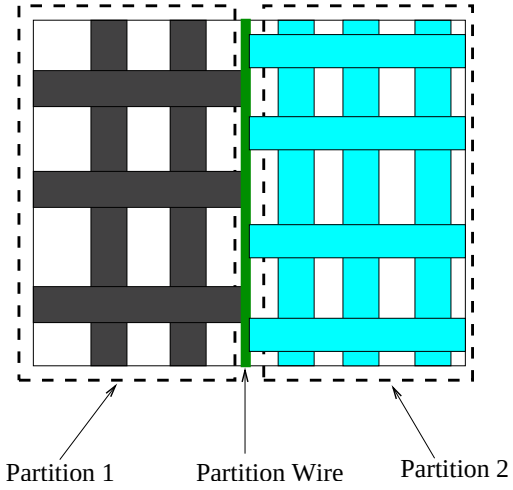


Figure 2.12: First level of partitioning: A chip divided into two partitions and the power grid constructed in the two partitions.

its specifications. These choices are made by constructing an initial grid in the two partitions, and then iteratively decreasing the pitches of power grid in the two partitions until there are no IR drop and EM violations. This choice of starting point for the initial wire widths and the pitches is determined empirically, or based on designer input, so that we begin not too far away from the final solution point in the search space.

The circuit analysis step, which detects constraint violations in the power grid, is performed using the macromodeling approached discussed in Section 2.5, coupled with a preconditioned conjugate gradient based iterative solver to speed up the power grid circuit simulation. Using the macromodeling idea, all nodes in the two partitions, except the port nodes, are abstracted away. The port nodes lie on the partition wire through which the two partitions connect to each other. Figure 2.13 depicts this situation, where the left and the right partitions are abstracted as macromodels connected to each other through the port nodes located on the partition wire. As seen in the figure, the power grid constructed in the two partitions is reduced to a system of two macromodels that connect to each other through the port nodes. Using the matrix algebra described in

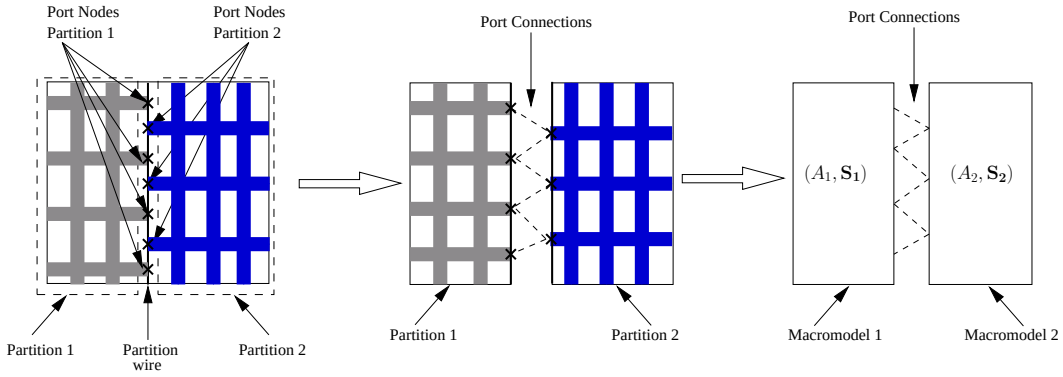


Figure 2.13: Power grid constructed in the two partitions changed to a system of macromodels. The macromodels connect with each other through the port nodes on the partition wire.

Section 2.5, we construct the macromodels given by the parameters $(A, \mathbf{S})$ for the two partitions. Following the hierarchical circuit analysis approach explained in Section 2.5, the macromodel parameters are stamped in the global system, $M\mathbf{X} = \mathbf{b}$, as given by MNA Equation (2.7).

The global system is solved, and subsequently, the node voltages within the two partitions are determined to check for any IR drop and EM violations. We improve the runtime of our procedure by solving the global system given by Equation (2.7) using an iterative solver: specifically, a conjugate gradient method with diagonal preconditioning. As will be explained in the following sections, we can reuse the solution of the power grid designed in the previous iteration to speed up the current iteration of the grid design procedure by employing an iterative linear solver to solve the global matrix of Equation (2.7).

In case a partition does not meet the voltage drop and current density specifications, the wire pitch of the partition is reduced by a factor β_1, and the process of creating the macromodels and solving the power grid system using the iterative linear solver is repeated. As an implementation detail, it is worth emphasizing that at this first level

of partitioning, we use unrealistically thick wires, e.g., wires having widths between 80 and 150 μm, and these will later (after further partitioning) be replaced by thinner wires through the grid refinement operation. The vias, connecting the top two metal layers, are assumed to have resistances proportional to the overlap wire area between the horizontal and vertical layers. Thus, during the grid refinement process the via resistances and number of vias are appropriately adjusted. Due to using thicker wires initially, the system size of each partition is fairly small, e.g., 1000 to 3000 nodes. As a consequence, the iterative process of constructing new macromodel parameters ($A, \mathbf{S}$) after decreasing the wire pitch and simulating the grid repeatedly is extremely fast. At the end of this step, we obtain a coarse power grid constructed in the two partitions that span the entire chip area.

Second Level of Partitioning

Recall that $Part_1$, the previous level of partitioning, used a vertical wire that divided the chip area in the left and right halves. We use the power grid constructed at the first partitioning level to guide the grid design at the next level, referred to as $Part_2$. The second level of partitioning has two steps, as depicted in Figure 2.11(c) and (d), referred to as $Part_{2_1}$ and $Part_{2_2}$, respectively. First, the power grid constructed in the left partition is ripped up. Next, a horizontal cut is made in the left partition by introducing a horizontal power grid partition wire in the middle of this region. As a result, the left partition is further divided into top-left and bottom-left partitions, as shown in Figure 2.14. As seen in Figure 2.11(c), $Part_{2_1}$ begins by adding child nodes to the parent nodes 2 and 3 in the partition_tree, or equivalently, growing the almost complete binary tree to the next level. Leaving the previously constructed grid in the right partition intact, a finer grid is designed in each of the two new partitions, i.e., the top-left and the bottom-left partitions.

The grid design procedure for these two partitions is similar to that employed in

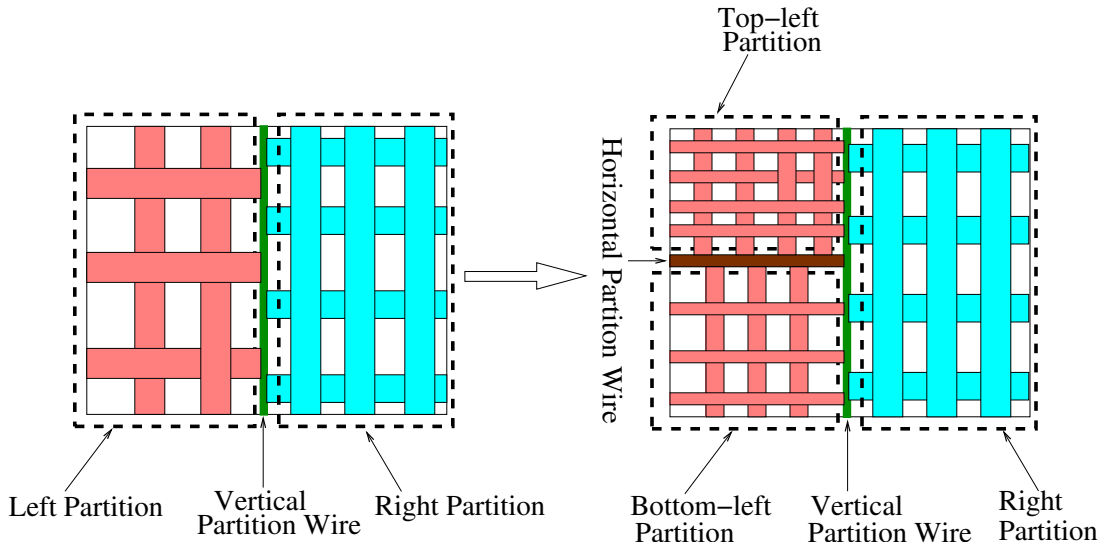


Figure 2.14: The second partitioning level in the power grid design procedure. The coarse grids in the left and the right partitions are refined in this level.

$Part_1$, where grids were constructed in the left and the right partitions. We begin the grid construction in the top-left and bottom-left partitions by using the notion of grid refinement to construct an initial grid whose wire width is chosen using a multiplicative factor, $\gamma_2 < 1$, on the wire width at the parent node, i.e., the wire width used for the grid in the left partition at the first level. In general we set, $w_k = \gamma_k \cdot w_{k-1}$, where w_k is the wire width for the k^{th} level of partitioning and γ_k is the multiplicative factor used when refining from the $(k-1)^{st}$ level of partitioning to the k^{th} level. The via resistances, assumed proportional to the overlap area between the horizontal and vertical wires, are appropriately adjusted at each partitioning level. This is precisely the grid refinement technique explained in Section 2.7.3 and shown in Figure 2.10, with the difference that we now also maintain a coarse macromodel for the right partition. The grid of the left partition, which had thicker wires and a larger pitch, is therefore replaced with the grids in partitions top-left and bottom-left that each have thinner wires but a smaller pitch. The macromodel parameters $(A, \mathbf{S})$ are calculated for the top-left and bottom-left partitions

and they are stamped in the global system of Equation (2.7), along with the macromodel parameters of the right partition which were previously computed in $Part_1$.

To solve the global matrix system of Equation (2.7), we use a preconditioned conjugate gradient iterative solver to speed up the solution of the linear system of equations. The simulation speed up is obtained by the observation that since the result of the grid refinement operation is designed to offer a similar effective resistance, it is reasonable to expect that the perturbation between the old system (i.e., the power grid design obtained at the end of the first level of partitioning), and the new system (i.e., the refined power grid for the left partition at the second partitioning level), would be small. Hence, we can use the voltages of port nodes located on the vertical cut wire, solved by the old global system of Equation (2.7), as the components of the initial guess vector X^0 corresponding to the same ports in the new system. The other components of the guess vector X^0, which correspond to the voltages of the newly introduced port nodes by addition of a partition wire, are set to be equal to V_{spec}. Thus, each component i of the initial guess vector used for the conjugate gradient method, after introducing each new horizontal or vertical partition wire is given by:

$$\mathbf{X_i^0} = \begin{cases} V_{port_{old}} : \text{i} \in \text{Port node of the old system} \\ V_{spec} : \text{i} \notin \text{Port node of the old system} \end{cases} \tag{2.40}$$

We find that reusing the solution of power grid design of the previous iteration for solving the new global matrix system improves the runtime of our supply net design procedure. The guess vector as formed by equation(2.40) provides a fairly good starting point for the conjugate gradient method, as a result it converges to the final solution in only a few steps. To improve the conditioning of the problem, we use a diagonal or a Jacobi preconditioner for the conjugate method. We pre-multiply the global system of Equation (2.7) by a diagonal matrix $\sum$ consisting of the reciprocals of the corresponding diagonal entries of the global MNA matrix M. Alternatively, we could have chosen Cholesky preconditioning, which is known to have better conditioning proper-

ties [Ber99]. However, as the step of solving the global system is a part of the inner loop in the grid design procedure, it is much cheaper to from a diagonal preconditioner as compared to the more expensive operation of finding the Cholesky factors of the global matrix M. After solving the global matrix system and determining the port voltages from the solution vector $\mathbf{X}$, using the hierarchical analysis technique, the IR drop and EM violations are detected for the power grid in the top-left and bottom-left partitions. The violations are corrected by inserting more wires in the partitions, by reducing the pitch by a factor β_2.

In addition to the voltage drop and the current density specifications, there is an additional requirement that the grids of the top-left and bottom-left partitions must meet. It is essential that the process of ripping up the original grid of the left partition and replacing it with the grids constructed in the top-left and the bottom-left partitions does not render the grid of the right partition ineffective in terms of meeting the specifications. To inspect whether the correctness of the grid in the right partition is maintained, we could completely solve the right partition power grid again. However, this would be costly in terms of runtime, as we would need to check the voltages of all of the nodes of the grid within the right partition, each time the pitches of the top-left or bottom-left partitions are decreased. To avoid this high simulation cost, we make use of the abstraction of the power grid of the right partition effectively. It is reasonable to expect that a very small change in port voltages of right partition would result only in a small change in the voltages of the internal nodes of the right partition. Thus, the voltages at only the port nodes of the right partition grid are evaluated. These voltages are compared with the port voltages of the right partition power gird obtained at the end of $Part_1$ and a grid violation, referred to as *previous_grid_violated*, is flagged if the maximum change in the port voltages is greater than a specified value, MAX_{spec}.

In the event of such a violation, the pitches of the top-left and bottom-left are further decreased, thereby increasing the number of wires in the power grid in order to maintain the correctness of the previously designed grid in the right partition. This process

ensures that the port voltages of the previously correct grid design are not significantly disturbed by the new power gird which replaces the previous grid.

The process of power grid design in $Part_{2_2}$ is similar to that of grid design in $Part_{2_1}$. In this case, as shown in Figure 2.11(d), the active partitions are the top-right and the bottom-right partitions. For $Part_{2_2}$, the old power grid system becomes the one designed in $Part_{2_1}$. The rest of the procedure of power grid design is essentially the same and can be derived from the explanation for constructing the power grid in $Part_{2_1}$.

Grid Refinement by Recursive Bipartitioning

In the process of growing the partition tree of Figure 2.11(e), when each time two child nodes are added to a parent node in the partition tree, the coarser power grid in the partition corresponding to the parent node is converted to finer grids in the two active partitions corresponding to the child nodes. As described in the previous section, the wire widths of the child nodes are some factor, $\gamma_k < 1$, of the widths of the parent nodes. The via resistances, for connections between the horizontal and vertical wires, are also recomputed for the new partitions according to the new overlap area between the horizontal and vertical layers.

The wire width is kept constant while designing the local grid for the two active partitions, and the search is restricted to choosing the optimal wire pitches only. Alternatively, we could have chosen to make the wire width also a design variable in the process of grid construction for the two partitions. However, changing the wire width repeatedly in the local grid design process would lead to significant changes in the elements of the conductance matrix from which the macromodel parameters $(A, \mathbf{S})$ are derived. Thus, the solution of the global MNA equation (2.7) could differ considerably from one inner loop iteration to the next. This could result in slow convergence or many inner loop iterations to find the best combination of wire widths and pitches that meet the reliability constraints. Moreover, at the initial partitioning levels we are building a

coarse solution to capture the essential features of the final power grid solution. Since using reduced wire pitches and thicker wires are expected to have similar effect on the equivalent resistance of the grid, we choose a strategy of fixing the wire width, and limiting the search space to find the optimal set of wire pitches, thus significantly reducing the number of optimizable parameters and the computational effort.

With the growth of the partition tree, the number of partitions in the current partitioning level increases exponentially with the number of levels.As a result, the number of port nodes and thus, the size of the global system of equation (2.7), grows rapidly. This adversely affects the runtime of the design procedure as in each iteration, following a pitch decrease in any one of the active partitions, it is necessary to construct the global matrix M, whose dimensions are rapidly increasing which leads to increase in number of conjugate gradient steps to solve the global system. To overcome this problem, we employ the *port approximation* technique suggested in Section 2.6.2, to reduce the macromodel sizes. By this approach, some of the port nodes located on the partition wires are collapsed. The port approximation scheme helps in controlling the fast increase of the global system of equation (2.7) at the cost of reasonable simulation errors. However, the accuracy of the final solution is not compromised since the port approximation technique is switched off during the final few iterations of the design procedure.

The addition of two new child nodes in the partition tree can only take place if the following are satisfied:

1. IR drop and EM constraints, for the new power grid being constructed in the two active partitions, are met. These violations are detected by the hierarchical circuit analysis step.

2. There were no previous_grid_violated flags set, as described in Section 2.7.4. These violations are detected by checking the port voltages of all of the neighboring partitions of the two active partitions. If the maximum change, between the new port voltages of the neighboring partitions and the port voltages of the

neighboring partitions before constructing the new grid in the active partitions, exceeds a specified value, MAX_{spec}, the previous_grid_violation flags are set to true.

3. The wire pitches in the two active partitions are greater than the minimum wire pitch, p_{min}.

In order to fix the violations 1 and 2, the pitch of one or both the active partitions is decreased, depending on which of the two active partitions exhibits these violations. The minimum pitch violation is an undesirable breakdown in our power grid design procedure, and occurs when the grid refinement operation does not work, i.e., the increase in the wire resistance by replacing a power grid having thicker wires with a power grid having thinner wires cannot be compensated by adding more wires in the replacement grid.

These minimum pitch violations cannot be fixed locally by modifying the grid in the active partitions. In this case, we must traverse the partition tree to the other tree elements that neighbor the active partitions, and add more wires in these neighboring partitions by reducing the wire pitches within them. For example, referring to Figure 2.11, if power grid being designed for partition 7 runs into a minimum pitch situation, more wires are added iteratively to partitions 5 and 6 until the constraints are met. Fixing the minimum pitch violations thus adversely affects the runtime of the grid design procedure. However, we found out empirically that if the width reduction factor at the k^{th} partitioning level, $\gamma_k < 1$, is chosen such that it is not too small, (empirical values correspond to $\gamma_k \in [0.65, 1)$), such breakdowns are very rare events.

We conclude the optimization procedure at the end of k levels of partitioning. The value of k is determined by specifying the minimum size of the partition by the following relation:

$$k = 2 \cdot \min\left(\left\lceil \log_2 \frac{CHIP_w}{PART_w} \right\rceil, \left\lceil \log_2 \frac{CHIP_l}{PART_l} \right\rceil\right) \tag{2.41}$$

where $CHIP_w$ and $CHIP_l$ are the chip dimensions, and $PART_w$ and $PART_l$ are the specified dimensions of the minimum partition size.

2.7.5 Post Processing for Wire Alignment

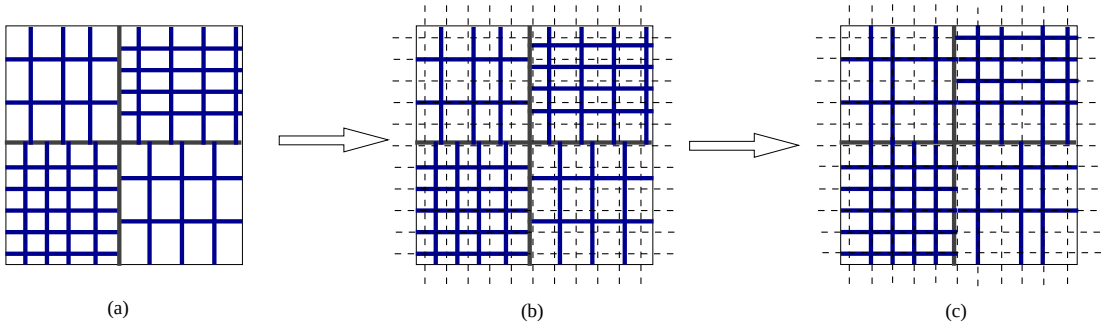


Figure 2.15: The post-processing step to align the power grid wires in different partitions. (a) Power grid wires in adjacent partitions are misaligned. (b) A minimum pitch virtual grid, shown with dashed lines is constructed over the entire layout area. (c) The power grid wires are moved to the nearest position on the virtual grid. The wires in adjacent partitions are better aligned now.

At the end of k-levels of partitioning, we have designed a power grid in each of the 2^k partitions of the chip, with potentially different wire pitches in each partition. As a result, the wires in the adjacent partitions maybe offset with respect to each other, as illustrated by the simple example in Figure 2.15(a), where a power grid with four partitions has misaligned wires in the adjacent partitions. This misalignment may lead to use of extra vias to maintain the electrical connectivity of the power grid wires in different metal layers.

To alleviate the misalignment, we introduce a post-processing step in our grid design procedure, illustrated through the example in Figure 2.15. We use the idea of the skeleton grid, described in Section 2.6.1, to align the wires in the adjacent partitions.

To rectify the misalignment in Figure 2.15(a), a skeleton grid, which is a uniform *virtual grid*, is superimposed over the entire layout area, whose pitch is chosen to be the minimum pitch of wires in all of the partitions. The virtual grid is a uniform grid with wires having a constant pitch throughout the chip area. By choosing the pitch of the virtual grid as the minimum of wire pitches of all the 2^k partitions, it is ensured that the virtual grid would at least have the same number of wires in any partition as the real local power grid. The virtual grid is represented by dashed lines in Figure 2.15(b). Next, the power grid wires in all partitions are moved to the nearest location on the virtual grid. The virtual grid thus acts as a place holder for the real power grid wires. As seen in Figure 2.15(c), the wires in the adjacent partitions are better aligned at the end of the post-processing step. To ensure that the small movement of power grid wires does not affect the correctness of the grid, we perform a complete simulation using the hierarchical analysis technique. Our experiments showed that the post-processing step hardly ever introduced any violations in the power grid circuit. In the rare cases where this was not true, the violations were easily fixed by adding more wires in the violating partitions. The extra wires are also added in such a way that they are placed on the virtual grid, to remove any possibility of misalignment.

2.7.6 The Complete Algorithm

The pseudo-code of the main loop of the power grid design algorithm is presented in Algorithm 1. We use the binary tree data structure to model the successive partitioning of the chip area. This is represented by the *tree* array in Algorithm 1. Each element of the tree array represents a node in the partition_tree of Figure 2.11, and contains the information about the partition dimensions, and the wire width and pitch of the power

Algorithm 1 Power Grid Design

1: **Power_Grid_Design**(*func_block_currents,power_pads_pos*)
2: /*Initialize the root node of the partition tree*/
3: tree[1].wire_width=W_{init};
4: tree[1].wire_pitch=P_{init};
5: tree[1].width=chip_width;
6: tree[1].length=chip_length;
7: i=2; k=$\lfloor log_2(i) \rfloor$;
8: cut_dir=0; /* cut_dir = 0/1 for a ver/horz cut*/
9: /*Begin Outer While Loop*/
10: **while** (i $<$ MAX_NUM_PARTITIONS) **do**
11: parent_index=i/2;
12: [$child_1$, $child_2$]=**Divide**(tree[parent_index],cut_dir);
13: /* Decrease the width, $\gamma_k \in (0, 1)$ */
14: $child_1$ .wire_width = γ_k * tree[parent_index].wire_width ;
15: $child_2$ .wire_width = γ_k * tree[parent_index].wire_width;
16: spec_met_flag_this_part_level = 0;
17: /*Begin Inner While Loop*/
18: **while** (spec_met_flag_this_part_level ==0) **do**
19: $child_1$.wire_pitch= β_{k_1} * tree[i].wire_pitch; $\beta_{k_1} \in [0, 1)$
20: $child_2$.wire_pitch= β_{k_2} * tree[i].wire_pitch; $\beta_{k_2} \in [0, 1)$
21: [$(A_1, \mathbf{S_1}), (A_2, \mathbf{S_2})$]=**make_pow_grid**($child_1, child_2$);
22: [$spec\_met\_flag_1, spec\_met\_flag_2$]=
solv_grid($(A_1, \mathbf{S_1}), (A_2, \mathbf{S_2})$)
23: **if** ($spec\_met\_flag_1$ AND $spec\_met\_flag_2$) **then**
24: spec_met_flag_this_part_level =1;
25: **end if**
26: **if** (!$spec\_met\_flag_1$) **then**
27: β_{k_1}=$\delta_k * \beta_{k_1}$; $\delta_k \in (0, 1)$ /*Decrease the pitch*/
28: **end if**
29: **if** (!$spec\_met\_flag_2$) **then**
30: β_{k_2}=$\delta_k * \beta_{k_2}$; $\delta_k \in (0, 1)$ /*Decrease the pitch*/
31: **end if**
32: **end while**
33: /*End Inner While Loop*/
34: prev_grid_viol_flag=1;
35: **while** (prev_grid_viol_flag ==1) **do**
36: [$prev\_grid\_viol\_flag$]=**chk_other_parts**($child_1, child_2$);
37: **if** (prev_grid_viol_flag == 0) **then**
38: β_{k_1}=$\delta_k * \beta_{k_1}$; $\delta_k \in (0, 1)$ /*Decrease the pitch*/
39: β_{k_2}=$\delta_k * \beta_{k_2}$; $\delta_k \in (0, 1)$ /*Decrease the pitch*/
40: **end if**
41: **end while**
42: /* Add two child nodes to the tree*/
43: tree[i]=$child_1$;
44: tree[i+1]=$child_2$;
45: i=i+2;
46: cut_dir=!cut_dir;
47: **end while**
48: /*End Outer While Loop*/
49: **post_processing_of_grid_to_align_wires()**;

grid constructed in the partition corresponding to the tree node. In lines 2 to 8 of the pseudo-code, the root node of the tree, which represents the full chip area, is initialized. The outer while loop, extending from lines 10 to 47, performs the recursive bipartitioning and grid construction process. Inside the outer loop, a parent partition is divided into two child partitions in lines 11 to 15. The **Divide** subroutine, shown on line 12, implements the task of dividing the chip area of the parent partition into two smaller sized partitions. The first inner while loop, in lines 18 to 32, performs the task of constructing the local power grid in the two active partitions, subject to reliability constraints. Within this loop, the steps of constructing the macromodel parameters $(A, \mathbf{S})$ for the two active partitions, as explained in Section 2.7.4, are contained in the routine **make_pow_grid**, listed on line 21. The steps of solution of the global system by preconditioned conjugate gradient method, and the solution of the partitions by the hierarchical approach, as described in Section 2.7.4, are performed by the routine **solv_grid**, shown on line 22. In case the local grid, constructed in the two active partitions, does not meet the reliability constraints, the pitches of the partitions are reduced by a factor δ_k, as shown in lines 26 to 31. The second inner while loop, shown in lines 35 to 41, ensures that the grid constructed in the active partition does not render the previously constructed grids in other partitions ineffective, as explained in Section 2.7.4. This is ensured by the routine **chk_other_parts**, shown on line 36, by checking the port voltages of neighboring partitions of the two active partitions. Lines 43 to 46, which are a part of the outer while loop, add the two child nodes to the parent node, after a satisfactory local grid has been designed in the two active partitions. At the end of the design procedure, the post-processing step, as described in Section 2.7.5 is employed to improve the alignment of power grid wires in the adjacent partitions. This task is performed by the subroutine **post_processing_of_grid_to_align_wires**.

In summary, the main features of the power grid design algorithm are:

1. The design procedure is based on a recursive bipartitioning scheme. At each step

a simple bipartitioning problem of figuring out the appropriate wire pitches of the two active partitions is solved.

2. Using the concept of grid refinement, the power grid is iteratively refined. The abstraction of grid constructed in previous levels of partitioning is used to guide the power grid design in the active partitions.

3. Using the circuit analysis step to detect the IR drop and EM violations in the inner loop ensures the accuracy of the design scheme. The speed up of this method over other power grid design schemes, relying on the explicit circuit analysis, is obtained by making the analysis step extremely fast. This is achieved by:

    - Controlling the circuit size by using very thick wires in the initial levels of partitioning. The wire width is successively reduced in each partitioning level.
    - Using the macromodeling technique for abstraction of power grid in different partitions and focusing on local analysis of the two active partitions.
    - Solving the global matrix system by a preconditioned conjugate gradient based iterative solver and using the starting guess vector of the voltages at the port nodes, as determined by the solution of the power grid system designed in the previous iteration.
    - Employing the port approximation technique during the intermediate partition levels.

4. A post-processing step at the end of the design procedure is employed to improve the wire alignment in adjacent partitions.

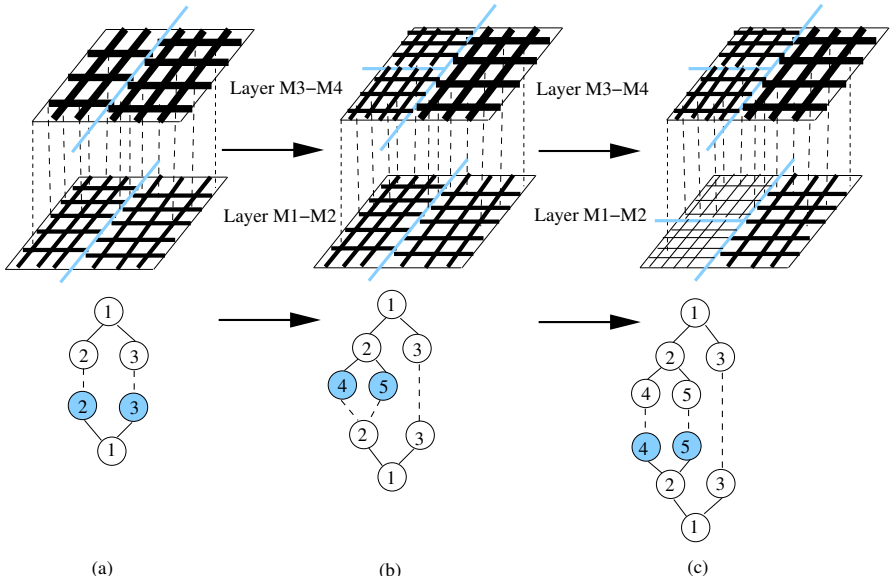


Figure 2.16: Extending the grid design procedure to multiple metal layers M1to M4. The dashed lines between the top two and the bottom two layers represent via connections between layers M2 and M3. (a) First level of partitioning has been completed for the metal layers M3-M4. Grids in the left and right partitions for the bottom two layers are being designed. (b) The top-left and bottom-left partitions for layers M3-M4 being processed in the second level of partitioning for the top two metal layers. (c) The top-left and bottom-left partitions for layers M1-M2 being processed in the second level of partitioning for the bottom two metal layers.

2.7.7 Extension to Multiple Layers

In the previous sections, we have described the proposed power grid design scheme to construct power grids in the top two metal layers. The wires added to the grid, in each iteration, are in the horizontal direction for the top layer and in vertical direction for the metal layer one beneath the top layer.

The same approach can be easily extended to design a power grid spanning multiple layers of metal. Figure 2.16 illustrates the procedure to design a power grid for a chip

having four metal layers, M1 to M4. The vertical dashed lines in the figure represent the via connections between layers M2 and M3. In the case of multiple metal layers, the levels of partitioning are defined with respect to each pair of consecutive metal layers. The recursive bipartitioning moves from layers M3-M4 to M1-M2 and back to M3-M4 for the next level of partitioning. For example, as a first step, the first level of partitioning for the layers M3-M4 would involve constructing the power grid in the left and right partitions for these two layers, with connections to a fixed uniform grid in the bottom layers. As a good starting point, the initial pitch of the uniform grid in M1-M2 may be chosen in such a way that the initial voltage drop is bound to be no more than some constant times the specified drop. Then, as seen in Figure 2.16(a), the power grid is refined in the left and right partitions of layers M1-M2. The grid wires in the layers M1-M2 have connections with the previously design power grid in the left and right sections of layers M3-M4. The step of checking for previous_grid_violated flags for the active partitions, now defined with respect to a pair of metal layers, would now entail evaluation of port voltages of the neighbors of the active partitions in the other pair of layers as well. For instance, while refining the power grid in the top-left and bottom-left partitions of layers M3-M4, as shown in Figure 2.16(b), the port voltages would be checked for the port nodes connecting to the right partition of layers M3-M4, and also for the ports connecting to the left partition of layers M1-M2. As seen in Figure 2.16, the almost complete binary partition trees are defined with respect to each pair of metal layers. The wires in different layers can be of different sizes as typically, the wires in top two metal layers are much wider than the ones in the intermediate metal layers. Thus, the width reduction factor, γ is defined with respect to each pair of layers. $\gamma_{k_{l1-l2}}$ represents the width reduction factor for the k^{th} partitioning level defined with respect to the pair of metal layers $l1 - l2$. Other steps in the proposed design procedure remain the same while designing a multi-layered power grid.

2.7.8 Experimental Results

The proposed power grid design scheme was implemented in C using a sparse matrix library [mes] for both computing the macromodels and solution to the global system by preconditioned conjugate gradient method, and design of several power networks were tested. Separate sparse matrices are used for each partition grids and for the nodes connecting the two partitions, so that when the grid in a previous partitioning level is ripped up the entire conductance matrix need not be recomputed. The input to our power grid design procedure is a floorplan with functional block current estimates and the locations and number of the power pads on the chip. The output is a non-uniform power grid that meets the IR drop, EM and minimum pitch constraints. We could find only two real benchmark floorplans [SSH$^+$03], [LHL04] for a microprocessor chip in which the functional block currents could be determined. These are the floorplans of ALPHA 21364 microprocessor chip. The block currents of the functional blocks in these floorplans were estimated from the given power consumption estimates of each functional block, in 130nm technology, using a V_{DD} of 1.2V. The functional block current I_{f_k}, of a block k is computed as $I_{f_k} = \mathrm{Power}_k / V_{DD}$, where Power_k is the total power consumption of block k. Due to the paucity of real full chip level benchmark floorplans, with functional block current estimates, we randomly generated floorplans and assigned realistic block currents to various functional blocks in the floorplans. The block currents were assigned by assuming the total power consumption of the chips to be between 40 to 80 Watts and distributing the total power consumed randomly between the various functional blocks. For each of our experiments, we assume an availability of 400 to 600 power pads, distributed either throughout the chip, as in the case of a flip-chip package, or 200 to 300 pads located on the chip periphery, as in the case of a wire-bond package. The power pads are assumed to be connected to the top metal layer which has wires running only in the horizontal direction.

The circuit parameter values, sheet resistivity (ρ_s), current density (σ) and mini-

mum wire pitches (p_{min}), were taken from [SIA01] and [Con00] for power delivery to a 2cm$\times$2cm chip in 130nm technology with V_{DD} = 1.2V. The voltage constraints for the power grids, i.e., V_{spec} was 1.08V, i.e., 90% of V_{DD}. The via specific resistances are chosen to be $0.001\Omega\mu^2$. The experiments were performed on P-4 processor, Linux machines with a speed of 2.4 GHz and 2GB RAM.

At the beginning of our optimization procedure, in the first level of partitioning, $Part_1$, the initial wire widths and wire pitches for the two partitions are chosen such that the worst case voltage drop is about twice the specified drop, e.g., 20%-25% of V_{DD}. This choice of starting point for the initial wire widths and the pitches selection has been empirically determined so that we begin not too far away from the final solution point in the search space. Choosing an initial pitch and width assignment corresponding to a much worse initial voltage drop, e.g., 50% of V_{DD}, would mean that the design procedure has to spend much more time to reach the feasibility region and find a point that meets the reliability constraints. On the other hand, choosing a starting point which is very near to the feasibility region, e.g., 10%-12% of V_{DD}, may lead to over utilization of wiring resources as the design heuristic may not have enough iterations to explore the search space, before it finds a feasible solution that has a wire width and pitch assignment which may be suboptimal in terms of wire area used. We assume equal pitches of the wires in horizontal and vertical directions within a partition, but, clearly this is not a restriction in the proposed scheme.

We construct the power grid by the proposed scheme for a set of eight benchmark floorplans, both for a flip-chip (FC) and a wire-bond (WB) case. Table 2.8 shows the results for these power grid constructions. For each experiment for both the FC and the WB case, corresponding to one row in Table 2.8, the initial power grid in the first partitioning level comprises very thick wires in the range of 60 to 100 μm. In subsequent partitioning levels, the width reduction factor, γ_k is assigned an appropriate empirically tuned value in the interval $[0.65, 1)$ so that at the end of k levels of partitioning, the final value of wire width for the power grid in 2^k partitions is between 2 to 6 μm. Moreover,

it was empirically found that by keeping the value of γ_k in the above interval minimizes the minimum pitch violations. We found that by choosing the value of γ_k in the interval $[0.65, 1)$, only about 5%-8% of iterations exhibit the minimum pitch violations, where the grids designed in the previously processed partitions have to be altered. The design procedure is terminated at the end of $k = 8$ levels of partitioning. The value of MAX_{spec} parameter, to flag the previous_grid_violations, was chosen to be 15mV.

Ckt	# of Blocks	# of Nodes		Wire Area (cm^2)		Runtime (sec)	
		Flip-Chip	Wire-Bond	Flip-Chip	Wire-Bond	Flip-Chip	Wire-Bond
pg-1	17	1557628	1635925	0.0834	0.0876	355	598
pg-2	17	1186124	1216726	0.0792	0.0824	431	704
pg-3	65	1261633	1376425	0.0781	0.0816	565	784
pg-4	80	1051237	1208613	0.0732	0.0776	521	791
pg-5	100	1217203	1343793	0.0756	0.0874	522	781
pg-6	124	1136898	1199516	0.0762	0.0824	625	817
pg-7	140	1648223	1703717	0.0904	0.1084	416	625
pg-8	162	1292815	1364712	0.0892	0.1032	433	618

Table 2.8: Results of power grids designed by the proposed scheme for both flip-chip and wire-bond cases.

The first two rows in Table 2.8 represent the power grid constructed for the two real benchmark floorplans of ALPHA 21364 chip. The other rows correspond to the power grid designed for the randomly generated floorplans. The second column in the table shows the number of blocks in the floorplan. The next two columns indicate the number of electrical nodes in the final optimized power grid circuit. For each circuit there are more than a million electrical nodes in the final circuit. The optimization is terminated when the worst voltage of all nodes in the final power grid circuit is greater than V_{spec}

and all branches meet the current density specifications at the end of $k = 8$ levels of partitioning and the post-processing step to align the wires. The worst voltage measured by performing an accurate simulation, at the end of the design procedure without the port approximation technique, verifies the accuracy of the final solution. The wire area consumed by the final power grid is listed in the fifth and the sixth column, for a flip-chip and a wire-bond case, respectively. The last two columns report the runtime for constructing the power grid by the proposed design procedure. The runtimes of the proposed power grid design procedure are in the range of about 6 to 11 minutes for the grids designed for the flip-chip case and about 10 to 14 minutes for the wire-bond case. The order of the runtimes obtained underscores the efficiency of the design algorithm, considering the fact that for each of the test cases in Table 2.8, the final power grid for both the flip-chip and the wire-bond case, spanning the entire chip area in two layers of metal, comprises more than a million electrical nodes.

As seen in Table 2.8, the proposed scheme performs better for the flip-chip case than the wire-bond case, both in terms of utilizing lower wire area and in its faster runtimes. This can be ascribed to the fact that the notion of locality in power grid design, which is one of the motivating factors of the proposed algorithm, is more pronounced in the case of a flip-chip package, where there are sufficient number of pads near the violating regions. For a wire-bond chip, the fact that the pads around the chip periphery are located far way from the violating regions that may be located at the center of the chip, could make local grid correction step for fixing the violations, a suboptimal choice. Hence, the procedure has to expend a larger amount of computational time and wiring resources to meet the reliability constraints for a wire-bond chip.

In the next set of experiments, we compare our bipartitioning-based power grid design scheme with the sensitivity based grid design heuristic presented in Section 2.6. We construct power grids for four randomly generated benchmark floorplans, using the two grid design procedures. Table 2.9 lists the results of this comparison between the two methods. The columns under *Prop Method 1* label, refer to the grid design procedure

Ckt	Wire Area (cm^2)		Runtime (sec)		% Saving in Wire Area for Method 1	Slow down of Method 1
	Proposed Method 1	Proposed Method 2	Proposed Method 1	Proposed Method 2		
Ckt-A	0.0745	0.07865	3516	604	5.6%	5.8×
Ckt-B	0.0842	0.0874	2898	632	3.8%	4.6×
Ckt-C	0.0912	0.0975	3219	592	6.9%	5.4×
Ckt-D	0.0796	0.0816	3478	684	2.5%	5.2×

Table 2.9: A wire area and runtime comparison of the two proposed power grid design methods.

proposed in Section 2.6. Similarly, the results for the second grid design procedure, presented in Section 2.7, are listed under the columns labeled *Prop Method 2*. For both the procedures, we assume a flip-chip scenario, with about 400-600 V_{DD} pads distributed throughout the chip area. For the first grid design method, we tessellate the chip area into 100 tiles, and after the port approximation step, keep 40 ports per tile. For the bipartitioning-based grid design algorithm, we use $k = 8$ levels of partitioning. As seen from the data in the table, the first method has about 3% to 6% better wire area utilization than the second method. However, the second power grid design procedure has a significant advantage over the first one, in terms of runtime. It is, on an average, about $5\times$ faster than the first technique. Therefore, we conclude that by paying a small wire area penalty, we can use the bipartitioning-based power grid design scheme, as a much more efficient alternative to the sensitivity-based grid design method of Section 2.6.

In another set of experiments, we compare the proposed power grid design algorithm with a previous grid design scheme [WMS05]. We implemented a simple version of the multigrid-based power grid optimization scheme of [WMS05] in C++ to compare the results of our proposed power grid design algorithm with this method. Table 2.10

Ckt	# of Wires Multigrid Scheme	Runtime (sec)		Wire Area (cm^2)		% Saving in Wire Area	% Saving in Run Time
		Prop Method	Multigrid Scheme	Prop Method	Multigrid Scheme		
Ckt-1	1000 × 1000	572	595	0.0733	0.0768	4.5%	3.8%
Ckt-2	1100 × 1100	584	663	0.0758	0.0810	6.4%	11.9%
Ckt-3	1150 × 1150	585	690	0.0773	0.0852	9.2%	15.2%
Ckt-4	1200 × 1200	591	713	0.0781	0.0854	8.5%	17.1%
Ckt-5	1250 × 1250	627	741	0.0808	0.0870	7.1%	15.4%
Ckt-6	1300 × 1300	692	781	0.0854	0.0946	9.7%	11.4%

Table 2.10: Results of power grids designed for flip-chip circuits by the proposed method and the multigrid-based scheme.

shows a comparison of the performance of the proposed power grid design algorithm with the multigrid-based technique of [WMS05]. The two schemes are used to design power grids for six randomly generated floorplans for a flip-chip case. The floorplans comprise 60 to 100 functional blocks with currents assigned randomly to each block so that the total power consumption of the chip is between 40 to 80 Watts. Some functional blocks are assigned about 3 to 4 times more power than the other blocks so that there are distinct high and low current density regions on the chip. The assignment of block currents, to model the high and low current density regions, follows from the observation that most full-chip microprocessor floorplans have about 30%-50% of chip area dedicated to caches which consume much less power than the other functional blocks, e.g., arithmetic and logic units [LHL04]. In our implementation of the multigrid-based method, only the wire widths are optimized by setting the weights corresponding to the decoupling capacitor cost and the congestion term to zero in the objective function of [WMS05]. A uniform power grid is constructed for the six cases using the multigrid-

based design scheme, with an initial constant wire width of 1.5μ for all wires. Next, by enumeration, the numbers of wires in the uniform grid topology is determined so that the initial voltage drop is about two times the specified drop. This initial starting point, in terms of the initial voltage drop, is the same as chosen for selecting the initial pitches in $Part_1$ of our proposed design procedure. The second column in the table 2.10 lists the number of horizontal and vertical wires used for the uniform grid construction. Following a series of network reductions, about 8 to 14 levels of reduction for each circuit, the top level power grid is reduced to a much smaller grid so that the problem size is sufficiently small. The wire sizing solution for reduced network is then obtained by solving a constrained nonlinear optimization problem by using a sequential quadratic programming software [LZT]. The back-mapping to the original network is performed by solving a series of linear programs as formulated in [WMS05].

The fourth and the fifth columns in the table show a comparison of the runtimes of the two schemes. For all of the six circuit examples, the total time taken by the multigrid-based scheme to perform the network reduction, solve the nonlinear optimization problem and solve a series of linear programs for back-mapping is greater than the proposed power grid design algorithm. On an average for the six example cases the proposed method is about 12% faster than the multigrid-based design algorithm. Columns six and seven show the wire area utilized for each of the example circuits by the two design methodologies. The wire area utilized by the proposed heuristic is about 5%-10% less than the grid design method of [WMS05]. In the multigrid-based design method, each column (row) of vertical (horizontal) wire is constrained to have the same wire width in order to reduce the number of design variables for efficiently solving the resulting nonlinear program. Since the width of all of the wire segments on a column (row) of the wire is determined by the highest current density blocks, the power grid has to be over-designed in the low current density regions of the chip. The proposed design algorithm is run for $k = 10$ levels of partitioning. By designing local power grids in each partition, it is ensured that the wiring resources are utilized in a judicious manner as per

the current density requirements.

To measure the quality of the solution given by the proposed scheme, in terms of the wire area utilized, we also implement an exact wire sizing scheme and compare our results with the exact scheme. The power grids by the exact scheme are constructed by formulating the IR drop and EM constraints as nonlinear constraints in terms of the wire widths as the optimization variable, and minimizing the wire area objective function. A sequential quadratic programming software [LZT] is used to solve the nonlinear problem to determine the wire sizes of the variables. The set up for implementing the exact sizing scheme is very similar to that of implementing the multigrid technique. The only difference is that wire width corresponding to each branch resistance is now treated as a separate variable as opposed to having one variable for the entire column (row) of a vertical (horizontal) wire, as done in the multigrid method. Since it is very inefficient to solve a nonlinear optimization problem for large power grid systems, we perform the comparison for grids of small sizes constructed for a toy chip of dimensions $300\mu \times 300\mu$. For our proposed approach, we use $k = 9$ levels of partitioning with an initial wire thickness of 10μ. We perform the comparison between the two schemes both for a flip-chip and a wire-bond chip.

Ckt	# of Wires Exact Scheme	Wire Area		Wire Area		Runtime (sec)		Runtime (sec)	
		Proposed Method	Exact Scheme	Proposed Method	Exact Scheme	Proposed Method	Exact Scheme	Proposed Method	Exact Scheme
		Flip-Chip		Wire-Bond		Flip-Chip		Wire-Bond	
Ckt-1	20×20	1.033	1.000	1.052	1.000	2	487	3	483
Ckt-2	30×30	1.041	1.000	1.061	1.000	4	622	6	630
Ckt-3	40×40	1.025	1.000	1.046	1.000	6	815	7	811
Ckt-4	50×50	1.044	1.000	1.072	1.000	9	1246	9	1232

Table 2.11: Results of power grids designed for flip-chip circuits by the proposed method and the exact wire sizing scheme.

Table 2.11 shows the results of the comparison. Columns three and four of the table represent the wire area utilized by the proposed and the exact method, normalized with

respect to the the exact sizing scheme. As seen in the table, for the four example circuits, our proposed method has an over utilization cost of about 2%-4% for the flip-chip case and about 4%-7% for the wire-bond case. The higher overhead of the wire-bond case is expected due to the lack of strong locality. However, as seen from the run time numbers in columns 8 and 10, the nonlinear programming solution for the exact sizing method becomes very inefficient and impractical to use as the number of optimization variables increase from a few hundreds in Ckt-1 to a few thousands in Ckt-4. Even though the wire area utilized by the exact method would be the most judicious, the high runtime prohibits the use of such a scheme to construct real power grids. As seen in the table, our proposed approach is extremely efficient and has a small wire area over-use cost as compared to the exact method.

In another set of experiments we study the effect of choosing different levels of partitioning for the power grids designed for the same input floorplan. Table 2.12 represents

SNo.	# of Partition Levels	Wire Area (cm^2)	Runtime (sec)
1	7	0.0812	560
2	8	0.0762	625
3	9	0.0756	688
4	10	0.0752	784

Table 2.12: Power grids designed for pg-6 floorplan by choosing different partitioning levels.

these experiments for constructing power grids for the pg-6 floorplan and assuming the flip-chip case. As seen in the table, the algorithm runs faster for smaller number of partitioning levels. This can be ascribed to the fact that by increasing the height of partition_tree of Figure 2.11(e), the number of partitions and consequently the size of the global system of Equation (2.7), even with the port approximation technique increases,

which results in greater simulation time in each iteration. The cost of using fewer partitioning levels is over-utilization of wiring resources. The number of wires required in the partition is determined by the the maximum of all current demands over the entire partition area. Since the partition size is larger for a smaller value of k, more wiring resources are wasted in the region within the large partitions where the current requirements are less than the maximum. By splitting the partitions into smaller sizes, the regions for different current demands can be isolated, and power grids with better area utilization can be constructed separately in these regions. However, beyond a point, the runtime penalty for increasing the granularity of partitions or the number of partitioning level, outweighs the savings in the wire area.

2.8 Conclusion

In this chapter of the thesis, we have presented two power grid design schemes, as techniques to address the issue of controllable type of environmental variations, in the form of voltage fluctuations on supply network wires. Our power grid design schemes constructs locally regular, globally irregular grids. Such a piecewise-uniform grid topology shows a significant reduction in the wire area used when compared to the area consumed by the uniform grid topologies. This design also aids in an easier routing scheme for the signal nets later in the design, as minimal book-keeping needs to be done for the proposed P/G architecture. Moreover, such a structured power grid design is easy to optimize. For our techniques, we use the hierarchical analysis method for the simulation of the power grid system. Including a simulator in the optimization loop ensures the accuracy of the optimized solutions.

For the first method, we have proposed a sensitivity-based heuristic to add wires in specific regions of the chip to meet the IR-drop and EM constraints. Including a congestion penalty term in the cost function helps in controlling the aggravation in congestion without compromising the local structured regularity of the supply grid. Experimental

results show that the grids designed using our proposed scheme save 12% to 23% of wiring area over other commonly used grid topologies. However, the runtime of our procedure is not very desirable and needs to be improved.

To overcome some of the inefficiency of our first power grid design method, we have presented a second considerably fast supply network design algorithm. The method is based on an iterative grid refinement scheme by recursive bipartitioning of the chip area. The concept of locality in power grid design is used to abstract away the details of some parts of power grid by the macromodeling technique. Using the grid abstractions, along with the strategy of constructing an initial coarse grid followed by a successive refinement of the grid, and reusing the solution of grid designed in the previous iteration as a starting guess point for the conjugate gradient linear solver, speeds up the circuit analysis step significantly. Experimental results on real and randomly generated realistic test cases show that the proposed power grid design algorithm is considerably fast and has efficient utilization of the wiring resources. Our proposed method is able to design power networks comprising thousands of wires, and more than a million nodes, in about 6 to 13 minutes of runtime. When compared to a multigrid-based power grid design scheme, it is found to save about 7% to 12% of wire area, and on an average is 14% faster. However, as compared to the first power grid design algorithm, the second grid design procedure spends about 3% to 7% more wire area, for the same benchmark floorplans, but on an average, is about $5\times$ faster than the first method.

Chapter 3

Robust Gate Sizing Techniques

In Chapter 2, we presented two power grid design schemes as techniques for reducing controllable variations in the supply levels. In this chapter, we address the problem of accounting for *uncontrollable* variations, arising mainly from the limitations of the manufacturing process. From a circuit design point of view, these process-driven uncertainties cannot be directly controlled by any specific circuit design techniques. However, the effects of these variations on the circuit performance can still be controlled. A designer can reduce the impact of these uncertainties by accounting for them, e.g., by maintaining sufficient margins in the design so that when these variations manifest themselves, the design margins ensure that the desired performance criteria are still met.

This method of designing the circuit robustly, by adding design margins, to safeguard against process variations and other uncertainties, can be regarded as a worst-case design methodology. Such schemes, typically consist of identifying a worst-case scenario in which the parameter uncertainties would be manifested to have the greatest impact on the circuit performance. After identifying such a scenario, the circuit is designed to meet the performance specifications for this worst-case occurrence of parameter variations. By keeping sufficient design margins, the effect of random process parameter variations are accounted for, thus ensuring robust circuit design. A critical step in this design paradigm is the identification of the worst-case situation. The use of an ad hoc method for this purpose could lead to excessively large design margins, and an overly pessimistic design. Moreover, identifying a possible but a highly improbable worst-case scenario would lead to extra guard-banding against the effect of variations, resulting in the design incurring excessive penalties and overheads.

In this chapter of the thesis, we propose a novel worst-casing methodology, by way of a robust gate sizing scheme, to design a circuit for the required timing yield. Our

method uses the statistical properties of the parameters of variations, such as the probability distributions and spatial correlations, to reduce the pessimism associated with conventional worst-case design schemes. Through our method, we provide flexibility to the user to specify a target timing yield, and perform gate sizing to achieve this specification. An early version of this work was published in [SNLS05].

3.1 Introduction to Robust Gate Sizing

The limitations of the manufacturing process in the current technologies leads to random variations in various circuit parameters such as the transistor width, channel length, and oxide thickness, which may cause a large spread in the circuit performance measures such as the delay and power. Since it is impossible to control process-driven variations, it is essential for the design tools to account for these uncertainties to enable the design of robust circuits that are as insensitive to the device parameter variations as possible.

The optimization of gate sizes offers a degree of flexibility in addressing this issue. The gate sizing problem determines an optimal set of transistor sizes, defined as the ratio of the transistor width (w) to the effective channel length (L_e), that minimize the area or power consumption of a combinational circuit, subject to meeting the specified delay constraints. Conventional gate sizing tools employ a static timing analysis (STA) routine to generate the delay constraints by adding intermediate variables at the output of each gate in the circuit, and then solve the resulting optimization problem to determine the widths of the devices in the circuit. The minimum length is chosen for all the devices.

However, due to the fact that the nominal designs are perturbed by the random process variations, a large number of chips may fail to meet the original delay specifications. This leads to a reduction in the timing yield of the circuit, defined as the fraction of total chips whose delay does not exceed the original specified value. An obvious way to increase the timing yield of the circuit is to design for the worst-case scenario, e.g., choose a delay specification of the circuit much tighter than the required delay. Unless this new

specification is appropriately selected, this could lead to large overheads in terms of the circuit area and the power, as the optimizer may have to aggressively size the critical as well as the non-critical paths. Hence, it is necessary to develop smart worst-casing methodologies in the presence of process uncertainties, that keep the area and the power budgets within reasonable bounds.

In this work, we present a novel worst-casing scheme, based on robust optimization theory. In our method, we modify the delay constraints to incorporate uncertainty in the parameters due to the process variations. An *uncertainty ellipsoid* method is used to model the random parameter variations, assuming normal distribution of parameters. Spatial correlations of intra-die parameter variations are incorporated in the optimization procedure. The resulting optimization problem is relaxed to be a geometric program (GP), and is efficiently solved using convex optimization tools. By using the well-known *Chi-square* probability distribution function, the desired timing yield can be parameterized into the optimization formulation. Our formulation is based on the principle of adding uncertainty related, parameter correlation-aware, margins to delay constraints at the output pin of each logic gate. However, by using these guard-bands for the delay constraints at the output of each node in the circuit graph[1], instead of the whole path delay, leads to a problem of overestimation of the effect of variations. We reduce this problem by employing a graph pruning technique to reduce the number of intermediate nodes in the circuit graph, and the corresponding arrival time variables in the optimization formulation. The use of variable size uncertainty ellipsoid at different topological levels of the circuit graph helps in further removing the extra timing margins in the constraints.

The organization of this chapter is as follows. We review the previous work on uncertainty-aware gate sizing in Section 3.2. Section 3.3 covers the preliminaries of ge-

[1]The graph obtained by modeling each pin of a gate as a vertex, and each pin-to-pin connection, in the whole circuit, as an edge is referred to as the circuit graph or the timing graph.

ometric programming, the traditional gate sizing formulation, the ellipsoid set and the Chi-square probability distribution. In Section 3.4, we present our formulation of the robust sizing problem, and use a simple example to explain the details of this formulations. Section 3.4.3 points out the problem of overestimation of the effect of variations in our robust formulation. The graph pruning technique and the use of variable amounts of timing margins at different topological levels of the circuits, as methods to reduce this pessimism in the robust formulation, are described in Section 3.4.4 and 3.4.5. Experimental results are presented in Section 3.5, and Section 3.6 concludes this chapter.

3.2 Previous Work

Traditional gate sizing methodologies [FD85], [SRVK93] solve the deterministic optimization problem of gate sizing without accounting for variations in parameters. These methods use posynomial delay constraints and formulate the problem as a geometric program. Section 3.3.2 reviews the formulation used in these conventional gate sizing works. While the method of [FD85] performs sizing based on a sensitivity-based heuristic, [SRVK93] offers an exact optimization algorithm to perform gate sizing, based on convex programming techniques. There have been several recent attempts to perform uncertainty-aware gate sizing to reduce the timing violations or increase the timing yield. In [BVSH02], the gate sizing problem is formulated as a nonlinear optimization problem with a penalty function added to improve the distribution of timing slacks. One of the first works on statistical gate sizing [JB00] proposes formulation of statistical objective and timing constraints, and solves the resulting nonlinear optimization formulation. In other works on robust gate sizing [CPR04, SSZ05, ACBZ05, CSS$^+$05], the central idea is to capture the delay distributions by performing a statistical static timing analysis (SSTA), as opposed to the traditional STA, and then use either a general nonlinear programming technique or a statistical sensitivity-based heuristic procedures to size the gates. In [RVW04], the mean and variances of the node delays in the circuit graph

are minimized in the selected paths, subject to constraints on delay and area penalty.

Some of the above mentioned variation-aware gate sizing works are heuristics [SSZ05, ACBZ05, CSS+05] without provable optimality properties. The sensitivity-based approaches optimize the statistical cost function in a local neighborhood, and cannot guarantee convergence to the globally optimal solution. Others rely on nonlinear nonconvex optimization techniques [RVW04, JB00, CPR04], which are either not scalable to practical circuits or may get stuck in locally optimal solutions. Some of these works [JB00, CPR04] ignore important statistical properties of varying parameters such as the spatial correlations.

In [MDO05], the authors present an interesting approach to optimize the statistical power of the circuit, subject to timing yield constraints under convex formulation of the problem as a second-order conic program. However, the formulation suffers from the same problem of overestimation of statistical nodal delay constraints as [SNLS05], which will be explained in Section 3.4.3, and we partially correct this by the techniques described in Section 3.4.4 and 3.4.5. More importantly, the solution in [MDO05] relies on a local search over the gate configuration space to identify a size that will absorb the slack assigned by the optimization solution. Such a method based on local searches has to assume that the delay of the gate depends only on the fixed local choices, e.g., a particular size and the fanout load of a gate. In reality, the gate delay is also a function of the slope of the signals at the input pins of the gate, which in turn are functions of the sizes of the fanin gates and the interconnect delay. Hence, although local search method of [MDO05] works well for simple delay models as functions of output load only, it is unlikely to work for a realistic delay model also considering input slews.

Recently a novel method for optimizing the binning yield of a chip was proposed in [DS06]. This method provides a binning yield loss function that has a linear penalty for delay of the circuit exceeding the target delay, and proves the convexity of this formulation. However, the method has to rely on an SSTA engine to evaluate the gradient of the binning yield loss function for optimization purposes. This could potentially make

the overall procedure considerably slow for many iterations of the optimization loop. As the objective function in the optimization formulation in this work is non-differentiable, the procedure could also run into some serious numerical problems while generating the subgradients of the objective function.

In this work, we propose a novel gate sizing technique based on robust optimization theory [BV04]. For simplicity, our implementation uses the Elmore delay based model, but our approach is applicable to any posynomial delay model, such as the rich class of generalized posynomial delay models proposed in [KKS98]. In our method, we first generate posynomial constraints by performing an STA. We then add *robust constraints* to the original constraints set by modeling the intra-chip random process parameter variations as Gaussian variables, contained in a constant probability density *uncertainty ellipsoid* [JW02], centered at the nominal values. The method of [XHL+05] also uses the ellipsoid uncertainty model, but for optimization of small size analog circuit. We use the well known Chi-square distribution tables to assign a timing yield value in our optimization constraints. Under the ellipsoid uncertainty model, the resulting optimization formulation is relaxed to be a GP, and is efficiently solved using the convex optimization tools. Furthermore, using a GP to perform robust gate sizing ensures that the optimizer finds a global minimum, which is not guaranteed in the case of a general nonlinear program. The relaxation of the robust counterpart of the conventional deterministic GP-based gate sizing solution as another GP is a major contribution of this work; in general, it is not true that the robust versions of convex programs are also convex programs [BV04].

Our robust gate sizing scheme is a type of worst-case design method, but by incorporating spatial correlations in the design procedure, we reduce some pessimism in the design. Spatial intra-die correlations between the parameter variations are incorporated in the optimization scheme by using a grid-based spatial correlation model used in [CS03] and [ABZ03]. In addition, we show that the nodal constraints formulation adds pessimism, and reduce some of this pessimism by employing the graph pruning

technique of [VC99]. Heuristic methods for assigning smaller timing margins at lower topological levels of the circuit graph, and increasing the guard-banding at higher levels, by employing different sized uncertainty ellipsoids, also helps in reducing the effects of this pessimism.

We focus on the intra-die variations in L_e and w parameters; however, the method can be easily modified to include inter-die variations. Process-driven variations in the interconnect widths and thickness can also be included in our method. The following sections in this chapter, describe in details the various steps of our robust gate sizing method.

3.3 Preliminaries

In this section, we will review some of the basic tools and formulations that we build on to obtain our robust optimization formulation.

3.3.1 Geometric Programming

A function is called a *monomial* function if it can be written in the form:

$$\begin{aligned} f(\mathbf{x}) &= c x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n} \\ &= c \prod_{i=1}^{n} x_i^{a_i} \end{aligned} \tag{3.1}$$

where $\mathbf{x} \in \mathbf{R}^{\mathbf{n}}_{++}$, $c > 0$ and $a_i \in \mathbf{R}$. The variables in a monomial function, and the coefficient c are strictly positive, and the exponents a_i can be any real numbers.

A sum of monomials is called a *posynomial* function. It can be written as:

$$f(x) = \sum_{j=1}^{k} c_j \prod_{i=1}^{n} x_i^{a_{ij}} \tag{3.2}$$

where $c_k > 0$. Posynomials are closed under addition, multiplication, and nonnegative scaling. Monomials are closed under multiplication and division.

From Equations (3.1) and (3.2), a geometric program can be defined as an optimization problem of the form:

$$\begin{aligned} \text{Minimize} \quad & f_0(x) \\ \text{Subject to} \quad & f_i(x) \leq 1, \quad i = 1, \cdots, m \\ & h_i(x) = 1, \quad i = 1, \cdots, p \end{aligned} \tag{3.3}$$

where $f_0, \cdots, f_m$ are posynomial function as in Equation (3.2), and $h_1, \cdots, h_m$ are monomial functions as in Equation (3.1).

Geometric programs are not, in general, convex optimization problems. However, by a simple transformation of variables, $x_i = e^{y_i}$ in the objective and the constraint functions of Equation (3.3), they can be converted to a convex program [BV04], and hence can be efficiently and globally solved using the convex optimization methods. A generalized geometric program (GGP) [KKS98], is an extension of the GP of Equation (3.3), and can also be similarly transformed to a convex program.

3.3.2 Deterministic Gate Sizing as a Geometric Program

The conventional deterministic gate sizing problem is formulated as:

$$\begin{aligned} \text{Minimize} \quad & Area = \sum_{i=1}^{n} a_i x_{i_0} \\ \text{Subject to:} \quad & \\ & \begin{cases} t_i \leq T_{spec} \quad \forall i \in PO \\ t_j + d_{ji}(\mathbf{X_0}) \leq t_i \quad \forall j \in fanin(i) \\ \vdots \\ x_{min} \leq x_{i_0} \leq x_{max} \quad \forall gate \ \ i \end{cases} \end{aligned} \tag{3.4}$$

where x_{i_0} represents the nominal size of the gate, a_i is some weighting factor such as the number of transistors in a gate cell, t_j are the intermediate input arrival time variables at the fanin of gate i, d_{ji} is the delay of gate i, from the j^{th} input pin to the output pin, as

a function of the vector $\mathbf{X_0}$ of the nominal gate sizes, T_{spec} is the specified target delay, x_{min} and x_{max} are the lower and upper bounds on the gate sizes, respectively.

Using the Elmore delay model[2], each gate i in the circuit can be replaced by an equivalent $R_{on_i}C_i$ element, where R_{on_i} represents the effective on resistance of the pull-up or the pull-down network, and the term C_i subsumes the source, drain and gate capacitances of the transistors in the gate. The expressions for R_{on_i} and C_i for a gate i are given by:

$$R_{on_i} = \frac{c_1 L_{e_i}}{w_i}, \qquad C_i = c_2 L_{e_i} w_i + c_3 \tag{3.5}$$

where, the constants c_1, c_2 and c_3 can be derived from [SRVK93]. Both the capacitances and the on resistance of the transistors in a gate are posynomial functions of the gate size, characterized by the widths w of the transistors in the gate. Consequently, the term $R_{on_i}C_i$ which is the equivalent delay contribution of gate i in the circuit is also a posynomial function of w.

From Equations (3.4) and (3.5), the delay constraints at each node of the circuit graph can be written as:

$$\begin{aligned} t_i &\leq T_{spec} \quad \forall i \in PO \\ t_j + \sum_l K_l \prod_k x_{k_0}^{a_{kl}} &\leq t_i \quad \forall j \in fanin(i) \end{aligned} \tag{3.6}$$

where, K_l is a constant coefficient of the l^{th} monomial term in the posynomial delay expression, and can be derived from (3.5), x_k represents the width of gate k , and a_k is the exponents of the k^{th} components of the $\mathbf{X_0}$ vector, $\in \{-1, 0, 1\}$. By substituting Equation (3.6) in Equation (3.4) for all gates in the circuit, the conventional transistor sizing is formulated as a GP optimization problem of Equation (3.3), having a posynomial

[2]Traditional gate sizing methods of [FD85] and [SRVK93] also use the Elmore delay. In any GP based formulation, the Elmore delay model is used for simplicity. Alternatively, generalized posynomial delay models [KKS98], which have a higher accuracy, can be used for the GP formulation.

objective function and posynomial constraints, which can be solved using the convex optimization techniques. In Section 3.4, we show how the robust version of the standard GP formulation, for the deterministic case, can be converted to another GP.

3.3.3 The Ellipsoidal Uncertainty Set

For any vectors Ω and $\Omega_0 \in R^n$, and a non-singular matrix $P \in R^{n \times n}$, an ellipsoid set U is defined as [JW02]:

$$U = \{\mathbf{\Omega} : (\mathbf{\Omega} - \mathbf{\Omega_0})^T P^{-1} (\mathbf{\Omega} - \mathbf{\Omega_0}) \leq \psi^2\} \tag{3.7}$$

If P is a symmetric and positive definite matrix, an alternative representation of (3.7) is realized by substituting, $P^{-1/2}(\mathbf{\Omega} - \mathbf{\Omega_0}) = \mathbf{u}$ as:

$$U = \{\mathbf{\Omega_0} + P^{1/2}\mathbf{u}| \ \|\mathbf{u}\|_2 \leq \psi\} \tag{3.8}$$

where $\|\mathbf{u}\|_2 = \mathbf{u}^T\mathbf{u}$ is the 2-norm of vector $\mathbf{u}$. For a symmetric and postive definite matrix P, the matrix $P^{1/2}$ can be computed by the Cholesky factors of P. The ellipsoid represents a n-dimensional region, where the vector Ω varies around the center point Ω_0. The vector $\mathbf{u}$ characterizes the movement of Ω around Ω_0.

Figure 3.1 illustrates the ellipsoid in R^2. The half-lengths of the axis of the ellipsoid are a factor ψ of the square roots of the eigenvalues, λ_1 and λ_2, of the matrix P, and the direction of the axis is given by the eigenvectors of P, e_1 and e_2.

Considering the vector $\mathbf{\Omega}$ to consist of random variables corresponding to the parameters of variations, with an associated covariance matrix given by P, and assuming that the parameters of variation follow a Gaussian distribution, the ellipsoid set described in Equations (3.7) and (3.8), can be used as a bounded model of variations. In particular, it can be shown that the constant probability density contours of a multivariate normal distribution represent an ellipsoid set. The joint probability distribution function (PDF) of the multivariate normal random vector $\mathbf{\Omega}$, with a covariance matrix P is:

$$f_\Omega(\mathbf{\Omega}) = \frac{1}{(2\pi)^{n/2}|P|^{1/2}} e^{\left\{-\frac{1}{2}(\mathbf{\Omega}-\mathbf{\Omega_0})^T P^{-1}(\mathbf{\Omega}-\mathbf{\Omega_0})\right\}} \tag{3.9}$$

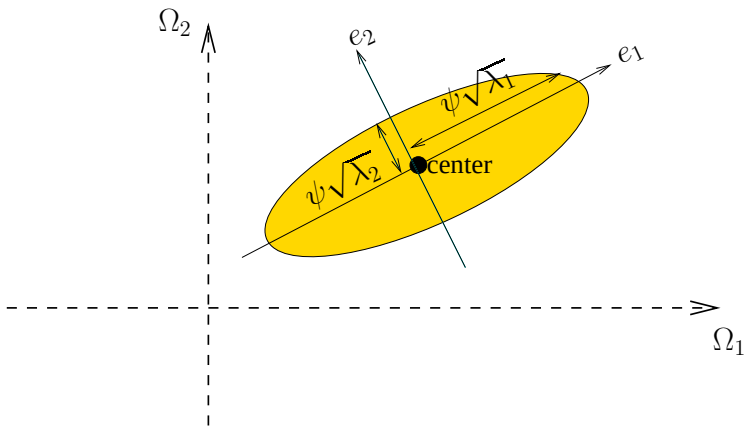


Figure 3.1: An uncertainty ellipsoid set in two dimensions. The ellipsoid set is used as a bounded model for multivariate normal parameter variations.

where $|P|$ is the determinant of the covariance matrix P, and n is the number of components in the variation vector $\mathbf{\Omega}$. It is clear from Equation (3.9), that the PDF of a multivariate normal distribution would be a constant c, if $(\mathbf{\Omega} - \mathbf{\Omega_0})^T P^{-1} (\mathbf{\Omega} - \mathbf{\Omega_0}) = c$. This relation represents precisely the surface of an ellipsoid given by Equation (3.7), with $c = \psi^2$. Since the covariance matrix P is symmetric and positive definite [JW02], we can also equivalently represent the constant probability ellipsoid as Equation (3.8). Thus from the discussion above, by assuming normality of parameter distribution, the ellipsoid set can be regarded as a high-dimensional region inside which the parameters randomly vary. This bounded model of parameter variations in the form of an ellipsoid set is referred to as an *uncertainty ellipsoid*. In Section 3.4, we use this uncertainty ellipsoid model to simplify our robust constraints and formulate the robust GP optimization problem.

3.3.4 Chi-square Distribution

If r_i are n independent normally distributed random variables with means μ_i and variances σ_i^2, the random variable $z = \sum_{i=1}^{n} (\frac{r_i - \mu_i}{\sigma_i})^2$ is distributed according to the Chi-

square distribution (χ^2_n), with n degrees of freedom [JW02]. The Chi-square distribution is a special case of gamma distribution, and for a random variable z following the Chi-square distribution, the cumulative density function (CDF) of z is given by [DS02]:

$$F(z;n) = \frac{\gamma(n/2, z/2)}{\Gamma(n/2)} \tag{3.10}$$

where Γ is the gamma function, and γ is the incomplete gamma function [DS02].

Referring back to Equation (3.7), it can be proved that the random variable $z = (\mathbf{\Omega} - \mathbf{\Omega_0})^T P^{-1} (\mathbf{\Omega} - \mathbf{\Omega_0})$ is χ^2_n distributed [JW02]. Therefore, the solid ellipsoid given by Equation (3.7) can be assigned a pre-specified amount of probability α as:

$$\alpha = F_{\chi^2_n}(\psi^2) \tag{3.11}$$

where F is the Chi-square CDF function given by Equation (3.10).

As will be explained in Section 3.4, we use the uncertainty ellipsoid to pad the deterministic delay constraints, and with the prespecified probability α given by the lower bound on timing yield specification, we define the size of the ellipsoid. This determines the amount of margin required for each delay constraint.

3.4 Variation-Aware Gate Sizing

3.4.1 Effect of Variations on Constraints

The deterministic posynomial constraints of (3.6) can be represented as:

$$t_j + f_{ji}(\mathbf{X_0}) \;\; \leq \;\; t_i \tag{3.12}$$

where $t_j + f_{ji}(\mathbf{X_0}) = \mathbf{t_j} + \sum_{\mathbf{l}} \mathbf{K_l} \prod_{\mathbf{j}} \mathbf{x_{jo}^{ajl}}$ represents the j^{th} constraint function, $\mathbf{X_0}$ is the vector representing the nominal gate sizes x_{0_i} for all gates. The conventional GP optimization assigns a set of optimal x_0 to the vector $\mathbf{X_0}$, so that each delay constraint is satisfied, i.e., $t_j + f_i(\mathbf{X_0}) \leq t_i$ for all constraints i, and the area objective is minimized.

However, due to the effect of process variations, the posynomial delay models of the gate can no longer be assumed to be deterministic quantities. Thus, the constraint inequalities at each node should be rewritten as:

$$t_j + f_{ij}(\mathbf{X_0}, \mathbf{\Omega}) \;\; \leq \;\; t_i \tag{3.13}$$

where $\mathbf{\Omega}$ is the random vector of perturbations around the nominal values of the parameters. For the cases when the new value of the constraint function $t_j + f_{ji}(\mathbf{X_0}, \mathbf{\Omega}) > t_i$, the effect of the random process variations leads to the original constraints being violated and a possible timing failure for the circuit.

Assuming that the random parameter perturbations around the nominal values are small, the new value of the gate delay model $f_i(\mathbf{X_0}, \mathbf{\Omega})$ can be approximated by a first order Taylor series expansion as:

$$\begin{aligned} f_{ji}(\mathbf{X_0}, \mathbf{\Omega_0} + \delta\mathbf{\Omega}) &= f_{ji}(\mathbf{X_0}, \mathbf{\Omega_o}) + \sum_j \left.\frac{\delta f_{ji}(\mathbf{X_0}, \mathbf{\Omega})}{\delta(\Omega_j)}\right|_{\Omega_{j_0}} (\Omega_j - \Omega_{j_0}) \\ &= f_{ji}(\mathbf{X_0}, \mathbf{\Omega_0}) + \nabla_{\mathbf{\Omega_0}} f_{ji}(\mathbf{X_0}, \mathbf{\Omega})\delta\mathbf{\Omega} \\ &= \sum_l K_l \prod_j x_{j_0}^{a_{jl}} + \nabla_{\mathbf{\Omega_0}}(\sum_l K_l \prod_j x_j^{a_{jl}} \delta\Omega) \end{aligned} \tag{3.14}$$

where $\nabla_{\mathbf{\Omega_0}}$ represents the gradient calculated at the nominal values of the parameters, and $\delta\mathbf{\Omega}$ represents the zero-mean random variation in the parameters such as transistor width, effective channel length and oxide thickness, around the nominal values. Note that the coefficient K_l also depends on the parameters, and therefore should be regarded as a function $K_l(\mathbf{\Omega})$ of the perturbation vector.

In (3.14) the term, $\nabla_{\mathbf{\Omega_0}}(\sum_l K_l \prod_j x_j^{a_j})\delta\mathbf{\Omega}$ is the variational term representing the effect of process variations, added to the nominal term $\sum_l K_l \prod_j x_{j_0}^{a_j}$. To safeguard against the uncertainty of process variations, it is necessary to meet the constraint, $t_j + f_i(\mathbf{X_0}, \mathbf{\Omega}) < t_i$, for the maximum value of the variational term. In other words:

$$t_j + \sum_l K_l \prod_j x_{j_0}^{a_{jl}} + \max_{\forall \delta\Omega \in U} (\nabla_{\mathbf{\Omega_0}}(\sum_l K_l \prod_j x_j^{a_{jl}} \delta\Omega) \;\; \leq \;\; t_i \tag{3.15}$$

Next, we show that by employing the concept of an uncertainty ellipsoid U, the constraint of (3.15) can be transformed to a set of posynomial constraints, so that the robust optimization formulation remains a GP, and can be efficiently solved. Our robust GP formulation is applicable for all cases where the original constraints are in the form of a generalized posynomial [KKS98].

We use the uncertainty ellipsoid to model the process variations that randomly perturb the transistor parameters around the nominal values for which they were designed. As the random vector $\mathbf{\Omega}$ of uncertain parameters varies around the nominal parameter vector Ω_0, the variations are considered to be bounded within the ellipsoid regions defined by (3.8). In other words, the variation $\delta\Omega$ from Ω_0 is given by $\delta\Omega = P^{1/2}\mathbf{u}$ with $\|\mathbf{u}\|_2 \leq \psi$.

Alternatively, we could have chosen the variation $\delta\Omega$ in the parameters to be bounded in an n-dimensional box given by $\Omega_{min} \leq \delta\mathbf{\Omega} \leq \Omega_{max}$. However, using the box as a model for bounded variation, ignores any correlation information between the random components of $\mathbf{\Omega}$, as each component can move independently inside a box, assuming any values between the minimum and maximum range. Thus, optimizing for a maximum variation in such a box region would translates to an overly pessimistic design. Moreover, an n-dimensional box modeling of parameter variations would be accurate only in the highly unlikely case when all parameters are statistically independent with respect to each other, and follow a uniform distribution. Most parameters have been observed to follow a distribution that resembles a Gaussian one. The advantage of using the ellipsoid uncertainty model is that it not only accurately models the region of variation for normally distributed parameters, any correlations between the parameters is directly captured by appropriately constructing the elements of the covariance matrix P. The covariance matrix can be derived from a spatial correlation model such as the ones used in [CS03] and [ABZ03].

In the next section, we show with the aid of a small example, the use of the uncertainty ellipsoid model in converting the constraint of (3.15) to a set of posynomial

constraints, and formulating the robust GP for gate sizing in the presence of process variations.

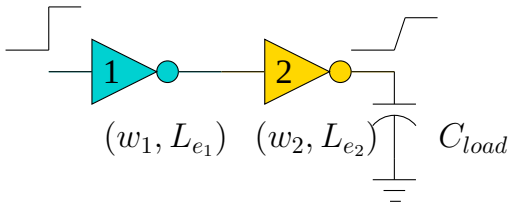


Figure 3.2: A simple example circuit to explain the geometric program formulation for robust gate sizing problem.

3.4.2 Robust GP formulation

We use a simple example to explain the procedure to incorporate the process variation effects in the delay constraints set. We use the toy circuit of Figure 3.2, comprising of just one driver gate and one load gate, for this illustration, but the idea can be generalized to arbitrarily large circuits. In this example, we consider the widths (w_1, w_2) and the effective channel lengths (L_{e_1}, L_{e_2}) of the two gates as the only varying parameters. The scheme can be directly extended to include other parameters.

Applying the Elmore delay model to the gates of circuit of Figure 3.2, and for simplicity, neglecting the interconnect delay and the effect of drain and source capacitances of the driver gate, the delay constraint for the circuit can be written as:

$$\frac{K_1 L_{e_1} L_{e_2} w_2}{w_1} + \frac{K_2 L_{e_2}}{w_2} \leq T_{spec} \tag{3.16}$$

where K_1 and K_2 are constants. As explained in Section 3.4, to ensure that the delay constraint of (3.16) is met under the effect of random process variations, the first order

Taylor series expansion of the constraint function results in the following relation:

$$\max_{\forall \delta w, \delta L_e \in U} \left(\frac{K_1 L_{e_{1_0}} L_{e_{2_0}} \delta w_2}{w_{1_0}} + \frac{K_1 L_{e_{2_0}} w_{2_0} \delta L_{e_1}}{w_{1_0}} + \frac{K_1 L_{e_{1_0}} w_{2_0} \delta L_{e_2}}{w_{1_0}} + \frac{K_2 \delta L_{e_2}}{w_{2_0}} - \frac{K_1 L_{e_{1_0}} L_{e_{2_0}} w_{2_0} \delta w_1}{w_{1_0}^2} - \frac{K_2 L_{e_{2_0}} \delta w_2}{w_{2_0}^2} \right) \leq T_{spec} - \frac{K_1 L_{e_{1_0}} L_{e_{2_0}} w_{2_0}}{w_{1_0}} + \frac{K_2 L_{e_{2_0}}}{w_{2_0}} + \tag{3.17}$$

where w_0 and L_{e_0} represent, respectively, the nominal values of the transistor w and L_e, and δw and δL_e are, respectively, the random variations in w and L_e. Employing the ellipsoid uncertainty model of (3.8) for the random parameter variations, leads to:

$$\begin{bmatrix} \delta w_1 \\ \delta w_2 \\ \delta L_{e_1} \\ \delta L_{e_2} \end{bmatrix} = \begin{bmatrix} (P^{1/2}\mathbf{u})_1 \\ (P^{1/2}\mathbf{u})_2 \\ (P^{1/2}\mathbf{u})_3 \\ (P^{1/2}\mathbf{u})_4 \end{bmatrix} \tag{3.18}$$

where P is the covariance matrix of the random vector $\mathbf{\Omega}$ consisting of the variations in gate w and L_e of the driver and the load gate of Figure 3.2, and $\mathbf{u}$ is the vector bounding the variation within the 4-dimensional ellipsoid centered at the nominal values of w and L_e, with $\|\mathbf{u}\|_2 \leq \psi$.

We introduce two vectors $\phi_\mathbf{1}$ and $\phi_\mathbf{2}$ to collect the positive and negative coefficients, respectively, of the variational parameters of (3.17) as:

$$\phi_\mathbf{1} = \begin{bmatrix} 0 \\ \frac{K_1 L_{e_{1_0}} L_{e_{2_0}}}{w_{1_0}} \\ \frac{K_1 L_{e_{2_0}} w_{2_0}}{w_{1_0}} \\ \frac{K_1 L_{e_{1_0}} w_{2_0}}{w_{1_0}} + \frac{K_2}{w_{2_0}} \end{bmatrix}, \phi_\mathbf{2} = \begin{bmatrix} \frac{-K_1 L_{e_{1_0}} L_{e_{2_0}} w_{2_0}}{w_{1_0}^2} \\ \frac{-K_2 L_{e_{2_0}}}{w_{2_0}^2} \\ 0 \\ 0 \end{bmatrix} \tag{3.19}$$

From the definitions in (3.18) and (3.19), (3.17) can be rewritten as:

$$\frac{K_1 L_{e_{1_0}} L_{e_{2_0}} w_{2_0}}{w_{1_0}} + \frac{K_2 L_{e_{1_0}}}{w_{2_0}} + \max_{\forall \mathbf{u}} \left(\langle P^{1/2}\phi_\mathbf{1}, \mathbf{u} \rangle + \langle P^{1/2}\phi_\mathbf{2}, \mathbf{u} \rangle \right) \leq T_{spec} \tag{3.20}$$

where $\langle a, b\rangle$ represents the inner product of vectors a and b. From the well-known result of the Cauchy Schwartz inequality[3]:

$$< a, b > \quad \leq \quad \|a\|_2 \cdot \|b\|_2 \tag{3.21}$$

and the fact that in the ellipsoid uncertainty model, $\|\mathbf{u}\|_2 \leq \psi$, a sufficient condition[4] for (3.20) is:

$$\frac{K_1 L_{e_{1_0}} L_{e_{2_0}} w_{2_0}}{w_{1_0}} + \frac{K_2 L_{e_{1_0}}}{w_{2_0}} + \psi\|P^{1/2}\phi_{\mathbf{1}}\|_2 + \psi\|P^{1/2}\phi_{\mathbf{2}}\|_2 \quad \leq T_{spec} \tag{3.22}$$

We then introduce two additional *robust variables* r_1 and r_2 as:

$$\begin{aligned} r_1 &= \psi\|P^{1/2}\phi_{\mathbf{1}}\|_2, \quad \text{i.e.,} \quad r_1^2 = \psi^2 {\phi_{\mathbf{1}}}^T P \phi_{\mathbf{1}} \\ r_2 &= \psi\|P^{1/2}\phi_{\mathbf{2}}\|_2, \quad \text{i.e.,} \quad r_2^2 = \psi^2 {\phi_{\mathbf{2}}}^T P \phi_{\mathbf{2}} \end{aligned} \tag{3.23}$$

The inequality of (3.22) is then replaced by the following relaxed constraints:

$$\frac{K_1 L_{e_{1_0}} L_{e_{2_0}} w_{2_0}}{w_{1_0}} + \frac{K_2 L_{e_{1_0}}}{w_{2_0}} + r_1 + r_2 \quad \leq \quad T_{spec} \tag{3.24}$$

$$\psi^2 {\phi_{\mathbf{1}}}^T P \phi_{\mathbf{1}} r_1^{-2} \quad \leq \quad 1 \tag{3.25}$$

$$\psi^2 {\phi_{\mathbf{2}}}^T P \phi_{\mathbf{2}} r_2^{-2} \quad \leq \quad 1 \tag{3.26}$$

As the optimizer tries to minimize the value of the robust variables r_1 and r_2, the relaxed inequality constraints of (3.25) and (3.26) would enforce the equality constraint of Equation (3.23).

The inequality of (3.24) is clearly a posynomial with the robust variables r_1 and r_2 added to the original variable list of the gate w and the intermediate arrival time variables t (not used in this example). By construction, all the elements of $\phi_{\mathbf{1}}$ are posynomials, and

[3] In our case, the equality in (3.21) also holds, as there are some points in the ellipsoid set which have $\langle P^{1/2}\phi_{\mathbf{1}}, \mathbf{u}\rangle = \|P^{1/2}\phi_{\mathbf{1}}\|_2 \cdot \|\mathbf{u}\|_2$.

[4] An equivalent condition for (3.20) is: $\left(\frac{K_1 L_{e_{1_0}} L_{e_{2_0}} w_{2_0}}{w_{1_0}} + \frac{K_2 L_{e_{1_0}}}{w_{2_0}} + \psi\|P^{1/2}(\phi_{\mathbf{1}} + \phi_{\mathbf{2}})\|_2\right) \leq T_{spec}$. However, this does not lead to the formulation of posynomial constraints of (3.25) and (3.26).

all the non-zero elements of $\phi_{\mathbf{2}}$ are negative of posynomials. The covariance matrix P is assumed to have all non-negative elements. This is a justifiable assumption because it is rare to find instances of parameters negatively correlated with each other. The spatial correlation models yield either zero or a positive correlation between the parameters. Thus, the quadratic terms ${\phi_{\mathbf{1}}}^T P \phi_{\mathbf{1}} = \sum_{i,j} P_{ij} \phi_{1_i} \phi_{1_j}$, and ${\phi_{\mathbf{2}}}^T P \phi_{\mathbf{2}} = \sum_{i,j} P_{ij} \phi_{2_i} \phi_{2_j}$ are a summation of monomials with positive coefficients. Consequently, the constraints of (3.25) and (3.26) are also posynomials. Hence, by following the procedure described in the above equations, we convert the non-robust posynomial constraint of (3.16) to a set of robust posynomial constraints of (3.24-3.26), by introducing two additional variables.

Next, we address the issue of assigning a timing yield parameter to the optimization formulation. As discussed in Section 3.3.4, we can assign a pre-specified probability α to the uncertainty ellipsoid model of variations by using the χ^2_n distribution. From Equation (3.11), we can determine ψ^2 as the upper $100\alpha^{th}$ percentile of the χ^2_n distribution from the standard tables of the Chi-square CDF. For instance, for the example circuit of Figure 3.2, corresponding to $\alpha = 0.9$ or 90%, the value of ψ determined from the χ^2_4 CDF tables, for the four-dimensional ellipsoid, is $\psi = 2.79$. The value assigned to ψ, determines the size of the uncertainty ellipsoid used to pad the nominal terms in the timing constraints. The pre-specified probability α serves as the lower bound on the timing yield, because the robust constraints formulated using the ellipsoid margin corresponding to such an α, would be satisfied for at least $\alpha\%$ of all cases. Since there are other points outside the ellipsoid set of the specified probability value that may not cause timing violations, the timing yield could be more than α.

For a general circuit, the procedure described for the example circuit of Figure 3.2 is repeated for each constraint. Thus, by addition of at most two additional variables for each constraint, robustness against the process uncertainties is added to original constraint set, while still maintaining the desirable posynomial structure of the constraints. By this procedure, we convert the conventional GP formulation of the gate sizing problem to a robust gate sizing problem, which is also a GP and hence, can be efficiently

solved using the convex optimization machinery.

3.4.3 Overestimation of Variations

The optimization formulation described in Section 3.4, adds margins to the deterministic constraints generated by an STA procedure. Due to the fact that separate margins are added at each node of the circuit graph, instead of the whole path, the resulting formulation could result in a large overestimation of the variational component of the circuit delay, which could lead to excessive design penalties.

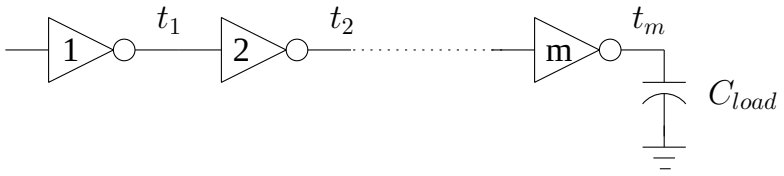


Figure 3.3: An example of a chain of inverters circuit to explain the problem of overestimation of variations in the robust GP formulation.

To understand the problem of this overestimation of variation, consider a simple example circuit consisting of m chain of inverters as shown in Figure 3.3. For this simple circuit, an STA module would generate the following block-based constraints:

$$\begin{aligned} d_1(\mathbf{X_0}) &\leq t_1 \\ t_1 + d_2(\mathbf{X_0}) &\leq t_2 \\ &\vdots \\ t_{m-1} + d_m(\mathbf{X_0}) &\leq t_m \\ t_m &\leq T_{spec} \end{aligned} \tag{3.27}$$

where d_i is the delay of the i^{th} inverter, which is a function of the vector of nominal gate sizes $\mathbf{X_0}$. By the method explained in Section 3.4, the equivalent robust constraints for

the example circuit of Figure 3.3, can be written as:

$$\begin{aligned} d_1(\mathbf{X_0}) + \max_{\forall \delta\Omega \in U} (\nabla_{\mathbf{\Omega_0}} d_1(\mathbf{X_0}, \mathbf{\Omega}) \delta\mathbf{\Omega}) &\leq t_1 \\ t_1 + d_2(\mathbf{X_0}) + \max_{\forall \delta\Omega \in U} (\nabla_{\mathbf{\Omega_0}} d_2(\mathbf{X_0}, \mathbf{\Omega}) \delta\mathbf{\Omega}) &\leq t_2 \\ &\vdots \\ t_{m-1} + d_m(\mathbf{X_0}) + \max_{\forall \delta\Omega \in U} (\nabla_{\mathbf{\Omega_0}} d_m(\mathbf{X_0}, \mathbf{\Omega}) \delta\mathbf{\Omega}) &\leq t_m \\ t_m &\leq T_{spec} \end{aligned} \tag{3.28}$$

It is easy to see that for the simple circuit of Figure 3.3, the delay is given by the whole path delay as $d_1(\mathbf{X_0}, \mathbf{\Omega}) + \cdots + d_m(\mathbf{X_0}, \mathbf{\Omega})$. Thus, the effect of variations can be accounted for by a simple robust constraint of the form:

$$\begin{aligned} d_1(\mathbf{X_0}) + \cdots + d_m(\mathbf{X_0}) + \max_{\forall \delta\Omega \in U} (\nabla_{\mathbf{\Omega_0}} (d_1(\mathbf{X_0}, \mathbf{\Omega})) + \\ \cdots + d_m(\mathbf{X_0}, \mathbf{\Omega}) \delta\mathbf{\Omega}) \leq T_{spec} \end{aligned} \tag{3.29}$$

For any m nonnegative functions, $y_1, \cdots, y_m$, the following inequality is well-known:

$$\max(y_1 + \cdots + y_m) \leq \max y_1 + \cdots + \max y_m \tag{3.30}$$

Therefore, for the variation terms in the constraints of (3.28) and (3.30), the following inequality holds:

$$\max_{\forall \delta\Omega \in U} (\nabla_{\mathbf{\Omega_0}} \sum_i d_i(\mathbf{X_0}, \mathbf{\Omega}) \delta\mathbf{\Omega}) \leq \sum_i \max_{\forall \delta\Omega \in U} (\nabla_{\mathbf{\Omega_0}} d_i(\mathbf{X_0}, \mathbf{\Omega}) \delta\mathbf{\Omega}) \tag{3.31}$$

It is clear from (3.28), (3.30) and (3.31), that the approach of adding the variational component of delay at each node leads to extra guard-banding.

Another way to understand the amount of pessimism introduced in the formulations is by realizing that the actual probability of failure, p_{fail1}, for the circuit of Figure 3.3 is given by:

$$p_{fail1} = Pr(d_1(\mathbf{X_0}, \mathbf{\Omega}) + \cdots + d_m(\mathbf{X_0}, \mathbf{\Omega})) > T_{spec} \tag{3.32}$$

On the other hand, the probability of failure, p_{fail2}, as computed by the padding of constraints at the each node in the circuit graph of Figure 3.3 is given by:

$$\begin{aligned} p_{fail2} &= [Pr(d_1(\mathbf{X_0}, \mathbf{\Omega}) > t_1)] \cup [Pr(t_1 + d_2(\mathbf{X_0}, \mathbf{\Omega}) > t_2)] \cup \\ &\quad \cdots \cup [Pr(t_{m-1} + d_m(\mathbf{X_0}, \mathbf{\Omega}) > T_{spec})] \end{aligned} \tag{3.33}$$

Clearly, from Equations (3.32) and (3.33), $p_{fail1} \leq p_{fail2}$. Thus, the robust GP formulation attempts to safeguard against a probability of timing failure that is greater than the actual failure probability, which could lead to extra design margins.

For a simple circuit similar to the one in Figure 3.3, it is trivial to trace the path delay, and then add margin to the whole path delay constraint. However, in general, the number of paths in a circuit graph can be exponential in the number of nodes. Therefore, enumeration of paths has a prohibitive cost for large circuits consisting of thousands of gates.

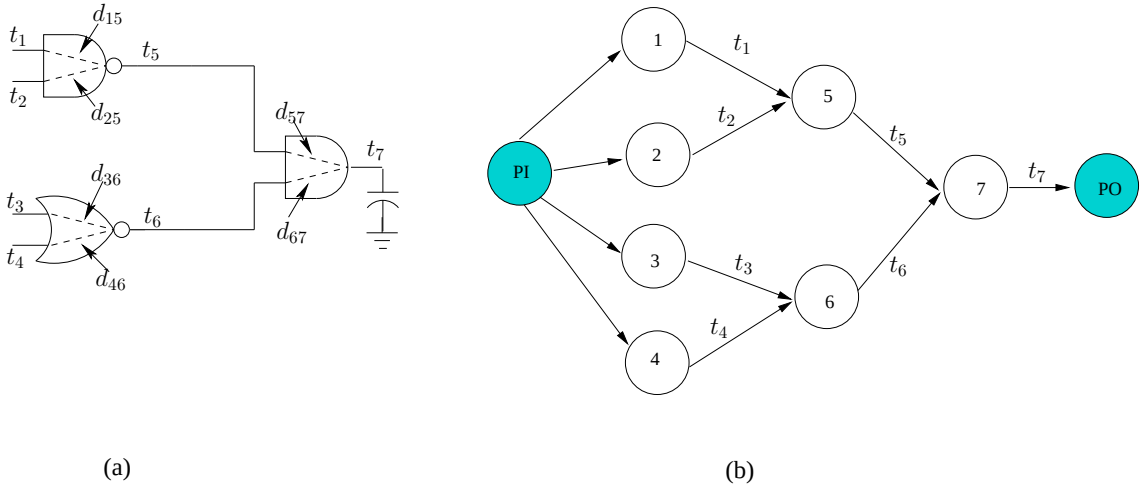
To reduce the problem of unnecessary padding at the intermediate nodes in the circuit, without incurring the exponential cost of formulating the path-based constraints, we employ a graph pruning technique proposed in [VC99]. The following section discusses this pruning method.

3.4.4 Graph Pruning

In [VC99], the authors propose a technique to reduce the number of variables, constraints and redundancy in the circuit optimization formulation, by removing the internal nodes and the original edges connected to them in the circuit graph. We apply this graph pruning technique to our method to reduce the pessimism in our gate sizing formulation.

This technique alters the delay constraints formulation by operating on the timing graph of the circuit. An initial timing graph of the circuit is constructed by representing each pin of a gate in the circuit as a vertex, and the connections between an input and an output pin of the same gate, and between an output pin of a gate and an input pin

of its fanout gate, as edges in the graph. The arrival time at a pin of a gate is used to annotate the edge originating at the node corresponding to that pin. Two additional nodes, representing the primary inputs (PI) and primary outputs (PO) are added to the vertex set of the graph. Figure 3.4 shows a simple circuit and its corresponding timing graph.



(a) (b)

Figure 3.4: A simple example circuit to illustrate the graph pruning method. (a) A two-level combinatorial circuit. (b) Timing graph for the circuit.

In the graph pruning method, the nodes of the graph are iteratively screened for a possible elimination by evaluating the cost of this node removal. The cost is typically expressed as some simple function of change in the number of variables and constraints in the optimization formulation, after the vertex under consideration is removed from the graph. If the evaluated cost is negative, implying a reduction in the problem size, the node is removed, and subsequently all incoming and outgoing edges of this node are also pruned from the graph.

The change in the formulation of delay constraints by a node removal can be understood by considering a segment of a circuit graph shown in Figure 3.5. In the above figure, we assume that node l meets the removal criterion according to the pruning cost.

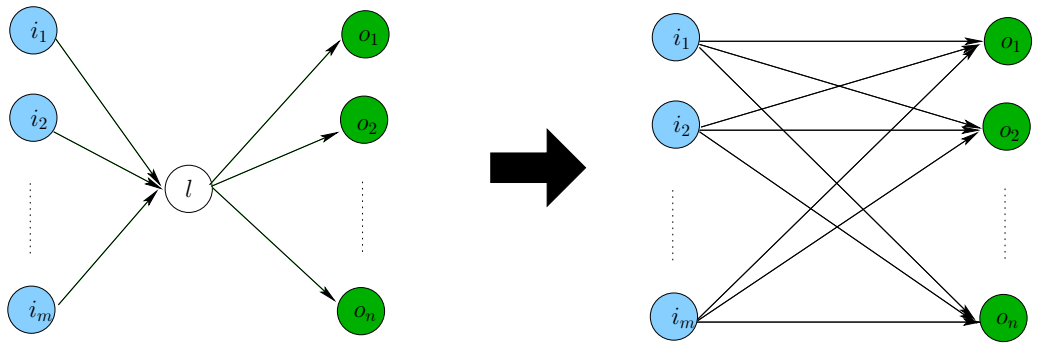


Figure 3.5: A segment of the timing graph of a circuit to illustrate the removal of a node in the graph pruning method. (a) The original graph segment. (b) The graph segment after pruning node l.

This node has m fanins, $i_1, \cdots, i_m$, and n fanouts, $o_1, \cdots, o_m$. The timing constraints for this graph segment before the node removal, as depicted by the graph segment of Figure 3.5(a) are:

$$\begin{aligned} t_{i_k} + d_{i_k,l} &\leq t_l \quad \forall k \in 1, \cdots, m \\ t_l + d_{l,o_j} &\leq t_{o_j} \quad \forall j \in 1, \cdots, n \end{aligned} \tag{3.34}$$

After eliminating node l, and the corresponding arrival time variable t_l, from the above constraint set, we obtain:

$$t_{i_k} + d_{i_k,l} + d_{l,o_j} \leq t_{o_j} \quad \forall k \in 1, \cdots, m, \quad \forall j \in 1, \cdots, n \tag{3.35}$$

These new constraints are shown graphically in Figure 3.5(b). The two sets of constraints in (3.34) and (3.35) are equivalent, and no timing information is lost in transforming from one set to the other. Since the pruning cost determines the nodes to be removed, a cost function constructed to reduce the problem size, e.g., a weighted sum of change in the number of variables and number of constraints, results in making the optimization formulation more compact after every pruning step.

Example of the Pruning Procedure

The application of the graph pruning method of [VC99] to reduce the pessimism in our optimization formulation can be best explained using a simple example circuit, and its corresponding timing graph. For this we refer back to the circuit of Figure 3.4. As shown in the figure, the arrival times at each pin of the logic gates are represented by variables $t_1, \cdots, t_7$. For simplicity, it is assumed that the interconnects have zero delay and that all primary inputs arrive at a time $t = 0$. The d_{ji} variables in Figure 3.4(a), represent the pin to pin delay of a logic gate. Figure 3.4(b) shows the corresponding timing graph for the example circuit. By employing an STA procedure, the delay constraints at the output of pin of each gate in the circuit of Figure 3.4(a) can be written as:

$$\begin{aligned} 0 &\leq t_i \quad i \in \{1,2,3,4\} \\ t_1 + d_{15}(\mathbf{X_0}, \mathbf{\Omega}) &\leq t_5 \\ t_2 + d_{25}(\mathbf{X_0}, \mathbf{\Omega}) &\leq t_5 \\ t_3 + d_{36}(\mathbf{X_0}, \mathbf{\Omega}) &\leq t_6 \\ t_4 + d_{46}(\mathbf{X_0}, \mathbf{\Omega}) &\leq t_6 \\ t_5 + d_{57}(\mathbf{X_0}, \mathbf{\Omega}) &\leq t_7 \\ t_6 + d_{67}(\mathbf{X_0}, \mathbf{\Omega}) &\leq t_7 \\ t_7 &\leq T_{spec} \end{aligned} \tag{3.36}$$

where $\mathbf{X_0}$ is the vector consisting of the sizes of the three gates of Figure 3.4(a), and $\mathbf{\Omega}$ is the random vector corresponding to the process uncertainties. From the discussion in Section 3.4.3, adding margins for each of the constraints of (3.36) can result in excessive guard-banding against the effect of variations, and hence a pessimistic design.

As described in the previous section, the circuit timing graph of Figure 3.4(b), and the corresponding constraints formulation of (3.36) can be altered by selectively removing nodes from the graph. Figure 3.6 illustrates the application of the graph pruning technique on the example circuit of Figure 3.4. For this specific example, the pruning

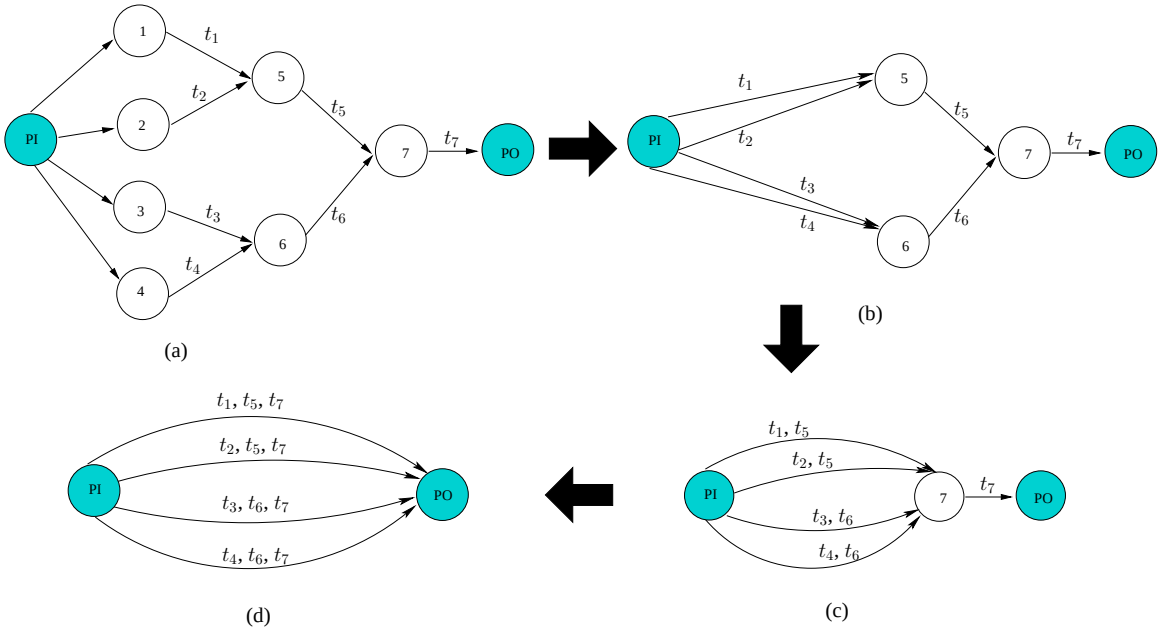


Figure 3.6: The graph pruning method applied to the example circuit of Figure 3.4. (a) The original circuit graph. (b) Graph after removing nodes 1, 2, 3 and 4. (c) Graph after removing nodes 5 and 6. (d) The final pruned graph.

cost chosen is simply the difference in the number of variable and constraints after removing a node from the graph. Figure 3.6(a) shows the graph obtained after eliminating nodes 1, 2, 3 and 4 in the original graph. Similarly, Figure 3.6(b) represents the graph after removing nodes 5 and 6, as well. The final pruned graph, obtained after removing all nodes except the PI and the PO nodes is shown in Figure 3.6(d). For each pruned node, a new edge is added between the fanin and fanout nodes of the removed node, and the new edge is annotated with the pruned arrival times. This annotation is required to generate the timing constraints at the end of the pruning procedure.

From the edge annotations, and the original constraints of (3.36), the constraints

corresponding to the final pruned circuit graph of Figure 3.6(d) can be written as:

$$\begin{aligned} d_{15}(\mathbf{X_0}, \mathbf{\Omega}) + d_{57}(\mathbf{X_0}, \mathbf{\Omega}) &\leq T_{spec} \\ d_{25}(\mathbf{X_0}, \mathbf{\Omega}) + d_{57}(\mathbf{X_0}, \mathbf{\Omega}) &\leq T_{spec} \\ d_{36}(\mathbf{X_0}, \mathbf{\Omega}) + d_{67}(\mathbf{X_0}, \mathbf{\Omega}) &\leq T_{spec} \\ d_{46}(\mathbf{X_0}, \mathbf{\Omega}) + d_{67}(\mathbf{X_0}, \mathbf{\Omega}) &\leq T_{spec} \end{aligned} \tag{3.37}$$

In the above set of constraints, the pruning method eliminates all nodes, except the ones corresponding to primary inputs and the primary output. Since all intermediate arrival time variables t_i are pruned, the above formulation does away with the problem of keeping redundant margins for the constraints at the output pin of each node. It should be emphasized that the example circuit of Figure 3.4 is an extremely simple case for which the pruning method can eliminate all intermediate nodes, and arrive at the path delay constraints of (3.37). Therefore, the problem of overestimation of effect of variation, as described in Section 3.4.3 is completely resolved for this example circuit. In general, for practical circuits, the graph pruning procedure could determine some nodes unsuitable for pruning, and some intermediate nodes could still remain in the final pruned circuit graph. However, due to the removal of many intermediate nodes, the pessimism in the robust optimization formulation is considerably reduced.

Practical Issues in Using Graph Pruning for the Robust GP Formulation

By removing a node with m fanins and n fanouts from the circuit graph, the change Δ_{con}, in the number of constraints is $\Delta_{con} = 2(mn - (m + n))$, and the change Δ_{var}, in the number of variables is $\Delta_{var} = -2$, as the variables corresponding to both rise and fall delays of the pruned node are eliminated. A pruning criterion can thus be established as some function $f_{cost}(\Delta_{con}, \Delta_{var})$, of change in the number of variables and constraint. The pruning procedure operates iteratively, in which the nodes with the lowest nonpositive f_{cost} are pruned in the first pass. After the first iteration, the

number of fanins and fanouts of the unpruned nodes are recalculated due to the addition of new edges in the pruned graph. This iterative method continued until all nodes in the graph produce a positive f_{cost}. At this point, no more nodes can be removed from the graph according to the given pruning metric. Typically, the pruning criterion is chosen as $f_{cost} = a.\Delta_{con} + b.\Delta_{var}$, where a and b are some normalized weighting factors. However, due to some practical problems in applying the graph pruning method to our formulation, we use a slightly modified pruning cost function. The following discussion explains these practical issues.

From (3.35), the number of d_{ji} terms, corresponding to the posynomial gate delay models, increase in every constraint during the pruning procedure. This results in the following problem for our robust GP formulation. Referring back to our robust GP method described in Section 3.4.2, we modify each delay constraint to include the terms corresponding to the maximum effect of variations inside the bounded uncertainty ellipsoid model. This is achieved by adding to each constraint, new robust variables r_1 and r_2, defined in Equation (3.23), and including additional constraints to the formulation, given by (3.25) and (3.26), as $\psi^2 {\phi_\mathbf{1}}^T P \phi_\mathbf{1} r_1^{-2} \leq 1$ and $\psi^2 \phi_\mathbf{2}^\mathbf{T} P \phi_\mathbf{2} r_2^{-2} \leq 1$. For constraints at each node of circuit graph, the vectors $\phi_\mathbf{1}$ and $\phi_\mathbf{2}$ are typically sparse, as these vectors consist of entries corresponding to a few parameters, affecting only a single gate delay. However, during the graph pruning method, as the intermediate nodes are removed, the number of d_{ji} terms increase in every constraint. Thus, the sparsity of $\phi_\mathbf{1}$ and $\phi_\mathbf{2}$ vectors is adversely affected. Moreover, as these vectors become dense, the number of monomial terms in the quadratic expansion of the constraints $\psi^2 {\phi_\mathbf{1}}^T P \phi_\mathbf{1} r_1^{-2}$, and $\psi^2 \phi_\mathbf{2}^\mathbf{T} P \phi_\mathbf{2} r_2^{-2}$ grow rapidly. As a result many constraints have monomial terms involving a large number of variables. Consequently, the constraint Jacobian matrix becomes very dense, which can considerably slow down the gradient computations required by the convex optimization methods, such as the interior point algorithm.

To overcome this issue of potential slow down of the gate sizing procedure, due to the increase in density of the constraint Jacobian matrix, we modify the pruning cost to

include a penalty term related to increasing the number of terms in the ϕ_1 and ϕ_2 vectors. We define $Mono_{num}$ as the maximum number of monomial terms in all the constraints affected by removing the node under consideration. The cost of pruning this node is then calculated as:

$$f_{cost} \quad = \quad a\Delta_{con} + b\Delta_{var} + c\max(Mono_{num} - Mono_{spec}, 0) \tag{3.38}$$

where c is a weight factor, and $Mono_{spec}$ is a user specified quantity to represent the maximum number of monomial terms allowed in each constraint. A higher value of $Mono_{spec}$ could result in more pruning, but at the cost of a potential slow down in obtaining the solution of the GP optimization problem. Thus, by adjusting the $Mono_{spec}$ parameter, the user can choose an engineering tradeoff between the runtime and the amount of pessimism reduction desired in the gate sizing procedure.

In the next section, we elaborate on another heuristic method to further reduce the pessimism in our formulation.

3.4.5 Using Variable Size Ellipsoids

The graph pruning procedure of [VC99], explained in Section 3.4.4, helps in eliminating many intermediate arrival time variables, and reduce the problem of variation overestimation in our formulation. However, as described in the previous section, it may not be possible to remove all intermediate nodes from the graph, and leave only the ones corresponding to the primary inputs and the primary outputs unpruned. The number of fanins and fanouts of a node increase monotonically during the pruning procedure. Therefore, for a given pruning cost of Equation (3.38), if a node is unsuitable for pruning in any iteration of the pruning method, i.e., it has a positive pruning cost, it will never be pruned under the same criterion. Due to the presence of the unpruned nodes in the circuit graph, the pessimism in our optimization formulation is not completely eradicated.

We present another method, to be employed after the graph pruning procedure, to further reduce the excessive margins from the timing constraints formulated at the un-

pruned nodes of the graph. This method is based on setting variable margins at different topological levels of the circuit. We use a simple example circuit consisting of just two inverters to explain this method.

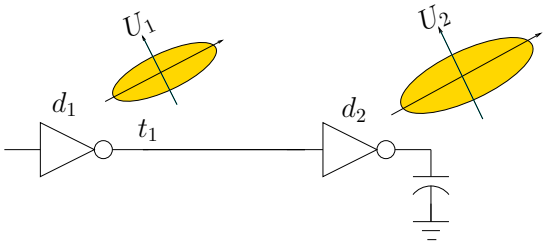


Figure 3.7: An example circuit to explain the use of variable size ellipsoids to reduce the pessimism in the robust GP formulation.

Consider the circuit of Figure 3.7 consisting of two inverter gates. For this simple circuit, the intermediate node, corresponding to the output pin of the first inverter, can be easily removed to formulate the path delay constraint. However, for the purposes of exposition of the method of using variable ellipsoids, we do not employ any pruning and formulate the constraints for this circuit as:

$$d_1(\mathbf{X_0}) + \max_{\forall \delta\mathbf{\Omega} \in U_1} (\nabla_{\mathbf{\Omega_0}} d_1(\mathbf{X_0}, \mathbf{\Omega}) \delta\mathbf{\Omega}) \leq t_1 \tag{3.39}$$

$$t_1 + d_2(\mathbf{X_0}) + \max_{\forall \delta\mathbf{\Omega} \in U_2} (\nabla_{\mathbf{\Omega_0}} d_2(\mathbf{X_0}, \mathbf{\Omega}) \delta\mathbf{\Omega}) \leq T_{spec} \tag{3.40}$$

We use different guard-bands for the constraints (3.39) and (3.40), by employing two uncertainty ellipsoids, U_1 and U_2 given by:

$$U_1 = \{\mathbf{\Omega} : (\mathbf{\Omega} - \mathbf{\Omega_0})^T P^{-1} (\mathbf{\Omega} - \mathbf{\Omega_0}) \leq \psi_1^2\} \tag{3.41}$$

$$U_2 = \{\mathbf{\Omega} : (\mathbf{\Omega} - \mathbf{\Omega_0})^T P^{-1} (\mathbf{\Omega} - \mathbf{\Omega_0}) \leq \psi_2^2\} \tag{3.42}$$

where $\psi_1 < \psi_2$. As explained in Section 3.3.4, we can use the CDF tables of the χ_n^2 distribution to associate probability values, α_1 and α_2 with the ellipsoids U_1 and U_2, respectively. As $\psi_1 < \psi_2$, it follows that $\alpha_1 < \alpha_2$.

A simple probabilistic analysis to achieve the timing yield of the circuit of Figure 3.7, provides insights into the idea of using variable ellipsoids. Using the bounded ellipsoid model for parameter variations, we first define two random variables β_1 and β_2 as:

$$\beta_1 = \max_{\forall \delta\mathbf{\Omega} \in U_1} (\nabla_{\mathbf{\Omega_0}} d_1(\mathbf{X}, \mathbf{\Omega})\delta\mathbf{\Omega}) - (_{\forall \delta\mathbf{\Omega}} d_1(\mathbf{X}, \mathbf{\Omega})) \quad (3.43)$$

$$\beta_2 = \max_{\forall \delta\mathbf{\Omega} \in U_2} (\nabla_{\mathbf{\Omega_0}} d_2(\mathbf{X}, \mathbf{\Omega})\delta\mathbf{\Omega}) - (_{\forall \delta\mathbf{\Omega}} d_2(\mathbf{X}, \mathbf{\Omega})) \quad (3.44)$$

The random variables defined in Equations (3.43) and (3.44), relate to the values α_1 and α_2 as :

$$\alpha_1 \leq Pr(\beta_1 > 0) \quad (3.45)$$

$$\alpha_2 \leq Pr(\beta_2 > 0) \quad (3.46)$$

By using a smaller ellipsoid U_1 to guard-band the timing constraint of (3.39), we associate a smaller probability α_1, as a lower bound on the chance that this small design margin would be sufficient to meet the constraint in the face of variations. However, even if the design margin is not sufficient to meet this constraint, corresponding to the case that $\beta_1 < 0$, by employing a larger ellipsoid U_2, and the corresponding bigger probability α_2, to pad the timing constraint of (3.40), we have a better chance to compensate for the violation of constraint (3.39). Mathematically, if A is the probabilistic event that constraint (3.39) is not met, and B is the event that the circuit fails to meet the specified delay, the following relation holds[5]:

$$\begin{aligned} Pr(B/A) = {} & Pr(\beta_1 < 0)Pr(\beta_2 > 0/\beta_1 < 0)Pr((|\beta_1| > |\beta_2|)/\beta_2 > 0, \beta_1 < 0) \\ & + Pr(\beta_2 < 0)Pr(\beta_1 < 0/\beta_2 < 0) \end{aligned} \quad (3.47)$$

The use of a larger ellipsoid U_2 with an associated lower bound probability $\alpha_2 \leq Pr(\beta_2 > 0)$, ensures that for the cases when $\beta_1 < 0$, the term $Pr(\beta_2 < 0)$ and the

[5]Since the parameters of the two inverters may be correlated, Equation (3.47) contains terms corresponding to conditional probabilities.

conditional probability term $Pr((|\beta_1| > |\beta_2|)/\beta_2 > 0, \beta_1 < 0)$ in Equation (3.47) are reasonably small. Therefore, the scheme of using a smaller design margin for a lower topological level, followed by a sufficiently large design margin for higher levels can still provide the necessary guard-banding to achieve the desired timing yield.

For a general circuit with k topological levels, we employ k uncertainty ellipsoids, $U_1, U_2, \cdots, U_k$, characterized by the constants, $\psi_1, \psi_2, \cdots, \psi_k$, with $\psi_1 < \psi_2 < \cdots < \psi_k$. Since it is extremely difficult to relate the individual ellipsoid sizes with the timing yield specification, we heuristically chose ψ_k to correspond to the lower bound on the timing yield α_k, and progressively decrease the constants $\psi_{k-1}, \cdots, \psi_1$. The value of ψ_k is determined from the tables of the χ^2_n distribution. The margins at logic levels, $1, \cdots, k-1$, are determined by setting:

$$\alpha_i = \alpha_k - \gamma.(k-i) \quad i = 1, \cdots, k-1 \tag{3.48}$$

where γ is an empirically determined factor. Using smaller timing margins at lower topological levels, as compared to choosing the same margin at all levels, corresponding to the lower bound on timing yield α_k, helps in reducing the pessimism in our formulation.

It should be noted that this scheme of using variable sized ellipsoids is employed for the unpruned nodes, only after the graph pruning step. The graph pruning method of [VC99], followed by the heuristic scheme of keeping variable guard-bands at different topological levels of the final pruned circuit, significantly reduces the problem of overestimation of variation in our gate sizing procedure.

3.4.6 Incorporating Spatial Correlations

We use the grid based spatial correlation model of [CS03] and [ABZ03] to incorporate the intra-die correlations between the parameters variations that exhibit spatial dependence, such as the transistor w and L_e.

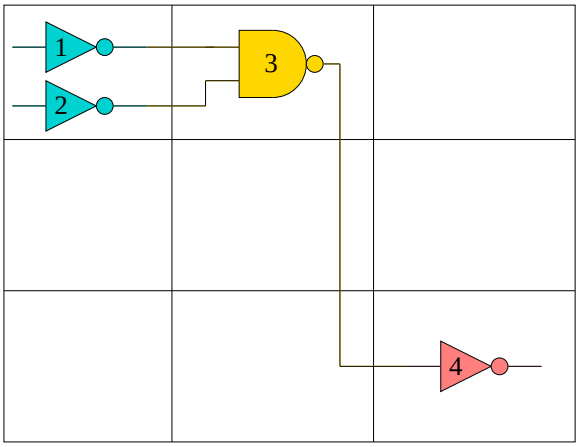


Figure 3.8: A grid based spatial correlation model. The layout is divided into a 3×3 grid. The gates in the same grid are assumed to have a perfect correlation. Gates in the nearby grids are assigned a high correlation factor, and the gates in far away grids are assigned a low or a zero correlation factor.

Figure 3.8 refers to such a model, where the layout area is partitioned into $m = 9$ grids. The widths (channel lengths) of the devices located in the same grid are assigned a perfect correlation factor, device widths (channel lengths) in nearby grids are assigned a high correlation factor, and the ones in far away grids have a low or zero correlation factor. As seen in Figure 3.8, gates $\{1,2\}$ have perfect correlation between their widths (channel lengths), gates $\{1,3\}$ and $\{2,3\}$ have high correlations, where as gates $\{1,4\}$ and $\{2,4\}$ are uncorrelated.

For a random vector $\mathbf{\Omega}$ representing the variations in w and L_e, and its corresponding covariance matrix P, the entry $P_{ij} = \sigma_i \sigma_j \rho_{ij}$ denotes the covariance between components i and j of $\mathbf{\Omega}$, where σ is the standard deviation of each random variable, and ρ_{ij} is the correlation factor between the random variables i and j. By employing the spatial correlation model of Figure 3.8, the correlation factor between all elements of $\mathbf{\Omega}$ is computed, and stamped out in matrix P. The ellipsoid uncertainty model, described in Section 3.3.3, then incorporates the impact of correlations in the robust optimization

formulation.

The following simple example explains how the correlations are captured by the uncertainty ellipsoid. Consider a simple constraint involving the transistor widths of two gates:

$$t_j + \frac{K_1 w_1}{w_2} \quad \leq \quad t_i \tag{3.49}$$

For simplicity, we assume that the gate widths, w_1 and w_2, are the only two varying parameters, and the other parameters are subsumed in the constant K_1. Furthermore, we assume that the gates are placed in the same grid of the spatial correlation model, hence, the variations in the two gate widths are same, i.e., $\delta w_1 = \delta w_2$. If the nominal gate sizes are also assumed to be identical, i.e., $w_{1_0} = w_{2_0}$, the effect of process variation cancels out in the numerator and denominator of (3.49), and no guard-banding is required. To verify that the ellipsoid uncertainty correctly incorporates this perfect correlation scenario, we apply our robust optimization procedure to the constraint of (3.49). Generating a first order Taylor series expansion of the constraint around the nominal values (w_{1_0}, w_{2_0}), and applying the ellipsoid uncertainty yields:

$$t_j + \frac{K_1 w_{1_0}}{w_{2_0}} + \max_{\forall \mathbf{u} \| \|\mathbf{u}\|_2 \leq \psi} \left(\frac{K_1 (P^{1/2}\mathbf{u})_1}{w_{2_0}} - \frac{K_1 w_{1_0} (P^{1/2}\mathbf{u})_2}{w_{2_0}^2}\right) \quad \leq \quad t_i \tag{3.50}$$

However, since we have perfect correlation between w_1 and w_2, the correlation factor, $\rho_{12} = \rho_{21} = 1$. Therefore, the correlation matrix P is given by:

$$P \quad = \quad \begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2 \\ \sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}$$

Furthermore, since the variations in w_1 and w_2, and the mean values are same, we must have $\sigma_1 = \sigma_2$. It then follows that for all vectors $\mathbf{u} = [u_1, u_2]$, which characterize the uncertainty ellipsoid, we have $(P^{1/2}u)_1 = \sigma_1^2 u_1 + \sigma_1\sigma_2 u_2 = (P^{1/2}u)_2 = \sigma_2^2 u_2 + \sigma_1\sigma_2 u_1$, and the variational term in (3.49) is:

$$\frac{K_1 (P^{1/2}\mathbf{u})_1}{w_{2_0}} - \frac{K_1 w_{1_0} (P^{1/2}\mathbf{u})_2}{w_{2_0}^2} \quad = \quad 0$$

Thus, the ellipsoid uncertainty model easily captures the effects of correlations between random variables, and incorporates the same in the optimization procedure. Incorporating the correlations in gate sizing optimization procedure reduces the pessimism involved with a worst-casing scheme, and provides opportunities for saving expensive design resources.

3.4.7 The Complete Sizing Procedure

The complete gate sizing procedure can be recapitulated by the following steps:

1. Generate the initial non-robust timing constraints by an STA procedure.

2. On the original circuit graph, employ the graph pruning method of [VC99], described in Section 3.4.4, to remove as many intermediate nodes as possible according to the pruning cost function of Equation (3.38).

3. For the final pruned graph, generate new timing constraints using the edge annotations in the final pruned graph.

4. Generate a first order Taylor series expression for each constraint at the nominal values of the parameters.

5. Employing the uncertainty ellipsoid model, transform each constraint to a set of robust constraints as described in Section 3.4.2. For this step, use variable size ellipsoids at each topological level of the circuit, as explained in Section 3.4.5.

6. Solve the resulting GP by using convex optimization tools.

The solution of the convex optimization problem provides the gate sizes for the circuit that minimize the area objective, subject to the specified timing yield constraints.

3.5 Experimental Results

The proposed robust gate sizing procedure was implemented in C++, and an optimization software [Mos] was used to solve the final GP. All experiments were performed on P-4 Linux machines with a clock speed of 3.2GHz, and 2GB of memory. The robust gate sizing technique was applied to the ISCAS 85 benchmark circuits. The cell library selected comprised inverters, and two and three input NAND and NOR gates. We assume capacitive loading for the gates. For simplicity we consider the variations in the transistor width, and the effective channel length as the only sources of variation. However, our approach can be easily extended to incorporate various other parameters of variation for the gate and interconnect delays. We use a simple Elmore delay model to generate posynomial gate delay models. Our approach can work just as well for any other posynomial based delay models, such as the ones based on generalized posynomials proposed in [KKS98].

We use the spatial correlation model of [ABZ03] and [CS03] to generate the elements of the covariance matrix P. To use these spatial correlation models, we first place the circuits using the placement tool Capo [CKM], and then divide the chip area into different number of grids, depending on the circuit size, so that each grid size is no greater than 50 $\mu \times 50\ \mu$. The standard deviations of the w and L_e parameters are chosen from [Nas00] for a $100\ nm$ technology node. The objective function chosen for the optimization is to minimize $Area = \sum_i a_i w_{i_0}$, where a_i is the number of transistors in gate i. For each circuit, the value of T_{spec} is chosen to be the point of 15% slack, i.e., $T_{spec} = D_{min} + 0.15(D_{max} - D_{min})$, where D_{min} and D_{max} are, respectively, the minimum and the maximum possible delays of the circuit, found by setting all gates to the minimum and the maximum size, respectively.

We implement the graph pruning technique of [VC99] to address the problem of overestimation of variation. As described in Section 3.4.4, we set the pruning cost of a node as $f_{cost} = a\Delta_{con} + b\Delta_{var} + c\max(Mono_{num} - Mono_{spec}, 0)$. For this cost function,

we choose $a = 1.5$, $b = 1$, $c = 1$. We choose different values for the term $Mono_{spec}$, that determines the maximum number of monomial terms allowed in each constraint. As described in Section 3.4.5, we employ smaller sized uncertainty ellipsoids at lower topological levels of the circuit, and progressively increase the ellipsoid size at higher logic levels. The size of the largest ellipsoid employed at the highest logic level k, characterized by ψ_k, is chosen to correspond to the lower bound on the timing yield specification, α_k. The value of ψ_k is determined from the tables of the χ^2_n distribution. The margins at logic levels, $1, \cdots, k-1$, are determined by using Equation(3.48) and choosing the factor γ to be in the interval of $[0.05, 0.10]$, which corresponds to a 5%-10% decrement from the value of α_k, that specifies the lower bound on the timing yield. The value of each ψ_i, corresponding to the α_i in Equation (3.48), is determined from the CDF tables of the Chi-square distribution.

In the first set of experiments, we compare the gate sizing solution obtained by our method with a deterministic gate sizing solution. The deterministic gate sizing is also formulated as a GP, using the formulation of 3.4, but it does not take into account the effect of parameter variations. For our robust optimization procedure, we set the lower bound on timing yield, $\alpha_k = 85\%$, and choose the value of $Mono_{spec} = 35$. To simulate the effect of parameter variations, we perform Monte Carlo analysis. We refer to the set of gate sizes obtained from the deterministic, and the robust optimization as $\mathbf{X_{0_{det}}}$ and $\mathbf{X_{0_{rob}}}$, respectively. Using these sizes, we generate $10,000$ samples each, from two multivariate normal distributions, $N_1(\mathbf{X_{0_{det}}}, P)$ and $N_2(\mathbf{X_{0_{rob}}}, P)$. Next, we perform an STA for each of these samples, and record the number of times the circuit meets the specified target delay. The timing yield of the two optimizations are then determined as $yield_{det} = n_{det} \times 100/M$, and $yield_{rob} = n_{rob} \times 100/M$, where n_{det} is the number of samples drawn from the $N_1(\mathbf{X_{0_{det}}}, P)$ distribution that meet the timing requirements, and n_{rob} is the number of samples drawn from the $N_2(\mathbf{X_{0_{rob}}}, P)$ distribution that meet the specified target delay. The total number of Monte Carlo samples is given by $M = 10000$. Table 3.1 contains the relevant data for this comparison.

Ckt	Gates	Deterministic Design			Robust Design		
		Area	$Yield_{det}$%	Runtime (sec)	Area	$Yield_{rob}$%	Runtime (sec)
C432	616	1.00	22.31%	3	1.12	99.91%	15
C499	1262	1.00	30.34%	2	1.18	99.94%	23
C880	854	1.00	28.46%	8	1.10	99.92%	18
C1355	1202	1.00	32.34%	12	1.15	98.89%	31
C1908	1636	1.00	35.14%	18	1.14	99.56%	159
C2670	2072	1.00	39.91%	30	1.17	99.83%	189
C3540	2882	1.00	33.31%	25	1.08	98.82%	212
C5315	4514	1.00	38.46%	43	1.12	98.76%	579
C6288	5548	1.00	37.45%	58	1.14	99.22%	742
C7552	6524	1.00	34.78%	90	1.17	99.13%	845

Table 3.1: A timing yield comparison of deterministic and robust gate sizing solutions.

The first column in Table 3.1 lists the benchmark circuit, and the number of gates in each circuit is shown in column two. The timing yield of the deterministically sized circuits, $Yield_{det}$, is listed in column four of the table. Since the non-robust gate sizing method does not take into account the effect of variations, the timing yield, as expected, is quite low for these circuits. Our robust sizing method, eliminates these timing violations by keeping adequate design margins. Column seven list the timing yield, $Yield_{rob}$, of the robustly sized circuits. It should be noted that a value of $\alpha_k = 85\%$, as a lower bound on the timing yield, is sufficient to provide an actual yield of about 99% for all benchmark circuits. The area overhead that the robust circuits have to employ to safeguard against the parameter variations is shown in sixth column of Table 3.1. At the cost of an area increase of about 8% to 18%, the robustly sized circuits are able to eliminate almost all timing violations. The runtimes of the deterministically, and robustly sized circuits are listed, respectively, in columns five and eight of the table. As seen in the table, the robust methods is much slower than the deterministic sizing procedure. The

steps of employing graph pruning, and the increased problem size of the robust gate sizing procedure due to the presence of robust variables and constraints lead to this relatively higher runtimes. However, the overall runtimes of the gate sizing method are very reasonable.

Ckt	Timing Yield for the Same Area Worst-Case (WC) and Robust (Rob) Designs											
	$\alpha_k = 0.55$			$\alpha_k = 0.65$			$\alpha_k = 0.75$			$\alpha_k = 0.85$		
	WC	Rob	Area	WC	Rob	Area	WC	Rob	Area	WC	Rob	Area
C432	45.63%	68.65%	1.05	86.78%	97.03%	1.08	91.62%	98.14%	1.10	93.12%	99.91%	1.12
C499	51.45%	63.45%	1.08	67.12%	74.28%	1.11	85.12%	97.01%	1.14	94.20%	99.94%	1.18
C880	52.36%	67.52%	1.03	77.38%	88.50%	1.06	88.42%	97.34%	1.08	92.38%	99.92%	1.10
C1355	55.78%	75.21%	1.08	66.17%	84.89%	1.11	82.66%	98.11%	1.13	91.43%	98.89%	1.15
C1908	50.67%	72.76%	1.06	70.69%	87.14%	1.10	84.53%	96.67%	1.12	93.89%	99.56%	1.14
C2670	56.32%	73.68%	1.08	72.86%	88.21%	1.11	89.23%	95.33%	1.14	92.34%	99.83%	1.17
C3540	60.22%	78.14%	1.02	76.15%	89.12%	1.04	89.32%	95.56%	1.06	94.14%	98.82%	1.08
C5315	55.81%	74.98%	1.05	75.50%	87.67%	1.08	90.56%	96.89%	1.10	93.45%	98.76%	1.12
C6288	55.39%	77.16%	1.07	69.79%	88.12%	1.10	85.78%	95.78%	1.12	91.91%	99.22%	1.14
C7552	49.08%	70.48%	1.08	66.21%	85.56%	1.12	83.89%	94.54%	1.15	90.11%	99.13%	1.17

Table 3.2: A comparison of the robust and worst case gate sizing designs using the same area.

We perform another series of experiments to compare our approach with a gate sizing methodology employing a conventional worst-case design approach. The worst-case designs are obtained by iteratively solving the standard GP, but for delay specifications tighter than the original required target delay, until the area of the worst-case design is the same as that of the robust design. These circuits are thus designed using an in-built guard-band, determined by the difference of the original target delay and the tighter delay specification. Furthermore, to explore the area-robustness tradeoff we vary the size of the largest uncertainty ellipsoid used, by choosing different values of the factor α_k, that determines the lower bound on the timing yield of the robustly sized circuits. For these experiments, as before, we set the values of $Mono_{spec} = 35$, to define the pruning cost function of Equation (3.38). Having sized these circuits, we perform Monte Carlo simulations to determine the timing yield of the worst-case and the robust circuits.

Table 3.2 lists the results of these experiments. As seen from the table, the num-

ber of timing violations reduces with increase in area, for both the worst-case and the robust circuits. However, in all cases, our robust design has a better timing yield than the worst-case design having the same area. On an average, the robust design has about 12% greater timing yield than the worst-case design having the same area. The better performance of our robust sizing solution is not surprising because of the fact that the spatial correlation information, stored in the P matrix, is used by the optimization scheme. The worst-case circuit is expected to have a large overhead, since designing by setting tighter delay specifications results in rendering critical some of the earlier non-critical paths. Therefore, the optimizer now has to aggressively size the gates on these paths, which results in greater transistor area than actually required. Since, the runtimes for our robust gate sizing solutions are not prohibitively high, the user can run the optimization for different values of α_k, to select the amount of robustness required against the process uncertainties, at the cost of additional chip area.

In the next set of experiments, we investigate the usefulness of the graph pruning method, and employing different sized ellipsoids, in reducing the pessimism in our robust formulation. We first employ graph pruning, and use variable sized ellipsoids to optimize the benchmark circuits. At the highest topological circuit level, we use the largest ellipsoid corresponding to a value of $\alpha_k = 0.65$. At the lower topological levels, we progressive decrease the ellipsoid size by choosing a lower α, as given by Equation (3.48). We use a value of $Mono_{spec} = 35$ to set the pruning cost according to Equation (3.38). These circuits are referred to as Rob_1 designs. Next, we optimize the benchmark circuits without any pruning, and using the same sized ellipsoids at all nodes, determined by the values of $\alpha_k = 0.65$. These optimized circuits are referred to as Rob_2 designs.

Table 3.3 contains the results of these experiments. The yields of the two designs, $Yield_{rob_1}$ and $Yield_{rob_2}$, are listed, respectively, in columns seven and ten of the table. The area employed by the Rob_1 and Rob_2 designs are shown, respectively, in columns six and nine of the table. As seen from this data in Table 3.3, the designs employing

Ckt	Gates	Deterministic Design			Rob_1 Design			Rob_2 Design		
		Area	$Yield_{det}$%	Runtime (sec)	Area	$Yield_{rob_1}$%	Runtime (sec)	Area	$Yield_{rob_2}$%	Runtime (sec)
C432	616	1.00	22.31%	3	1.08	97.03%	15	1.15	98.32%	14
C499	1262	1.00	30.34%	2	1.11	74.28%	23	1.17	76.78%	21
C880	854	1.00	28.46%	8	1.06	88.50%	18	1.14	90.23%	16
C1355	1202	1.00	32.34%	12	1.11	84.89%	31	1.20	85.34%	27
C1908	1636	1.00	35.14%	18	1.10	87.14%	159	1.22	89.12%	123
C2670	2072	1.00	39.91%	30	1.11	88.21%	189	1.24	89.03%	158
C3540	2882	1.00	33.31%	25	1.04	89.12%	212	1.17	90.32%	181
C5315	4514	1.00	38.46%	43	1.08	87.67%	579	1.23	89.32%	398
C6288	5548	1.00	37.45%	58	1.10	88.12%	742	1.24	90.45%	587
C7552	6524	1.00	34.78%	90	1.12	85.56%	845	1.27	87.29%	693

Table 3.3: A comparison of robust gate sizing solutions, with and without using graph pruning and variable size ellipsoids.

the heuristic techniques of graph pruning, and using variable size ellipsoids use about 7% to 15% lesser circuit area compared to the design without any pruning, and using a constant size ellipsoid. The timing yields of Rob_2 designs are only slightly better, $< 2\%$ for all circuits, compared to the timing yields of Rob_1 design. This indicates that employing the graph pruning method, and the strategy of keeping variable guard-bands for the timing constraints, leads to considerable pessimism reduction in our optimization formulation, without a significant loss in the timing yield of the circuit. The runtimes for the Rob_2 designs are smaller compared to Rob_1 designs. This is due to the fact the robust constraints of (3.25) and (3.26) have fewer monomial terms for the procedure not employing any pruning compared to the one that prunes some intermediate nodes. As a result, the constraint functions are sparser for the former method, which helps in speeding up the optimization. The absence of the graph pruning step also makes the procedure for Rob_2 design run faster.

In the last set of experiments, we explore the tradeoff obtained by tuning the pruning cost function by changing the value of the $Mono_{spec}$ term, which regulates the maximum number of monomials allowed in a constraint. This term in the pruning cost of Equation (3.38) helps in preventing the constraint Jacobian matrix from becoming immoderately dense. Table 3.4 contains the results of these experiments. As seen in the table, as the

Ckt	$Mono_{spec} = 20$			$Mono_{spec} = 35$			$Mono_{spec} = 50$		
	Area	$Yield$	Runtime (sec)	Area	$Yield$	Runtime (sec)	Area	$Yield$	Runtime (sec)
C432	1.09	97.58%	15	1.08	97.03%	15	1.07	96.89%	17
C499	1.11	74.89%	22	1.11	74.28%	23	1.10	74.10%	25
C880	1.07	88.91%	18	1.06	88.50%	18	1.05	87.78%	20
C1355	1.12	85.12%	29	1.11	84.89%	31	1.10	83.67%	33
C1908	1.10	87.89%	147	1.10	87.14%	159	1.09	86.57%	172
C2670	1.13	88.95%	176	1.11	88.21%	189	1.10	87.34%	231
C3540	1.06	90.05%	200	1.04	89.12%	212	1.04	88.78%	294
C5315	1.09	88.34%	504	1.08	87.67%	579	1.07	86.89%	681
C6288	1.13	89.57%	657	1.10	88.12%	742	1.08	87.34%	920
C7552	1.14	86.78%	784	1.12	85.56%	845	1.10	84.12%	1027

Table 3.4: A comparison of the robust gate sizing designs obtained by changing the pruning cost function of Equation (3.38).

value of $Mono_{spec}$ term is increases, the runtime of the procedure increases. For the larger benchmark circuits, the slow down of the optimizer is significant, e.g., for C6288 circuit, the runtime increases by almost 40% by increasing the value of the $Mono_{spec}$ term from 20 to 50. This is due to the fact that for larger circuits, with thousands of constraints, the sparsity of the large constraint matrix has a greater impact on the speed of the convex optimization tool. Although, the runtime of the robust optimization method increases, for higher values of $Mono_{spec}$ term, there is also a greater reduction of pessimism in the formulation, due to more aggressive pruning. This results in lesser use of the circuit area for a higher valuer of $Mono_{spec}$ term. For example, for C6288 circuit, there is a 5% reduction in area by increasing the value of $Mono_{spec}$ from 20 to 50. The timing yield is not significantly impacted by changing the value of the $Mono_{spec}$ term. Based on this runtime and reduction in circuit area tradeoff, the user can appropriately set the value of $Mono_{spec}$ term to be employed in the pruning cost function of Equation (3.38).

3.6 Conclusion

In this chapter of the thesis, we have presented an optimization method to perform gate sizing, as a technique to reduce the impact of uncontrollable process variations. Our procedure is a worst-casing methodology that tries to keep smart design margins to safeguard against the effect of variations. To enable efficient optimization, we employ various reasonable and realistic assumptions. Assuming a multivariate normal distribution for the process-driven parameter variations, an uncertainty ellipsoid set is employed as a bounded model for these variations. This uncertainty ellipsoid, defined by the appropriate covariance matrix of the varying parameters, incorporates the effect of spatial correlations in the optimization set up. The multivariate Gaussian assumption for parameter distributions allows the use of Chi-square CDF tables to specify a lower bound on the timing yield of the circuit. Using posynomial delay models, the optimization formulation for the gate sizing procedure is relaxed to a geometric program, that is solved using convex optimization tools. We use first order Taylor series expansions of these posynomial delay functions, to generate the variational terms of the timing constraints.

In the optimization procedure, we use the results of well-known Cauchy Schwartz inequality to add guard-bands to protect against the worst-case effect of variations. To reduce the pessimism associated with the node-based formulation, we employ the techniques of graph pruning and heuristically choosing variable sized ellipsoids at different topological levels of the circuit. We use Monte Carlo analysis to verify the results of our optimized designs. Experimental results show that for the same transistor area, the circuits sized by of our robust optimization approach have, on an average, 12% fewer timing violations as compared to the gate sizing solutions obtained via the traditional, deterministically based guard-banding method.

Chapter 4

Statistical Timing Analysis Incorporating Correlated Non-Gaussian Parameters

In Chapter 3, we presented an optimization method in the presence of process-driven uncontrollable variations. In this chapter, we present a statistical timing method, to determine the timing characteristics of a circuit, in the presence of process parameter variations. The proposed method can predict the timing yield of the circuit by evaluating the circuit delay probability distribution functions, and may be used inside an optimization engine to achieve the desired timing yield.

4.1 Introduction to SSTA

As transistor and interconnect geometries shrink, the reduced level of control over the chip fabrication process results in significant levels of variation in process parameters such as the effective channel length, gate width, gate oxide thickness, dopant concentration, and interlayer dielectric thickness. These variations create randomness in the behavior of circuit-level electrical parameters, such as gate and interconnect capacitances, transistor on-resistances, threshold voltages and via resistances. The prediction of chip timing characteristics in the face of these process-driven random parameter uncertainties remains a challenging problem.

Traditionally, to safeguard against this variability, a static timing analysis (STA) procedure is employed at different process corners, and margins are introduced in the design based on the STA results. This worst case design, corresponding to the process corners, where the gate and wire delays are at their extreme levels, ensures that the design would work for any other values of gate and interconnect delays. However, with increasing levels of variations, the corner-based method becomes impractical and computationally

expensive. The number of process corners that must be considered grows exponentially as the number of uncertain parameters increase. Moreover, the corner-based method does not utilize any statistical information about the variations of parameters, such as the correlations between the process variables arising from the spatial proximity of the manufactured transistors on chip, or from the structural properties of the circuit such as path reconvergences, and hence can result in overly pessimistic and suboptimal designs. The results of variation-aware timing are eventually required to be used for a circuit optimization tool. Since, the multi-corner-based methodology produces overly pessimistic estimates of a circuit timing characteristics, any optimization tool using these results could lead to a design employing much more resources than actually required. This would adversely impact the other performance measures of the circuit, such as the circuit power.

Monte Carlo simulation provides an alternative means to measure the probability distributions of the delay of a circuit. This method is based on a sampling and simulation framework. In an iterative process, a sample of the uncertain process variables is drawn from the underlying distributions of the variables, and an STA is performed. For each sample, the result of the STA is recorded, and the probability distributions of the timing characteristic is inferred by binning each delay value into discrete bins, corresponding to some delay ranges. However, for sufficient accuracy, the Monte Carlo methods require thousands of samples. Since each delay value, corresponding to one sample, has an expensive cost of one STA method, the overall Monte Carlo simulation technique becomes extremely prohibitive for even medium-size circuits comprising thousands of gates.

As a result, the field of statistical static timing analysis (SSTA) has recently become an active area of research. An SSTA procedure aims at efficiently predicting the probability distribution function (PDF) and the cumulative distribution function (CDF) of the delay. In other words, SSTA evaluates the statistical distributions of the delay from the statistical information of sources of variation. A computationally efficient SSTA

algorithm facilitates the easy prediction of timing yield, and can be used within an optimization engine to robustly optimize the circuit in the presence of parameter variations.

In the next section, we review some previous SSTA methods, and in the other sections of this chapter, we explain our proposed extension to the statistical timing algorithms to efficiently incorporate non-Gaussian parameters of variation. An early version of this work was published in [SS06a].

4.2 Previous Work

Existing SSTA algorithms have many flavors: they may be path-based or block-based; they may assume Gaussian or non-Gaussian distributions; they may be parameterized in expressing all delay variables in terms of underlying parameters or not; they may incorporate spatial correlations due to physical proximity or not; and so on. In [DK03], the authors provide a non-parameterized method to perform SSTA in a block-based manner. This method is based on performing statistical operations of the assumed independent arrival time and random variables, by piecewise-linear modeling of CDF of variables. The authors of [OB04] present another non-parameterized SSTA procedure to estimate the bounds on the circuit delay PDF and CDF. In contrast, parameterized methods for SSTA provide a convenient framework for analyzing the relationship between the statistical information of the sources of variation to that of the circuit delay distributions, and are more useful in practice. A parameterized model also enables efficient computation of the statistical sensitivities of the circuit delay with respect to the varying parameters [LLCP05, ZSSN05, XZVV06].

Practical parameterized SSTA algorithms are block-based in nature, i.e., they propagate the distributions of the delay from the primary inputs to the primary outputs of a circuit using a PERT-like (Program Evaluation and Review Technique) [KC66] traversal of the circuit graph. One of the exceptions is a path-based SSTA method proposed in [AMK+05]. In this work, the authors provide a simple procedure to perform statistical

timing analysis using a path-based scheme, as a post-processing step, after identifying a sufficiently large number of critical paths by a deterministic STA. The parameterized block-based SSTA algorithms [CS03, VRK+04, LLP04, ZSLP05, ZCH+05] provide efficient methods for performing statistical timing analysis, under the assumption of normality of parameter distributions. In [CS03], a novel SSTA procedure is proposed by approximating all delay and arrival time random variables as linear functions of correlated parameters. By assuming that the random vector, comprising of the parameters of variations, has all its components following a Gaussian distribution, a principal component analysis (PCA) transformation techniques is employed to generate another random vector comprising of components which are statistically independent Gaussian random variables. A similar work [VRK+04] assumes Gaussian modeling of parameters and linear delay representation to perform efficient SSTA. Both these works, [CS03] and [VRK+04], use Clark's closed-form formulae [Cla61] to approximate the maximum of two Gaussian random variables as another Gaussian random variable. The authors of [LLP04] also propose a linear Gaussian SSTA procedure by simplifying the computations involving a set of correlated normal variables, using the PCA method. The algorithms presented in [ZSLP05] and [ZCH+05] provide techniques for performing SSTA using quadratic delay models of Gaussian parameters.

For all of the abovementioned Gaussian SSTA algorithms, the assumption of normality of process variations lends itself rather well for generating closed-form expressions for the delay and arrival time PDFs. Although correlation and statistical dependence between random variables tends to increase the complexity of SSTA, recent work has presented efficient techniques for handling such correlations under Gaussian distributions, using PCA to perform a simple variable transformation. This transformation enables efficient SSTA, representing delays and arrival times as functions of a new set of orthogonal, statistically independent Gaussian random variables.

However, the normality assumption is not always valid [DDK05], and it is well known that some process parameters deviate significantly from a Gaussian distribution.

For example, via resistances exhibit an asymmetric probability distribution [CZNV05], and the dopant concentration density is also observed to be well modeled by a Poisson distribution: a normality assumption may lead to significant sources of errors in SSTA. Some recent works [CZNV05, KS05] propose SSTA methods that do away with the assumptions of normality for the parameter distributions, but to the best of our knowledge, no prior approach is scalable to handle large number of non-Gaussian parameters, or has presented an efficient SSTA solution under correlated non-Gaussian parameter distributions. In [CZNV05], the solution to tackle uncorrelated non-Gaussian parameters employs a numerical integration technique. However, the method of numerical integration in higher dimensions has an exponential computational complexity with respect to the number of non-Gaussian parameters. Thus, the method can efficiently handle only a few non-Gaussian sources of variation, and the runtime does not scale well with the number of such sources. The SSTA framework of [KS05] is general enough to consider both Gaussian and non-Gaussian parameters of variations, as long as the non-Gaussian parameters are uncorrelated. However, the technique relies on a regression strategy that requires a Monte Carlo simulation in the inner loop of the SSTA procedure. Such a technique is unlikely to scale well for large circuits with numerous sources of variations.

From the discussion of the existing SSTA methods in this section, the procedures can be broadly classified into the following four categories:

1. *Linear, Gaussian SSTA:* These methods employ a linear delay representation and assume normality of parameter distributions. Some examples of the techniques that offer an efficient and an accurate solution within this class of SSTA algorithms are [CS03, VRK+04, LLP04, AMK+05].

2. *Nonlinear, Gaussian SSTA:* These SSTA algorithms use a nonlinear delay model, in particular, a quadratic representation of all gate delay and arrival time variables, but still assume that all parameters as Gaussians. The works of [ZSLP05] and [ZCH+05] fall into this category of SSTA methods.

3. *Linear, non-Gaussian SSTA:* This class of SSTA procedures consists of techniques that do away with the Gaussian assumption for all parameters, but still employ a first order delay model. Our SSTA method, presented in this paper, is the only efficient and scalable known work for this class of algorithms.

4. *Nonlinear, non-Gaussian SSTA:* These SSTA methods are a superset of the other three classes, and cover the most general case for performing statistical timing analysis. Such SSTA procedures not only use a general nonlinear delay model, they also allow the parameters to be non-normally distributed. The methods of [CZNV05] and [KS05] are two such examples of these general SSTA algorithms. However, as mentioned before, these works rely on computationally expensive techniques, and are not scalable to a large number of variables. In fact, even the application of these methods to a simpler case of linear representation (the subset of class 3 SSTA methods, as described above) is just as inefficient. Thus, the quest for an efficient SSTA technique for a nonlinear delay form that includes non-Gaussian parameters of distribution, remains an unsolved research problem.

Most SSTA methods focus only on evaluating the delay and arrival time distributions, and do not usually include input signal transition time information in their procedures. However, in general, it is possible to extend these algorithms to incorporate slew distributions, as suggested in [CS05].

4.3 Outline of the SSTA Procedure

The main steps in our SSTA algorithm are:

1. **Preprocessing to obtain an independent set of basis variables:** We employ a technique known as independent component analysis (ICA) [Bel, HO99, HO00, MP99] as a *preprocessing step*, with the goal of transforming the random vector of correlated non-Gaussian components to a random vector whose components

are statistically independent. We then compute moments of the independent components from the moments of the non-Gaussian parameters. We orthogonalize the Gaussian parameters separately, performing PCA as in [CS03]. Together, we refer to this set of independent variables as the *basis set*.

2. **Moment matching-based PDF evaluation:** Next, we represent the gate delays as a linear canonical function of the basis set. From the moments of the basis set, we compute the moments of the gate delay variables. Finally, we translate the moments into an approximating PDF for the delay variables, using a Padé approximation-based moment matching scheme, as proposed in [LLGP04].

3. **Correlation-preserving statistical operations:** We process the circuit in a block-based manner, in topological order, computing the statistical sum and max operations at every step to compute the extracted PDFs of the arrival time variables. These variables are stored in terms of the linear canonical form through a moment-matching procedure.

To the best of our knowledge, this is the only work that can handle a large number of Gaussian and non-Gaussian process parameters with correlations. The correlations are described using a grid structure, similar to that used in [CS03], which handles Gaussian distributions only. For a circuit with $|G|$ gates and a layout with g spatial correlation grids, the complexity of our approach is $O(g|G|)$, similar to the Gaussian case in [CS03].

In our implementation, we consider the effective channel length, L_e, the transistor width W, and the dopant concentration, N_d as the sources of variation. The parameters L_e and W are modeled as correlated sources of variations, and the dopant concentration, N_d, is modeled as an independent source of variation. The same framework can be easily extended to include other parameters of variations.

Interval i ($[lb, ub)$ nm)	$Pr(\hat{L_e} = \frac{L_e - \mu_{L_e}}{\sigma_{L_e}}) \in i)$
[-4.0.-3.0)	0.000
[-3.0,-2.6)	0.000
[-2.6,-2.3)	0.006
[-2.3,-1.9)	0.022
[-1.9,-1.5)	0.072
[-1.5,-1.2)	0.092
[-1.2,-0.8)	0.128
[-0.8,-0.4)	0.106
[-0.4,-0.1)	0.120
[-0.1,0.3)	0.108
[0.3,0.7)	0.122
[0.7,1.1)	0.110
[1.1,1.4)	0.070
[1.4,1.8)	0.036
[1.8,2.2)	0.002
[2.2,2.5)	0.002
[2.5,2.9)	0.004
[2.9,3.3)	0.000
[3.3,3.6)	0.000
[3.6,4.0)	0.000

Table 4.1: A cumulative frequency table for 500 randomly generated values of L_e with $\mu_{L_e} = 65\ nm$ and $\sigma_{L_e} = 5.2\ nm$.

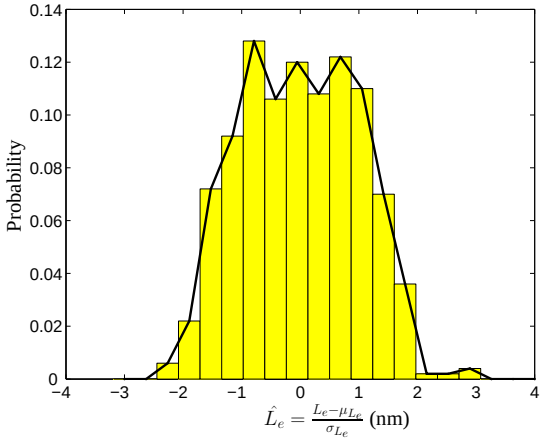


Figure 4.1: A frequency histogram of the $\hat{L_e}$ values listed in Table 4.1.

4.4 Generating Moments from Process Data

It is important to note that our algorithm requires minimal input information: rather than relying on closed-form distribution of variational parameters, the knowledge of their moments is sufficient for our scheme to generate the circuit delay distribution. This is a desirable property for an SSTA method, as it is typically difficult to extract precise distributions from process data, and it is more realistic to obtain the moments of the parameter variations from a process engineer. For instance, given the measurements of

a particular parameter X across N chips[1], k^{th} moment of X, denoted by $m_k(x)$, where x represents a sample point, can be easily computed as $m_k(x) = \sum_x x^k Pr(X = x)$. The probability $Pr(X = x)$ can be calculated by binning[2] all the measured values of X in some small discrete intervals $[lb, ub)$, and then dividing the frequency of values in each bin by the total number of samples N. This process is much easier than trying to fit an accurate closed-form PDF expression for the measured values of parameter X across all N sample points, given by the value of X in each of the N chips.

k	$m_k(\hat{L_e})$
1	0.0000
2	1.0000
3	0.0384
4	2.2733
5	0.6174
6	7.9839
7	6.2385
8	39.3220
9	56.2410
10	245.0898
11	485.8118
12	1.7515×10^3
13	4.1178×10^3
14	1.3447×10^4
15	3.4592×10^4
16	1.0711×10^5
17	2.8938×10^5
18	8.7014×10^5
19	2.4167×10^6
20	7.1479×10^6

Table 4.2: A table showing the first twenty moments of $\hat{L_e}$ values listed in Table 4.1.

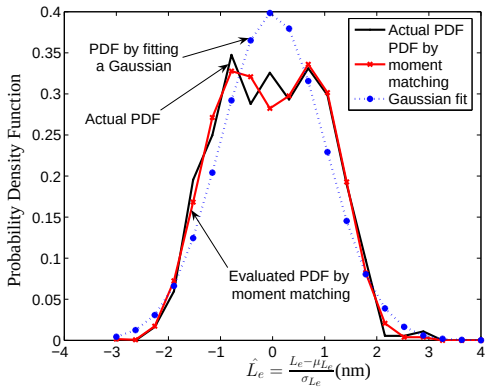


Figure 4.2: PDF of $\hat{L_e}$ values listed in Table 4.1.

[1]For simplicity, we ignore the intra-die variation for parameter X in this discussion.

[2]Binning sample points in intervals simplifies the computation by reducing the dimensionality of total number of sample points. Alternatively, it is also possible to use the raw process data to compute the moments by assigning a discrete probability, $Pr(X = x)$, to each sample point.

To understand the moment generation process, consider values of the effective channel length (L_e) as shown in Table 4.1. The table contains 500 randomly generated values of L_e with a mean of $65\ nm$, and a standard deviation of $5.2\ nm$. These L_e values can be thought of as measurements across $N = 500$ chips, similar to the ones expected to be extracted from the real wafer data. Table 4.1 is the cumulative frequency table for the zero-mean, unit-variance variable $\hat{L_e}$ values, derived by subtracting from L_e values, the sample mean (μ_{L_e}), and scaling the result by the reciprocal of the sample standard deviation (σ_{L_e}). The probabilities of occurrence of the random variable $\hat{L_e}$ in each discrete interval or bin in the range $[-4, 4]$, shown in column one of Table 4.1, is computed by simply dividing the frequency of $\hat{L_e}$ in the particular bin by the total number of measured points, in this case $N = 500$. Figure 4.1, depicts the frequency histogram of the $\hat{L_e}$ values listed in Table 4.1. The solid dark line in Figure 4.1, corresponds to the PDF[3] of $\hat{L_e}$. As seen in the figure, it is extremely difficult to fit a closed-form expression that would closely match this PDF.

However, the moments of the $\hat{L_e}$ values can be easily computed by using the relation, $m_k(\hat{l_e}) = \sum_{\hat{l_e}} \hat{l_e}^k Pr(\hat{L_e} = \hat{l_e})$, where the values of $Pr(\hat{L_e} = \hat{l_e})$ are shown in the second column of Table 4.1. The first twenty such moments are listed in Table 4.4. The only inputs required by our SSTA procedure are these moments of the varying parameters. As will be explained in Section 4.9, using the moments as input, the moment matching-based PDF evaluation method can generate closed-form PDF expressions. Figure 4.2 shows the actual PDF of $\hat{L_e}$, the PDF corresponding to fitting a Gaussian distribution to the data of Table 4.1, and the PDF obtained by using the moment matching-based PDF evaluation scheme. As seen from the figure, using the moments information, it is possible to derive the PDF of $\hat{L_e}$ that matches closely with the actual PDF.

[3]It is trivial to derive the PDF of L_e from the PDF of $\hat{L_e}$, as will be discussed in Section 4.9.

4.5 Non-Gaussianity in SSTA

The circuit delay distribution depends on a number of parameters such as the effective channel length, transistor width, metal thickness, interlayer dielectric thickness, dopant density, and the oxide thickness. As pointed out in Section 4.1, not all parameters of variations can be accurately modeled by a normally distributed random variable. Moreover, these non-Gaussian parameters may be correlated to each other due to the effect of spatial proximity. As a result, the approximation of parameters as normal distributions, followed by performing a Gaussian SSTA, may lead to significant inaccuracies in the PDF and CDF of the circuit delay.

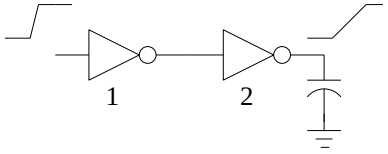


Figure 4.3: A simple circuit example to illustrate the effect of non-Gaussian parameters on the PDF of the circuit delay.

To illustrate the effect of such non-Gaussian parameters on the delay distribution, we use a toy circuit, shown in Figure 4.3. We assume W_i and L_{e_i} for each inverter i to be the random parameters of variation. Using a first order Taylor series approximation, the delay of this circuit can be written as:

$$D = \mu + a_1.W_1 + a_2.W_2 + b_1.L_{e_1} + b_2.L_{e_2} \tag{4.1}$$

where $a_1, a_2, b_1,$ and b_2 are the sensitivities of the delay with respect to the zero-mean randomly varying parameters W_1, W_2, L_{e_1}, and L_{e_2}, respectively, and μ is the nominal delay of the circuit. Next, we perform a simple Monte Carlo simulation to evaluate the PDF of the circuit by considering the following four scenarios:

Case 1: $\{W_1, W_2\}$ are modeled as uniformly distributed random variables in $[-\sqrt{3}\sigma_W, \sqrt{3}\sigma_W]$, and $\{L_{e_1}, L_{e_2}\}$ are assumed to be Gaussian random variables with a normal distribution

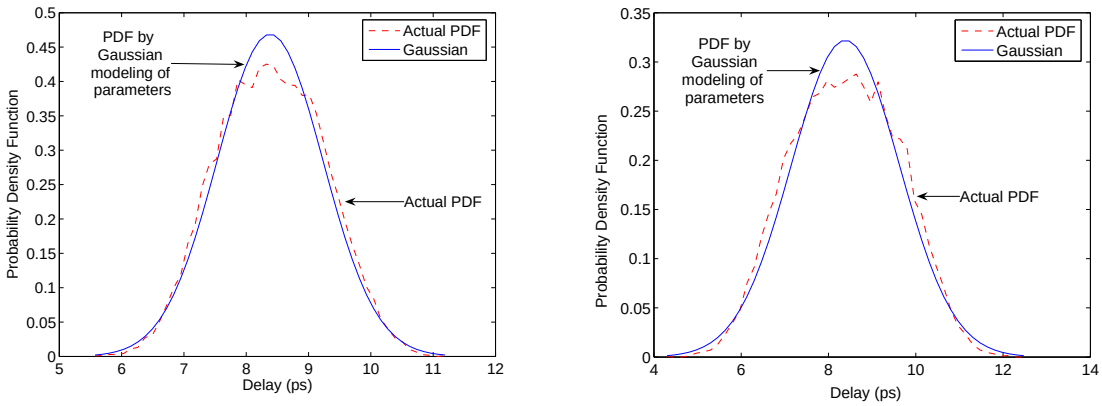
$N(0, \sigma_{L_e})$. Furthermore, all parameters are assumed to be statistically independent with respect to each other. Figure 4.4(a) illustrates the PDF of the circuit delay for this case.

Case 2: Employing the same model for the distributions of W and L_e parameters as above (Case 1), but assuming that W_1 is perfectly correlated with W_2, and L_{e_1} is perfectly correlated with L_{e_2}. The circuit delay PDF for this case is shown in Figure 4.4(b).

Case 3: $\{L_{e_1}, L_{e_2}\}$ are modeled as uniformly distributed random variables in $[-\sqrt{3}\sigma_{L_e}, \sqrt{3}\sigma_{L_e}]$, and $\{W_1, W_2\}$ are assumed to be Gaussian random variables with a normal distribution $N(0, \sigma_W)$. Furthermore, all parameters are assumed to be statistically independent with respect to each other. Figure 4.5(a) shows the PDF of the circuit delay for this case.

Case 4: Employing the same model for the distributions of W and L_e parameters as above (Case 3), but assuming that W_1 is perfectly correlated with W_2, and L_{e_1} is perfectly correlated with L_{e_2}. The circuit delay PDF for this case is illustrated in Figure 4.5(b).

The dashed curve in Figures 4.4 and 4.5, show the actual PDF of the circuit delay obtained by performing a Monte Carlo simulation, and correctly modeling W (for Cases 1 and 2) and L_e (for Cases 3 and 4) parameters, as uniformly distributed random variables, while the solid curve is the PDF obtained if the non-Gaussian variables were also modeled as Gaussian variables with the same mean and standard deviation as the uniformly distributed variables. Figures 4.4(a) and 4.5(a) show the PDFs for the cases where all of the parameters are considered to be statistically independent with respect to each other, while Figures 4.4(b) and 4.5(b) show the PDFs when W_1 is considered to perfectly correlated with W_2, and L_{e_1} is assumed to be perfectly correlated with L_{e_2}. In each case, it is seen that the circuit delay PDF deviates from a Gaussian distribution due to the presence of the non-Gaussian random variables. However, the deviation from a normal distribution is most significant in Figure 4.5(b). The following two reasons explain this significant non-Gaussian behavior of the circuit delay PDF:



(a) (b)

Figure 4.4: PDF of the delay of the example circuit of Figure 4.3, when $\{W_1, W_2\}$ are modeled as uniformly distributed, and $\{L_{e_1}, L_{e_2}\}$ are modeled as normally distributed random variables for (a) uncorrelated and (b) correlated W and L_e process variables.

1. The delay model used for the circuit of Figure 4.5 in these experiments, given by Equation (4.1), contains terms b_1 and b_2, corresponding to the sensitivities of L_{e_1} and L_{e_2}, that outweigh the terms a_1 and a_2, corresponding to the sensitivities of W_1 and W_2. In particular, $|b_1| = 5.2|a_1|$, and $|b_2| = 9.8|a_2|$. Therefore, for the experiments for Cases 1 and 2, corresponding to the PDF curves of Figures 4.4(a) and 4.4(b), the effect of the Gaussian parameters $\{L_{e_1}, L_{e_2}\}$ dominates the effect of the non-Gaussian parameters $\{W_1, W_2\}$, and the circuit delay PDF does not significantly aberrate from a normal distribution.

   For the experiment for Case 4, corresponding to the PDF curve in Figure 4.5(b), $\{L_{e_1}, L_{e_2}\}$ are modeled as uniformly distributed variables, therefore in this case, the non-Gaussian parameters dominate the normally distributed $\{W_1, W_2\}$ parameters, and the circuit delay PDF shows significant divergence from a Gaussian one.

2. For both Cases 3 and 4, $\{L_{e_1}, L_{e_2}\}$ are modeled as non-Gaussian variables. How-

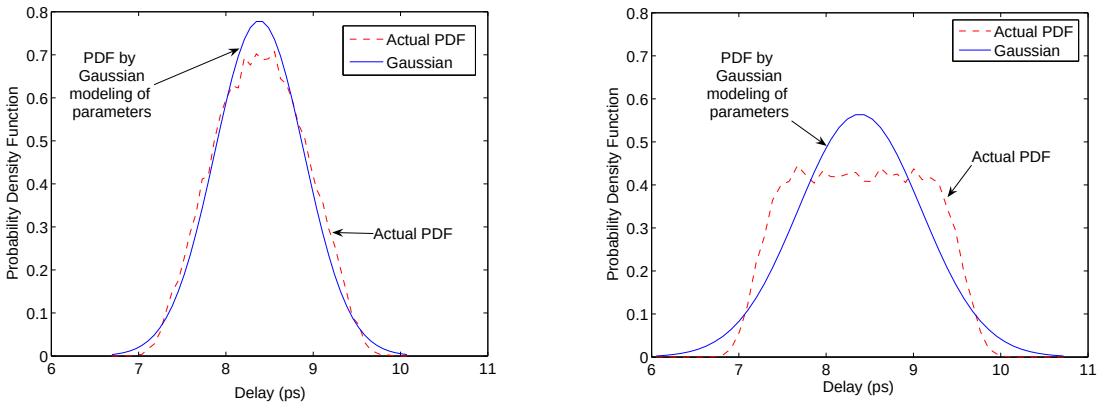


Figure 4.5: PDF of the delay of the example circuit of Figure 4.3, when $\{L_{e_1}, L_{e_2}\}$ are modeled as uniformly distributed, and $\{W_1, W_2\}$ are modeled as normally distributed random variables for (a) uncorrelated and (b) correlated W and L_e process variables.

ever the Monte Carlo PDF for Case 3, shown in Figure 4.5(a), assumes statistical independence of parameters. This PDF has a much closer match to a Gaussian distribution, compared to the one shown in Figure 4.5(b), that assumes perfect correlation between W_1 [L_{e_1}] and W_2 [L_{e_2}] parameters. The intuition for the significant change from a normal PDF, for the correlated case, can be arrived at by appealing to the Central Limit Theorem, according to which the addition of independent variables makes them "more Gaussian," but this is not necessarily true for correlated random variables.

For real circuits, where many parameters are correlated due to the presence of the inherent spatial and structural correlations, the presence of non-Gaussian parameters, the sensitivities of which could potentially outweigh the Gaussian ones, implies that the circuit delay may deviate significantly from a normal distribution.

4.6 Delay Representation

To incorporate the effects of both Gaussian and non-Gaussian parameters of distribution in our SSTA framework, we represent all delay and arrival times in a linear form as:

$$D = \mu + \sum_{i=1}^{n} b_i.x_i + \sum_{j=1}^{m} c_j.y_j + e.z = \mu + \mathbf{B}^T\mathbf{X} + \mathbf{C}^T\mathbf{Y} + e.z \tag{4.2}$$

where D is the random variable corresponding to a gate delay or an arrival time at the input port of a gate, x_i is a non-Gaussian random variable corresponding to a physical parameter variation, b_i is the first order sensitivity of the delay with respect to the i^{th} non-Gaussian parameter, y_j is a parameter variation modeled as a Gaussian random variable, c_j is the linear sensitivity with respect to the j^{th} Gaussian parameter, z is the uncorrelated parameter which may be a Gaussian or a non-Gaussian random variable, e is the sensitivity with respect the uncorrelated variable, n is the number of correlated non-Gaussian variables, and m is the number of correlated Gaussian variables. In the vector form, $\mathbf{B}$ and $\mathbf{C}$ are the sensitivity vectors for $\mathbf{X}$, the random vector of non-Gaussian parameter variations, and $\mathbf{Y}$, the random vector of Gaussian random variables, respectively. Note that we assume statistical independence between the Gaussian and non-Gaussian parameters: this is a reasonable assumption as parameters with dissimilar distributions are likely to represent different types of variables, and are unlikely to be correlated.

The value of the mean delay μ is adjusted so that the random vectors $\mathbf{X}$ and $\mathbf{Y}$ are centered, i.e., each component x_i and y_i is a zero-mean random variable. The uncorrelated random variable z is also centered. Note that in the representation of Equation (4.2), the random variables x_i are correlated with each other and may be of any underlying non-Gaussian distribution. Unlike the delay models of [VRK+04, CS03], we do not constraint the parameter distributions to be Gaussian. The canonical model of equation (4.2) is similar to the model of [CZNV05] without the nonlinear terms. The slight difference is that the uncorrelated parameter z is not constrained to be a Gaussian

variable.

4.7 Independent Component Analysis

For reasons of computational and conceptual simplicity, it is useful to work with a set of statistically independent random variables in the SSTA framework. If the components of random vector $\mathbf{X}$ were correlated Gaussian random variables with a covariance matrix $\sum$, a PCA transformation $\mathbf{R} = P_x\mathbf{X}$ would yield a random vector $\mathbf{R}$ comprising of Gaussian uncorrelated random variables [CS03]. Since for a Gaussian distribution, uncorrelatedness implies statistical independence[4], the components of $\mathbf{R}$ are also statistically independent.

However, such a property does not hold for general non-Gaussian distributions. In Equation (4.2), the random vector $\mathbf{X}$ consists of correlated non-Gaussian random variables, and a PCA transformation, $\mathbf{S} = P_x\mathbf{X}$, would not guarantee statistical independence for the components of the transformed vector $\mathbf{S}$. Since the PCA technique focuses only on second order statistics, it can only ensure uncorrelatedness, and not the much stronger requirement of statistical independence.

Independent component analysis [Bel, HO99, HO00, MP99] is a mathematical technique that precisely accomplishes the desired goal of transforming a set of non-Gaussian correlated random variables to a set of random variables that are statistically as independent as possible, via a linear transformation. ICA has been an active area of research in the area of signal processing, feature extraction and neural networks due to its ability to capture the essential structure of data in many applications.

[4]Two random variables X and Y are uncorrelated if $E[XY] = E[X]E[Y]$, while they are independent if $E[f(X)g(Y)] = E[f(X)]E[g(Y)]$ for any functions f and g. For instance, if X and Y are independent, then $E[X^iY^j] = E[X^i]E[Y^j]$. For Gaussian distributions, uncorrelatedness is identical to independence. For a general non-Gaussian distribution, independence implies uncorrelatedness, but not vice versa.

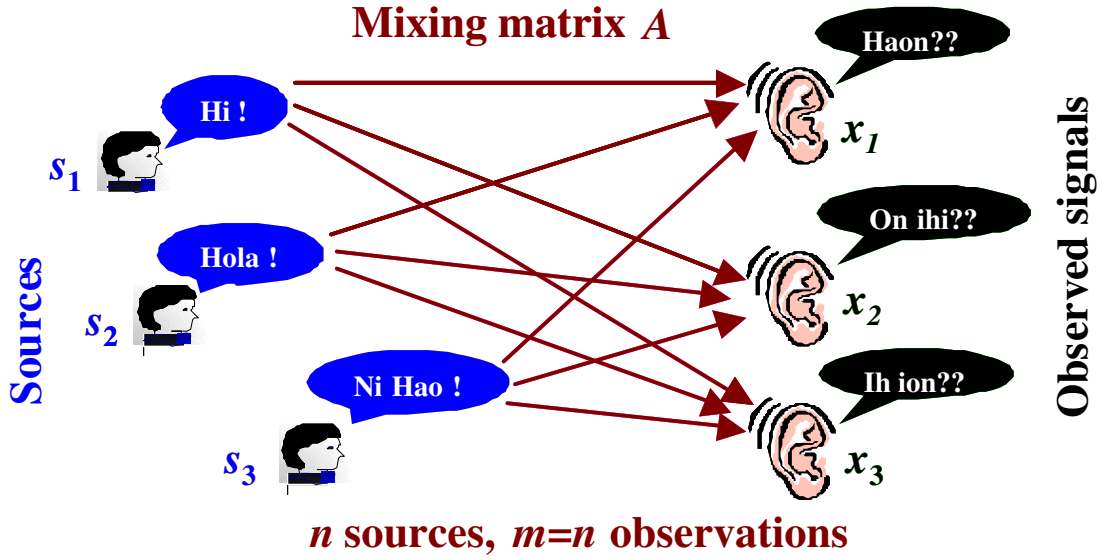


Figure 4.6: The cocktail party problem to illustrate the independent component analysis set up.

4.7.1 The Cocktail Party Problem

The ICA principle can be explained by the *cocktail party problem* example illustrated in Figure 4.6. The set up shown in the figure, consists of n speakers, who can be regarded as independent sources, and n receivers, represented by the ears in Figure 4.6. The speakers or the independent sources emit independent speech signals, but their simultaneous speech results in interferences of the independent signals. As shown in Figure 4.6, due to the interference or mixing of the independent speech signals, the signals observed by the receivers are no longer independent. The amount of mixing of the independent speech signals may be derived form elements of a mixing matrix A, which could depend on metrics such as the distance of each speaker from the receiver. Mapping the cocktail party problem set up back to the ICA problem, the ICA set up consists of having a vector **S** consisting of n statistically independent components, $s_1, \cdots, s_n$, and observations of n linear mixtures, $x_1, \cdots, x_n$, of the n independent components.

The observed components can be thought of as the correlated non-Gaussian random variables $\mathbf{X}$ in Equation (4.2), produced by a linear mixing of the elements of a vector $\mathbf{S}$ of independent random variables, as follows:

$$\mathbf{X} = A\mathbf{S} \tag{4.3}$$

where A is the $n \times n$ *mixing matrix*.

The problem of ICA is to estimate the elements of the unknown mixing matrix A, and the samples of statistically independent components $s_1, \cdots, s_n$, as accurately as possible, given only the samples of the observed vector $\mathbf{X}$. Equation (4.3) can be alternatively written as:

$$\begin{aligned} \mathbf{S} &= W\mathbf{X} \text{ where} \\ s_i &= \mathbf{W_i^T X} = \textstyle\sum_{j=1}^{n} w_{ij} x_i \;\; \forall i = 1, \cdots, n \end{aligned} \tag{4.4}$$

In the above equation, W is the inverse of the unknown mixing matrix A. Algorithms for ICA estimate the vectors $\mathbf{W_i}$ that maximize the non-Gaussianity of $\mathbf{W_i^T X}$ by solving a nonlinear optimization problem. Typical measures of non-Gaussianity are kurtosis, negentropy, and mutual information; for a comprehensive reference on ICA, see [Bel, HO99, HO00, MP99].

For our SSTA algorithm, we use ICA as a preprocessing step to transform the correlated set of non-Gaussian random variables $x_i, \cdots, x_n$ to a set of statistically independent variables $s_i, \cdots, s_n$, by the relation $\mathbf{S} = W\mathbf{X}$ of Equation (4.4). In practice, ICA estimates the mixing matrix A and its inverse matrix W, which yield the components, $s_i, \cdots, s_n$, which are statistically as independent as possible. For the purposes of application of ICA transformation in our SSTA algorithm, we will consider the vector $\mathbf{S}$ to consist of truly statistically independent components. Experimental results, presented in Section 4.12, validate this assumption.

Like principal components, the independent components of vector $\mathbf{S}$ are mathematical abstractions that cannot be directly observed. Similar to the PCA procedure, which

requires normalization of N(μ, σ) variables to N(0,1) variables, the ICA methods also require centering and whitening of the components of vector $\mathbf{X}$, i.e., prescaling the variables to have zero mean and unit variance [HO00]. For a specific grid, the independent components of the non-Gaussian random variables must be computed just once, and this can be carried out as a precharacterization step. In other words, ICA need not be recomputed for different circuits or different placements of a circuit. *Thus, the ICA preprocessing step does not impact the runtime of the SSTA procedure.*

One of the requirements of the ICA technique is that all of the original source of independent sources, $s_1, \cdots, s_n$, should be non-Gaussian. Therefore, in the delay model of Equation (4.2), we must treat the correlated non-Gaussian random variables $\mathbf{X}$, and the correlated Gaussian random variables $\mathbf{Y}$, separately. The ICA technique is applied to non-Gaussian parameters $\mathbf{X}$, and a PCA transformation is applied to Gaussian variables $\mathbf{Y}$, to obtain a set of statistically independent non-Gaussian variables $\mathbf{S}$, and a set of independent Gaussian variables $\mathbf{R}$. We then substitute the respective transformation matrices A and P_y in Equation (4.2) to arrive at the following *canonical delay model*:

$$\begin{aligned} D &= \mu + \mathbf{B'^T S} + \mathbf{C'^T R} + e.z \\ &= \mu + \sum_{i=1}^{n} b'_i.s_i + \sum_{j=1}^{m} c'_j.r_j + e.z \end{aligned} \tag{4.5}$$

where $\mathbf{B'^T} = \mathbf{B^T}A$, [$\mathbf{C'^T} = \mathbf{C^T}P_y^{-1}$] is the new sensitivity vector with respect to the statistically independent non-Gaussian components, $s_1, \cdots, s_n$ [Gaussian principal components $r_1, \cdots, r_m$].

4.7.2 Generating Samples of Correlated Non-Gaussian Variables

The ICA method requires, as inputs, the samples of the correlated non-Gaussian parameters. If these samples are readily available from the process data, they can be directly provided to the ICA module to generate the estimates of the mixing matrix A, and the samples of the independent components, $s_1, \cdots, s_n$. However, if instead of the

Algorithm 2 Generate Correlated Non-Gaussian Samples

1: /*Inputs: Correlation matrix Q ($n \times n$), mean vector $\mu_{\mathbf{X}}$ ($n \times 1$), CDF of x_j parameter $F_j(x_j), \forall j = 1, \cdots, n$*/

2: /*Output: Matrix $Corr(NUM\_SAMPLES \times n)$ as samples of correlated non-Gaussian variables*/

3: /*Step1 : Generate samples of multivariate normal distribution $N(\mu, Q)$*/

4: i=1;

5: **while** $(i < NUM\_SAMPLES)$ **do**

6: $Z(i)$=**mvnrnd**(μ, Q);

7: i=i+1;

8: **end while**

9: /*Step2: Map the multivariate normal samples to a multivariate uniform samples in [0,1]*/

10: U=**normcdf**(Z);

11: /*Step3: Apply inverse CDF transformation to samples in each column of U*/

12: j=1;

13: **while** $(j < n)$ **do**

14: $Corr(j)$=$\mathbf{F_j^{-1}}(U)$;

15: j=j+1;

16: **end while**

samples of correlated parameters, the closed-form PDFs of the non-Gaussian sources of variation are provided, we must first generate samples of the parameters from the given PDF expressions[5]. To model the correlation between the non-normal parameters, $x_1, \cdots, x_n$, the chip area is first tiled into a grid, as in [CS03], and the correlation matrix, Q, associated with $\mathbf{X}$ is determined. The matrix Q and the mean vector $\mu_{\mathbf{X}}$ is used to generate the samples of the correlated non-Gaussian variables by employing the method of normal copulas [Sim]. Algorithm 2 shows the pseudo-code of this method, which is based on performing a series of correlation preserving transforms on a set of random numbers.

The procedure consists of three main steps. In the first step, spanning lines 4–8, samples from a multivariate normal distribution, $N(\mu_{\mathbf{X}}, Q)$, are generated. As will become clear in the next steps, these set of Gaussian random numbers are used to generate the required non-normal numbers having a mean vector $\mu_{\mathbf{x}}$, and the correlation matrix Q. The function call **mvnrnd** generates these samples. In the next step, shown on line 10, the normal samples are mapped to a multivariate uniform distribution in the range [0,1]. The transformation function **normcdf** is simply the CDF of the standard normal distribution. The following relations prove that for a single standard normal random variable y, with a CDF denoted by $F_y(y)$, a transformation $u = F_y(y)$ results in a uniformly distributed variable u in the range [0,1].

$$F_u(u_0) = Pr(u \leq u_0) = Pr(F_y(y) \leq u_o) = Pr(y \leq F_y^{-1}(u_0)) = F_y(F_y^{-1}(u_0)) = u_0 \quad (4.6)$$

Thus, the CDF of u is $F_u(u_0) = u_0$, which is same as the CDF of a uniformly distributed random variable in the range [0,1]. In our case, Z comprises of samples of multivariate normal distribution. Thus, each component of random vector associated with Z, has a marginal distribution of a standard normal. Therefore, the function mapping $U = \mathbf{normcdf}(Z)$, maps each normally distributed component of the random vector

[5]As will be explained in Section 4.12, we use the method of generating correlated non-Gaussian random numbers, described in this section, for our experimental set up that assumes, as inputs, well-known closed-form PDFs for parameters $x_1, \cdots, x_n$.

associated with Z, into a uniformly distributed variable in the range [0,1]. The statistical dependence between the generated samples still remains after the transformation. The subroutines for generating samples of multivariate normal distribution (**mvnrnd( )**), and the CDF of normal distribution (**normcdf( )**) are commonly available in standard mathematical software packages, such as [MRM] and [MRG].

The last step in Algorithm 2, shown in lines 12–16, consists of transforming the multivariate uniform samples in U to the individual non-Gaussian marginal distributions. The transformation function is F_j^{-1}, which is the inverse of the CDF of the j^{th} non-Gaussian random variable. For example, if the j^{th} non-Gaussian parameter x_j is uniformly distributed in the range $[lb, ub]$, $F_j^{-1}(x) = lb + (ub - lb)x$. It is easy to prove that mapping uniformly distributed random numbers on interval [0,1], by a function which is an inverse CDF $F^{-1}(x)$ of a particular distribution, produces random numbers which have a distribution as given by CDF $F(x)$ [DS02]. Since samples in each column of the matrix U, are mapped by the required inverse CDF function F_j^{-1}, the correlation structure between the columns of U is preserved after the transformation. The output of the algorithm produces a matrix $Corr$, with $NUM\_SAMPLES$ rows and n columns. Each column of this matrix contains samples of a non-Gaussian parameter drawn from the required distribution. The columns are correlated with each other according to the original linear correlation matrix Q, and their sample mean is the same as the original mean vector $\mu_{\mathbf{X}}$.

Following the steps described in Algorithm 2, we generate samples of correlated non-Gaussian parameters. These samples are required as input to the ICA methods, which generate the ICA transformation matrix A in Equation (4.3).

4.8 Preprocessing to Evaluate the Moments of the Independent Components

The inputs required for our SSTA technique correspond to the moments of parameters of variation. Consider a process parameter represented by a random variable x_i: let us denote its k^{th} moment by $m_k(x_i) = E[x_i^k]$. We consider three possible cases:

Case I: If the closed-form of the distribution of x_i is available, and it is of a standard form (e.g., Poisson or uniform), then $m_k(x_i)\ \forall\ k$ can be derived from the standard mathematical tables of these distributions.

Case II: If the distribution is not in a standard form, then $m_k(x_i)\ \forall\ k$ may be derived from the moment generating function (MGF), if a continuous closed-form PDF of the parameter is known. If the PDF of x_i is the function $f_{x_i}(x_i)$, then its moment generating function $M(t)$ is given by

$$M(t) \quad = \quad E[e^{tx_i}] = \int_{-\infty}^{\infty} e^{tx_i} f_{x_i}(x_i) dx_i \tag{4.7}$$

The k^{th} moment of x_i can then be calculated as the k^{th} order derivative of $M(t)$ with respect to t, evaluated at $t = 0$. Thus, $m_k(x_i) = \frac{d^k M(t)}{dt^k}$ at $t = 0$.

Case III: If a continuous closed-form PDF cannot be determined for a parameter, the moments can still be evaluated from the process data files as:

$$m_k(x_i) \quad = \quad \sum_x x^k Pr(X_i = x) \tag{4.8}$$

where $Pr(x_i = x)$ is the probability that the parameter x_i assumes a value x. This moment generation process is explained in Section 4.4.

Given the underlying process variables and their moments, the next step after performing ICA is to determine the moments of the independent components, $s_i, \cdots, s_n$, from the moments of the correlated non-Gaussian parameters $x_i, \cdots, x_n$. The moments of the parameters, $E[x_i^k]$, are the inputs to the SSTA algorithm.

We now refer back to the ICA transformation of Equation (4.3), $\mathbf{X} = A\mathbf{S}$ and rewrite the relationship by taking the expectation of both sides as:

$$\begin{aligned} E[x_1^k] &= E[(a_{11}s_1 + a_{12}s_2 + \cdots a_{1n}s_n)^k] \\ E[x_2^k] &= E[(a_{21}s_1 + a_{22}s_2 + \cdots a_{2n}s_n)^k] \\ \vdots &\quad \vdots \\ E[x_n^k] &= E[(a_{n1}s_1 + a_{n2}s_2 + \cdots a_{nn}s_n)^k] \end{aligned} \tag{4.9}$$

where a_{ij} is an element of the mixing matrix A obtained via ICA. In the above equation, the left hand side, which is the k^{th} moment of each component of $\mathbf{X}$, is known. The right hand side can be simplified by performing an efficient multinomial expansion using the idea of binomial moment evaluation presented in [LLGP04]. The moments are computed successively, starting from the first to the second to the third, and so on. For example, after all of the first moments have been computed, the second moment of each s_i can be computing by rewriting Equation (4.9) using $k = 2$ as

$$\begin{aligned} E[x_1^2] &= \sum_{i=1}^{n} a_{1i}^2 E[s_i^2] + 2\sum_{i=1}^{n}\sum_{j=i+1}^{n} a_{1i}a_{1j}E[s_i]E[s_j] \\ E[x_2^2] &= \sum_{i=1}^{n} a_{2i}^2 E[s_i^2] + 2\sum_{i=1}^{n}\sum_{j=i+1}^{n} a_{2i}a_{2j}E[s_i]E[s_j] \\ \vdots &\quad \vdots \\ E[x_n^2] &= \sum_{i=1}^{n} a_{ni}^2 E[s_i^2] + 2\sum_{i=1}^{n}\sum_{j=i+1}^{n} a_{ni}a_{nj}E[s_i]E[s_j] \end{aligned} \tag{4.10}$$

The only unknowns in the above equation are the second moments, $E[s_i^2]$, of each s_i, and these can be calculated easily.

In general, while solving for the k^{th} moment of s_i using Equation (4.9), all of the $(k-1)$ moments are known from previous computations. Moreover, since the components of $\mathbf{S}$ are independent, we can perform the operation $E[s_i^a s_j^b] = E[s_i^a]E[s_j^b]$, and efficiently apply the binomial moment evaluation scheme. As indicated by Equation (4.10), the computation of the k^{th} moment of the independent components, $s_i, \cdots, s_n$, requires the

solution of an $n \times n$ system of linear equations. Thus, to compute $2M$ moments of the independent components, we must solve $2M$ systems of linear equations corresponding to (4.9) for $k = 1, \cdots, 2M$. However, since this is a part of the preprocessing phase, it may be carried out off-line for a specific technology, and it does not contribute to the complexity of the SSTA algorithm.

Note that while ICA does provide the W matrix, it is not easily possible to use $\mathbf{S} = W\mathbf{X}$ to find the moments of the s_i variables. This is because the binomial moment evaluation procedure requires the random variables to be statistically independent, which is true for the s_i variables but not the x_i variables.

4.9 Moment-Matching-based PDF Extraction

To compute the PDF/CDF of the delay or arrival time random variable we adapt the probability extraction scheme, *APEX*, proposed in [LLGP04]. Given $2M$ moments of a random variable as inputs to the *APEX* algorithm, the scheme employs an asymptotic waveform evaluate (AWE) technique to match the $2M$ moments in order to generate an M^{th} order linear time invariant (LTI) system. The scheme then approximates the PDF [CDF] of a random variable by an impulse response $h(t)$ [step response $s(t)$] of the M^{th} order LTI system. The details of the *APEX* algorithm can be found in [LLGP04].

We return to the example of Figure 4.3 to explain moment matching-based PDF evaluation method. To compute the delay PDF for the example, we must first calculate $2M$ moments of D from Equation (4.1). Assuming (W_1, W_2) to be perfectly correlated identical Gaussian random variables, and (L_1, L_2) to be perfectly correlated, and uniformly distributed identical random variables (Case 4 of Section **??**), we have:

$$\hat{D} \;=\; a.W + b.L_e \tag{4.11}$$

where $\hat{D} = D - \mu$, $a = a_1 + a_2$ and $b = b_1 + b_2$. Assuming W and L_e as statistically independent variables, the k^{th} moment of $\hat{D}$ can be computed by using the binomial

expansion formula as:

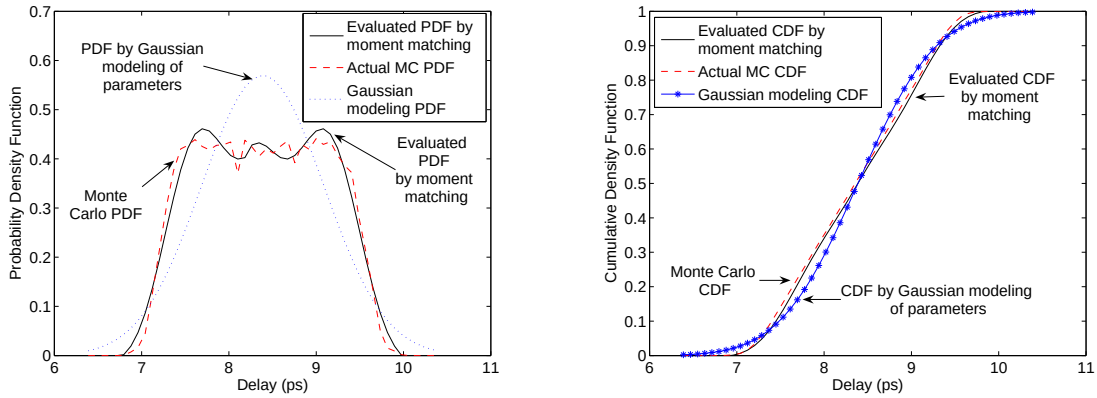
$$m_k[\hat{D}] = \sum_{i=0}^{k} \binom{k}{i} a^i b^{k-i} m_i(W) m_{k-i}(L_e) \tag{4.12}$$

where all of the k moments of W and L_e are known from the underlying normal and uniform distributions. Since the normal and uniform distributions used in this example are both well-studied, their moments can be obtained from mathematical tables. Having computed $2M$ moments of $\hat{D}$ from Equation (4.12), we can now employ the AWE-based PDF evaluation scheme to approximate the PDF and CDF of $\hat{D}$ by an impulse response as:

$$f_{\hat{D}}(\hat{d}) = \begin{cases} \sum_{i=1}^{M} \hat{r_i}.e^{\hat{p_i}.\hat{d}} & \hat{d} \geq 0 \\ 0 & \hat{d} < 0 \end{cases} \tag{4.13}$$

$$F_{\hat{D}}(\hat{d}) = \begin{cases} \sum_{i=1}^{M} \frac{\hat{r_i}}{\hat{p_i}}(e^{\hat{p_i}.\hat{d}} - 1) & \hat{d} \geq 0 \\ 0 & \hat{d} < 0 \end{cases} \tag{4.14}$$

where $\hat{r}$ [$\hat{p}$] are the residues [poles] of the LTI approximation.



(a) PDF (b) CDF

Figure 4.7: Extracted PDF and CDF for the delay of the example circuit.

Figure 4.7 shows the evaluated delay PDF ($f_D(d) = f_{\hat{D}}(d + \mu)$) and CDF ($F_D(d) =$

$F_{\hat{D}}(d+\mu)$) of the circuit of Figure 4.3 using $M = 10$ moments. The evaluated PDF matches closely with the Monte Carlo simulation; the match for the CDF is even better.

We can generalize the PDF evaluation idea, illustrated in the above example, to compute the PDF (CDF) of any random delay variable expressed in the canonical form of Equation (4.5). For such a delay variable with $l = m + n + 2$ terms, the binomial moment evaluation procedure can be employed to calculate the $2M$ moments, as long as all l variables in the delay expression are statistically independent. The canonical form expression of Equation (4.5) satisfies this independence requirement by construction.

We have enhanced the PDF evaluation algorithm in [LLGP04] for better numerical accuracy and stability. Instead of evaluating the PDF of a random variable D directly, we first prescale it by defining a new random variable $\hat{D} = \frac{D-\mu_D}{\sigma_D}$, and evaluate the PDF of $\hat{D}$. Without the prescaling step, the higher order moments of D can become extremely large (or extremely small) and affect the numerical accuracy of the moment computation. We compute the flipped PDF of $(-\hat{D})$, and reconstruct the final PDF from the flipped and the original PDF to avoid numerical errors due to the final value theorem, as in [LLGP04]. The PDF and CDF of D is retrieved from the PDF of $\hat{D}$ by using the relationship:

$$\begin{aligned} f_D(d) &= \frac{1}{\sigma_D} f_{\hat{D}}\left(\frac{d-\mu_D}{\sigma_D}\right) \\ F_D(d) &= F_{\hat{D}}\left(\frac{d-\mu_D}{\sigma_D}\right) \end{aligned} \tag{4.15}$$

In general, given the moments of the independent components, precharacterized as in Section 4.8, we can compute the moments of the delay and arrival time random variables from Equation (4.5). The moments of an $N(0,1)$ Gaussian distribution corresponding to each principal component, $r_1, \cdots, r_m$, are well known as:

$$m_k(r_i) = \begin{cases} 1 & k = 0 \\ 0 & k = 1, 3, 5, \cdots \\ 1 \cdot 3 \cdot 5 \cdots (k-1) & k = 2, 4, 6, \cdots \end{cases} \tag{4.16}$$

The moments of the uncorrelated process parameter z can be easily computed using the techniques in Section 4.8. As we will see in Section 4.10, during the SSTA propagation, the role of z in the canonical form is to serve as a place holder for the moments of the uncorrelated part, and these moments will be propagated further. For each gate, given the moments of all random variables $s_1, \cdots, s_n$, $r_1, \cdots, r_m$, and z, which are all statistically independent with respect to each other, we may use the binomial evaluation method to compute the $2M$ moments of the gate delay; a similar procedure will be used to compute the arrival times in the canonical form in Section 4.10.

4.10 SSTA Procedure

From the theory explained in the previous sections, we now have the ability to evaluate the PDF and the CDF of the delay and the arrival time random variables, expressed in the linear canonical form, as a function of Gaussian and non-Gaussian parameters of variation. In this section, we describe our SSTA framework. It is well known that the arrival time propagation procedure, operating in topological order on the circuit graph, involves the atomic operations of "sum" and "max." We will show how these atomic operations can be performed to produce a result that can be represented in the canonical form of Equation (4.5).

4.10.1 The "sum" Operation

The sum operation to add two arrival time or delay random variables, expressed in the linear canonical form of Equation (4.5), is mostly straightforward. Consider two random variables, D_1 and D_2 expressed as:

$$\begin{aligned} D_1 &= \mu_1 + \sum_{i=1}^{n} b'_{i_1}.s_i + \sum_{j=1}^{m} c'_{j_1}.r_j + e_1.z_1 \\ D_2 &= \mu_2 + \sum_{i=1}^{n} b'_{i_2}.s_i + \sum_{j=1}^{m} c'_{j_2}.r_j + e_2.z_2 \end{aligned} \tag{4.17}$$

The sum $D_3 = D_1 + D_2$ can be expressed in the same canonical form as:

$$D_3 \quad = \quad \mu_3 + \sum_{i=1}^{n} b'_{i_3}.s_i + \sum_{j=1}^{m} c'_{j_3}.r_j + e_3.z_3 \tag{4.18}$$

where $\mu_3 = \mu_1 + \mu_2$, $b'_{i_3} = b'_{i_1} + b'_{i_2}$, and $c'_{i_3} = c'_{i_1} + b'_{i_2}$.

The one difference here, as compared to the Gaussian case (e.g., in [CS03]), relates to the computation of the uncorrelated non-Gaussian parameter, $e_3.z_3$. The random variable $e_3.z_3 = e_1.z_1 + e_2.z_2$, serves as a place holder to store the moments of $(e_1.z_1 + e_2.z_2)$. In other words, rather than propagating an uncorrelated component z in the canonical form, we propagate its $2M$ moments.

4.10.2 The "max" Operation

The PDF of the maximum of the two *independent* random variables U and V, given by $T = \max(U, V)$, can be simply computed as:

$$f_T(t) = F_U(t)f_V(t) + F_V(t)f_u(t) \tag{4.19}$$

where f represents the PDF of each random variable, and F its CDF. If U, V are not only independent, but can also be expressed in the canonical form of Equation (4.5), then the PDF and CDF of T can be easily computed using the PDF evaluation technique described in Section 4.10, in a closed-form using Equation (4.19).

However, in general, two arrival time random variables A_1 and A_2, expressed in the canonical form of Equation (4.5), *do not* satisfy the independence requirement above, as they may both have nonzero coefficients associated with an s_i and/or an r_i variable. Fortunately, it is possible to work around this by using a simple technique that permits the application of Equation (4.19) to compute the PDF of random variable $A_{max} =$

$\max(A_1, A_2)$. Let us begin with the canonical expressions for A_1 and A_2:

$$\begin{aligned} A_1 &= \mu_1 + \sum_{i=1}^{n} b'_{i_1}.s_i + \sum_{j=1}^{m} c'_{j_1}.r_j + e_1.z_1 \\ A_2 &= \mu_2 + \sum_{i=1}^{n} b'_{i_2}.s_i + \sum_{j=1}^{m} c'_{j_2}.r_j + e_2.z_2 \end{aligned} \tag{4.20}$$

The operation $A_{max} = \max(A_1, A_2)$ can be now simplified as:

$$A_{max} = W + \max(U, V) \tag{4.21}$$

where

$$\begin{aligned} W &= b'_{1_2}.s_1 + c'_{1_2}.r_1 + \sum_{i=2}^{n} b'_{i_1}.s_i + \sum_{j=2}^{m} c'_{j_1}.r_j \\ U &= \mu_1 + (b'_{1_1} - b'_{1_2}).s_1 + (c'_{1_1} - c'_{1_2}).r_1 + e_1.z_1 \\ V &= \mu_2 + \sum_{i=2}^{n} (b'_{i_2} - b'_{i_1}).s_i + \sum_{j=2}^{m} (c'_{i_2} - c'_{i_1}).r_i + e_2.z_2 \end{aligned} \tag{4.22}$$

The above representation of the max operation ensures that the random variables U and V involved in the max operation, $\max(U, V)$, are statistically independent as they do not share any variables[6].

Therefore, from Equations (4.19) and (4.21), we can write $A_{max} = W + T$. Clearly, from Equation (4.22), W is available in the canonical form, and our next task is to express T in the form of Equation (4.5) as well, since this would permit us to write A_{max} in the canonical form.

To achieve this, we employ the idea of tightness probability [VRK+04], to express $T = \max(U, V)$ as:

$$T = \mu_T + \sum_{i=1}^{n} b'_{i_T}.s_i + \sum_{j=1}^{m} c'_{j_T}.r_j + e_T z_T \tag{4.23}$$

[6]Note that this is a sufficient condition for independence since all variables in the expressions of U and V, obtained from the ICA and the PCA transforms are statistically independent.

Our discussions in the previous sections provide us with all of the machinery required to efficiently compute the tightness probability, $p_{U>V} = Pr(U > V)$. We define a random variable $\hat{Q} = V - U$, and use the sum operation defined in Section 4.10.1 to express the random variable $\hat{Q}$ in the canonical form. Next, employing the technique described in Section 4.9, we compute the $2M$ moments of random variable $\hat{Q}$, and evaluate the CDF, $F_{\hat{Q}}(\hat{q})$, as a step response of the approximated LTI system using the following relationship:

$$\begin{aligned} F_{\hat{Q}}(\hat{q}) &= \sum_{i=1}^{M} \frac{\hat{r}_i}{\hat{p}_i}(e^{\hat{p}_i.\hat{q}} - 1) \quad (\hat{q} \geq 0) \\ &= 0 \qquad\qquad\qquad\qquad (\hat{q} < 0) \end{aligned} \tag{4.24}$$

where $\hat{r}$ and $\hat{p}$ are the residues and poles of the approximated M^{th} order LTI system. The tightness probability $p_{U>V}$ is simply given by the CDF of $\hat{Q}$ evaluated at $\hat{q} = 0$, since $Pr(U > V) = Pr(\hat{Q} \leq 0) = F_{\hat{Q}}(0)$.

Unlike [CZNV05], this method does not require the computationally expensive technique of numerical integration in high dimensions for non-Gaussian parameters. The ability to compute the tightness probability $p_{U>V}$ analytically, from the evaluated CDF of ($\hat{Q} = V - U$), makes the SSTA procedure very efficient and allows us to process a large number non-Gaussian variables.

Having computed the tightness probability, $p_{U>V}$, the sensitivities b'_{i_T}, c'_{i_T}, and z_T of $T = \max(U, V)$ in Equation (4.23) can be written in terms of the sensitivities of U and V. Specifically:

$$\begin{aligned} b'_{i_T} &= p_{U>V}.b'_{i_U} + (1 - p_{U>V}).b'_{i_U} \quad \forall i = 1, \cdots, n \\ c'_{j_T} &= p_{U>V}.c'_{j_V} + (1 - p_{U>V}).c'_{j_V} \quad \forall j = 1, \cdots, m \end{aligned} \tag{4.25}$$

Recall that the uncorrelated parameter term in Equation (4.23) is a place holder for the moments of the uncorrelated parameter: the moments of z_T can also be computed using the tightness probability: z_T assigned the moments of the random variable $(p_{U>V}.e_U.z_U +$

$(1 - p_{U>V}).e_V.z_V)$. The adjustment of the sensitivity term e_T will be explained later in this section.

The use of tightness probabilities is only a heuristic and suffers from problems of accuracy. Therefore, to reduce the error in the heuristic, we compute the mean μ_T in Equation (4.23) and the variance of T, σ_T^2, exactly from the PDF of T. In order to achieve this, we use Equation (4.19): note that this is applicable since U and V are independent by construction. Using the closed-form PDF, $f_T(t)$, we can compute μ_T from the first principles as $\mu_T = E[\max(U, V)] = \int_{-\infty}^{\infty} t f_T(t) dt$.

The last term left to compute is e_T, the coefficient term of the uncorrelated random variable z_T. We compute this term so that we match the variance of the closed-form PDF of T, $f_T(t)$, alluded to above, with the variance of canonical representation of Equation (4.23). The variance can be computed from $f_T(t)$ as:

$$\sigma_T^2 = \int_{-\infty}^{\infty} t^2 f_T(t) dt - (E[\max(U, V)])^2 \tag{4.26}$$

Having matched the variance term in Equation (4.26) to the variance in the expression Equation (4.23), all of the terms required to represent $T = \max(U, V)$ back to the canonical form are known. As a final step, referring back to Equation (4.21), we perform the sum operation between W and $T = \max(U, V)$ to complete the computation of $A_{max} = \max(A_1, A_2)$.

4.11 Time Complexity Analysis

The steps to generate the ICA mixing matrix A, the PCA transform, and the moments of the independent components $s_i, \cdots, s_n$ do not affect the online runtime of the procedure. These preprocessing steps have a one time precharacterization cost. Hence, the computational cost of the main steps in the SSTA procedure is comprised of the circuit graph traversal, and the sum and max operations.

The sum operation has a time complexity of $O(n + m)$, where n is the number

of non-Gaussian independent components and m is the number of Gaussian principal components.

The main steps in the max operation consists of computing moments of the delay variables, PDF evaluation by the AWE-based method, and calculating the mean and the variance terms to express the result of max operation back to a canonical form. The cost of computing $2M$ moments using the binomial moment evaluation procedure is $O(M(n+m))$. The PDF evaluation involves the solution of a linear $M \times M$ system of linear equations, described by a Hankel matrix, is $O(M^3)$; in practice, M is upper-bounded by a small constant, and excellent solution are obtained for $M \leq 10$. The mean and the variance terms are computed by one dimensional numerical integration and can be calculated in constant time. Thus, the complexity of the max operation is $O(m+n)$. For a layout with g spatial correlation grids, $m+n = O(g)$. Therefore, both the sum and the max operation have a complexity of $O(g)$.

In the PERT-like traversal of the circuit graph, for each gate we must change the delay representation of Equation (4.2) to that of Equation (4.5). In particular, we require the new sensitivity vectors $\mathbf{B'^T} = \mathbf{B^T}A$, [$\mathbf{C'^T} = \mathbf{C^T}P_y^{-1}$]. The dimensions of the ICA transformation matrix A is $n \times n$, and the PCA transformation matrix P_y is $m \times m$. However, the original sensitivity vectors $\mathbf{B^T}$ and $\mathbf{C^T}$ are typically sparse because a gate, in a particular grid, would fanout to other gates in not more than k different grids[7], with $k << Min(m,n)$. Therefore, the cost of computing the new sensitivity vectors, $\mathbf{B'^T}$ and $\mathbf{C'^T}$ by the multiplication of a sparse vector and a dense matrix is $O(m+n) = O(g)$.

For a circuit graph with V nodes and E edges, the overall time complexity of the SSTA procedure is $O(g(V+E))$. Therefore, the time complexity for our SSTA procedure, incorporating both Gaussian and non-Gaussian parameters, is the same as that of SSTA techniques considering only Gaussian variables [CS03, VRK+04]. However, the complexity constant for our procedure is higher due to the steps of moments evaluation

[7] In the case of a gate driving a global wire which spans many grids, it is highly likely that the global wire would be buffered.

and PDF extraction, and this is not surprising since [CS03, VRK$^+$04] can be reduced to special cases of our solution.

4.12 Experimental Results

The proposed SSTA algorithm was implemented in C++, using the *MinSSTA* code [CS03], and tested on edge-triggered ISCAS89 benchmark circuits. All experiments were performed on Pentium-4 Linux machines with a clock speed of 3.2GHz and 2GB of memory. The *FastICA* package [Hyv05] and the *Icasso* software [HH03], were used to obtain the ICA transform of Equation (4.3). To generate samples of correlated non-Gaussian parameters, required as inputs to the *FastICA* code, we use the method of *normal copula* [Sim], as described in Section 4.7.2. For all the experiments, we generate 5000 samples of each non-Gaussian parameter to feed to the ICA module. We use the Elmore delay model and the first order Taylor series terms to represent the canonical delay model of Equation (4.2). However, clearly this is not a restriction, as our canonical form is similar in form to that in [CS03, VRK$^+$04], and any analytical or numerical delay model may be used, as long as the sensitivities of the delay with respect to the varying parameters can be computed.

We consider the effective channel length, L_e, the transistor width W, and the dopant concentration, N_d as the sources of variation. The parameters L_e and W are modeled as correlated sources of variations, and the dopant concentration, N_d, is modeled as an independent source of variation. The same framework can be easily extended to include other parameters of variations. For simplicity, our current implementation ignores the effect of the input signal transition time on the delay at the output port of the gate. However, according to the technique described in [CS05], our SSTA procedure can also be extended to incorporate and propagate the distributions of the signal transition times. As described in [CS05], it is possible to express slope at the output pin of the gate as a probability weighted sum of distributions of the slope from all input pins to the output

pin of the gate. In our SSTA framework, we can efficiently compute these weights as closed-form probabilities, using the AWE-based PDF extraction scheme.

We use the grid-based model of [CS03] to generate the spatial correlations for the W and L_e parameters. Due to the lack of access to any real wafer data and process data files, we do not have the required information to realistically model the parameter distributions. We consider the following two cases for modeling the W and L_e parameters: **Case 1**: W of gates in each grid are modeled as non-Gaussian parameters, and L_e are modeled as Gaussian variables. Section 4.12.1 discusses the SSTA results for this case. **Case 2**: L_e of gates in each grid are modeled as non-Gaussian parameters, and W are assumed to be normally distributed variables. Section 4.12.2 discusses the SSTA results for this case.

For both cases, the independent parameter N_d is assumed to follow a Poisson distribution. The μ and σ values of the parameters are based on the predictions from [Nas00]. For $90nm$ technology, we use $\mu_W = 150nm$, $\mu_{L_e} = 60nm$, $\sigma_W = 7.5nm$ and $\sigma_{L_e} = 4nm$. For the independent parameter N_d modeled as a Poisson random variable, we use $\mu_{N_d} = 10 \times 10^{17} cm^{-3}$ for both nmos and pmos. We test our SSTA procedure by comparing our results for each benchmark with 10,000 Monte Carlo (MC) simulations based on the same grid model. The samples of correlated non-Gaussian parameters for Monte Carlo simulations are also generated using the method of normal copula, as described in Section 4.7.2.

4.12.1 SSTA results for Case 1

For these experiments, we model W of gates in each grid as non-Gaussian parameters, and L_e of gates in each grid as Gaussian parameters. For the correlated non-Gaussian W parameters, we randomly assign to W in each grid either a uniform distribution in $[\mu_W - \sqrt{3}.\sigma_W, \mu_W + \sqrt{3}.\sigma_W]$, or a symmetric triangular distribution in $[\mu_W - k.\sigma_W, \mu_W +$

$k.\sigma_W]$, given by:

$$\begin{aligned} f_W(w) &= \frac{2(w-a)}{(b-a)(c-a)} \quad a \leq w \leq c \\ f_W(w) &= \frac{2(b-w)}{(b-a)(b-c)} \quad c < w \leq b \end{aligned} \tag{4.27}$$

where $a = \mu_w - k.\sigma_w$, $c = \mu_w$, and $b = \mu_w + k.\sigma_w$. The number k is chosen so that the variance of the symmetric triangular distribution described in Equation (4.27) is the same as σ_w^2.

Benchmark			Error ($\frac{SSTA-MC}{MC}\%$)				Error ($\frac{MC_{Gauss}-MC}{MC}\%$)			
Name	# Cells	# Grids	μ	σ	95% Pt	5% Pt	μ	σ	95% Pt	5% Pt
s27	13	4	0.13%	0.22%	0.13%	0.57%	0.26%	0.54%	0.24%	0.81%
s1196	547	16	0.29%	0.59%	0.97%	0.83%	0.66%	1.22%	1.57%	1.35%
s5378	2958	64	-0.53%	-1.32%	-1.34%	-1.56%	0.93%	2.03%	1.93%	2.05%
s9234	5825	64	0.91%	1.81%	1.29%	-1.31%	0.87%	1.95%	2.59%	2.61%
s13207	8260	256	1.77%	2.24%	2.39%	3.03%	2.26%	3.35%	3.55%	3.11%
s15850	10369	256	1.98%	2.51%	3.14%	3.79%	2.89%	3.82%	3.51%	3.09%
s35932	17793	256	1.15%	2.82%	3.78%	3.67%	1.56%	2.56%	4.12%	4.26%
s38584	20705	256	1.71%	3.29%	3.59%	3.87%	2.09%	3.89%	4.22%	4.17%
s38417	23815	256	1.51%	3.68%	3.50%	3.61%	2.05%	4.35%	4.93%	4.88%
Avg Abs Err	-	-	1.11%	2.05%	2.24%	2.47%	1.51%	2.63%	2.96%	2.93%

Table 4.3: A comparison of results of the proposed SSTA method with Monte Carlo simulation . W parameters are modeled as non-Gaussian variables, and L_e parameters are modeled as Gaussian variables.

Table 4.3 shows a comparison of the results of the Monte Carlo (MC) simulations with our SSTA procedure for each benchmark circuit. We compare the mean (μ), the standard deviation (σ), the 95% and the 5% quantile points of the delay distribution obtained from our SSTA scheme with those generated from the Monte Carlo simulations, as the metrics of accuracy. As seen in Table 4.3, the results of the proposed SSTA scheme are quite close to that of Monte Carlo analysis. The average of the absolute errors, across the nine benchmark circuits, shown in the last row of Table 4.3, is 1.11% for μ, 2.05 % for σ, 2.24% for the 95% point, and 2.47% for the 5% quantile point.

We also compare the actual Monte Carlo results with the ones obtained by incorrectly modeling the non-normal W parameters as Gaussian variables, and then performing a Monte Carlo analysis, termed as MC_{Gauss}. Columns eight to eleven of Table 4.4 report the errors for comparison between the actual Monte Carlo results, and the ones obtained by Gaussian modeling of all parameters. As seen in the table, the errors for assuming an incorrect Gaussian distribution for W parameters, *does not* result in significant errors, implying that the circuit delay PDF does not significantly deviate from a Gaussian distribution. It should be noted that for our gate delay models, the coefficients of the L_e terms are greater than the coefficients of the W terms by a factor of about $5\times$ to $12\times$. Since the sensitivities of the Gaussian L_e terms outweigh the sensitivities of the non-Gaussian W terms, the circuit delay PDF is dominated by the normal parameters, and does not significantly diverge a normal distribution.

4.12.2 SSTA results for Case 2

For these experiments, we model L_e of gates in each grid as non-Gaussian parameters, and W of gates in each grid as Gaussian parameters. For the correlated non-Gaussian L_e parameters, we randomly assign to L_e in each grid either a uniform distribution in $[\mu_{L_e} - \sqrt{3}.\sigma_{L_e}, \mu_{L_e} + \sqrt{3}.\sigma_{L_e}]$, or a symmetric triangular distribution, similar to the one described by Equation (4.27), but replacing W by L_e.

Table 4.4 shows a comparison of the results of the Monte Carlo simulations with our SSTA procedure for each benchmark circuit. As seen in Table 4.4, the results of the proposed SSTA scheme are quite close to that of Monte Carlo analysis. The average of the absolute errors, across the nine benchmark circuits, is 0.99% for μ, 2.05 % for σ, 2.33% for the 95% point, and 2.36% for the 5% quantile point. These errors are reasonably small as compared to the accuracy penalty paid by assuming the incorrect normal distribution modeling of L_e parameters. Columns eight to eleven of Table 4.4 show the error incurred when modeling the non-Gaussian L_e parameters as normally distributed

Benchmark			Error ($\frac{SSTA-MC}{MC}$%)				Error ($\frac{MC_{Gauss}-MC}{MC}$%)			
Name	# Cells	# Grids	μ	σ	95% Pt	5% Pt	μ	σ	95% Pt	5% Pt
s27	13	4	-0.09%	-0.34%	-0.75%	0.79%	0.56%	3.23%	8.56%	2.04%
s1196	547	16	-0.23%	-0.67%	-0.87%	-0.53%	0.84%	8.82%	11.27%	2.21%
s5378	2958	64	0.31%	1.12%	1.21%	1.28%	0.98%	10.23%	10.91%	1.21%
s9234	5825	64	0.82%	1.78%	1.32%	-1.48%	1.88%	15.32%	15.28%	-1.83%
s13207	8260	256	1.58%	2.34%	-2.54%	2.89%	2.96%	28.13%	18.34%	-2.13%
s15850	10369	256	1.85%	-2.12%	3.36%	3.61%	2.63%	22.12%	17.62%	3.16%
s35932	17793	256	-1.07%	2.78%	4.01%	3.57%	2.34%	26.71%	19.17%	3.31%
s38584	20705	256	1.65%	-3.56%	3.89%	3.91%	2.21%	25.67%	18.28%	2.95%
s38417	23815	256	1.34%	3.78%	3.37%	3.22%	2.81%	34.62%	21.63%	2.51%
Avg Abs Err	-	-	0.99%	2.05%	2.33%	2.36%	1.91%	19.42%	15.67%	2.37%

Table 4.4: A comparison of results of the proposed SSTA method with Monte Carlo simulation . L_e parameters are modeled as non-Gaussian variables, and W parameters are modeled as Gaussian variables.

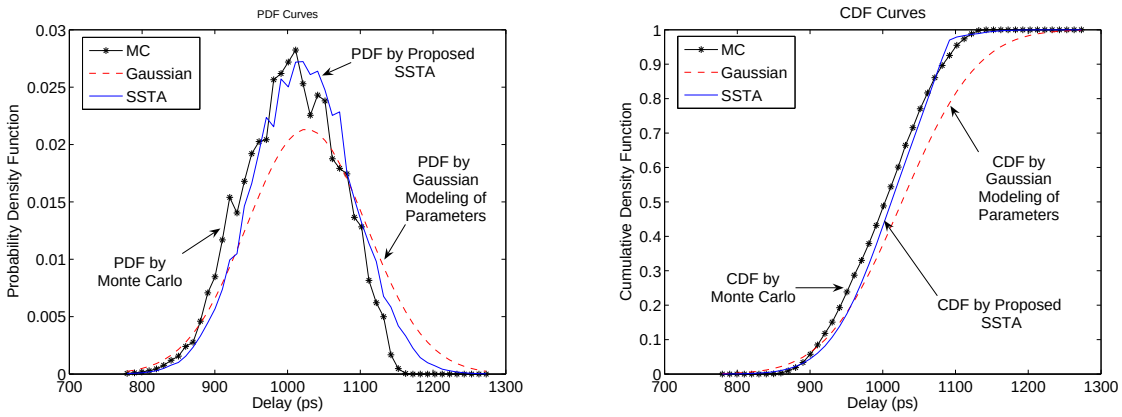
random variables and performing Monte Carlo simulations, termed as MC_{Gauss}, for each benchmark circuit. For instance, for the largest benchmark circuit s38417, when assuming that the non-Gaussian L_e parameters follow Gaussian distributions, the error observed is 2.81% for μ, 34.62% for σ, 21.63 % for the 95% point and 2.51% for the 5% point. Unlike, the results in Section 4.12.1, modeling the non-Gaussian L_e parameters as normally distributed ones, leads to significant inaccuracy in the circuit delay PDF. Due to the fact that the sensitivities of the non-Gaussian L_e terms outweigh the sensitivities of the Gaussian W terms, the correlated non-Gaussian parameters have a dominating effect on the circuit delay distribution, causing it to significantly aberrate from a normal distribution.

Table 4.5 compares the runtime performance of our proposed SSTA algorithm with that of a Gaussian SSTA procedure [CS03], and the Monte Carlo simulations. As expected, our SSTA procedure is considerably faster than the Monte Carlo simulations, but has a higher runtime cost as compared to a Gaussian SSTA [CS03], due to the additional feature of handling non-Gaussian variables. On an average our procedure is $33\times$ faster

Benchmark			CPU Time (sec)		
Name	# Cells	# Grids	$SSTA_{Gauss}$ [CS03]	SSTA	MC
s27	13	4	0.0	1.1	6.0
s1196	547	16	1.2	8.3	634.2
s5378	2958	64	17.1	41.6	3214.4
s9234	5825	64	20.3	137.9	4756.6
s13207	8260	256	108.6	303.6	8532.1
s15850	10369	256	110.8	410.8	9587.8
s35932	17793	256	315.2	761.4	10156.5
s38584	20705	256	322.4	910.6	18903.3
s38417	23815	256	377.3	1235.6	22398.5

Table 4.5: A runtime comparison the proposed SSTA with Gaussian SSTA and Monte Carlo simulation.

than Monte Carlo method, but about $3\times$ slower than the Gaussian SSTA algorithm. Our approach can handle a large number of correlated and independent non-Gaussian parameters. The number of grids chosen for each benchmark circuit, shown in the third column of Table 4.5, is equal to the number of correlated Gaussian and non-Gaussian variables. The number of independent non-Gaussian variables is the same as the number of cells in a circuit. For instance, the SSTA procedure for the circuit s13207 processes 256 correlated Gaussian variables, 256 correlated non-Gaussian variables, and 8260 independent non-Gaussian variables in about 5 mins of online runtime. Thus, our procedure scales well with the number of non-Gaussian parameters. The runtime reported in Table 4.5 does not include the time spent for the preprocessing steps of Sections 4.7 and 4.8, which are carried out only once for a process and a given discretization. For the largest benchmark s38417, the preprocessing time taken to generate the ICA matrix A, and to compute the moments of the independent components is 3.5 hours.



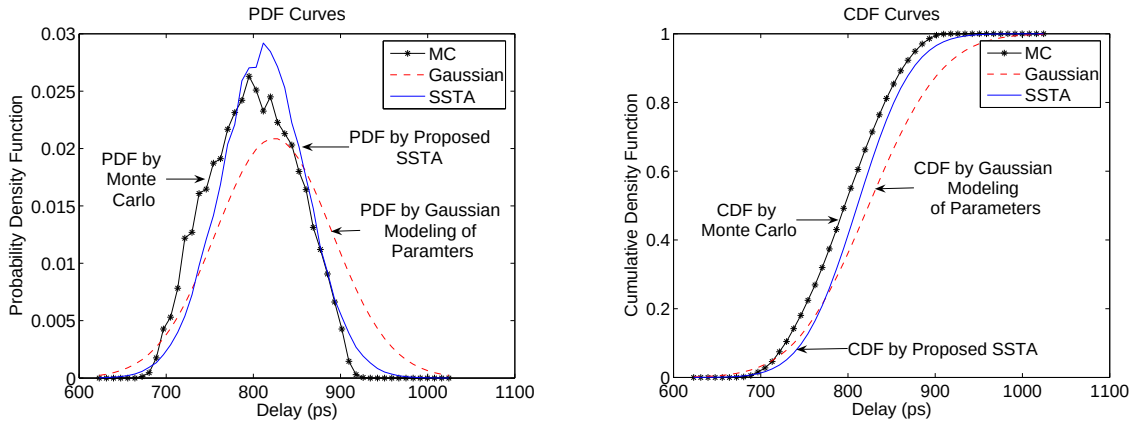
(a) PDF comparison (b) CDF comparison

Figure 4.8: A comparison of SSTA and Monte Carlo distribution for circuit s13207.

In Figures 4.8 and 4.9, the PDF and CDF plots for the benchmark circuits s13207 and s38417 are provided. As seen in the figures, the PDF and the CDF as predicted by the proposed SSTA scheme matches well with the Monte Carlo PDF and CDF. The dashed curves in Figures 4.8 and 4.9, represent the case when the L_e parameters are incorrectly modeled as Gaussian variables with the same μ_{L_e} and σ_{L_e} as the original non-Gaussian parameters. The plots in these figures show that in the presence of correlated non-Gaussian parameters, the real circuit delay distribution deviates significantly from the one obtained by assuming normality for parameters. The distribution functions evaluated by SSTA approach are able to match, within reasonably small errors, the real distribution functions.

4.13 Conclusion

In this chapter of the thesis, we have presented a statistical timing analysis method as a variation-aware timing analysis technique. Our novel and efficient SSTA algorithm incorporates correlated parameters, both Gaussian and non-Gaussian. Our approach is



(a) PDF comparison

(b) CDF comparison

Figure 4.9: A comparison of the results of SSTA and Monte Carlo for circuit s38417.

based on PDF evaluation by matching the moments of the delay variables. We have used the independent component analysis technique in our SSTA framework to handle correlations between the non-Gaussian parameters. A time complexity analysis of our procedure shows that it is linear in the number of grids and the number of gates in the circuit. Hence, our scheme provides a scalable solution to the problem of performing SSTA in the presence of many correlated non-Gaussian parameters. Experimental results validate our hypothesis that performing a Gaussian SSTA, in the presence of dominating strongly non-Gaussian parameters of variation, could result in significant inaccuracies in estimating the PDF and CDF of the circuit delay. Our proposed SSTA procedure is able to match the real PDF and CDF of the delay much more closely, and produces the delay distributions with reasonably small errors compared to the Monte Carlo distributions, and is much faster than the Monte Carlo analysis. Applying our SSTA method to the nine benchmark circuits, the average of the absolute errors is 0.99% for μ, 2.05 % for σ, 2.33% for the 95% point, and 2.36% for the 5% quantile point.

Chapter 5

Summary

The certainty of uncertainties, in the operating environment and the fabrication process of VLSI circuits, poses enormous challenges in front of the circuit design community to continue the desired growth of the semiconductor industry. To enable the circuit designers to combat this beast of environmental and manufacturing process fluctuations, the electronic design automation tools must be variation-aware.

In this thesis, we have presented such uncertainty-aware computer-aided design techniques focusing on three important issues in circuit design: power grid design, gate sizing, and timing analysis. The summary of our research efforts and contributions for each of these problems are as follows:

- We have presented two topology optimization power grid design schemes to reduce the voltage drop variations in the P/G network of wires. Our schemes optimize a special locally regular, globally irregular structure of the power grid, that we have proposed. Experimental results show that such a power grid structure offers considerable savings in the wire area utilized, compared to other commonly used grid topologies. Moreover, this piecewise-uniform structure is relatively easier to optimize, and is expected to aid signal net routing, as minimal amount of book-keeping is required to account for the locations and the widths of the P/G wires. Although, the grids designed using our first power grid design procedure, show considerable saving of expensive wiring resources, the procedure itself is not very efficient. As a much more efficient alternative, we have proposed a second, considerably fast algorithm to design the power grid. This method is able to design power grids comprising millions of nodes, and thousands of wires is a reasonably small amount of time. The wire area used by the power grids designed using the second method is not significantly greater than the ones designed by the

first technique. Using these power grid design procedures, the user can design a high-performance power grid that meets the reliability constraints of IR drop and EM.

- We have proposed a novel worst-casing methodology to perform gate sizing in the presence of process variations. Our method is based on using the statistical information of the varying parameters to keep sufficient, but not excessive, margins in the timing constraints that can guard-band against the worst-case variation effects. We formulate the uncertainty-aware gate sizing problem as a GP, and solve it efficiently using convex optimization tools. By incorporating the effect of spatial correlations in the optimization framework, we are able to mitigate some of the pessimism involved in a worst-casing methodology. We use various heuristic techniques to reduce the conservatism in our method. Experimental results demonstrate that, compared to our proposed method, a naive worst-casing scheme, based on keeping a deterministically-based guard-band, produces substantially more expensive designs, employing extra resources.

- We have presented an efficient and an accurate statistical timing analysis procedure that incorporates correlated parameters, both Gaussian and non-Gaussian. Prior to this work, there were no known SSTA methods that could efficiently handle a large number of non-Gaussian variables. The time complexity of our SSTA procedure is $O(n * N_G)$, where n is the number of grids the chip layout is divided into, and N_G is the number of gates in the circuit, which is the same linear complexity as of the Gaussian SSTA algorithms. Thus, the procedure is scalable to process a large number of non-Gaussian parameters. In our method, we use independent component analysis to handle the correlated non-normal variables, and employs moment matching-based technique to predict the probability distribution of the circuit delay. We demonstrate the accuracy of our SSTA procedure by verifying it against Monte Carlo simulations. The errors for our method are

reasonably small compared to Monte Carlo analysis.

Future extensions of the research work presented in this thesis may consist of:

- In addition to the static voltage drop variations on the power grid wires, the transient voltage droop problem on these wires also causes serious reliability issues. Efforts to develop efficient simultaneous decoupling capacitor placement and topology optimization algorithms, to control this transient voltage noise would be well spent.

- The leakage power of a circuit is extremely sensitive to the process variations. Therefore, the profitability of a chip is affected by both the timing yield and the power yield of a circuit. Performing circuit optimization under probabilistic constraints for both timing and power, could lead to interesting opportunities.

- The linear delay model used in the proposed SSTA algorithm may prove to be inaccurate in future, with the amount of variations increasing significantly in future technologies. Extending the current non-Gaussian SSTA methods to nonlinear models remains an open problem, and requires further investigation.

- One of the ways to perform circuit optimization, in the presence of variations, is to use an SSTA engine to generate some notion of a probabilistic critical set of paths. The designers can then spend the extra design resources to make these paths less sensitive to variations. Developing an automation tool for such an analysis remains a challenging, but an important problem to solve.

- There are a number of factor that could lead to a statistical circuit optimization methods win over the deterministic-based guard-banding approaches. These factors could be, e.g., amount of path correlations, logic dominated or interconnect dominated circuit structure, power-delay tradeoff nonlinearities, systematic or random variations, etc. Understanding these factors clearly, would be extremely

helpful for the designers to adjust their deterministic margins to incorporate the appropriate statistical attributes, and reduce the conservatism in their designs. Studies for such an analysis are exciting problems to explore.

In conclusion, the variations in VLSI circuit design are here to stay. This thesis attempts to provide CAD methods to either directly reduce these uncertainties or to control their effect on the circuit performance. It is hoped that work presented in this thesis, provides a platform to propel further research in this domain, that culminates into producing industry standard electronic design automation tools.

BIBLIOGRAPHY

[ABZ03] A. Agarwal, D. Blaauw, and V. Zoltov. Statistical Timing Analysis for Intra-Die Process Variations with Spatial Correlations. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 900–907, 2003.

[ACBZ05] A. Agarwal, K. Chopra, D. Blaauw, and V. Zolotov. Circuit Optimization Using Statistical Static Timing Analysis. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 338–342, 2005.

[AK98] V. Axelrad and J. Kibarain. Statistical Aspects of Modern IC Design. In *Proceedings of 28th European Solid-State Device Research Conference*, 1998.

[AMK+05] C. Amin, N. Menezes, K. Killpack, F. Dartu, U. Choudhury, N. Hakim, and Y. I. Ismail. Statistical Static Timing Analysis: How simple can we get? In *Proceedings of ACM/IEEE Design Automation Conference*, pages 652–657, 2005.

[Bel] T. Bell. An ICA page – papers, code, demos, links. Available at `http://www.cnl.salk.edu/tony/ica.html`.

[Ber99] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1999.

[BKD04] S. Borkar, T. Karnik, and V. De. Design and Reliability Challenges in Nanomter Design. In *Proceedings of ACM/IEEE Design Automation Conference*, page 75, 2004.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[BVGY01] S. Boyd, L. Vandenberghe, A. E. Gamal, and S. Yun. Design of Robust Global Power and Ground Networks. In *Proceedings of ACM International Symposium on Physical Design*, pages 60–65, 2001.

[BVSH02] X. Bai, C. Visweswariah, P. N. Strenski, and D. J. Hathaway. Uncertainty-Aware Circuit Optimization. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 58–63, 2002.

[Cai88] H. Cai. Multi-pads Single Layer Power Net Routing in VLSI Circuit. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 183–188, 1988.

[Chi04] E. Chiprout. Fast Flip-Chip Power Grid Analysis via Locality and Grid Shells. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 485–488, 2004.

[CKM] A. Caldwell, A. B. Kahng, and I. Markov. Capo: a large-scale fixed-die placer. available at `http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/Placement.`

[CL97] H. H. Chen and D. D. Ling. Power Supply Noise Analysis Methodology for Deep-Sub-micron VLSI Chip Design. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 638–643, 1997.

[Cla61] C. E. Clark. The Greatest of a Finite Set of Random Variables. *Operations Research*, 9:145–162, March-April 1961.

[Con00] J. Cong. An Interconnect-Centric Design Flow for Nanometer Technologies. *the IEEE*, 89:477–480, 2000.

[CPR04] S. H. Choi, B. C. Paul, and K. Roy. Novel Sizing Algorithm for Yield Improvement under Process Variation in Nanometer Technology. In *Pro-*

ceedings of ACM/IEEE Design Automation Conference, pages 454–459, 2004.

[CS03] H. Chang and S. S. Sapatnekar. Statistical Tming Analysis Considering Spatial Correlations Using a Single PERT-like Traversal. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 621–625, 2003.

[CS05] H. Chang and S. S. Sapatnekar. Statistical Timing Analysis Considering Spatial Correlations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24:1467–1482, September 2005.

[CSS+05] K. Chopra, S. Shah, A. Srivastava, D. Blaauw, and D. Sylvester. Parametric Yield Maximization using Gate Sizing based on Efficient Statistical Power and Delay Gradient Computation. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 1023–1028, 2005.

[CZNV05] H. Chang, V. Zoltov, S. Narayan, and C. Visweswariah. Parameterized Block-Based Statistical Timing Analysis with Non-Gaussian Parameters, Nonlinear Delay Functions. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 71–76, 2005.

[DDK05] H. Damerdji, A. Dasdan, and S. Kolay. On the Assumption of Normality in Statistical Static Timing Analysis. In *Proceedings of TAU*, pages 2–7, 2005.

[DK03] A. Devgan and C. Kashyap. Block-based Static Timing Analysis with Uncertainty. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 607–614, 2003.

[DMS89] R. Dutta and M. Marek-Sadowska. Automatic Sizing of Power Ground (P/G) Networks in VLSI. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 783–786, 1989.

[DS02] M. H. Degroot and M. J. Schervish. *Probability and Statistics*. Addison Wesley, Boston, MA, 2002.

[DS06] A. Davoodi and A. Srivastava. Variability Driven Gate Sizing for Binning Yield Optimization. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 959–964, 2006.

[FD85] J. Fishburn and A. Dunlop. TILOS: A Posynomial Programming Approach to Transistor Sizing. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 326–328, 1985.

[GKSY03] P. Gupta, A. B. Kahng, D. Sylvester, and J. Yang. Performance-driven OPC for Mask Cost Reduction. In *Proceedings of IEEE Internation Symposium on Quality Electroninc Design*, pages 270–275, 2003.

[HH03] J. Himberg and A. Hyv*ä*rinen. Icasso:Software for Investigating the Reliability of ICA Estimates by Clustering and Visualization. In *Proceedings of IEEE Workshop on Neural Networks for Signal Processing*, pages 259–268, 2003.

[HO99] A. Hyv*ä*rinen and E. Oja. Independent Component Analysis: A Tutorial. `http://www.cis.hut.fi/aapo/papers/IJCNN99_tutorialweb/`, 1999.

[HO00] A. Hyv*ä*rinen and E. Oja. Independent Component Analysis:Algorithms and Applications. *Neural Networks*, 13:411–430, 2000.

[Hyv05] A. Hyv*ä*rinen. Fast ICA. Available at `http://www.cis.hut.fi/projects/ica/fastica/`, 2005.

[JB00] E. T. A. F. Jacobs and M. R. C. M. Berkelaar. Gate Sizing Using a Statistical Delay Model. In *Proceedings of IEEE Design Automation and Test in Europe*, pages 283–291, 2000.

[JW02] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, Upper Saddle River, NJ, 2002.

[KC66] T. Kirkpatrick and N. Clark. PERT as an Aid to Logic Design. *IBM Journal of Research and Development*, 10(2):135–141, June 1966.

[KKS98] K. Kasamsetty, M. Ketkar, and S. S. Sapatnekar. A New Class of Convex Functions for Delay Modeling and their Application to the Transistor Sizing Problem. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19:779–788, Jul 1998.

[KNH95] H. Kriplani, F. Najm, and I. Hajj. Pattern Independent Maximum Current Estimation in Power and Ground Buses of CMOS VLSI Circuits : Algorithms, Signal Correlations, and Their Resolution. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 998–1012, 1995.

[KS05] V. Khandelwal and A. Srivastava. A General Framework for Accurate Statistical Timing Analysis Considering Correlations. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 89–94, 2005.

[KWW04] S-D. Kim, H. Wada, and J. C. S. Woo. TCAD-Based Statistical Analysis and Modeling of Gate Line Edge Roughness Effect on Nanoscale MOS Transistor Performance and Scaling. *IEEE Transactions on Semiconductor Manufacturing*, 17:192–200, May 2004.

[LC01] S. Lin and N. Chang. Challenges in Power-Ground Integrity. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 651–654, 2001.

[LHL04] W. Liao, L. He, and K. Lepak. Temperature-Aware Performance and Power Modeling. In *Technical Report 04-250, UCLA Engr, Los Angeles, California*, 2004.

[LLCP05] X. Li, J. Le, M. Celik, and L. Pileggi. Defining Statistical Sensitivity for Timing Optimization of Logic Circuits with Large-Scale Process and Environmental Variations. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 844–852, 2005.

[LLGP04] X. Li, J. Le, P. Gopalakrishnan, and L. Pileggi. Asymptotic Probability Extraction for non-Normal Distributions of Circuit Performance. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 2–9, 2004.

[LLP04] J. Le, X. Li, and L. T. Pileggi. STAC: Statistical Timing Analysis with Corrleation. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 343–348, 2004.

[LTKS02] J. Lou, S. Thakur, S. Krishnamoorthy, and H. S. Sheng. Estimating Routing Congestion Using Probabilistic Analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21:32–41, Jan 2002.

[LZT] C. Lawrence, J. Zhou, and A. Tits. User's gude for cfsqp version 2.4. Institute for Systems Research, University of Maryland, College Park, MD, Tech. Rep. TR-94-16rl, 1996.

[MDO05] M. Mani, A. Devgan, and M. Orshansky. An Efficient Algorithm for Statistical Power under Timing Yield Constraints. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 309–314, 2005.

[mes] Meschach: Matrix computations in C. Available at `http://www.netlib.org/c/meschach`.

[MK92] T. Mitsuhashi and E. S. Kuh. Power and Ground Network Topology Optimization for Cell Based VLSIs. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 524–529, 1992.

[Moo65] G. E. Moore. Cramming more components onto integrated circuits. *Electronic Magazine*, 38:114–117, April 1965.

[Mos] MOSEK Optimization Software. Available at `http://www.mosek.com`.

[MP99] R. Manduchi and J. Portilla. Independent Component Analysis of Textures. In *Proceedings of IEEE Conference on Computer Vision*, volume 2, pages 1054–1060, 1999.

[MRG] Mathematica Reference Guide. Available at `http://documents.wolfram.com/v5/TheMathematicaBook/MathematicaReferenceGuide/index.html`.

[MRM] Matlab Reference Manual. Available at `http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.shtml`.

[Nas00] S. Nassif. Delay Variability: Sources, Impact and Trends. In *Proceedings of IEEE International Solid State Circuit Conference*, pages 368–369, 2000.

[OB04] M. Orshansky and A. Bandyopadhyay. Fast Statistical Timing Analysis Handling Arbitrary Delay Correlations. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 337–342, 2004.

[OP98] J. Oh and M. Pedram. Multi-pad Power/Ground Network Design for Uniform Distribution of Ground Bounce. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 287–290, 1998.

[PC06] S. Pant and E. Chiprout. Power Grid Physics and Implication for CAD. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 199–204, 2006.

[PRV95] L .T. Pillage, R. A. Rohrer, and C. Visweswariah. *Electronic Circuit and System Simulation Methods*. McGraw-Hill, New York, NY, 1995.

[RVW04] S. Raj, S. B. K. Vrudhala, and J. Wang. A Methodology to Improve Timing Yield in the Presence of Process Variations. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 448–453, 2004.

[SG03] P. Saxena and S. Gupta. On Integrating Power and Signal Routing for Shield Count Minimization in Congested Regions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24:437–445, Apr 2003.

[SGS00] H. Su, K. H. Gala, and S. S. Sapatnekar. Fast Analysis and Optimization of Power/Ground Networks. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 477–480, 2000.

[SHSN02] H. Su, J. Hu, S. S. Sapatnekar, and S. Nassif. Congestion-driven Codesign of Power and Signal Networks. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 64–69, 2002.

[SIA01] Semiconductor Industry Association. International Technology Roadmap for Semiconductors, 2001. Available at `http://public.itrs.net/Links/2001ITRS/Home.htm`.

[SIA05] Semiconductor Industry Association. International Technology Roadmap Semiconductors, 2005. Available at `http://public.itrs.net/Links/2005ITRS/Home.htm`.

[Sim] Simulating Dependent Random Variables Using Copulas. Available at `http://www.mathworks.com/products/statistics/`.

[SNLS05] J. Singh, V. Nookala, T. Luo, and S. Sapatnekar. Robust Gate Sizing by Geometric programming. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 315–320, 2005.

[SRVK93] S. Sapatnekar, V. B. Rao, P. M. Vaidya, and S. M. Kang. An Exact Solution to the Transistor Sizing Problem for CMOS Circuits Using Convex Optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12:1621–1634, Nov 1993.

[SS05] J. Singh and S. S. Sapatnekar. Congestion-Aware Topology Optimization of Structured Power/Ground Networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24:683–695, May 2005.

[SS06a] J. Singh and S. Sapatnekar. Statistical Timing Analysis with Correlated Non-Gaussian Parameters using Independent Component Analysis. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 155–160, 2006.

[SS06b] J. Singh and S. S. Sapatnekar. A Partition-based Algorithm for Power Grid Design using Locality. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25:664–677, April 2006.

[SSH+03] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-Aware Microarchitecture. In *Proceedings of the International Symposium on Computer Architecture*, 2003.

[SSZ05] D. Sinha, N. V. Shenoy, and H. Zhou. Statistical Gate Sizing for Timing Yield Optimization. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 1037–1042, 2005.

[TS01] X. D. S. Tan and C. J. R. Shi. Fast Power/Ground Network Optimization Based on Equivalent Circuit Modeling. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 550–554, 2001.

[TSL03] X. D. S. Tan, C. J. R. Shi, and J. C. Lee. Reliability-Constrained Area Optimization of VLSI Power/Ground Networks Via Sequence of Linear Programmings. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems Conference*, 22:156–161, Dec 2003.

[VC99] C. Visweswariah and A. R. Conn. Formulation of Static Circuit Optimization with Reduced Size, Degeneracy and Redundancy by Timing Graph Manipulation. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 244–252, 1999.

[VRK+04] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan. First-Order Incremental Block-Based Statistical Timing Analysis. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 331–336, 2004.

[WBG04] J. Westra, C. Bartels, and P. Groeneveld. Probabilistic Congestion Prediction. In *Proceedings of ACM International Symposium on Physical Design*, pages 204–209, 2004.

[WC02] T. Wang and C. C. Chen. Optimization of the Power/Ground Network Wire-Sizing and Spacing Based on Sequential Network Simplex Algorithm. In *Proceedings of IEEE International Symposium on Quality Electronic Design*, pages 157–162, 2002.

[WHC+01] X. Wu, X. Hon, Y. Ca, C. K. Cheng, J. Gu, and W. Dai. Area Minimization of Power Distribution Network Using Efficient Nonlinear Programming Techniques. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 153–157, 2001.

[WMR04] Z. Wang, R. Murgai, and J. Roychowdhury. Automated Accurate Macromodelling of Digital Agressors for Power/Ground/Substrate Noise Prediction. In *Proceedings of IEEE Design Automation and Test in Europe*, pages 824–829, 2004.

[WMS05] K. Wang and M. Marek-Sadowska. On-chip Power Supply Network Optimization using Multigrid-based Technique. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24:407–417, Mar 2005.

[XHL+05] Y. Xu, K. L. Hsiung, X. Li, I. Nausieda, S. Boyd, and L. Pileggi. OPERA: Optimization with Ellipsoidal Uncertainty for Robust Analog IC Design. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 632–637, 2005.

[XZVV06] J. Xiong, V. Zolotov, N. Venkateswaran, and C. Visweswariah. Criticality computation in Parameterized Statistical Timing. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 63–68, 2006.

[ZCH+05] L. Zhang, W. Chen, Y. Hu, J. A. Gubner, and C. C.-P. Chen. Correlation-Preserved Non-Gaussian Statistical Timing Analysis with Quadratic Tim-

ing Model. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 83–88, 2005.

[ZPS+00] M. Zhao, R. V. Panda, S. S. Sapatnekar, T. Edwards, R. Chaudhry, and D. Blaauw. Hierarchical Analysis of Power Distribution Networks. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 482–486, 2000.

[ZPS+06] M. Zhao, R. Panda, S. Sundareswaran, S. Yan, and Y. Fu. A Fast On-Chip Decoupling Capacitor Budgeting Algorithm Using Macromodeling and Linear Programming. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 217–222, 2006.

[ZSLP05] Y. Zhan, A. J. Strojwas, X. Li, and L. T. Pillegi. Correlation-Aware Statistical Timing Analysis with Non-Gaussian Delay Distributions. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 77–82, 2005.

[ZSSN05] Y. Zhan, A. J. Strojwas, M. Sharma, and D. Newmark. Statistical Critical Path Analysis Considering Correlations. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 699–704, 2005.