Tuning for yield : towards predictable deep-submicron manufacturing

Tuning for Yield

Towards predictable deep-submicron manufacturing

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de Rector Magnificus, prof.dr. R.A. van Santen, voor een commissie aangewezen door het College voor Promoties in het openbaar te verdedigen op dinsdag 13 juli 2004 om 16.00 uur

door

Srinath Robin Naidu

geboren te Bangalore, India

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr.ir. R.H.J.M. Otten
en
prof.Dr.-Ing. J.A.G. Jess

Druk: Universiteitsdrukkerij Technische Universiteit Eindhoven

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Naidu, Srinath R.

Tuning for yield : towards predictable deep-submicron manufacturing / by Srinath Robin Naidu. - Eindhoven : Technische Universiteit Eindhoven, 2004.
Proefschrift. - ISBN 90-386-1603-1
NUR 959
Trefw.: grote geintegreerde schakelingen ; CAD / digital systemen ; CAD / stochastische analyse / Monte-Carlo-methoden / convex programmeren.
Subject headings: integrated circuit yield / design for manufacture / delay estimation / Monte Carlo methods / convex programming.

Summary of Thesis

Process variations in modern semiconductor fabs are beginning to impact the performance of manufactured chips. The impact is at two levels - the process variations either cause functional faults in the chip, or they cause a parametric fault in the manufactured chip wherein the chip in question does not meet performance specifications regarding clock frequency or power consumption. Functional faults in the chip prevent the chip from being sold at all, but parametric faults are milder in that the faulty chips can still be sold, albeit at a lower price. Considerable attention has been paid to the functional fault problem thus far, but relatively little attention has been paid to the second problem.

In the presence of process variations, a natural question that can be asked is the percentage of chips meeting a given performance specification. For example, one could ask what percentage of manufactured chips have a clock frequency greater than 1GHz. In fact one might desire to calculate the entire probability distribution curve for all relevant performances. The availability of such a probability distribution curve would enable the semiconductor manufacturer to devise an accurate pricing strategy for manufactured chips *before* actually manufacturing them. The problem of generating this curve for a given chip, given a model of process variations, is called the *yield estimation* problem. A natural corollary to this problem is the *yield optimisation* problem where the object is to change circuit parameters so that yield may be increased.

This thesis attempts to address both the yield estimation as well as the yield optimisation problem. The first step towards the solution of both of these problems is to accurately model the impact of process variations on the performance of circuit elements. Semiconductor companies are still in the process of modeling the impact of process variations on the behaviour of

circuit elements, so one must take recourse to assuming a realistic process variation model. Two models are considered in this thesis. The first assumes that the delays of the circuit elements are *independent* of each other. The yield estimation problem is then shown to be a variant of the probabilistic PERT problem that has been the subject of study in the operations research area. The problem is theoretically hard in a deep computational sense. We provide upper and lower bounds on the true probability distribution curve of circuit delay using a novel method of discretisation of the probability density functions of the individual gate delays.

We then turn our attention to a different model of process variations - one that fully takes into account correlations due to path sharing as well as correlations due to gate delays depending on the same set of global parameters such as the length of the transistor gate, thickness of oxide and so on.In this framework the yield estimation problem is transformed into one of integrating a joint probability density function over a polytope (convex region in an appropriate space bounded by hyperplanes). This problem, like the probabilistic PERT problem discussed in the previous paragraph, is also of high complexity and there exists no constant-factor approximation algorithm that estimates yield. We introduce the novel concept of approximating the volume of the polytope by the maximum volume ellipsoid that fits in it. Instead of integrating over the whole polytope, we consider the simpler problem of integrating over the ellipsoid, to obtain a lower bound on the true yield. When the number of dimensions (or global parameters) gets large, the ellipsoid becomes a poor approximation of the polytope, and there is a substantial yield loss due to the omission of the polytope corners. Therefore we change tack, and propose a novel Monte-Carlo algorithm to estimate yield. The Monte-Carlo algorithm uses the maximum volume ellipsoid to determine a sampling probability density function that reduces the variance of the yield estimate.

Finally we conclude by demonstrating the true power of the ellipsoid approximation of the polytope feasible region, namely the ability of the ellipsoid to represent the *shape* of the feasible region. In particular the smallest axis of the ellipsoid represents the direction in which to expand the polytope so as to increase the yield by the largest amount. The increase in the volume of the polytope is achieved by adjusting the nominal delays of the hyperplanes, or in circuit terms, adjusting the nominal delays of paths in the circuit represented by those hyperplanes. The nominal delays of the paths in turn depend on the nominal delays of the gates in the path. We provide a simple method-

ology for tuning the delays of the gates to achieve the required change in the nominal delays of the selected paths.

Samenvatting

Procesvariaties in moderne halfgeleiderfabrieken beginnen prestaties van de gefabriceerde halfgeleiders schakelingen (chips) te benvloeden. De gevolgen zijn merkbaar op twee niveaus: oftewel de procesvariaties veroorzaken functionele fouten in de chip, of ze veroorzaken parametrische fouten waarbij de gefabriceerde chip de specificaties wat betreft de klokfrequentie of vermogensdissipatie niet haalt. Bij functionele fouten is de chip onverkoopbaar. Parametrische fouten zijn minder desastreus, omdat de defecte chips nog steeds verkoopbaar zijn, zij het tegen een lagere prijs. Tot dusver is veel aandacht besteed aan functionele fouten, terwijl relatief weinig aandacht is besteed aan het tweede probleem.

Een voor de hand liggende vraag die gesteld kan worden bij procesvariaties is het percentage van de chips die aan een gegeven specificatie voor de prestaties voldoen. Zo zou men zich bijvoorbeeld kunnen afvragen welk percentage van de gefabriceerde chips een klokfrequentie haalt van meer dan 1GHz. Eigenlijk zou men de kansdichtheidsverdeling voor alle relevante opbrengstvariabelen willen berekenen. De beschikbaarheid van zo'n kansdichtheidsverdeling stelt de fabrikant in staat een accurate prijsstelling voor de gefabrieerde chips te maken, nog voor deze chips daadwerkelijk gefabriceerd zijn. Het probleem van het genereren van zo'n verdeling voor een chip, gegeven een model van de procesvariaties, heet het opbrengstafschattingsprobleem. Een natuurlijke uitbreiding op dit probleem is het probleem van opbrengstoptimalisatie; hierbij is het doel de parameters van de schakeling zodanig te veranderen dat de opbrengst toeneemt.

Dit proefschrift beschouwt zowel opbrengstafschatting als opbrengstoptimalisatie. De eerste stap in de richting van een oplossing voor deze problemen is een nauwkeurige modellering van de invloed van procesvariaties op de elementen in de schakeling. De halfgeleiderindustrie bevindt zich vooralsnog

in het stadium waarin de invloed van procesvariaties op elementen van de schakeling nog niet volledig gemodelleerd is, dus men moet zijn toevlucht nemen tot de aanname van een realistisch model voor procesvariaties. Twee modellen woren beschouwd in dit proefschrift. Het eerste neemt aan dat de vertragingen van elementen van de schakeling onafhankelijk zijn van elkaar. In dit geval zal worden aangetoond dat het opbrengstafschattingsprobleem een variant is van het probabilistisch PERT probleem, hetgeen reeds onderwerp van onderzoek is in de operations research. Dit probleem is fundamenteel moeilijk in de zin van de theorie van berekenbaarheid. We berekenen boven- en ondergrenzen voor de daadwerkelijke kansverdeling van de vertraging van de schakeling. We maken hierbij gebruik van een nieuwe methode, waarbij we de kansdichtheidsfunctie van de individuele vertragingen van de elementen van de schakeling discretiseren.

Vervolgens richten we onze aandacht op een ander model van procesvariaties - n die correlaties door het delen van paden door de schakeling en correlaties in de vertraging van elementen van de schakeling door dezelfde verzameling globale paremeters in beschouwing neemt. Voorbeelden van zulke globale parameters zijn onder andere lengte van de transistor en dikte van het oxide. In deze context wordt het opbrengstafschattingsprobleem getransformeerd naar het probleem van het integreren van een gezamenlijke kansdichtheidsfunctie over een polytoop (een convexe regio in een ruimte begrensd door hypervlakken). Dit probleem is, net zoals het probabilistisch PERT problem van de vorige alinea, van hoge complexiteit en er bestaat geen algoritme dat de opbrengst afschat met een constante factor. We introduceren een nieuw concept waarbij we het volume van de polytoop benaderen door de ellipsode met een maximaal volume die hierin past. In plaats van het integreren over de hele polytoop, beschouwen we het integreren over de ellipsode. Dit levert een ondergrens op voor de werkelijke opbrengst. Als het aantal dimensies (globale parameters) groot wordt, wordt de ellipsode een slechtere benadering van de polytoop, en is er een substantiel verlies in opbrengst door het weglaten van hoeken van de polytoop. Om dit te ondervangen schakelen we over op een Monte-Carlo algoritme om de opbrengst te bepalen. De Monte-Carlo algoritme gebruikt de ellipsode met het maximale volume om een bemonstering van de kansdichtheidfunctie te bepalen die de variatie van de opbrengst reduceert.

Tenslotte eindigen we met het demonstereren van de werkelijke kracht van de benadering door een ellipsode van de polytope toegestane regio, namelijk

de mogelijkheid van de ellipsode om de vorm van de toegestane regio te benaderen. Met name de kleinste as van de ellipsode representeert de richting in welke de polytoop moet worden uitgebreid om de opbrengst zo veel mogelijk te verhogen. De toename in het volume van de polytoop wordt behaald door het aanpassen van de nominale vertragingen van de hypervlakken. In termen van de schakeling betekent dit het aanpassen van de nominale vertraging van de paden door de schakeling gerepresenteerd door de hypervlakken. De nominale vertraging van de paden hangt op zijn beurt weer af van de nominale vertraging van de vertraging van de elementen van de schakeling op het pad. We presenteren een simpele methode om de vertraging van de elementen van de schakeling te veranderen teneinde de gewenste verandering in de nominale vertraging van de geselecteerde paden te bewerkstelligen.

Acknowledgements

When I began my Ph.D contract in September 1999, I had no idea that the next four years would be as exciting as they have turned out to be. First I must thank Prof Jochen Jess who provided me with the opportunity of working in the the ES group at the Eindhoven University of Technology, even though he knew that my previous work was unrelated to the subject in which I was to carry out research. I have had the pleasure of conducting many useful conversations with him.

I thank Professor Ralph Otten for assuming the responsibility of guiding my thesis, and providing me with some penetrative insights from time to time. I have benefited greatly from his insights into how a thesis should be organised. The summer of 2002 was a particularly fruitful time for me in terms of research, and a large portion of the credit for this goes to the incredible energy, talent and organisation skills of Dr Chandu Visweswariah of IBM T.J Watson Research Center. It was his observation that the delays of gates are much closer to being perfectly correlated rather than being completely independent that led to the results of the latter half of this thesis. He also helped me create a basic software framework to test my ideas, and was available for consultations even after the end of his sabbatical period at the university.

The last four years in which I have been in the ES group have been tumultuous. This period witnessed a sea-change in the composition of the group, and a redefining of its research focus. This period would have been even more unsettling if I did not have the pleasure of getting to knowing a number of people, and having many interesting conversations with them during the coffee breaks. I must especially mention Etienne Jacobs, Jeroen Rutten and Marc Geilen for informing me about various aspects of Dutch culture; Calin Ciordas, Aleksandr Beric, Qin Zhao and Carlos Alba Pinto for a lot of interesting information relating to their own countries. I thank them for

ensuring that coffee breaks were never dull.

By its very nature, research is a roller-coaster affair with long periods of comparative inactivity followed by short bursts of inspiration. It is crucial to avoid being completely disheartened during the barren period. I am thankful to my parents whose love and encouragement could always be counted upon, and my brother Srikanth, whose interest in my work and advice helped me keep things in perspective, especially during bad times. Let me conclude by especially acknowledging the contribution of my father whose own research career first inspired me to pursue a career in research.

Contents

Chapter 1

Introduction

1.1 Variability

Variation of process parameters in modern semiconductor fabs has now assumed such significance that it must be taken into account to ensure that the fab returns a profit. The topic of variability in the manufacture of digital integrated circuits has just started to receive a lot of attention [9] ,[69], [70], [71], [72], [56]. It is recognized that variability must be handled all across the design process and not merely at the manufacturing stage. Variability causes "yield" loss where yield of a semiconductor process can be defined as follows:

$$Y = \frac{\text{Number of Working Circuits}}{\text{Total Number of Manufactured Circuits}}. \tag{1.1}$$

We need to define what we mean by a "working" circuit, as the variation of manufacturing parameters usually affects chips in one of two ways: (a) the variability causes "catastrophic" yield loss, i.e., manufactured chips may be functionally incorrect and hence cannot be sold or (b) the variability causes "parametric" yield loss, i.e., the manufactured chips may not be as fast as they were designed to be, or may consume more power than they were designed to consume, but the chips can still be sold albeit at a lower price.

It is easy to see why the first kind of yield loss is the one that has received the most attention to-date. Functional faults in chips are caused by deposition of excess metal to link wires that were not supposed to be linked (bridging faults), or the non-deposition of metal leading to opens. A standard method

of analysing how susceptible a given layout is to "catastrophic" yield loss is to study its critical area. This is an examination of the geometry of the layout to see where a blob of extra metal is likely to cause a bridging fault, or where the absence of metal is likely to break a connection. Further, dust particles could land at arbitrary locations and cause shorts and opens depending on the conductivity and size of the particles. Techniques to handle catastrophic yield loss include critical area minimisation, redundant via-insertion and wire bending/spacing.

The parametric yield loss problem has begun to assume importance, especially in the competitive microprocessors market where there is significant market advantage to be had in designing fast chips or limiting power consumption. Parametric yield loss has for long been in the domain of analog synthesis and so the methods to handle it are also drawn from that domain. These include design centring and design for manufacturability. Figure 1.1 shows in a nutshell variability, and its impact on circuit behaviour.

The quality of a semiconductor processing line and its profitability are determined by the number of manufactured chips that meet the manufacturing specifications. The manufacturing specifications include such basic ones as requiring the circuit to operate correctly according to its functional specifications. Other specifications include a limit on power consumption and the operating clock frequency of the chip. The clock frequency of a manufactured chip is determined in part by the maximum propagation delay of logic blocks contained within the chip. Determining the propagation delay of a circuit is the traditional timing analysis problem. At this point, it is important to state that timing violations in a chip do not always lead to a parametric fault; sometimes they can lead to functional faults as well. There are two types of timing violations in a chip: set-up violations and hold violations. Set-up violations occur when the propagation delay of a logic block between a set of latches exceeds the clock period, preventing the correct signal value from being latched at the next clock pulse. In this case, slowing down the clock will increase the clock period and enable the right signal value to be latched. Hold violations, on the other hand occur when logical changes initiated by the current clock pulse at a particular set of latches race through the combinational logic and become available at the inputs of the subsequent set of latches in time to be clocked by the current clock pulse. This is wrong because the intention is for the next clock pulse to latch the signals initiated by the current clock pulse. Hold violations cannot be taken care of by slow-

ing the clock down. Instead wither additional stages of combinational logic must be introduced between sets of latches, or clock skew must be reduced so that it does not compare with the propagation delay of the logic block. Thus only set-up timing violations lead to parametric faults.

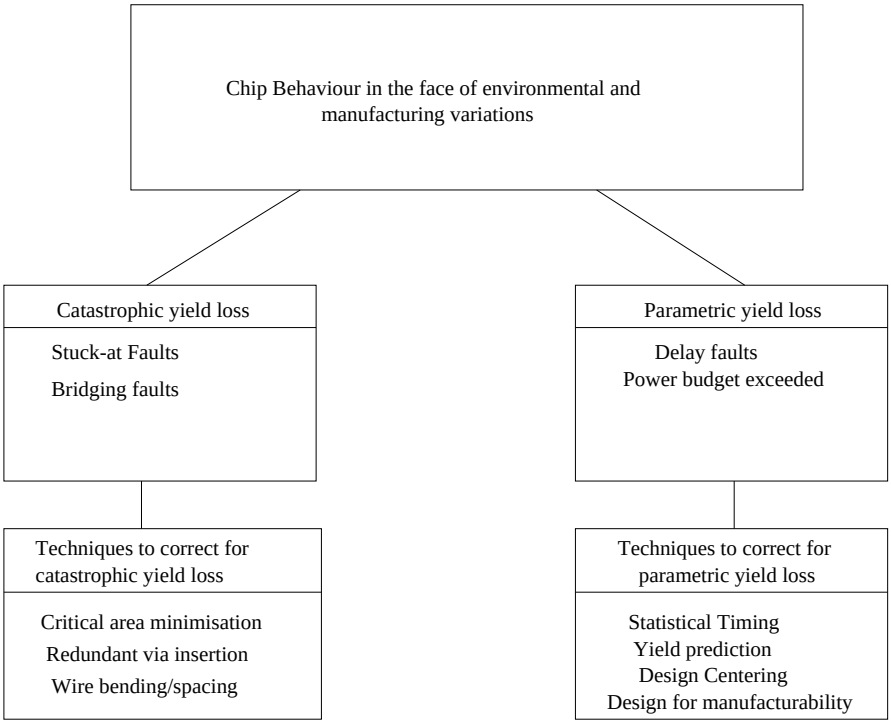


Figure 1.1: Variability and its impact on manufacturing.

The problem of determining the timing of a chip in the face of uncertainty has traditionally been solved by *worst-case timing analysis*, a linear time procedure that would determine the timing performance of a chip by letting components assume their worst case delays. The underpinning of this technique was that if the circuit worked correctly under the most pessimistic conditions, then it would work well under normal conditions. Therefore designing for extreme conditions would automatically take care of the nominal case as well. A combination of factors in recent times has led to a resurgence in research

interest in the field statistical modelling and design of integrated circuits. The first factor has been the shrinking feature sizes which have increased the importance of manufacturing variations. At the same time, there has also been a decrease in the amount of tolerance of integrated circuits to manufacturing variations. The more stringent performance requirements mean that the designer has less room to manoeuvre and the lower tolerance of circuit delay to manufacturing variations means that there is considerable difference between the worst-case solution and the average-case solution. The design solution suggested by worst-case analysis may violate the high performance requirements while the one suggested by accurate statistical techniques might still be able to meet these requirements. The need to accurately characterise manufacturing variations and their impact on design therefore becomes more acute. It is inevitable that statistical methods will need to be employed to bolster the design process in the face of manufacturing variations.

It must be noted that statistical design and analysis in the context of semiconductor manufacturing has been going on for many years. There has been significant research effort but very little in terms of industrial implementation. The main reasons for this according to [27] are the complexity of most statistical design techniques, the orthogonality of statistical design techniques to the worst-case methods used in industry making their adoption in an industrial environment very hard, the difficulty of constructing accurate process variation models, and until recently the perceived poor cost-benefit ratio of such techniques. Building statistical circuit models is an expensive process, since we need to have an elaborate understanding of semiconductor process conditions, and their interaction. This data is typically very hard to collect. Even when the data is collected, it remains to analyse the data and build statistical models. The complexity of modern semiconductor manufacturing processes means that one is quickly overwhelmed by the sheer amount of data in the system. The main shortcoming of statistical methods has been noted by [27] to be their sheer complexity. They also conflict with the worst-case techniques currently used in the industry. Coupled with the cost of implementation, this means that statistical methods have up until now been regarded as being unviable.

In the next few sections we will describe current worst-case (static) timing analysis methods and their shortcomings in greater detail. Then we shall describe *statistical circuit modelling* and describe very briefly the sources of process variation.

1.2 Worst-case circuit modelling

A worst case circuit modelling approach assumes that the worst process and operating conditions exist simultaneously and determines the delay of each circuit element under these conditions. Then we perform a static timing analysis to determine the output delay. Static timing analysis is a linear time procedure where the circuit is traversed in a breadth-first fashion to determine the output delay.

The basis of this technique is that if a circuit functions according to design specifications under extreme operating conditions, then it will operate under normal conditions as well. This intuitive technique is currently very popular in industry because of the ease with which it can be implemented but it suffers from some drawbacks. The first of these drawbacks, already alluded to in the previous section, is the pessimism it induces in design. This is typically due to the fact that it is highly unlikely that all process parameters will assume their worst-case values at the same time; in other words, the assumed worst-case process corner may not fall in the feasible region. The other drawback is that worst-case methods cannot provide information to the designer about the sensitivity of the design to various process parameters, which can potentially be very useful in guiding the designer to a more robust design.

Although worst-case analysis might seem easier than computing the entire probability distribution curve, [9] argues that in fact the effort required is more than what is needed to compute the yield. This is because the methods that compute worst-case process parameters need the complete statistics of the circuit performance of interest. The saving grace of worst-case methods is that structurally similar circuits tend to have the same worst-case process parameters, so that the cost of finding the worst-case parameters can be amortized across several circuits.

Worst-case approaches may be found in the works of [1], [20]. The work of [68] examines the impact of worst-case circuit modelling on predicting circuit performance.

1.3 Statistical circuit modelling

Statistical Circuit models do not make any assumptions about the combination of process conditions which causes the worst performance. Instead these models attempt to characterise each process parameter statistically, and then use circuit simulators to propagate the process statistics to the circuit responses. The first step in constructing statistical models is to characterise the underlying process variations.

Process Disturbances:
A process disturbance is defined by [27] as any random phenomenon which causes a change in the physical characteristics of the manufactured product. In the context of semiconductor manufacturing processes, two different types of process disturbances have been considered by researchers: (1) *Defects*: Defects are isolated events in the manufacturing process which usually lead to catastrophic yield loss i.e., chips that do not work at all. Common defects are spot defects such as spots of metal causing a short between a pair of wires, which could lead to bridging faults and opens which could lead to stuck-at faults. Researchers have come up with critical area models that determine the sensitivity of a layout to spot defects [61]. All in all, this is a relatively well-studied problem.
(2) *Parametric variations*: These are random variations in operating conditions such as temperature and supply voltage, or in material properties such as channel length of individual transistors, thickness of gate oxide, or in optical properties of semiconductor manufacturing equipment. Typically parametric variations do not cause chips to fail completely; rather they cause chips to have have higher power dissipation or greater clock frequency than the intended target. There are several types of parametric variations. They can broadly be characterised as

(a) *Between-die variations*: These parameters are constant across a single die but vary from die-to-die assuming different values for dies at different locations on the wafer. They are also known as inter-die variations. In [9], inter-die variations are broken up mathematically as follows:

$$P_{interdie} = P_{fab-to-fab} + P_{lot-to-lot} + P_{wafer-to-wafer} + P_{die-to-die} \qquad (1.2)$$

In the above equation $P_{interdie}$ refers to the total variation in some parameter P between two nominally identical die that could have been drawn from the same wafer, or from two different wafers from the same lot, or from two

different lots or even from two different fabs. Each contributing term in the above equation can be thought to arise from different physical sources.

(b) *Intra-die variations*: These variations cause a loss of matched behaviour between devices on the same chip. The intra-die variations may be further classified into two-types:

(i)*feature scale variations*: These are random variations from one location to another within the same die. An example of this type of variation is the number of dopant atoms at any location in the die.

(ii)*die scale variations*: These are variations that are deterministic within a die. An example of this type of variation is the variation in printed line widths caused by aberrations in the stepper lens.

(iii) *Operating Variations*: These are variations in the operating conditions of a chip. They are typically temperature variations, power supply variations, temperature gradients and signal noise.

Some variations may have components at both the die level as well as the wafer level. An example is the ILD thickness that shows a smooth variation across a wafer and a more random variation within a die. Given the complexity of the manufacturing variations, the challenge in statistical circuit modelling is to come up with a statistical model that can model all the variations and yet efficiently propagate the statistics to the level of the circuit responses.

The variation in a certain parameter p_i for a circuit element i can be written as follows [27]:

$$p_i = p_{interdie}(d_i) + p_{die}(x_i, y_i, d_i) + p_{feature}(d_i) \tag{1.3}$$

Here (x_i, y_i) describe the location of circuit element i. The vector d_i is a vector of design parameters such as the lengths of transistors. The component $p_{interdie}$ represents the between die variations in the parameter and is a function of the design vector d_i while p_{die} represents within-die variations and $p_{feature}$ represents feature-scale variations within a die. Inter-die and intra-die variations are shown in Figure 1.2.

Process variations take place not only at the level of devices in modern integrated circuits but also at the level of the interconnect. Indeed, the seriousness of interconnect variations on the clock skew of modern microprocessors can be gleaned from [56], [45].

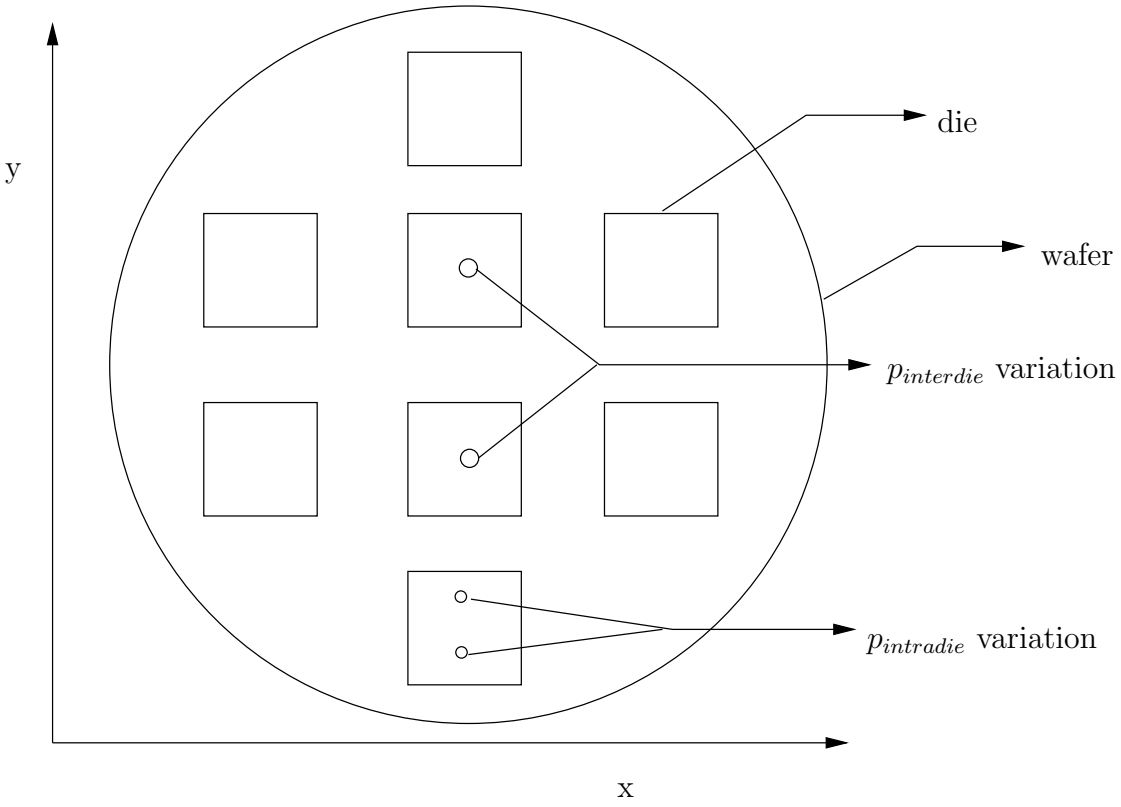


Figure 1.2: Conceptual illustration of inter and intradie variations.

1.4 Modelling process variations

Process variations can be modeled in a vector space using one of the following sets of parameters [27]:
(a) Process parameters such as diffusion times and oven temperatures.
(b) Physical parameters such as the widths of metal lines and the channel doping of transistors.
(c) Model parameters such as the parameters of the BSIM3 transistor model.
(d) Electrical parameters such as the resistance of metal lines and the saturation current of transistors.

The model parameters can be easily derived from the electrical parameters. Electrical parameters can also be easily obtained from mature manufacturing processes. However, the electrical parameters bear a complex relationship with each other which complicates the task of constructing an accurate model of them. The relationship between process parameters, on the other hand, is relatively simple. In fact diffusion times and oven temperatures are statistically independent which is an advantage. However, there are many process parameters and they are typically difficult to measure. Accurate and complete process and device models are needed to construct circuit delay models. In order to model process variations there is a need to make many measurements that capture all the important sources of variation. However, the manufacturing process is a continuously evolving process and the required process data are difficult to obtain. This makes modelling in process parameter space an unattractive proposition.

Process variations are generally modeled in physical or model parameter space. The relationships between physical parameters are much simpler than the relationships between electrical parameters. Many parameters of the BSIM3 model are very closely related to physical or model parameters making the mapping task very easy. For example, channel doping concentration and oxide thickness are more simply related than electrical parameters such as saturation current and threshold voltage. It is easy to characterise model parameters from the BSIM3 model from measured electrical data.

1.5 Building statistical models

Simulation or direct measurement can be used to obtain data to build statistical models of model parameters. It must however be kept in mind that data collection cannot proceed in the ideally desired manner of measuring data on mature processes that are later used to manufacture circuits. Rather processes are continuously developed and measurements on production wafers must proceed in parallel with manufacturing chips. It is here that simulation and so-called short-loop experiments come into play.

Direct parameter extraction is described in [79]. Statistical models are derived from process specifications in [83]. In this work, principal component analysis is used to reduce the complexity of statistical models. The basic

idea of principal component analysis is to describe the statistical information contained in n parameters using m independent parameters, where m is much smaller than n. This reduces the dimensionality of the space. In [62], an approach is described to extract process parameters from electrical test data.

The literature lists several attempts to derive statistical models of CMOS circuits. In [2], the objective is to relate individual transistor delays to geometrical and noise parameters. The approach of [99] attempts to statistically characterise digital IP libraries.

The work of [74] takes a different approach and eschews the use of principal component analysis citing the reason that the statistical data is too heterogeneous to permit the use of principal component analysis. Therefore they propose a "direct sampling" methodology, wherein they avoid the use of erroneous statistical inferencing and rely as far as possible on the data itself. They characterise a statistically significant number of data sites and build a device parameter set for each site. For each physically distinct site, they transform the collected data to a SPICE [46] set. Then they conduct SPICE simulations on a few illustrative sample circuits (such as an inverter) with the collected SPICE data sets.

There are several ways to model the joint probability density function of process parameters. The most obvious way is to model the distribution as a multi-variate normal distribution, but other possibilities including a nested distribution are possible [34] in order to better handle correlations between these parameters, such as the well-known dependence between threshold voltage and gate-oxide thickness.

1.6 Modelling circuit responses to process variations

The output delay of a circuit can be seen to be a function of the design parameters, and the process and environmental parameters. If we denote delay of the circuit by z then we can express z as

$$z = F(d, p), \tag{1.4}$$

where d is the vector of design parameters (the lengths and widths of individual transistors), and p is the set of process parameters. The function F is conceptual and can be thought of as a simulator or a response surface model constructed from measured data.

The traditional means of calculating circuit responses to process variations has been the tool, SPICE [46]. SPICE is a circuit simulator and would appear to be the ideal candidate to propagate process variations to the output of circuits. An approach is presented in [90] that incorporates statistical information into SPICE models. The simulator solves differential equations representing the relationships between circuit parameters and is very expensive computationally. However, it is used as a gold standard in the industry. The high computational requirements of SPICE have led to the development of less accurate simulators which save computation time. These simulators use response-surface models which are polynomial approximations to circuit data, as a replacement to the complex equations used in SPICE. For example a response surface model suggested in [27] uses a quadratic approximation to model design parameters and a linear approximation to model process and operating parameters:

$$r = \alpha + \beta^T d + \frac{1}{2} d^T \tau d + \xi^T p + \epsilon(d, p). \tag{1.5}$$

In the above equation d stands for the design parameters and p stands for the process variations. The use of linear response surface models to perform statistical design analysis of large logic circuits is illustrated in [59] which also studies the impact of process variation on circuit performance. The construction of response-surface models has been the subject of much study. Latin Hypercube sampling [28] attempts to place points throughout the parameter space and thus construct more accurate models. Response-surface model construction can also be time-intensive. Let us present a concrete example of obtaining a linear model of gate delay variation with respect to L_{eff}.

An elaborate derivation is given in [76] to model the dependence of gate delay on the length of the gate. We shall recount the salient details of the analysis of [76]. According to the compact gate delay model, we can write the delay of an individual CMOS gate as follows:

$$d = \frac{C_L V_{dd}}{n} (I_{dn}^{-1} + I_{dp}^{-1}). \tag{1.6}$$

In the above equation, I_{dn} and I_{dp} are drain currents of NMOS and PMOS transistors, V_{dd} is the supply voltage, n = 3.7 and C_L is the load capacitance. The saturation current is expressed as follows:

$$I_{dsat} \approx L_{eff}^{-0.5} T_{ox}^{-0.8} (V_{dd} - V_t). \tag{1.7}$$

Assuming that $L_{eff} \approx L_{gate} = L$ we can write $C_L \approx L.W.C_{ox}$. Finally we arrive at

$$d = kL^{1.5}. \tag{1.8}$$

If we imagine a gate to be driving another gate as in a path of gates, then we must interpret equation (1.6) in terms of C_L being the load capacitance, and I_{dn} and I_{dp} being the drain currents of the driving gate. In light of this (1.8) must be interpreted as follows:

$$d = kL_{driver}^{0.5} L_{load}. \tag{1.9}$$

The path delay is simply the sum of the delays of the gates in the path and can be written as follows:

$$d = k. \sum_{i=1}^{i=n} L_i^{0.5} L_{i+1}. \tag{1.10}$$

We can now write the first-order expansion of the delay of a gate around its nominal point as follows:

$$d_i = d_0 + \frac{\partial d}{\partial L_i} \Delta L_i + \frac{\partial d}{\partial L_{i+1}} \Delta L_{i+1}. \tag{1.11}$$

Differentiating (1.9) we obtain

$$d_i = d_0 + \frac{0.5 d_0}{L_0} \Delta L_i + \frac{d_0}{L_0} \Delta L_{i+1}, \tag{1.12}$$

where d_0 and L_0 are nominal values of the delay and length of the gate respectively.

The above derivation can be performed with regard to the other global sources of variation to obtain a linear model for gate delay variation. We shall revisit this model in Chapter 4.

1.7 Outline of this thesis

This thesis deals with the problem of determining the probability distribution of circuit delay for a combinational circuit given certain assumptions on the nature of the random phenomena affecting the delay of individual circuit components. As described previously, there are many different kinds of manufacturing variations and considerable research effort has been devoted to quantifying them. However, the most complicated statistical model that one can come up with that covers the entire gamut of manufacturing variations does not lend itself to efficient implementation at the level of statistical timing analysis. In practice therefore we must make trade-offs between the accuracy of the statistical model and the efficiency of its implementation in a circuit consisting of thousands of circuit elements. This thesis is an effort in this direction.

In chapter 2, we review existing literature on the statistical modelling of delay in a combinational circuit. We introduce the concept of a timing graph. Then we relate the timing analysis problem for combinational circuits to the PERT (Performance Evaluation and Review Technique) problem. This latter problem is well-studied in the operations research literature. We study how the results of this body of work impinge on our own research.

In chapter 3, we consider a relatively simple model that assumes that paths in the circuit are correlated only due to path sharing(i.e., the delays of gate elements are independent random variables). We come up with a statistical timing method to determine the circuit delay distribution curve. The model is shown to be very time-consuming for real circuits, but has some redeeming features such as being able to handle any sort of distribution for the gates. We also show how to come up with upper and lower bound curves on the real probability distribution of circuit delay.

In chapter 4, we consider a more refined delay model. This is the model that we employ for the remainder of the thesis. We show how this model captures two different types of correlation - that induced by path sharing and that induced by spatial correlation of component delays. The implications of this model are considered and its weaknesses discussed. We show how the problem of calculating the circuit delay distribution can be reduced to computing the weighted volume of a polytope in an appropriately defined *parameter* space.

In Chapter 5, we begin by exploring an elegant algorithm to solve the yield estimation problem defined in Chapter 4. The problem involves integrating a function over a polyhedral domain and is known to be computationally difficult. Hence any algorithm that computes the result to within a bounded error will have to be of exponential complexity. We discuss the exponential, but elegant, algorithm of Cohen and Hickey. We also discuss the use of numerical integration algorithms.

Chapter 6 marks the heart of the thesis. We start by discussing algorithms to find the maximum volume ellipsoid that can be inscribed within the yield polyhedron. The ellipsoid is then used as a surrogate for the yield polyhedron in both direct and indirect ways. As a direct surrogate, the ellipsoid is taken to be the yield body itself, and integration is performed over it. While integration over an irregular polyhedron is a complex task, there are good algorithms for integrating over an ellipsoid. However, the yield value returned by integrating over an ellipsoid is only a lower bound on the true yield. Therefore we change tack, and investigate how to use the ellipsoid in an *indirect* way to enhance Monte-Carlo methods. Two novel ways of using the ellipsoid to construct Monte-Carlo methods of low variance are described. The second of these methods is intended for use especially at low yields (and very fast circuits).

In Chapter 7, we return to the theme of integrating over the ellipsoid. We try to address the main problems of numerical integration techniques discussed in Chapter 6 - namely the rapid increase in the number of points needed for integration as the number of dimensions in the integral increases. Specifically, we discuss the low variance Monte-Carlo spherical integration method of Alan Genz [31],[32]. The method's various redeeming features are discussed. We show that it is polynomial in the number of dimensions, a highly desirable property considering that we may have to deal with a lot of dimensions to fully characterise random phenomena on the chip.

In Chapter 8, we show how to use the eigenvectors and eigenvalues of the maximum volume ellipsoid that fits in the feasible region, to tune the circuit so that its yield improves. The eigenvector corresponding to the smallest eigenvalue is shown to be the most promising direction of yield improvement under certain conditions while the eigenvector corresponding to the largest eigenvalue is the least promising direction for yield improvement.

Chapter 2

The PERT problem

2.1 Introduction

The complexity of modern manufacturing processes and the amount of statistical information that characterise them means that there could be many different statistical timing models that we could use to characterise the delay of each circuit element. The choice of statistical model directly impacts the complexity of the statistical timing procedure. In this chapter, we shall lay the groundwork for the remaining chapters. We shall introduce the concept of a timing graph for static timing analysis [39] and show how it is closely related to the project completion diagram of the PERT problem. The PERT problem is a longstanding area of research in the field of operations research and we review the literature from operations research to see how it might impact our own problem.

A brief history of research into the PERT problem is presented followed by a theoretical result that has negative implications for the problem we intend to solve. We end the chapter by reviewing the different techniques used in statistical timing analysis, and consider the pros and cons associated with these techniques.

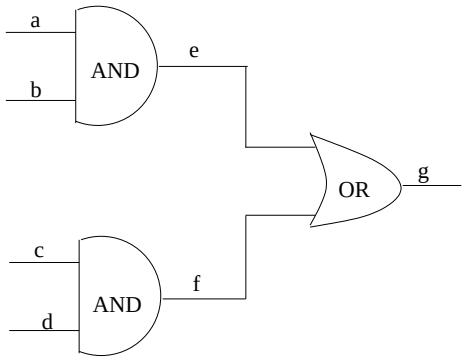


Figure 2.1: A simple combinational circuit.

2.2 Timing graph

Consider the logic network shown in Figure 2.1. For every gate in the network there exists a signal propagation delay from every input of the gate to the output of that gate. This signal propagation delay is usually different for different input-output combinations, and also depends on whether the signal itself is on a rising or falling transition. Further signal propagation delay depends upon the slew of the input signal, or in other words the slope of the rising or falling transition. The timing analysis problem is to determine a lower and upper bound on the delay of the entire circuit given the input-output delays for every gate.

Given the dependence of gate delay on the logical values of the inputs and the nature of the input transitions, it might seem that timing analysis should be performed for all possible logic inputs. However, this is computationally prohibitive as there are 2^n different input combinations possible where n is the number of inputs. Therefore both static timing analysis and statistical static timing analysis ignore the effects of logic on gate delay. This simplification makes the static timing analysis problem tractable but as we will see later, the statistical timing problem remains non-trivial.

We can represent the timing information contained in a logic network in the form of a *timing graph* shown in Figure 2.2. For real circuits, timing edges are also introduced for each wire delay but this is not shown in Figure 2.2. The timing graph can be defined as follows:

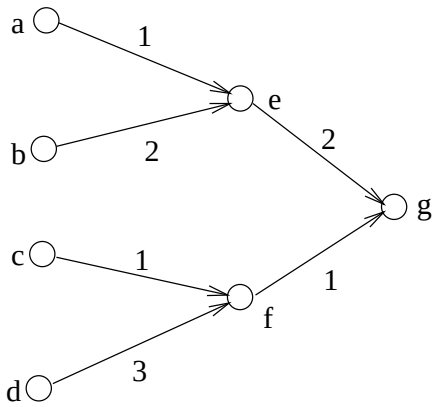


Figure 2.2: Timing graph corresponding to the circuit in Figure 2.1.

Definition 2.1: A timing graph G corresponding to a logic network C consists of a set V of nodes and a set E of edges such that every signal line in C is represented as a node in V and every input-output pair of every gate in C is represented as an edge in G. The signal propagation delay associated with an input-output pair is represented as a weight on the corresponding edge in G.

Figure 2.2 represents a timing graph corresponding to the logic network of Figure 2.1. With a slight modification, it could just as easily be a project diagram in the context of the well-known operations research concept called PERT (Performance Evaluation and Review Technique). PERT diagrams have "tasks" and "task durations" and precedence relations between different tasks. The problem in PERT is to determine the project completion time. There would thus seem to be an analogy between PERT and our own problem of determining the output delay of a given logic network. We shall explore PERT in more detail, and study long-standing techniques used to solve this problem and see how they might be relevant to our situation.

2.3 The PERT problem

A variety of approaches have been proposed in the literature to tackle the problem of statistical timing analysis. There has been a research effort in the

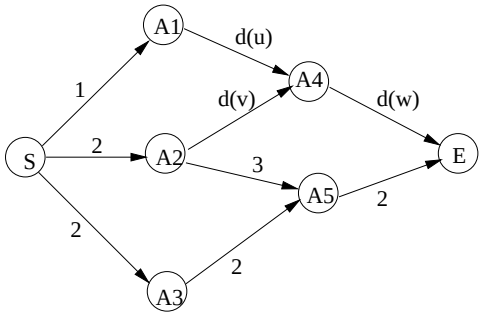


Figure 2.3: A PERT task graph.

domain of operations research (PERT) to characterise the project completion time probability distribution. The domain of this research is very close to the problem considered in this thesis; however, the assumptions do not always carry through from one domain to another.

Let us begin by defining the PERT problem more accurately. PERT refers to the Performance Evaluation Review Technique and was developed by operations researchers to deal with the growing complexity of project management. A graph is defined where the edges represent project tasks and the nodes points at which some tasks end and others begin. A task duration is associated with each task. The problem for the operations researcher is then to determine if the project network so defined has an acceptable completion time. A task graph relevant to PERT is shown in Figure 2.3. [6] is a recent survey on the techniques used to solve the PERT problem.

2.4 The PERT problem and Combinational logic networks

As has already been mentioned in the introduction to this chapter, there is a strong similarity between PERT networks and (combinational) logic networks. This similarity becomes stronger when one ignores the logic of the gates in a logic network. This simplification, without which even static timing analysis would be a computationally prohibitive task, is widely assumed in the literature dealing with statistical timing analysis of digital circuits.

There are some notational differences between PERT networks and logic networks but most of them are superficial. Most PERT networks have a source node and a sink node, whereas logic networks have many primary inputs and usually more than one output. One could transform a logic network into a PERT network by creating a fictitious source node and joining it to all the primary inputs of the circuit. Similarly one can create a fictitious sink node and join all outputs to it. Thus there is a clear correspondence between requiring that all signals arrive at their outputs in the logic network by a certain time, and requiring that the PERT project completion time not exceed a given time. The PERT literature has two ways of specifying a PERT diagram: either the nodes represent tasks and the edges represent the precedence relations between tasks, or the edges represent tasks and the nodes determine the point when some tasks end and new ones begin. The latter model corresponds very closely (except for source and sink nodes) to our notion of a timing graph but this correspondence is not crucial to establish the equivalence between PERT and the timing analysis problem in logic network. This is because [6] has shown that PERT networks with delays on their nodes can be transformed into networks with delays on their arcs and vice-versa.

The PERT problem has been studied in a wide variety of contexts. Researchers have studied the distribution of project completion time assuming project delay distributions to be Gaussian, beta, exponential or gamma distributions. Discrete distributions have also been the subject of study [58].

2.5 Solving the PERT problem

Solving the PERT problem in the deterministic case is easy - indeed, one can adopt the same approach as used in static timing analysis. First one levelizes the network. Starting at the lowest level (closest to the source node), one determines the earliest start time of a task as the maximum of the completion time of the tasks preceding the given task. Then one repeats this process for nodes at successive levels, and thus one can arrive at the sink node. Figure 2.3 shows a node $A4$ in a PERT network with weighted edges leading into it from nodes $A1$ and $A2$. Let us denote the tasks on these edges as u and v. There exists a task w leading from node $A4$ to node E. The earliest time at which a task leading out of $A4$ can start is given by:

$$T_{A4} = max(T_{A1} + d(u), T_{A2} + d(v)). \tag{2.1}$$

Here $d(u)$ and $d(v)$ represent the durations of the tasks u and v respectively. The above equation denotes a recurrence where the start time at a given node is established in terms of the start times at its predecessor nodes.

The probabilistic case of the above equation is considerably more involved because the basic variables become random variables instead of simple deterministic variables. The operations of addition and maximisation, trivial in the deterministic case, become more involved in the probabilistic case as we are concerned with the addition of random variables and maximisation of several random variables. The output project completion time becomes a projection completion time distribution. But if the addition and maximisation operations on random variables were the only complication, probabilistic PERT would still remain a tractable problem. The real killer is the appearance for the first time of *correlation* between random variables. It is correlation which makes the probabilistic PERT problem truly hard, as the discussion in the next section will further dwell upon. In this thesis, we shall be concerned with solving the probabilistic PERT problem in the context of statistical timing analysis problem for digital integrated circuits.

Given the complexity of the PERT problem even in the case of simple discrete distributions (which we shall postpone to the next section), many researchers [65], [26], [52], [49], [82] have turned their attention to providing good upper and lower bounds for the distribution of the project completion time. We shall describe the salient aspects of the work of these researchers. First we shall need some definitions:

Definition 2.2 [87]: (Stochastic ordering) A random variable X is said to be stochastically smaller than another random variable Y in the strong sense, if $F_X \geq F_Y$, where F_X is the cumulative distribution function of X, i.e., $F_X(z) = P(X \leq z)$ and F_Y is the cumulative distribution function of Y. X would be stochastically larger than Y if $F_X \leq F_Y$. The concept of stochastic ordering is illustrated in Figure 2.4.

An alternate definition of stochastic ordering in the strong sense is the following:

Definition 2.3 [87]: X is said to be stochastically smaller than Y in the strong sense if the expectation $E(g(X)) \leq E(g(Y))$ for all increasing func-

tions g such that both expectations exist. A weaker form of stochastic ordering is the following:

Definition 2.4 [87]: X is said to be stochastically smaller than Y in the increasing and convex sense if the expectation $E(g(X)) \leq E(g(Y))$ for all increasing and convex functions g such that both expectations exist.

An early attempt at statistical timing analysis for digital circuits was made by [65]. This work recognises the complexity of the problem and makes clear that even if one were to assume relatively simple distributions for the delays of the tasks, one still has no guarantee of obtaining a closed form solution to the problem of determining the output delay distribution curve (here project completion time and output delay should be taken to mean the same thing). The paper makes very general assumptions about the correlation between task completion times - no correlation structure is assumed. With these assumptions, [65] derives an upper bound on the expected tardiness of the project completion time as a function of time t, $E(max(0, C_{max}(p) - t))$ where C_{max} is the project completion time.

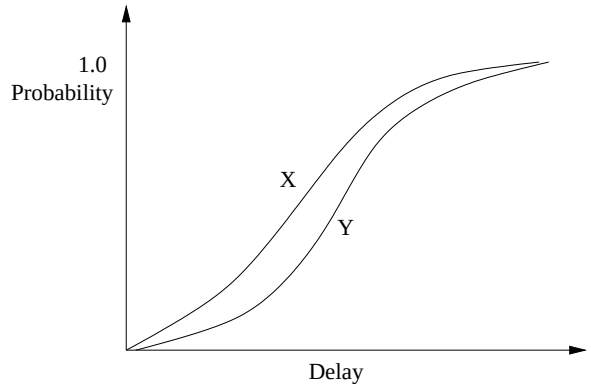


Figure 2.4: Stochastic Ordering. In the figure the cumulative distribution curve marked X is stochastically smaller than the curve marked Y because $P(X \leq t) > P(Y \leq t)\ \forall t$.

While the bounds of Nadas [65] are weak because they are based on convex stochastic ordering, the bounds of Kleindorfer [52] are based on stronger stochastic ordering. Given random variables X and Y, Kleindorfer's method makes use of the following set of inequalities:

$$F_X.F_Y \leq F_{max(X,Y)} \leq min(F_X, F_Y). \tag{2.2}$$

The method first topologically sorts the nodes of the PERT network. For $i, 1 \leq i \leq n$ let i denote an activity which starts at node u_i and ends at node v. Then given upper bound distributions for the start times of the activities i, Kleindorfer's method constructs an upper bound

$$F_V = \prod_{i=1}^{i=n} [F_{U_i} * F_i]. \tag{2.3}$$

In order to understand the above equation let us look at what happens at the level of random variables representing completion times of all activities at a node. Let U_i represent the start time of activity i. This is also the completion time of all activities leading upto the node U_i. Then the completion time of all activities leading upto V, or the start time of any activity beginning at V is given by

$$V = max(U_i + d(i)), 1 \leq i \leq n, \tag{2.4}$$

where $d(i)$ denotes the duration of the task i. The convolution operation of distribution functions represents the addition of the corresponding random variables. For random variables U_i and $d(i)$, the distribution function corresponding to $U_i + d(i)$ is given by $F_{U_i+d(i)} = F_{U_i} * F_{d(i)}$ where the $*$ operator denotes convolution. It is here that the crux of Kleindorfer's method may be found: the max operation over several random variables is replaced by the product operation over the corresponding distribution functions in accordance with the basic bounds described above. The operation we have described for a single node is carried out in topological order for the entire network, giving us a stochastic upper bound for the project completion time distribution. For the stochastic lower-bound, the product operator is replaced by the point-wise minimum operator.

Dodin [26] considers reducing the given PERT network to a series-parallel network to which the basic bounds of equation (2.2) are applied.

2.6 Theoretical complexity of the PERT Problem

It is useful to study the complexity of the PERT problem in a deep computational sense because it gives us a direct insight into our own problem. Hagstrom [36] has shown that the problem of determining even a single point on the completion-time distribution of a PERT network, in general, is #P-complete even if we assume that each task has a two-point distribution (i.e., a discrete distribution with two points in the sample-space). The complexity may be reduced in case the project tasks have more complicated distributions since the encoding of more complicated distributions takes more space than for simpler distributions. However, there is compelling evidence that the PERT problem remains hard even then, since there is an exponential number of equally critical paths in the network. Further even if project tasks have independent distributions (a hugely simplifying assumption in the case of PERT and digital circuits), the paths will still be correlated on account of path sharing. This will complicate the task of finding the output delay distribution.

2.7 Previous research in statistical timing analysis

Most methods in statistical timing analysis in the context of digital circuits can be divided into two broad categories: path-based approaches and block-based approaches. Path-based approaches follow in the tradition of research into PERT networks - critical paths starting from primary inputs to outputs are identified according to well-known algorithms and then statistical timing analysis is performed on these paths. The conceptual framework of path-based approaches is shown in Figure 2.5. Note that the path collection process is separated from the actual statistical-timing process. This clear division of labour permits us to use sophisticated path-collection algorithms in conjunction with sophisticated statistical timing algorithms. Path-based methods are also well-adapted to handling correlations between gate delays and path sharing. The downside of using path-based methods is the sheer number of paths that the designer must contend with for even a moderate-

sized circuit.

Nadas [65] was among the first to propose a path-based approach to solve the statistical timing analysis problem in all its generality. His work assumes that the correlation dependence between gate delays is unknown and also takes into account correlations due to path sharing. Unfortunately, with these general assumptions, only very weak bounds can be derived for the probability distribution of output delay. The work of [39], [40] must also be included in the list of path-based statistical timing approaches. The efficient generation of the top K critical paths in a circuit is central to path-based approaches; this topic is covered by [97],[98]. Two flavours of the critical path generation problem are presented: the first collects all paths with delays greater than a given threshold and the second collects the top K paths in decreasing order of path delay.

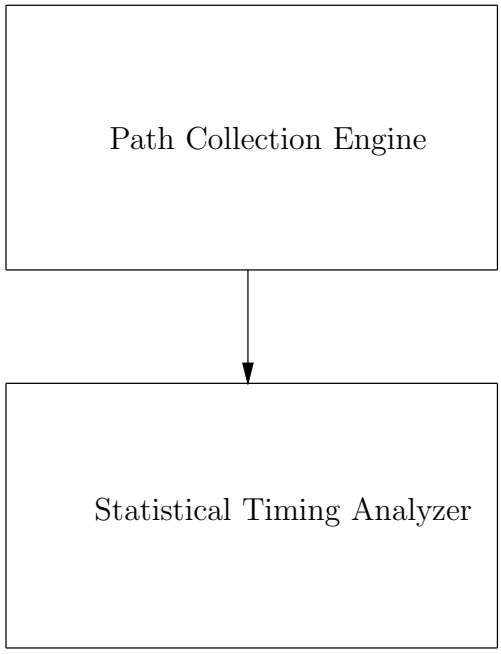


Figure 2.5: Conceptual illustration of path-based approaches.

Block-based methods, on the other hand do not generate paths in the circuit but work through a process similar to depth-first search to calculate the final probability distribution. Delay equations are established for individual gates

and the final output-delay distribution is calculated by a tree-walk. The main advantage here is undoubtedly the fact that millions of critical paths are not generated, but the significant downside is the inability to handle correlations in a computationally efficient manner. As will be shown subsequently, these methods are exponential even if we were to restrict ourselves to correlations due to path-sharing. The basic idea behind block-based methods is illustrated in Figure 2.6. [18] represents an approach to block-based statistical timing. [47], [48], represent block-based approaches to statistical timing analysis that can additionally focus on false-path removal which is not covered in this thesis.

The papers [47], [48] use the "delay lumped at a gate paradigm," which is however not a serious restriction in their analysis. However, they do make the strong assumption that gate delays are bounded random variables. Their method of statistical timing analysis essentially consists of a symbolic simulation of the circuit with delays sampled from the respective distributions, together with a pruning technique. More recent block-based approaches include the bounds approach of [3] which tries to construct an upper and lower bound on the circuit delay distribution. Novel block-based approaches based on discrete computations are described in [55], [66], [67]. The papers [55] and [67] take into account reconvergent path correlations but not correlations due to delay dependencies.

There has been a fair amount of research dealing with the problem of removing false paths from the analysis. False paths are those which are never sensitized by the logic of the circuit, and can affect the delay distribution if they are critical. [47], [48], [91], [54] have all looked at the false-path problem and come up with methods to remove them from consideration.

It has been the focus of many researchers to accurately model correlations in the circuit, see for example [92], [55], [66], [67], [53]. Most researchers in PERT assume that the activity durations are independent. On the other hand, this assumption is not so common among researchers in statistical static timing analysis. This means that researchers in statistical static timing analysis try to contend with correlations between paths due to path sharing as well as correlations between paths due to correlations between gate propagation delays. The complexity of this problem has made Monte-Carlo analysis a popular choice with some researchers.

Monte-Carlo analysis is well known to handle any kind of statistical prob-

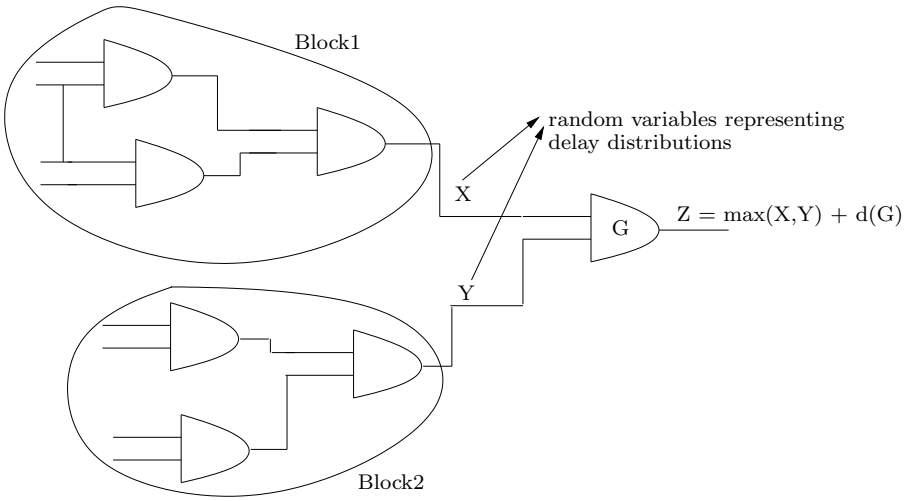


Figure 2.6: Conceptual illustration of block-based approaches.

lem - it is a brute force method that never fails, and in some cases may be the only recourse available. The method consists of several trials, each of which is a full-scale circuit simulation. In each circuit simulation, each signal propagation delay is sampled from its distribution, and then a static timing analysis is performed to get the output delay. For example [10], [15] follow this approach. The procedure is repeated over thousands of trials, and the output delay distribution is deduced from the collection of output delays. The main advantage of Monte-Carlo analysis is that it is linear time, and can handle any sort of problem. The main disadvantage is that one cannot ascertain (without a knowledge of the result!) the number of trials needed to bound a probabilistic error. Note that we cannot do better than a probabilistic error, because of the #P-complexity. Let us describe Monte-Carlo analysis for a circuit.

Suppose we define a delay requirement for the output of a circuit. We intend to determine the percentage of manufactured circuits that are likely to pass this delay requirement. Let the pass percentage be p. In order to determine p, we conduct n simulations of the circuit. In each simulation we sample the delays of every arc in the timing graph (let us assume that the arc delays are independent random variables) and perform a static timing analysis with the sampled delay values. Let us define random variable Z_i for the trial i, and

set it to 1 if the output delay in this trial passes the delay requirement, and set it to 0 otherwise. Next let us define the random variable Z as follows:

$$Z = \frac{Z_1 + Z_2 + \ldots + Z_n}{n}. \tag{2.5}$$

We can show that Z obeys a binomial distribution with mean $\mu = p$ and variance, $\sigma^2 = \frac{p(1-p)}{n}$. We can now use Chebyshev's inequality to determine a lower bound on the number of trials n needed to ascertain p with a specified error and at a specified confidence:

$$Pr(|Z - p| \geq \epsilon p) \leq \frac{(\sigma)^2}{(\epsilon p)^2}. \tag{2.6}$$

The left-hand side of the above equation represents the probability of error of the estimate Z where error is measured as the absolute value of the deviation of Z from the true value p. Chebyshev's inequality provides an upper bound on the probability of the error. In order to ensure that we have a confidence of $100c$ percent in the estimate Z with an accuracy of 100ϵ percent we must ensure that the worst-case probability of error does not exceed $(1 - c)$. In other words, the right-hand side of the above equation must be less than or equal to $(1 - c)$. Substituting for σ^2 in the right-hand side of the above equation, we have

$$\frac{(1-p)}{np(\epsilon)^2} \leq (1 - c). \tag{2.7}$$

Setting $c = 0.99$ and $\epsilon = 0.01$ this translates to

$$n \geq \frac{1-p}{p} \times 10^6. \tag{2.8}$$

Thus the number of trials needed for a certain accuracy and confidence in the distribution is dependent on the yield which is the quantity that is sought to be estimated. If we take the yield $p \approx 0.9$ then we will need about 100,000 trials. The interesting thing about this result is that the number of trials for a certain confidence and accuracy needed does not depend on the size of the circuit directly, but instead depends on the yield itself. This represents the main shortcoming of Monte-Carlo analysis - if the quantity sought is a very small probability then one will need a large number of simulations

to approximate its value with low relative error and high confidence. The reader may be tempted to look at the form of equation 2.8 and conclude that for very low yields p, it might be advisable to redefine the Monte-Carlo experiment and try to estimate the probability of failure $q = (1 - p)$. Then $\frac{1-p}{p}$ in 2.8 becomes $\frac{1-q}{q}$ which is a much smaller quantity. Thus it appears that a mere redefinition of the Monte-Carlo experiment could reduce the number of trials needed. However, this is not the case. The key here is to note the role of ϵ, the relative error. If the probability of success is 1 percent, and $\epsilon = 0.01$, then the absolute error made in the estimate of p is required to be no larger than $\epsilon p = 10^{-4}$. To achieve this same error in the probability of failure experiment, ϵ cannot be 1 percent, as this would make the absolute error $\approx 10^{-2}$. Therefore in the probability of failure experiment, a new relative error of magnitude equal to one-hundredth the relative error in the probability of success experiment must be considered. This smaller relative error cancels the gain obtained by working with the probability of failure rather than that of success.

The above analysis shows that essentially the variance σ of the binomial random variable Z controls the error characteristics of the simulation. It is instructive to note that the error of the simulation scales as $n^{-1/2}$ where n is the number of trials. In other words, if were to simulate the circuit for $100n$ trials, then the simulation error would decrease to a tenth of its value at n trials. This is independent of the dimension of the problem (i.e., number of gates in the circuit) and illustrates a fundamental property of Monte-Carlo simulation - the error rate scales as $n^{-1/2}$ *independent of dimension* [38]. In contrast numerical integration methods based on constructing a uniform grid for sampling suffer from a $n^{-1/d}$ error-scaling rate, which is clearly unsatisfactory at higher dimensions.

2.8 Quality of statistical timing methods

Statistical timing methods differ from each other in terms of the modelling assumptions they make, the quality of the results they generate and ability to produce information useful for synthesis. In this section we shall describe these aspects in greater detail, and for the rest of the thesis we shall judge the algorithms we develop in terms of the quality criteria outlined in this section.

Model Sophistication: It is not easy to construct a statistical model for the interaction between process variations and the delays of gates on a chip. There are many timing models in existence in literature, and they vary from assuming gate delays to be independent random variables to those that assume correlations between gate delays based on an underlying dependence on the same process parameters. Some models assume that gate delays vary linearly with process variations, which is reasonable if the process variations are small. Other methods try to model the non-linear effects of process variation on circuit behaviour, with increased complexity.

Computational Complexity: Depending on the models they employ statistical timing methods could be linear in the size of the circuit, or they could be exponential. The latter class of methods usually correspond to more realistic timing methods, and capture structural correlations between delays of the gates. The statistical timing problem considering correlations is a hard problem in a deep computational sense.

Error theory: It is reasonable to expect that the output delay distribution produced by the statistical timing method may differ from the true distribution. It is crucial to know the nature of the error. Most analytical methods of computing the distribution tend to not have a very good error theory. Usually these methods perform recursive calculations, and errors are made at every stage of the recursion. It is usually hard to assess how these errors propagate through the circuit. The final output delay distribution curve may look similar to one generated by exhaustive Monte-Carlo simulation, but the confidence in the accuracy of the curve generated by analytical techniques is not very great. Monte-Carlo based methods have a good error theory - no matter what the current estimate of yield is, it can always be improved by increasing the number of simulations.

Tuning information: The real benefit of statistical timing tools can justifiably be measured in terms of their ability to provide information for tuning a circuit to improve yield. Some statistical timing methods are better equipped to handle this problem than others as information relevant to tuning is computed during statistical timing analysis. Block-based timing methods typically are unable to compute the statistical timing cumulative distribution curve accurately. However, they possess the nice property of being amenable to incremental timing analysis - if a small change is introduced into the circuit by changing the delay of a certain gate, then only the portion of the circuit impacted by the gate in question needs to be recomputed. This means that

existing information can be re-used to perform optimisation. The inaccuracy of block-based statistical timing might make some optimisation moves unproductive, and others more productive. Path-based statistical timing methods, in contrast, can perform more accurate statistical timing but they are not amenable to incremental statistical timing. If the distribution of a certain gate delay is changed, then new paths will have to listed and statistical timing performed on the whole set of paths again in order to determine the new statistical timing curve. However, since path-based statistical timing is more accurate, the yield improvement directions are likely to be much more reliable which means that relatively few optimisation moves will be needed to achieve good results.

Chapter 3

A simple timing model

In this chapter we work with a simple timing model and derive some useful results. The timing model we use is one where the delays of the gates are independent random variables - this means that the correlation between path delays is due only to path sharing. This model is identical to the one taken in PERT research, and it has been shown by Hagstrom [36] that determining the cumulative distribution function of the output delay distribution is a #P-complete problem.

Let us represent the delay of an edge in the timing graph by a triangular distribution. A triangular distribution is chosen because it is "Gaussian-like" in the sense that it possesses a peak, but we are also interested in the fact that the distribution has a finite domain. Figure 3.1 displays a triangular distribution. We seek to answer the following question: Given that the delay of each edge in a timing graph is a random variable obeying a triangular distribution, determine the cumulative distribution function of the output delay of the timing graph.

It is instructive to note that if we take each gate delay distribution to be a triangular distribution, then we can in theory compute the maximum of two random variables in closed form. However, the distribution of the maximum quickly becomes more and more cumbersome as we traverse deeper and deeper into the circuit. Therefore we shall not take this route. A more appealing route suggests itself when we take into account that the domain of the distribution is finite - we can *discretise* the distribution. After discretisation the maximum and addition operations become much simpler as they

will need to be performed over discrete random variables.

In Section 3.1, we describe the basic technique as applied to fanout-free circuits, namely circuits where the output of each gate feeds only to at most one other gate. We will also introduce certain operations designed to improve the efficiency of the procedure. In the next section, we show how to extend the technique to circuits that are not fanout-free. The infeasibility of the technique will become clear for circuits that have a large amount of path sharing. In the final section, we borrow on the notion of stochastic ordering introduced in Chapter 2, and show how to develop upper and lower bounds on the delay distribution curve at the output.

3.1 The distribution used

The arrival time of a signal at the output of gate i (with two-inputs) can be modelled as

$$Z_i = max(X_i, Y_i) + G. \tag{3.1}$$

In the above equation we ignore the fact that there could be different delays from inputs to outputs. A more realistic formulation is the one given below:

$$Z_i = max(X_i + d(X_i \rightarrow Z_i), Y_i + d(Y_i \rightarrow Z_i)), \tag{3.2}$$

where X_i and Y_i are the latest arrival times of signals at the inputs of the gate, $d(X_i \rightarrow Z_i)$ is the propagation delay from input X_i to Z_i and likewise for $d(Y_i \rightarrow Z_i)$. Z_i is the latest arrival time at the output of the gate. This formulation of arrival time of the output essentially means that we assume that any transition on either of the inputs gets transferred to the output and we are calculating the latest time beyond which there are no further transitions at the output (i.e., the time at which the output becomes stable). Since the operations involved in (3.2) and (3.1) are maximisation and convolution, we stick with the simpler (3.1) in the following.

We model the delay of a gate as a random variable obeying a triangular distribution centered around a mean in the interval (mean - a, mean + a) as in Figure 3.1. Our choice of a triangular distribution is motivated by the fact that the delay of a gate cannot range from -∞ to +∞ but instead must

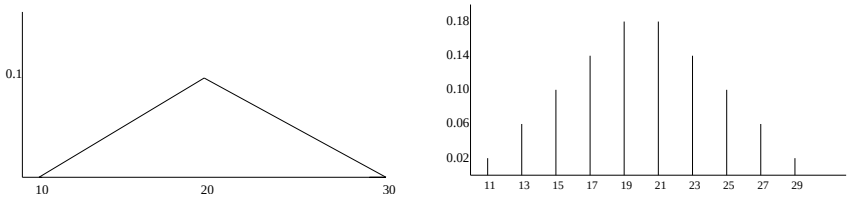


Figure 3.1: A triangular distribution and its discretised form.

have finite minimum and maximum values, and that the delay distribution must have a single peak. We would like to emphasize that the techniques in this paper can be extended to any finite-domain distribution. Consider the distribution shown in the figure below in the range (10ps,30ps) centered at 20ps.

In order to discretise the distribution we divide the interval (10,30) into 10 equal parts (10,12),(12,14)...(28,30). We concentrate the probability of the random variable lying in any one of these intervals in an impulse lying at the center of that interval. The height of an impulse is the area of the probability distribution curve over the corresponding interval range. For example the impulse corresponding to the range (10,12) is of height 0.02 which is the probability that the delay lies in the interval (10,12). The idea here is that if one were to divide the interval into a sufficiently large number of sub-intervals, then the discretised distribution will represent in a fairly accurate manner the continuous distribution. Since we calculate a train of impulses to represent the probability distribution of the delay at each gate of the circuit, we call this the "impulse-train" approach.

3.2 Computing delay random variables

In order to compute the delay using the formula of (3.1) we need to compute a maximisation followed by a convolution. Let X and Y be discrete random variables with sample spaces $S(X)$ and $S(Y)$ of sizes m and n respectively. Let $S(X) = \{p_i, 1 \leq i \leq m\}$ and $S(Y) = \{q_i, 1 \leq i \leq n\}$.

Then Z = max(X,Y) is a random variable represented by at most m+n impulses, and we can compute the probability that $Z = k$, $Pr(Z = k)$ as

follows:

$$Pr(Z = k) = Pr(X < k)Pr(Y = k) + Pr(X = k)(Pr(Y < k) + Pr(Y = k)). \quad (3.3)$$

Here

$$Pr(X < k) = \sum_{p_i<k} Pr(X = p_i). \quad (3.4)$$

The distribution for the random variable Z = X + Y can be computed by performing a discrete convolution as follows:

$$Pr(Z = k) = \sum_{p_i \in S(X)} Pr(X = p_i)Pr(Y = k - p_i). \quad (3.5)$$

Note that $Pr(Z = k)$ is non-zero only for those values of k that are given by the sum of some pair of values in the sample spaces of X and Y.

It must be noted that the equations for the maximisation and convolution operations are merely the discrete versions of their continuous counterparts. The discretisation of the distributions enables us to carry out the maximisation and convolution operations efficiently for any type of distribution as opposed to an analytical approach that attempts to compute closed-form expressions for the delay. In [8], an analytical approach is proposed assuming that the gate delays are Gaussian and noting that the maximum of two Gaussian random variables is approximately Gaussian. However, this approximation is only valid for certain situations and it is not clear how the distortions will propagate through the circuit.

3.3 Computational complexity

Let X and Y be two discrete random variables with sample space sizes $S(X) = m$ and $S(Y) = n$. Then the maximisation operation takes O(mn) time and the the random variable $Z = max(X, Y)$ has a sample space of size O(m+n). Thus the maximisation operation does not cause a blow-up in the sample-space size. However, if $Z = X + Y$, then Z has a sample space size of O(mn) and it takes O(mn) time to compute it. Thus in case of repeated convolution, there is a potential for a blow-up in the number of impulses.

In practice, the blow-up does not materialise because the number of distinct values in the sample space of Z is fewer than the theoretical number, mn. In fact, if we were to perform uniform discrete convolution where both sequences to be convolved have the same period, then the number of distinct values in the sample space of Z would actually be of O(m+n). However in a practical setting we are not always able to perform uniform discrete convolution. For instance, if the gate distributions were asymmetric then it might become necessary to position each impulse at the centroid of the corresponding strip and not at its midpoint in order to ensure that the mean of the discretised distribution is the same as the mean of the original continuous distribution. This would make the sequence non-uniform. For signal lines deep inside the circuit, the delay distribution becomes asymmetric even if we started with symmetric distributions. Generally these delay distributions rise sharply and fall off slowly due to repeated maximisation of random variables. This is all the more reason to sometimes use non-uniform discretisation at least for blocks deep inside the circuit. One way to get around the problem of a blow-up in the number of impulses is to ensure that the impulses are positioned at the integer value closest to the centroid of the strip. Then the convolution operation would result in output impulses positioned at integer values and the number of impulses would only grow according as the spread of the output distribution defined as the difference between the maximum value of the output random variable and the minimum value of the output random variable. Thus the number of impulses at the output would again be of O(m+n).

Even if we could ensure that the number of impulses at the output of each gate in the circuit is of the order of the sum of the numbers of impulses at the inputs of the gate, we could end up with nodes with a fairly large number of impulses (such as nodes close to the primary output in a circuit with large depth). To reduce the number of impulses for such nodes, we propose two techniques, with contrasting properties, to combine impulses such that the mean of the distribution is preserved and the variance is only slightly changed. In the first technique, we combine the two impulses of height m and n into a single impulse of height $m + n$ located at $d = \frac{mx+ny}{m+n}$, which is merely the "center of mass" of the two impulses. While this operation preserves the mean it does change the variance of the distribution. The change in variance can be computed as follows:

$$\delta v = \left(\frac{mx + ny}{m + n}\right)^2 (m + n) - (mx^2 + ny^2). \tag{3.6}$$

After some manipulation, we have $\delta v = -(\frac{mn}{m+n})(y - x)^2$. If we assume that $m > n$, then δv is bounded by $\delta v > -(n(y - x)^2)$. Thus in a practical scenario, we can combine impulses until the penalty in terms of variance is no greater than a small percentage of the original variance. This technique causes a decrease in the variance of the resulting distribution. We propose a second technique that increases the variance of the resulting distribution while reducing the number of impulses. In this technique, the impulse situated at y of magnitude q is combined into the impulses located at x and z such that the mean of the distribution is preserved. To do this, the impulse at x of magnitude p is increased to $p + \frac{q(z-y)}{(z-x)}$ while the impulse at z of height r is increased to $r + \frac{q(y-x)}{(z-x)}$. The change in variance is computed as follows:

$$\delta v = q(z - y)(y - x). \tag{3.7}$$

Experimental evidence indicates that the combination of these two strategies does result in a reduction in the number of impulses without affecting the final delay distribution too much. However one cannot indiscriminately combine impulses using the above two techniques. An effort should be made to make sure that the resulting impulses are located at convenient values (such as integral values) for efficient further computation.

3.4 Reconvergent fanout

Circuits with reconvergent fanout cause the above analysis to become complicated. This is because we lose an important characteristic of fanout-free circuits, namely that the inputs to any given gate are logically independent. A number of techniques have been proposed in the literature to deal with logical correlations. One such technique is the supergate technique first proposed by [81]. The technique consists of first decomposing the original circuit into sub-circuits such that the inputs to each sub-circuit are logically independent. The specific circuit decomposition scheme we use is drawn from [14]. As in [14] we define the supergate of a node in the circuit to be the

minimal sub-circuit in the transitive fanin of the node such that the sub-circuit's fanins are logically independent. This technique consists of building a *lc-graph* corresponding to the given circuit. The vertices of the lc-graph are all the signals of the circuit, and an edge is inserted between two vertices if and only if either the signal lines corresponding to the two vertices are both inputs to some gate or one of them is an input to some gate while the other is the output of the same gate. This procedure has the effect of creating a local clique corresponding to each gate of the circuit. Theorem 1 of [14] establishes that the bi-connected components of the lc-graph are maximal supergates (i.e., supergates that are not properly contained in a larger supergate). Figure 3.4 shows how a simple circuit is partitioned into supergates.

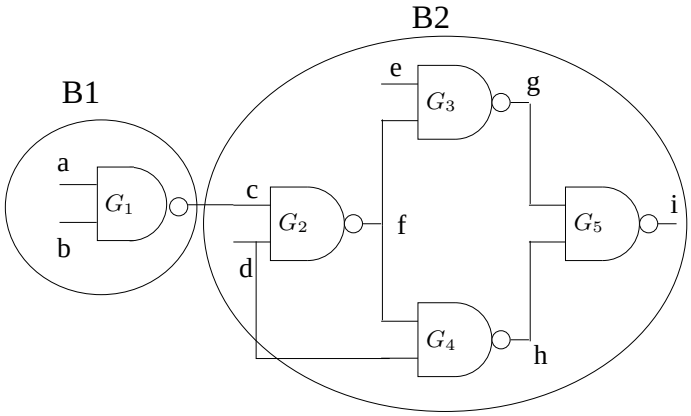


Figure 3.2: A simple circuit with reconvergent fanout and supergate blocks B1 and B2.

The associated lc-graph of the circuit is shown in Figure 3.3. Note that the node c is an articulation point in this graph (i.e., vertex whose removal disconnects the graph). Below we present the basic algorithm used to compute the output distribution for a supergate S given in the form of a set of SuperGate expressions.

```
SuperGateCompute(S, BayesFactor){
  Q = RepeatSet(S);
  if(Q == NULL){
    S' = Evaluate(S, Q = NULL);
    Scale(SuperGateLocalDist, BayesFactor);
```

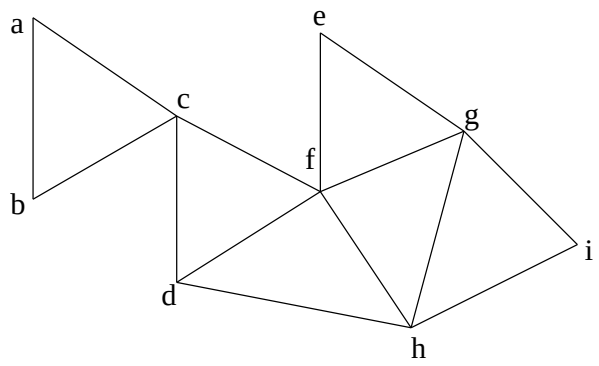


Figure 3.3: lc-graph for circuit of Figure 3.4.

```
    Combine(SuperGateGlobalDist, SuperGateLocalDist);
    return;
  }
  if(Q != NULL){
    T = NewTuple(Q);
    BayesFactor = BayesFactor * P(Q = T);
    S' = Evaluate(S, Q = T);
    SuperGateCompute(S', BayesFactor);
    return;
  }
}
```

We describe how the above algorithm works in the case of our example. We start with the set of supergate expressions S representing all the gates of the supergate. In our example, for the supergate B2, S is given by $\{f = max(c,d)+G_2, g = max(e,f)+G_3, h = max(f,d)+G_4, i = max(g,h)+G_5\}$ where $G_2 \ldots G_5$ are the gate delay distributions of the respective gates. Given a set of Supergate expressions S, the base set of variables is the set of variables that do not occur on the left hand side of any equation in S. $RepeatSet(S)$ returns a set of variables which occur more than once in the set S but never on the left hand side of any equation in S. For our example set S,the base set is $\{c,d,e\}$ and $RepeatSet(S)$ returns the set $\{d\}$. $NewTuple(Q)$ returns a previously unused tuple from the set of tuples associated with the distributions of the random variables in Q. The set $S' = Evaluate(S, Q = T))$ is got by substituting each variable of Q by the corresponding constant value from T, and then evaluating the set of

supergate expressions where possible. This means that if we find the supergate expression of the form $Z = max(X, Y) + G_i$ where X and Y are either base set variables or constants, we can evaluate the expression, and remove it from the list. In our example $S' = Evaluate(S, d = k)$ results in the set $\{g = max(e, f) + G_3, h = max(f, k) + G_4, i = max(g, h) + G_5\}$ as we could remove the supergate expression $f = max(c, k) + G_2$. We perform the above steps in each recursive call until we reach a point where $RepeatSet(S)$ is the empty set. Thereafter, $RepeatSet(S) = NULL$ in successive invocations of SuperGateCompute and we arrive at a point where there is only one expression left S. At this point we can actually evaluate the supergate output. Having done so, we scale the impulse-train corresponding to the supergate output by the computed $BayesFactor$ using the function $Scale(SuperGateLocalDist, BayesFactor)$. We then combine the locally computed distribution with a global distribution maintained at the output. Note that the functions $Scale$ and $Combine$ only perform their tasks when $SuperGateLocalDist$ is available.

The recursion tree for our example is of depth 3. Let us evaluate BayesFactor for one path in this recursion tree. In the first call of SuperGateCompute(S, 1), Q turns out to be the set {d}. The second call to SuperGateCompute sets Q to be {f}. In the third call Q turns out to be NULL. At this point there is only one expression left in S, i.e., $i = max(g, h) + G_5$. The BayesFactor used to scale $SuperGateLocalDist$ for the supergate output i is given by $BayesFactor = P(f = T_2/d = T_1) * P(d = T_1)$. The probability magnitude at each sample space point in the impulse-train of i prior to scaling corresponds to the probability $P(i/f = T_2, d = T_1)$. Scaling each impulse in the train by the BayesFactor computed gives us the joint probability $P(i, f = T_2, d = T_1)$ according to Bayes product rule $P(i, f = T_2, d = T_1) = P(i/(f = T_2, d = T_1)) * P(f = T_2/d = T_1) * P(d = T_1)$. Clearly when we are finished with all paths in the recursion tree, we will have computed in the resulting impulse train for the supergate output i, the distribution for i without any dependencies.

3.5 More complex supergates

For supergates with many stems, the impulse-train approach becomes very impractical and begins to become much more expensive than Monte Carlo

simulation. Fortunately we can combine Monte Carlo simulation with the impulse-train approach. The idea here is that we sample the discrete distributions at the inputs of the supergate and perform a Monte-Carlo simulation by sampling the distributions at the gates of the circuit, and collecting the samples at the output. The work of [55] alludes to a hybrid approach but does not discuss how to construct a distribution out of the samples collected at the output. We divide the spread of the distribution into several frequency bins and determine the number of samples that fall into each bin. The probability magnitude for each bin is set to the ratio of the number of samples belonging to that bin to the total number of samples. We locate the probability at the centroid of each bin.

3.6 Upper and lower bounds on the delay distribution

We have not so far provided any indication of the error incurred in the discretisation process and how this error propagates through the circuit. Intuitively it seems as if the smaller the unit of discretisation, the lower the error as the continuous distribution is represented far more accurately. It turns out that we can use a result of Stoyan [87] to obtain exact upper and lower bounds on the distribution of the output delay.

We state below the theorem of Stoyan [87] that will be of use in developing the bounds of this section:

Theorem 3.1 Let G_1 and G_2 be two timing graphs with exactly the same topology but with corresponding edges in the timing graphs labelled by different random variables. Let the ith timing edge in G_1 be labelled by X_i and the the same edge in G_2 be labelled by Y_i. Then if X_i is stochastically smaller than Y_i, for every edge i then T_{G_1}, the random variable representing the output delay distribution of G_1 is stochastically smaller than T_{G_2}.

In other words, if F_{G_1} is the cumulative distribution function of G_1 and F_{G_2} is the cumulative distribution function of G_2, then $F_{G_1}(t) > F_{G_2}(t)$. This result is intuitive because it just tells us that if all the the random delays of all the edges of a timing graph were replaced by "faster" random variables then the circuit would become "faster".

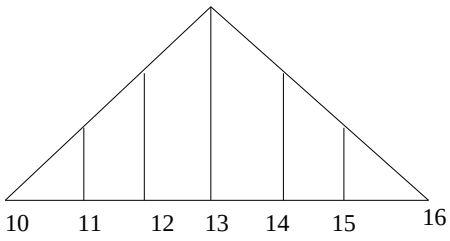
It remains to show that we can use the above result to develop lower and upper bounds. Note that we are calculating an upper and lower bound on the so-called "late-mode" arrival time at the primary output. We can calculate similar curves for "early-mode" arrival time at the primary output. Essentially given a random variable X having a continuous density function, we must create a random variable Y such that Y is stochastically smaller than X. Since we are in the discrete domain, we would like Y to have a discrete density function. For the upper bound, Y must be stochastically larger than X. Fortunately there is a novel way of ensuring these conditions by choosing the points at which to discretise the distribution. In Figure 3.4, three types of discretisations of the gate delay distribution are studied. In Figure 3.4b, each impulse is centred at the midpoint of its interval. A distribution *stochastically smaller* than the original triangular distribution results when the impulses are centred at the left endpoint of each interval as shown in Figure 3.4c. In other words the staircase function representing the cumulative distribution function for the discrete density shown in Figure 3.4c is an upper bound on the cumulative distribution function for the triangular density function shown in Figure 3.4a. Figure 3.4d shows a discrete density function that results in a distribution *stochastically larger* than the original cumulative distribution function.

The algorithm to produce a lower bound and upper bound delay distribution curve consists of replacing the delay random variable of each timing edge by its corresponding discretised lower (upper) bound delay and then running the algorithm described previously in this chapter to determine the output delay distribution. We show computational results on an example circuit in the set of figures below. In Figure 3.5, the upper and lower bounds are pretty loose owing to the use of a large discretisation interval. The discretisation interval is reduced to generate Figure 3.6, and the upper and lower bounds become much tighter.

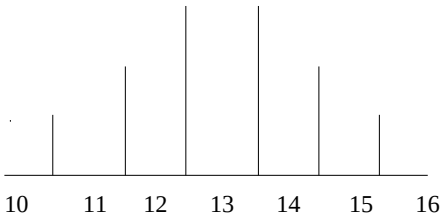
Note that the upper and lower staircase functions are not just shifted versions of each other. In the general case, they could be substantially different.

3.7 Discussion

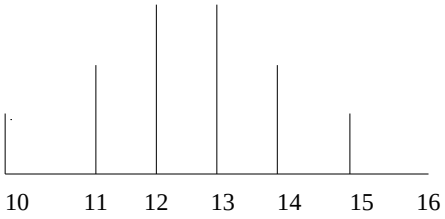
It is instructive to explore the complexity of the impulse-train (or discretisation) approach to the timing analysis problem. The problem is linear time



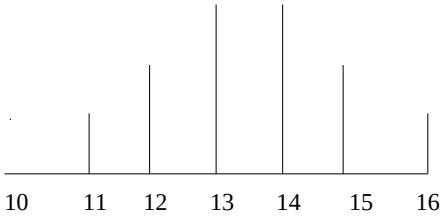
(a) original triangular distribution



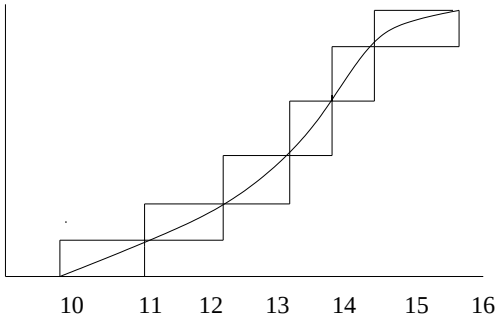
(b) normal discretisation



(c) concentrating impulses at left end of each interval



(d) concentrating impulses at right end of each interval



(e) staircase functions representing upper and lower bounds on the true cumulative distribution function

Figure 3.4: Graphical illustration of derivation of upper and lower bounds.

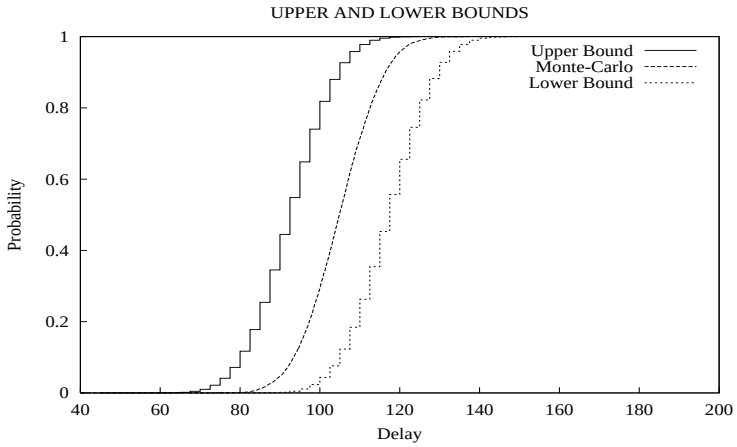


Figure 3.5: Upper and lower bounds on an actual circuit. Number of impulses used to discretise the domain of the distribution is 4.

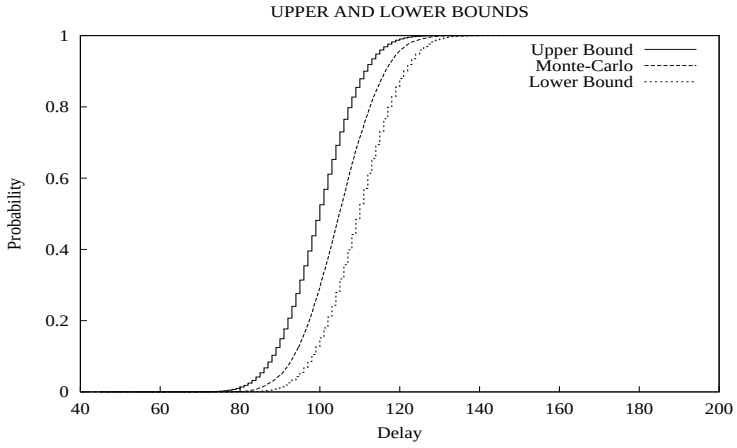


Figure 3.6: Upper and lower bounds on an actual circuit. Number of impulses used to discretise the domain of the distribution is 10.

only for tree-circuits. Let us assume that we use uniform discretisation of the same step size to discretise the gate delay distributions. For two delay random variables X and Y with densities represented by m and n impulses respectively, $X + Y$ can be computed in $O(mn)$ time, and is represented by $O(m + n)$ samples. Similarly $max(X, Y)$ can be computed in $O(mn)$ time and can be represented by $O(m+n)$ samples. Therefore the entire procedure can be completed in linear time. This pretty picture is disturbed by the use of non-uniform discretisation in which case the number of impulses needed to represent the result of an addition operation becomes $O(mn)$, so there is a possibility of a blow-up in space requirements.

The situation becomes much more complicated for circuits with reconvergent fanout. The key to complexity here is the Bayes product rule computation. From the description of the algorithm, the depth of the recursion tree is controlled by the number of fanout points in a supergate and the number of leaves in the recursion tree is exponential in the number of levels of the recursion tree. Thus the algorithm becomes exponential in the number of fanout points in a supergate. This ultimately is the undoing of this algorithm - it is impractical for circuits having even a moderate amount of reconvergent fanout.

Let us situate the statistical timing method described in this chapter in terms of the quality criteria outlined in the last section of Chapter 2. The criteria mentioned were model sophistication, computational complexity, error theory and tuning information. The statistical model we employ is a simple one - it assumes that the gate delays are independent. However, correlations due to path sharing are taken into account. The computational complexity of the impulse-train method is high; in case there is a lot of reconvergent fanout, the method is exponential in the number of fanout points in the circuit. The basic method of situating impulses at the centre of each delay interval does not lend itself to the development of a suitable error theory because it is unclear how the discretisation error propagates itself through the circuit. It seems intuitive that using a lot of impulses to represent the probability density of each gate should lead to a low-error estimate of the final output probability distribution. We can develop an error theory of sorts by resorting to the upper and lower bounds computation of Section 3.6. The staircase functions of Figure 3.4 are exact upper and lower bounds on the true output delay distribution curve, and the staircase functions approach each other as the number of impulses used to represent each gate delay distribution

increases. Finally as regards the last criterion of tuning information, it is difficult to extract tuning information as a byproduct of running the impulse-train method.

Chapter 4

A more refined timing model

In the last chapter, we looked at a simple timing model and showed how to derive lower and upper bounds on the delay distribution of the output delay. The main problem with this timing model is that it is not realistic in that it ignores the correlation between delays of the edges in the timing graph, although it does take into account the correlation between paths due to path sharing. It turns out that most approaches in the literature do not provide a unified framework to handle both types of correlation and have perhaps for this reason not so far been widely used in industry. The impulse-train approach of [66], [67], [55], the method of bounds [3] can all be seen as approaches that can handle path sharing alone, while that of [30] can be seen as one that handles correlation between gates alone.

This chapter focuses attention on a timing model that was first suggested many years ago [96] and then abandoned because it appeared to be intractable. We reexamine the model in this thesis and propose some new solution techniques to deal with the intractability. The rest of this thesis consists of analysing this model, presenting experimental results, and then devising a technique for synthesis based on it.

4.1 Timing edge correlation models

It has long been known that the delays of timing edges in a timing graph are not independent but are closely tracked. In other words, if one of the gates

shows a particular kind of behaviour (fast or slow), then surrounding gates show a similar kind of behaviour. The precise nature of this tracking has been established for MOS circuits, but the studies of [9], [69], [70] which detail systematic variations in parameters, corroborate this situation for CMOS circuits.

Even if we were to allow the possibility of correlation between delays of edges in the timing graph, we are still faced with the question of characterising this correlation. Some authors, for example, [53] take a "gates talking to each other" approach. They let each timing edge (note that here we assume each gate has a single timing edge, but that does not really matter) take on a Gaussian delay distribution and then define a correlation matrix where the $(i, j)th$ entry gives the correlation coefficient between the delays of gates i and j. The choice of the Gaussian distribution is not only a matter of convenience - it is actually crucial in the sense that specifying correlation with any other distribution would be more complex than providing just a correlation matrix.

The main problem with the above approach is that the size of the correlation matrix is quadratic in the number of edges in the timing graph. It is likely to be pretty difficult to populate the correlation matrix, as we need to be able to ascertain the timing behaviour of every edge with respect to every other edge.

The "gates talking to each other" approach is costly, so we need another alternative. This is the "global phenomena talking to gates" alternative. The concept is represented in Figure 4.1. The underpinning of this approach is that there are really only a few *global* parameters which cause the delays of all gates to vary from their nominal values. The delays of gates are correlated because the same global phenomena affect the gates across the chip. The true worth of the model occurs when the number of global parameters is in the order of 10-20, and we believe that this is the case with modern manufacturing processes. There are few main process parameters $T_{ox}, L_{eff}, ILDthickness$ which affect all gates. We shall show later that location dependence can be taken care of by using more auxiliary parameters.

4.2 Linear model and yield formulation

Once we accept the idea that there are only a few global parameters that cause the delays of the gates to vary from their nominal values, it remains to

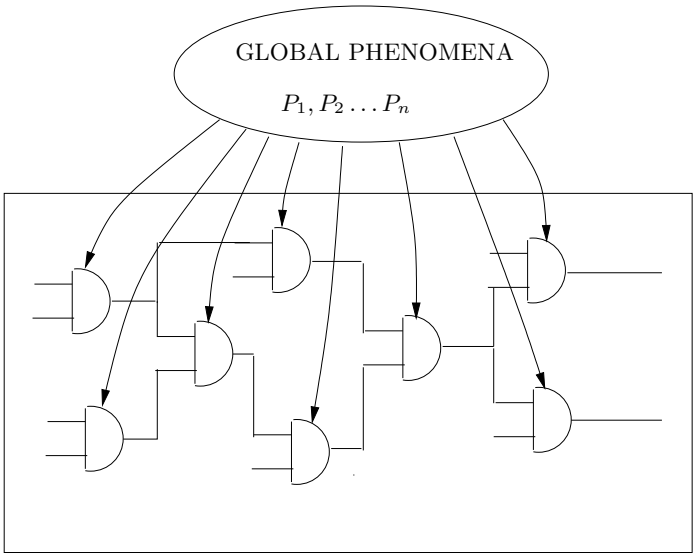


Figure 4.1: Global phenomena impacting all gates.

actually characterise the effect of the variation of the global parameters on the delays of the gates. For reasons of tractability, we shall assume a linear model i.e., the delay of a gate varies linearly with each global parameter. As we shall see even this "first-order" model is not easy to solve, and calls for special techniques. Below we shall present mathematical formalism that will serve us for the remainder of this chapter and subsequent chapters.

Since the edge delays in the timing graph are linear in the variations of the global parameters, we can write the following equation:

$$d_{edge} = \mu_{edge} + Q\Delta z. \tag{4.1}$$

The above equation is written in matrix form. The (i, j)th entry of the matrix Q represents the sensitivity of the delay of the edge i to the jth global parameter. The vector Δz represents the variations in the global parameters, μ_{edge} is a vector of nominal edge delays, and d_{edge} represents the vector of all edge delays. If there are n global parameters and m timing edges in the timing graph, the vectors d_{edge}, μ_{edge} are both $m \times 1$ column vectors, Q is a $m \times n$ matrix, and Δz is a $n \times 1$ column vector.

Now that we have a means of expressing delays of the edges in terms of their nominal delays and variations in the global parameters, we can express the delays of the paths. To do this, we introduce the notion of a topological path matrix C. The $(i,j)th$ entry in this matrix is 1 if path i contains edge j, and 0 otherwise. If there are P paths, and m timing edges in the timing graph, C is a $P \times m$ matrix. We can write the path delay vector d_{path} as follows:

$$d_{path} = Cd_{edge} = C\mu_{edge} + CQ\Delta Z. \tag{4.2}$$

Let $C\mu_{edge} + CQ\Delta z = \gamma + R\Delta z$. Then for the circuit to have a performance no greater than η, we require

$$\gamma + R\Delta z \leq [\eta\ \eta\ \cdots\ \eta]^T \tag{4.3}$$

where the right-hand side is a vector with η in all positions. In z-space, the space of global parameters, we can express the above as follows:

$$Rz \leq [t_1\ t_2\ \cdots\ t_P]^T \tag{4.4}$$

The above defines a linear system of inequalities in parameter space as shown in Figure 4.2. Each inequality is a hyperplane and all the inequalities together define a convex feasible region. To obtain the actual yield, we must merely integrate the joint probability density function of all the global parameters over the feasible region. This can be expressed as

$$\int\int\int_R f(z_1, z_2 \ldots z_n) dz_n dz_{n-1} \ldots dz_1, \tag{4.5}$$

where R is the feasible region and $f(z_1, z_2, \ldots z_n)$ is the joint probability density function of the global parameter random variables, which we assume is available.

We repeat this procedure over the entire range of performances to construct the cumulative distribution function of the output delay distribution. Integration of a joint probability density function over a feasible region has been considered in [85], [96]. The work of [75] uses a linear model for the parameter variations, but computes bounds on the probability distribution of output delay using the theory of stochastic processes. This work assumes that the joint probability distribution of global parameters is Gaussian.

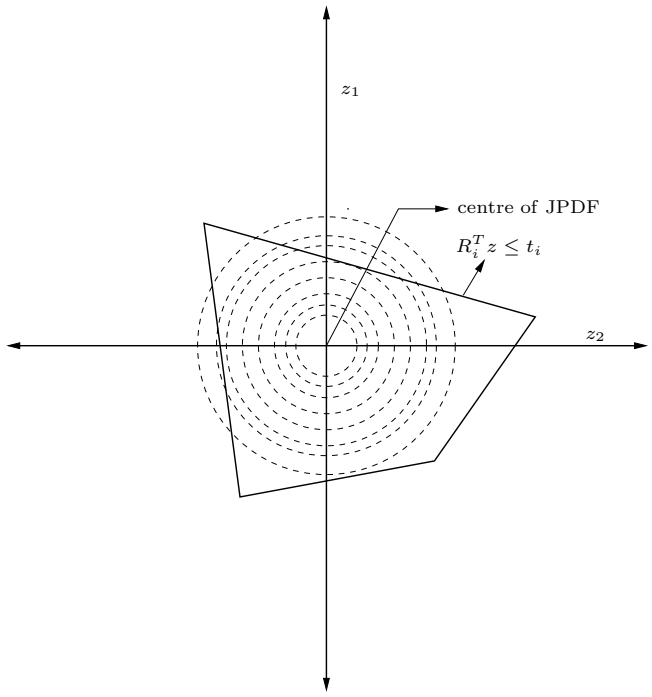


Figure 4.2: Feasible region defined by hyperplanes.

The reader may be tempted to think that it is possible to generate a convex feasible region as outlined in equation (4.3) that identifies the circuits which have a delay no less than η. It might appear that we can write

$$\gamma + R\Delta z \geq [\eta\ \eta\ \cdots\ \eta]^T, \tag{4.6}$$

where the right-hand side is a vector with η in all positions to generate the required convex feasible region. However, closer examination reveals that the above system of inequalities is too restrictive to determine the number of circuits that will have a delay no less than η. Indeed it is not necessary for all paths in the circuit to have a delay greater than η for the whole circuit to have a delay greater than η; it is sufficient for one path to have a delay greater than η to make the whole circuit have a delay greater than η. In other words, a single path may act as a bottleneck. Another way of looking at this is to see that the complement of a convex feasible region is not necessarily a convex region.

4.3 A simple lower bound on timing yield

We shall derive a simple lower bound on timing yield in this section by approximating the feasible region from below. In (4.2) the matrix CQ has entries which are unit dependent since the various global parameters z_i have different units. We shall redefine the ith path delay equation as follows:

$$d_{path} = \gamma_i + p_{i1}\frac{\delta z_1}{z_1^{nom}} + p_{i2}\frac{\delta z_2}{z_2^{nom}} + \ldots + p_{in}\frac{\delta z_n}{z_n^{nom}}. \tag{4.7}$$

Substituting for $\delta z_i = z_i - z_i^{nom}$, and defining $u_i = \frac{z_i}{z_i^{nom}}$ we can rewrite the path delay as follows:

$$d_{path} = c_i + p_{i1}u_1 + p_{i2}u_2 + \ldots + p_{in}u_{in}. \tag{4.8}$$

with $c_i = \gamma_i - \sum_{j=1}^{j=n} p_{ij}$ as the constant contribution. Note that each u_i is positive as each global parameter is a positive physical quantity (such as temperature, length etc). Unlike the entries of the original R matrix, the p_{ij} formed in the above process have the same units (of time).

Since there are many paths in a circuit, with each of them contributing a hyperplane, the feasible region is formed by the intersection of a large number of hyperplanes. One can consider a simpler region described only by $n+1$ hyperplanes where n is the number of global parameters, such that integrating the joint density function over this simpler region yields a lower bound on the timing yield. Let the set S^+ denote the set of rows of P which have positive largest values in them. Similarly let S^- denote the set of rows of P which have negative largest values in them (i.e., all values in such a row are negative, so the largest value is also negative). We can compare the elements in a row of P because they are all of the same units. Let us define the matrix M as follows:

$$M = \begin{bmatrix} v_1 & v_1 & \ldots & v_1 \\ v_2 & v_2 & \ldots & v_2 \\ v_3 & v_3 & \ldots & v_3 \\ \vdots & \vdots & \vdots & \vdots \\ v_N & v_N & v_N & v_N \end{bmatrix}$$

Since all the random variables u_i are positive, we can easily see that

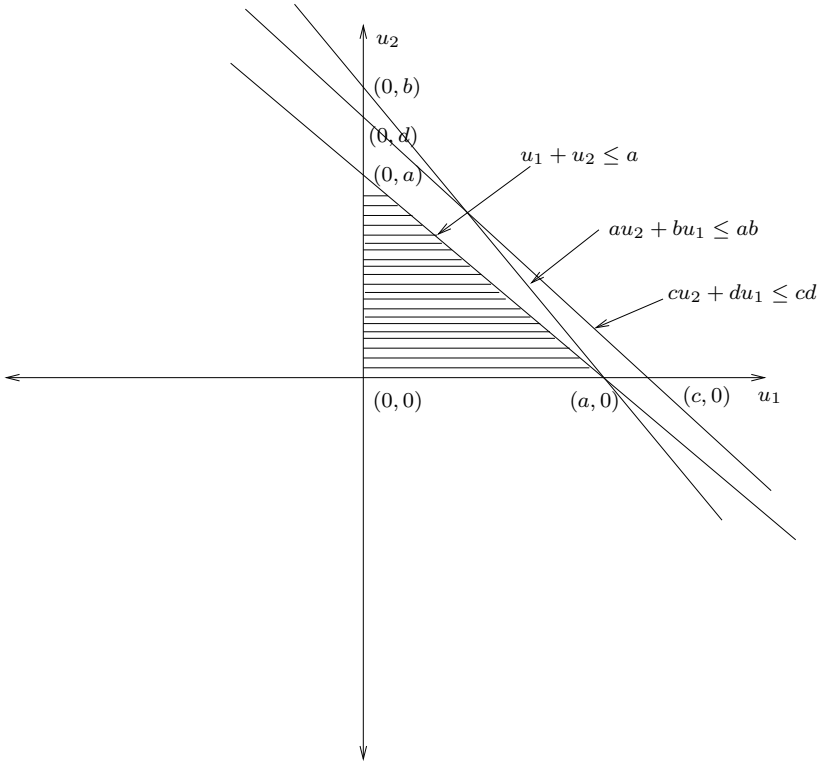


Figure 4.3: Simplex approximation to feasible region.

$$c + Pu \leq c + Mu. \tag{4.9}$$

.

Thus requiring that $c + Mu \leq \eta$ will automatically ensure that all paths in the circuit will have a delay no greater than η. Expanding each equation in $c + Mu \leq \eta$, we get the following inequalities for those rows which have positive largest values:

$$(u_1 + u_2 + \ldots + u_n) \leq ((\eta - c_i)/v_i) \tag{4.10}$$

$\forall v_i \in S^+$.

We get inequalities of the following kind for all those rows which have negative elements:

$$(u_1 + u_2 + \ldots + u_n) \geq ((\eta - c_j)/v_j) \tag{4.11}$$

$\forall v_j \in S^-$.

The set of inequalities in (4.10) can be rewritten as follows:

$$(u_1 + u_2 + \ldots + u_n) \leq \min_j(\eta - c_j/v_j), v_j \in S^+. \tag{4.12}$$

The set of inequalities in (4.11) can be written as follows:

$$(u_1 + u_2 + ...u_n) \geq \max_j(\eta - c_j/v_j), v_j \in S^-. \tag{4.13}$$

Note that any vector u that satisfies the above inequalities also satisfies the inequality $c + Pu \leq t$ and is thus in the feasible region of circuit operation, but not vice-versa. Thus integrating the joint density function over the approximation to the feasible region specified by the conjunction of (4.12) and (4.13) gives us a lower bound on the timing yield. The simplex approximation to a two-dimensional feasible region (assuming that the maximum value in each row is positive) is shown in Figure 4.3. If D is the random variable describing the latest arrival time at the primary output, we have the following:

$$P(D \leq \eta) \quad \geq \quad I_1 - I_2, \tag{4.14}$$

where

$$I_1 \quad = \quad \int_0^{K_1} \int_0^{K_1 - u_1} \ldots \int_0^{K_1 - \sum_{j=1}^{j \leq n-1} u_j} f(u_1, u_2, \cdots, u_n) du_n du_{n-1} ... du_1, \tag{4.15}$$

and $K_1 = min_j \frac{\eta - c_j}{v_j}, v_j \in S^+$. Further

$$I_2 \quad = \quad \int_0^{K_2} \int_0^{K_2 - u_1} \ldots \int_0^{K_2 - \sum_{j=1}^{j \leq n-1} u_j} f(u_1, u_2, \cdots, u_n) du_n du_{n-1} ... du_1 \tag{4.16}$$

where $K_2 = max_j \frac{\eta - c_j}{v_j}, v_j \in S^-$. Note that I_1 and I_2 are integrals over simplices which are nice geometrical structures for which a variety of numerical integration algorithms are available. We expect that the bound given by (4.14) will be somewhat loose in practice because the difference between the minimum and maximum relative sensitivity in each row of the matrix P may be relatively large.

4.4 Estimating the integral

For the moment let us remain in the space of normalised global parameters where each $u_i = \frac{z_i}{z_i^{nom}}$. We shall now examine the problem of integrating the joint probability density function over a convex polyhedron formed by the path delay inequalities.

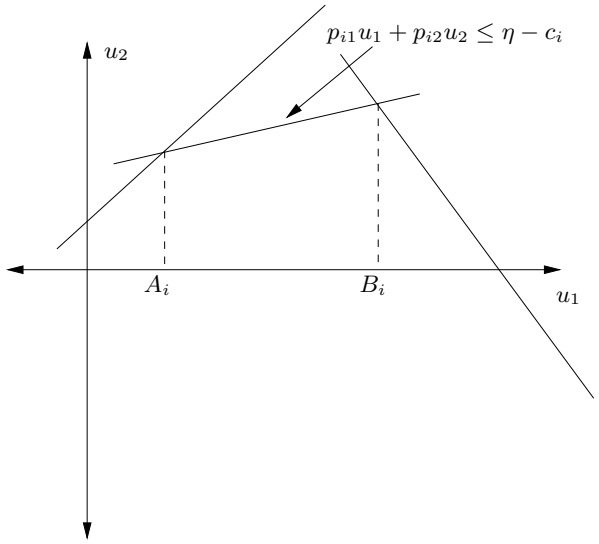


Figure 4.4: The active region for the hyperplane marked $p_{i1}u_1 + p_{i2}u_2 \leq \eta - c_i$ is the region on the u_1 axis between the points A_1 and B_1.

The method of hyperplanes essentially reduces the problem of estimating yield to the calculation of n-dimensional multiple integrals. The region of integration is a convex polyhedron in n-space. If the polyhedron is in 2-space (with two global parameters u_1 and u_2), then we can easily perform the integration. Essentially we consider the hyperplanes one by one and for a given hyperplane, determine the region of the x-axis over which that hyperplane is active. This means that for the ith hyperplane, we solve the

following linear programme to get the lower bound on the active region:

$$\begin{aligned} &\min u_1 \text{ such that} \\ c_1 + p_{11}u_1 + p_{12}u_2 &\leq t \\ c_2 + p_{21}u_1 + p_{22}u_2 &\leq t \\ &\cdot \\ &\cdot \\ c_i + p_{i1}u_1 + p_{i2}u_2 &= t \\ &\cdot \\ &\cdot \\ c_N + p_{N1}u_1 + p_{N2}u_2 &\leq t \\ u_1 &\geq 0 \\ u_2 &\geq 0. \end{aligned} \tag{4.17}$$

To find the upper bound for the active region for the ith hyperplane we simply change the objective function in the above linear programme to max u_1. Let A_i denote the lower bound on the variable u_1 for the active region corresponding to the ith hyperplane and B_i denote the upper bound on the variable u_1 corresponding to the active region for the ith hyperplane. The active region for the ith hyperplane is shown in Figure 4.4. Then we can write

$$P(D \leq t) = \sum_{i=1}^{N} (-1)^{t_i} \int_{A_i}^{B_i} \int_{0}^{(t-c_i-p_{i1}u_1)/p_{i2}} f(u_1, u_2) du_2 du_1. \tag{4.18}$$

In the above equation the $t_i = 1$ if the active interval (A_i, B_i) on the u_1 axis for the ith hyperplane lies on the infeasible side of that hyperplane, and $t_i = 2$ if the active interval (A_i, B_i) is on the feasible side of the hyperplane. For example in Figure 4.4, the active intervals for all the three hyperplanes shown lie on the feasible side of the hyperplanes. Therefore $t_i = 2$ for all the hyperplanes. Note that since each u_i is positive, the entire feasible region lies strictly in the first quadrant.

4.5 The general case

It is almost certain that the number of global parameters to consider is greater than two. Therefore we must use techniques for multi-dimensional integration. In general there are no easy ways to perform multi-dimensional integration; closed-form formulae are usually not available. Numerical integration !numerical is a standard method for integrating functions in one or two variables. The problem with numerical integration is that the formulae are meant for specific domains such as an n-dimensional cube, sphere, tetrahedron etc. If a given feasible region can be recast into one of these special regions, and the number of dimensions is not too large, then numerical integration is the method of choice.

It might be expected that the dimensionality of the model is crucial to an efficient solution. Throughout this thesis we shall treat the number of dimensions of the parameter space to be a variable. This is because current literature does not throw adequate light on how many significant process parameters exist in a modern manufacturing line. Since the model described in this chapter is path-based, there is also the question of coming up with a set of relevant paths that bound the feasible region. The number of critical paths in even a moderate-sized circuit is very large, and it will be helpful to prune the list of critical paths so as to identify the significant paths. In the next section we shall examine a method to prune the set of critical paths in a circuit.

4.6 Path filtering

We must address the path explosion problem that will doubtless occur in large circuits. There are many millions of paths leading from primary inputs to outputs in even moderately-sized circuits, and even if we were to look at only "critical" paths - or those with a delay large enough to impact the delay of the whole circuit - we may arrive at several thousand paths. The statistical timing procedures we shall outline in subsequent chapters are not linear in the number of paths; some of them are cubic in the number of paths. Therefore we benefit by reducing the number of paths as much as possible.

For real circuits, there are a large number of paths that have very similar characteristics owing to a large amount of path sharing. The scenario we

construct divides the chip area into sub-regions and associate a unique set of parameters to each sub-region. The set of parameters associated with each sub-region characterise gate delay behaviour for that sub-region. This arrangement is a half-way measure between assuming complete independence of gate delays from each other on the one hand, to perfect tracking on the other. Neither of the extreme options is realistic for modern chip manufacturing processes. The sub-regions are illustrated in Figure 4.5.

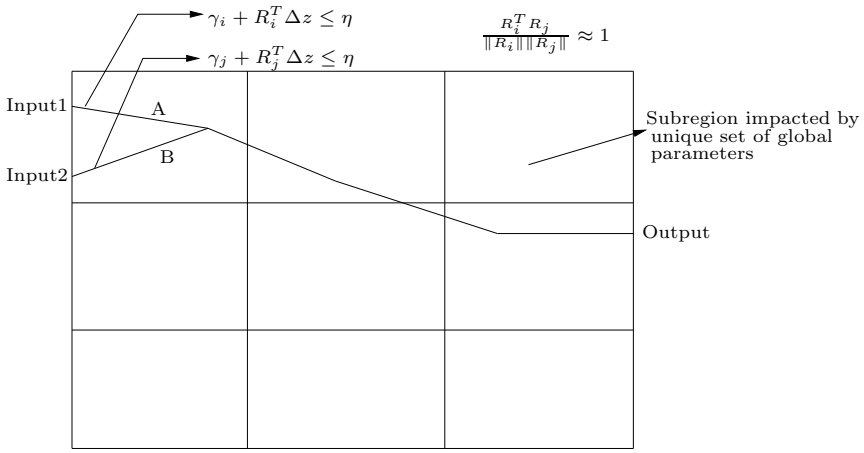


Figure 4.5: Chip area divided into sub-regions. Paths A and B with path vectors R_i and R_j share almost all their gates. Therefore the angle between them is almost zero.

Paths marked A and B in Figure 4.5 depend on the same set of parameters since they pass through the same sub-regions, and must also have nearly the same coefficients since they share so many gates. This suggests that we can club these two paths together, and thus reduce the number of paths. Thus we can augment the path selection process so that paths are selected not only according to high nominal delay. The other measures we suggest include checking to see if the normal to the path vector is oriented at an angle not represented by any other paths already in the list. The idea behind both of these measures is to select paths that, in some extreme circumstances become limiting to circuit performance. Some paths can exhibit markedly different behaviour to some global parameters, than paths already in the list. This will be brought out by the angle criterion whereby we include the candidate path in the list of all paths if the minimum value of its angle to any path in

the current set of selected paths is greater than some threshold value. The cosine of the angle that a path with path vector R_i makes with a path R_j can be expressed as:

$$\cos\gamma = \frac{R_i^T R_j}{\|R_i\|\|R_j\|}. \tag{4.19}$$

Let us express the preceding ideas in terms of an algorithm which can be viewed as a preprocessing step that takes a large number of paths and produces a smaller number of "representative" paths. The basic algorithm can be expressed as follows:

```
SelectRepresentativeDirections-I{
  DirectionSet[0] = SensitivityMatrix[0];
  SizeOfDirectionSet = 1;
  while(SizeOfDirectionSet < numPaths){
    GlobalMinimumCosine = 1.0;
    for(i=0; i < TotalNumberOfPathsInList; i++){
      CurrentMaximumCosine = -1.0;
      for(j=0; j < SizeOfDirectionSet; j++){
        CurrentCosine = CosineValue(DirectionSet[j], SensitivityMatrix[i]);
        if(CurrentCosine > CurrentMaximumCosine){
          CurrentMaximumCosine = CurrentCosine;
        }
      }
      if(CurrentMaximumCosine <= GlobalMinimumCosine){
        GlobalMinimumCosine = CurrentMaximumCosine;
        PathIndex = i;
      }
    }
    SizeOfDirectionSet++;
    DirectionSet[SizeOfDirectionSet] = SensitivityMatrix[PathIndex];
  }
}
```

The idea encapsulated in the above pseudo-code is simple: at each step, we simply pick a path that makes the largest angle with respect to all the paths selected upto that point. The innermost *for* loop finds the smallest angle that a candidate path makes to all the directions in *DirectionSet*. The variable *GlobalMinimumCosine* is updated if the smallest angle that the current candidate path makes with respect to all the paths in *DirectionSet* is larger than the smallest angle that any other candidate path upto that point makes with the selected paths. The value of *CurrentMaximumCosine* at the end of

the inner *for* loop measures the "degree of agreement" that the given candidate path has with all the paths selected so far. If this value is 1.0, it means that the direction represented by the candidate path is already represented in the set of paths selected so far. When the outer *for* loop is completed, *GlobalMinimumCosine* records the maximum "degree of disagreement" that any candidate path has with the set of paths in *DirectionSet*, and *PathIndex* records the identity of the candidate path that has this "degree of disagreement." As such it is natural that this path be included in the set of paths selected so far. If C is the number of paths required to be selected and m is the total number of paths in the system, then the procedure above can be seen to take $O(mC^2n)$ time. A closer look at the basic algorithm outlined above shows that it is possible to improve it significantly: in the innermost *for* loop the cosines of the angles made by a candidate path with all the paths in the selected set are calculated repeatedly, when they need to be calculated just once. This means that having calculated a relevant cosine value once, we must save it in memory for later use. To do this, we create an array *CurrentMaxCosineArray* with *TotalNumberOfPathsInList* elements. The ith element in this array is intended to carry the maximum cosine value that the ith path in the total set of paths makes with all the directions in *DirectionSet*. Each element in this array is initialised to -1.0. With this modification, the above pseudo-code can be rewritten as follows:

```
SelectRepresentativeDirections-II{
  for(i=0; i < TotalNumberOfPathsInList; i++){
    CurrentMaximumCosineArray[i] = -1.0;
  }
  DirectionSet[0] = SensitivityMatrix[0];
  j=0;
  while(j < numPaths){
    GlobalMinimumCosine = 1.0;
    for(i=0; i < TotalNumberOfPathsInList; i++){
      CurrentCosine = CosineValue(SensitivityMatrix[i], DirectionSet[j]);
      if(CurrentCosine > CurrentMaximumCosineArray[i]){
        CurrentMaximumCosineArray[i] = CurrentCosine;
      }
      if(CurrentMaximumCosineArray[i] <= GlobalMinimumCosine){
        GlobalMinimumCosine = CurrentMaximumCosineArray[i];
        PathIndex = i;
      }
    }
    j++;
```

```
    DirectionSet[j] = SensitivityMatrix[PathIndex];
  }
}
```

The pseudo-code of *SelectRepresentativeDirections-II* can be seen to run in $O(mCn)$ time. When the last direction is selected in the pseudo-code above, the direction makes an angle of atleast $\cos^{-1}(GlobalMinimumCosine)$ with all the other selected directions. Further none of the paths outside the selected set makes an angle greater than $\cos^{-1}(GlobalMinimumCosine)$ with the paths in the selected set. Therefore any given path outside the selected set lies within a "cone" of $\cos^{-1}(GlobalMinimumCosine)$ of some selected path. It remains to define the feasible region boundary in each cone, i.e., select the path among all those lying in the cone that actually forms the boundary. This can be done by computing the distances of the paths from the origin. When the origin is on the feasible side of a hyperplane, the distance is positive, and when it is on the infeasible side of a hyperplane, the distance is negative. Let us consider two paths with normal vectors a_i and a_j such that

$$\begin{aligned} R_i^T \Delta z &\leq b_i \\ R_j^T \Delta z &\leq b_j \end{aligned} \tag{4.20}$$

are the hyperplane equations corresponding to the two paths. Let the two normal vectors R_i and R_j point in the same direction. Let the origin be on the feasible side of both these hyperplanes, meaning that b_i and b_j are both positive. The path that is closest to the origin in terms of the absolute value distance will form the feasible region boundary. If $\frac{b_i}{\|R_i\|} < \frac{b_j}{\|R_j\|}$ then the distance from the origin to path i is less than the distance to path j. Therefore path i forms the feasible region boundary. If the origin lies on the infeasible side of both hyperplanes, then b_i and b_j will both be negative, and the path which is furthest away in absolute distance from the origin will form the feasible region boundary. In this case the feasible region boundary is once again decided by $min(\frac{b_i}{\|R_i\|}, \frac{b_j}{\|R_j\|})$. Minimising the signed distance to each hyperplane will ensure the selection of the right path for the feasible region boundary. The algorithm below performs the task of picking paths according to the distance criterion:

```
FindFeasibleRegionBoundary{
```

```
  Eta = RequiredPerformance;
  for(i=0; i < SizeOfDirectionSet; i++){
    MinimumDistanceSoFar = Infinity;
    for(j=0; j < TotalNumberOfPathsInList; j++){
      Direction = CosineValue(DirectionSet[i], SensitivityMatrix[j]);
      if((Direction >= GlobalMinimumCosine)){
        Distance = (Eta - NominalDelay[j])/Length(SensitivityMatrix[j]);
        if(Distance <= MinimumDistanceSoFar){
          MinimumDistanceSoFar = Distance;
          CurrentMinIndex = j;
        }
      }
    }
    FinalPathFilteredSet[CurrentMinIndex] = 1;
  }
}
```

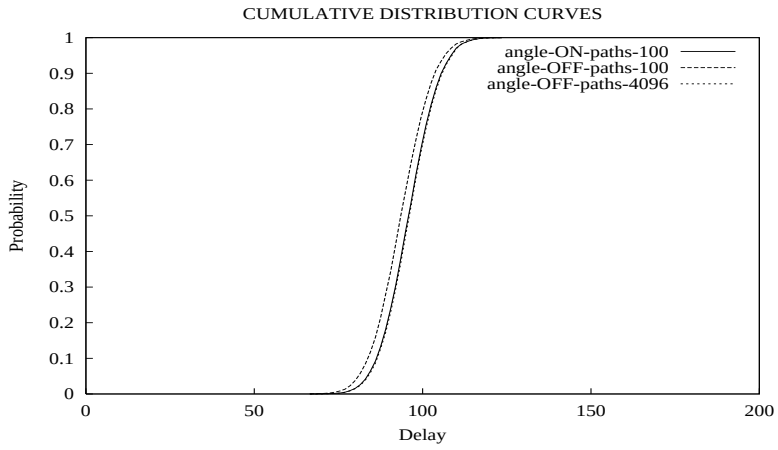


Figure 4.6: Cumulative Distribution curves obtained by considering the top 100 critical paths, 100 critical paths obtained by using the angle criterion and considering all critical paths. There is good agreement between the curve obtained by considering 100 paths selected using the angle criterion and the curve obtained by considering all paths.

The above discussion is intended for the case of computing yield for a single performance. As the performance values change, the angles between the hyperplanes do not change since the hyperplanes continue to have the same normal vectors. However, the distances of the hyperplanes from the origin

will change, and in any given cone, a different hyperplane may form the boundary of the feasible region as the required performance value changes. Therefore it is necessary to run *FindFeasibleRegionBoundary* once for each performance, but it is sufficient to run *SelectRepresentativeDirections* only once.

The results of applying the angle criterion to filter paths are shown in Figure 4.6. Three curves are shown in this figure. The cumulative distribution curve assuming the polyhedron is approximated by the first 100 paths in order of decreasing nominal delay is shown as "angle-OFF-paths-100". The cumulative distribution curve obtained by selecting 100 paths according to the angle criterion outlined above is shown as "angle-ON-paths-100". This curve corresponds fairly closely to the real cumulative distribution curve obtained by taking all 4096 paths in the system.

The path filtering algorithm outlined above has one weakness in that it is performance dependent. Since we are typically interested in the whole performance curve, it is useful to investigate whether we can modify the algorithm above so that path filtering is done such that it remains accurate for all performances. The reason the path filtering algorithm outlined above is performance-dependent is that in a given selected direction, different paths may become critical for different performances. This phenomenon occurs especially when the lengths of sensitivity vectors that point in the same direction are different, as shown in Figure 4.7 shows. This figure shows two paths A and B which are represented as $x + y \leq -1$ and $2x + 2y \leq -1$ respectively. When the origin lies on the infeasible side of both hyperplanes, path A is the critical path. However, when the origin gets on the feasible side of both hyperplanes, path B becomes critical. If path filtering had been done keeping the first situation in mind, then the result would be inaccurate for performances representing the second situation (when path B becomes critical). A straightforward way to tackle this problem is to divide the performance range of interest into several smaller subregions, and picking paths that are critical in each subregion. The pseudo-code *SelectPathsInEachRepresentativeDirection* below does precisely this. It should be seen as a replacement for *FindFeasibleRegionBoundary*. Note that we must run *SelectRepresentativeDirections* once just for the basic performance-dependent path filtering algorithm to get the set of representative directions.

```
SelectPathsInEachRepresentativeDirection(){

  BinSize = (RightPerformance - LeftPerformance)/(NumberOfBins);
  for(i=0; i < SizeOfSelectedDirectionSet; i++){
    for(p=0; p < NumberOfBins; p++){
      Eta = LeftPerformance + p*BinSize;
      MinimumDistanceSoFar = Infinity;
      for(j=0; j < TotalNumberOfPathsInList; j++){
        Direction = CosineValue(DirectionSet[i], SensitivityMatrix[j]);
        if((Direction >= GlobalMinimumCosine)){
          Distance = (Eta - NominalDelay[j])/Length(SensitivityMatrix[j]);
          if(Distance <= MinimumDistanceSoFar){
            MinimumDistanceSoFar = Distance;
            CurrentMinIndex = j;
          }
        }
      }
      FinalPathFilteredSet[CurrentMinIndex] = 1;
    }
  }
}
```

A novel method for "all-performances-at-once" path filtering has been recently proposed in a private communication by Lou Scheffer [80]. The method uses a Monte-Carlo method of selecting a subset of paths that represent the feasible region. The selection process involves a set of trials, in each one of which the JPDF of process parameters is first sampled to obtain a point in process space. Then the path with the largest delay for the selected process parameter vector is selected. When this procedure is run for N trials, it will result in the selection of N paths. The logic behind the selection process is established as follows: if a path is critical a fraction c of the time, then it has a probability c of being selected in an arbitrary trial, and a probability $1-c$ of not being selected in a trial. The probability of not selecting this path in N trials is $(1-c)^N$. When c is small relative to 1, the probability of non-selection can be written as e^{-Nc}. For $c = 0.1$, the probability of non-selection in a 100 trials is roughly e^{-10}, a very small quantity indeed.

The problem with the above Monte-Carlo method of selecting paths is that it is not particularly good at picking paths that occur a very small fraction of the time. Let us imagine that there exists a path which forms a boundary of the feasible region only when the performance requirement is less than n (i.e., for very fast circuits). In order to increase the volume of the feasible region

for this performance requirement, any optimisation procedure would need to identify the path in question since it bounds the feasible region. However, the path filtering scheme of Scheffer would identify this path with low probability, thus making it unviable for use as a method for optimisation. In fact, any "all-performances-at-once" path filtering technique would suffer from a similar drawback. In order to optimise yield for a stringent performance requirement, one must seek recourse to the "performance-at-a-time" path filtering procedure outlined in this section. Since this kind of path filtering takes into account the boundary of the feasible region at the performance of interest, n, it will identify the path in question as a bottleneck, and the tuning procedure can shift this path to increase yield. The shifting of the path can be accomplished by adjusting the nominal delay of the path. We shall address this topic in Chapter 8.

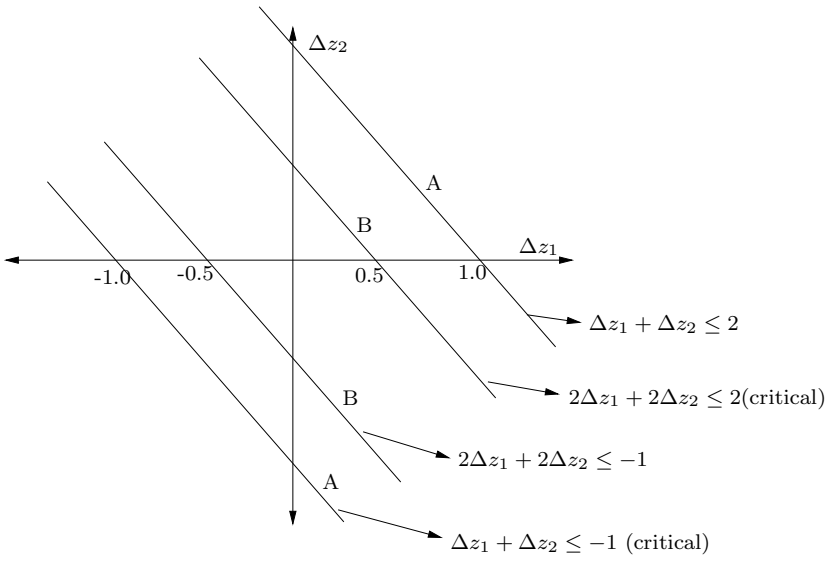


Figure 4.7: Different situations in which different paths are critical. When the origin is infeasible for paths A and B, path A is critical. When the origin is feasible for both paths, path B becomes critical.

4.7 The joint probability density function of global parameters

It is customary to take the joint probability density function of global parameters to be a multi-variate normal of the following form:

$$f(\Delta z) = e^{-\frac{1}{2}(\Delta z)^T A(\Delta z)} \tag{4.21}$$

where A is the correlation matrix. Note that since we are dealing with the changes in global parameters, the mean of the distribution is the zero-vector. Let us recall the feasible region given by (4.3):

$$\gamma + R\Delta z \leq [\eta\ \eta\ \cdots\ \eta]^T. \tag{4.22}$$

The yield integral for any given performance is given by

$$Y = \int\int \ldots \int_R e^{-\frac{1}{2}(\Delta z)^T A(\Delta z)} dz. \tag{4.23}$$

We can re-express this integral so that the integrand represents a multivariate normal whose components are independent. The advantage of the new formulation is that it is much easier to sample from a multivariate normal density whose components are independent as opposed to one whose components are not independent. To achieve the reformulation, we must observe that the correlation matrix A can be diagonalised as follows:

$$A = P\Lambda P^{-1}, \tag{4.24}$$

where P is the eigenmatrix ($P^T = P^{-1}$) and Λ is the diagonal matrix of eigenvalues.

Let us transform the variables of integration as follows:

$$\Delta z = B\Delta v. \tag{4.25}$$

Then the joint probability density function can be rewritten as

$$f(\Delta z) = e^{-\frac{1}{2}(\Delta v)^T B^T AB(\Delta v)}. \tag{4.26}$$

For the JPDF to become multivariate normal with independent components, $B^T AB$ must become the identity matrix. From (4.24) we can see that this is achieved when

$$B = P\Lambda^{-\frac{1}{2}}. \tag{4.27}$$

The region of integration R gets transformed to R^*. R is an arbitrary polyhedron in space and R^* is another arbitrary polyhedron. Thus the angle criterion of the previous section, which can be used to filter hyperplanes in R can also be used in R^*.

4.8 Performance space formulation

The approach we have outlined in this chapter until now can be regarded as a parameter space approach because the feasible region was expressed in process parameter space. The feasible region was an irregular polyhedron in parameter space. In contrast, and especially with the transformation of the previous section, the joint probability density function of global parameters could be expressed as a multi-variate normal density with independent components. In this section, we shall demonstrate a complementary approach which results in a complex joint probability density function which must be integrated over a nice feasible region.

In this approach a random variable is created for each path P and the feasible region formed by the following set of inequalities:

$$\begin{aligned} 0 &\leq P_1 \leq \eta \\ 0 &\leq P_2 \leq \eta \\ &\vdots \\ 0 &\leq P_N \leq \eta. \end{aligned} \tag{4.28}$$

In other words we require each path to have a delay greater than zero and less than η for the whole circuit to have a delay no greater than η. The feasible region is shown in Figure 4.8.

The JPDF of path delays $f(P_1, P_2 \ldots P_N)$ is however much more complicated. Even for tree-circuits (no re-convergent fanout) where the gate delays are independent, the path delays are not independent owing to the large amount

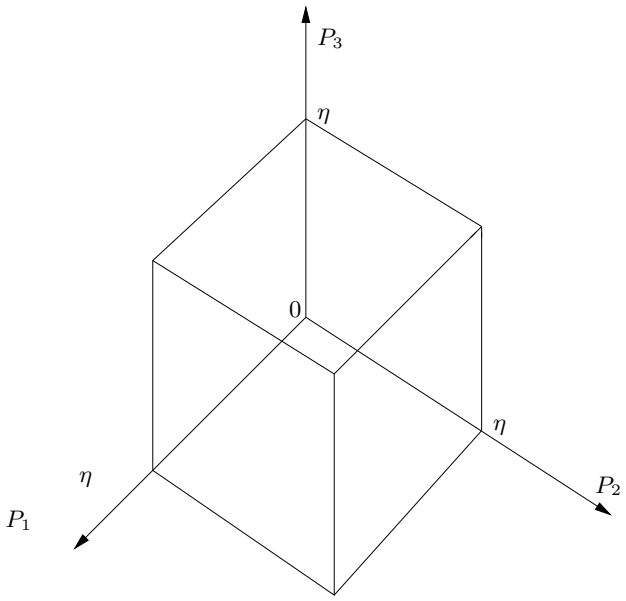


Figure 4.8: Feasible region for performance space integration.

of path sharing. If the path delays are taken to be normal random variables, then the joint probability density function becomes a multivariate normal with a non-diagonal correlation matrix of size $N \times N$ where N is the number of paths in the system.

Chapter 5

Integration over a polytope

In this chapter we present methods to calculate the integral that gives us the yield for a particular performance. We examine first a "brute-force" method proposed by Cohen and Hickey [16] of complexity exponential in the number of dimensions but linear in the number of paths of the system. This method is therefore applicable when we can reduce the number of dimensions to a small number. The error bounds produced by the method are valuable in that they are not probabilistic. There exist a surfeit of numerical integration routines to calculate integrals in special multi-dimensional regions such as a sphere, cube etc, but there are no known routines for general polytopes. We shall review a few numerical integration routines for certain special regions and make greater use of them in succeeding chapters.

We shall also look more carefully at Monte-Carlo methods. Two variants will be examined. One kind of Monte-Carlo is the "all-performances-at-once" Monte Carlo whose main advantage is the small amount of computation time needed. We explore the weaknesses of this method and propose a "performance-at-a-time" Monte-Carlo method that, at the cost of greater computation time, addresses some of these weaknesses.

5.1 Cohen and Hickey method

The basic Cohen and Hickey algorithm is intended to calculate the volume of a given polytope. Our interest is in the weighted volume of a polytope, rather

than the volume itself. We begin with a discussion of the algorithm to compute volume. This is a "brute force" algorithm that divides up the polytope of interest into small parallelepipeds and counts the number of parallelepipeds falling within the polytope as an estimate of the volume of the polytope. It relies crucially on the following fact: if all vertices of a n-parallelepiped lie inside a polyhedron, then all points within the n-parallelepiped lie within the polytope. Therefore checking feasibility of the corners of a n-parallelepiped will allow us to determine whether the parallelepiped lies entirely within the polytope. The algorithm proceeds recursively and starts with a large parallelepiped that clearly contains the feasible region (in our context, this could be the 4-sigma box which almost surely contains the feasible region for any performance value). Then it divides the given parallelepiped into 2^n smaller parallelepipeds, each of side equal to half the side of the original box. Then it proceeds to establish if each of these smaller parallelepipeds is contained within the polytope. For every parallelepiped completely contained within the polytope, it adds the volume of the parallelepiped (trivially calculated) to the running volume counter, and does not further subdivide the parallelepiped. For parallelepipeds that are partially contained within the polytope, it first checks to see whether the size of the parallelepiped is already the smallest user-defined size. If not, the parallelepiped is further sub-divided and the procedure repeated. On the other hand, if the parallelepiped is of the user-defined smallest size, then the volume of the intersection of the parallelepiped with the polytope is determined in one of two ways: (a) in the first method, for each of the 2^n vertices of the parallelepiped, the algorithm checks whether the vertex belongs in the feasible region or not. Finally the contribution of the box to the total volume is taken to be $k/2^n$ where k is the number of vertices found to belong in the feasible region. (b) in the second method, a Monte-Carlo analysis is performed within a box to establish the fraction of the volume of the parallelepiped that falls within the polytope.

The procedure is illustrated for an arbitrary polytope in Figure 5.1. We shall now give the pseudo-code of Cohen and Hickey's method from [16] and follow it up by an analysis of its complexity. First we shall need some notation:

x: an n-dimensional vector representing the bottom left corner of a box
Δ_l: integer chosen by the user to represent the grid size in the lth direction.
$maxd$: maximum depth of recursion
d: the depth of recursion
X^d: points of the form $v_1, v_2, ..., v_n$ where $v_i = l_i + k\frac{(u_i - l_i)}{\Delta_i^d}$

C_x^d: the parallelepiped whose bottom left corner is the point x and whose ith side is of length $(u_i - l_i)/\Delta_i^d$
$\left\|C^d\right\|$: the volume of C_x^d for any x. The volume is simply the product of all the sides of the cubes i.e., $\prod_{i=1}^{i=n} \frac{(u_i - l_i)}{\Delta_i^d}$.

Procedure Vol(d, **x**)
begin
if d ¡ maxd **then**
for all x_j such that $C_{x_j}^{d+1}$ subset of C_x^d **do**
begin
if $C_{x_j}^{d+1} \subset P$ **then**
V ← V + Volume($C_{x_j}^{d+1}$)
else
Vol(d+1, x_j)
end
end
end of Vol

To set the computation off, we call Vol(0, $(l_1, l_2, ..., l_n)$.

In the worst case, the algorithm explores $O((\prod_{i=1}^{i=n} \Delta_i)^{maxd})$ nodes. This is because when we divide a parallelepiped at a certain level of recursion, we create $\prod_{i=1}^{i=n} \Delta_i$ nodes. If each $\Delta_i = 2$, then this means we create 2^n smaller parallelepipeds when we divide a given parallelepiped. In this case, checking if all the vertices of a given parallelepiped belong in a feasible region requires $2^n mn$ multiplications as an n-dimensional parallelepiped has 2^n vertices, there are m inequalities defining the polytope, and for any inequality we have to perform $O(n)$ multiplication operations to determine if a given vertex satisfies it. More generally when each Δ_i may be different from 2, this leads to an overall complexity of $O(mn(\prod_{i=1}^{i=n} \Delta_i)^{maxd})$, or when all the Δ_is are equal to 2, an overall complexity of $O(2^{n(maxd)}mn)$.

Let us now investigate what is the minimum number of smallest parallelepipeds that intersect the surface of the given polytope, as this number will provide some understanding of the difference in the volume estimates provided by the upper and lower bounds of the Cohen and Hickey algorithm. The example we consider is that of a regular polygon of n sides with a very large n, so that we might essentially consider the polygon to be a sphere in n dimensions of radius, say r (this simplifies the analysis). Let us restrict

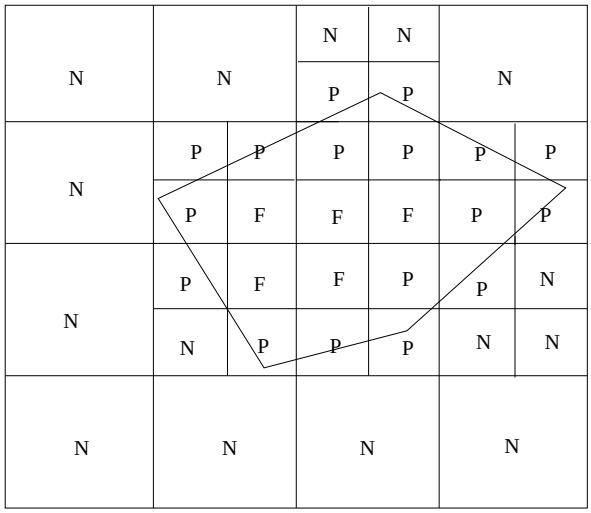


Figure 5.1: Cohen and Hickey method in action. Squares marked F fall entirely within the polytope and are fully counted; squares marked P fall partly within the polytope and are partially counted; squares marked N fall entirely outside.

our analysis to two dimensions. Figure 5.2 shows how we go about our task. We first construct a square of side $l = 2r$ that contains the circle, and then proceed to recursively carve up the area of the circle. Let k be the number of levels of recursion, and each Δ_i of the algorithm above be equal to 2. Then the smallest parallelepiped has a side of length $l/2^k$. We note that a lower bound on the volume of any polytope can be computed from the sum of the volumes of all parallelepipeds that are *fully* contained within it. Also the sum of all the *full* parallelepipeds and all the smallest parallelepipeds that intersect the boundary of the circle will give us an upper bound on the area of the circle. A lower bound on the minimum number of smallest parallelepipeds intersecting the surface , say p can be seen to be

$$p > \frac{2\pi r}{\frac{l}{2^k} * 4}.$$

This is because the parallelepipeds cover the surface and the maximum circumference of intersection is no greater than the perimeter of the smallest

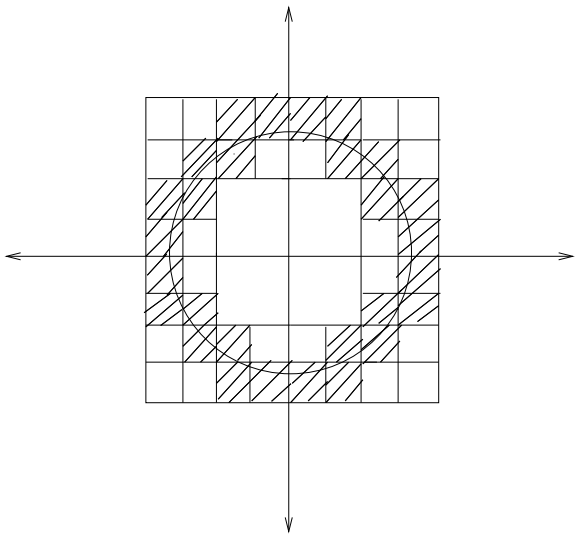


Figure 5.2: Computing the volume of a sphere; the shaded cubes are partial cubes whose volume is approximated.

parallelepiped. Substituting $r = l/2$, we obtain

$$p \geq \frac{\pi 2^k}{4}.$$

The total area of all the parallelepipeds of the smallest size that intersect the boundary of the circle is $\frac{\pi 2^k}{4}\frac{l^2}{2^{2k}} = \frac{\pi l^2}{2^{(k+2)}}$. If V represents the area obtained by summing up the area of the circle ($V = \pi r^2$), then the area contained in the parallelepipeds intersecting the boundary can be seen to be $\frac{V}{2^k}$. Thus the difference in the area estimates of the lower and upper bounds of the Cohen and Hickey algorithm in two dimensions can be seen to be $\frac{V}{2^k}$, in typical cases where we can assume that each parallelepiped that intersects the surface has a non-zero portion of its volume inside the polytope.

The above equation confirms the intuition that an increase in the number of recursion levels will give us better estimates of the the volume of the polytope. For d dimensions, one can show that the error becomes $\frac{V(d-1)}{2^k}$, showing that for the same number of recursion levels, Cohen and Hickey's algorithm will perform poorer at higher dimensions. The algorithm of Cohen and Hickey can be adapted to calculating the weighted volume of an arbitrary polytope.

This application is discussed at length in [44]. It must be remarked here that the basic algorithm of Cohen and Hickey can be speeded-up considerably through the adoption of a few tricks. For example, if a particular path turns out to be infeasible at all vertices, then the recursion can be stopped immediately, as further sub-division of the parallelepiped in question will not yield any smaller boxes that are feasible for the given path. Secondly, if a particular path is feasible for all vertices of given parallelepiped, then in further recursion for that parallelepiped, the particular path's feasibility for a vertex need not be checked, as any such vertex is bound to be feasible for the path. For a more detailed discussion of these tricks, the reader may consult the original paper of [16], as also [44].

5.2 Computing the entire yield curve by a Monte-Carlo method

The brute-force algorithm of Cohen and Hickey is impressive for low dimensions, as the computational results at the end of this chapter show. The exponential complexity of Cohen and Hickey's approach is not surprising - we know from [19] that estimating the volume of an arbitrary polytope (closed bounded polyhedron) is a #P-complete problem. Cohen and Hickey's approach provides us a lower bound of the true volume to within a constant factor and therefore cannot take polynomial-time. The complexity of the problem ensures that for an efficient solution, we must take recourse to Monte-Carlo methods.

We can use Monte-Carlo in two ways to calculate the entire yield curve:
(a) We divide the entire range of performances at which we wish to compute yield into bins $B_1, B_2, ..., B_n$. Then we generate a very large number of samples in the 3σ box around the mean of the joint probability density function of the process parameters and then for each of the sampled points, calculate circuit delay and augment the bin-count for the bin in which the calculated circuit delay belongs. When we are finished with all the samples, we divide each bin-count by the total number of samples and thus obtain the probability of circuit delay belonging to each bin. Let us call this the "binning-Monte-Carlo procedure."
(b) A more costly alternative is to define for each performance a feasible region bounded by path hyperplanes as in the beginning of this chapter, and

then perform Monte-Carlo integration for each yield integral.

The advantage of this formulation ("performance-at-a-time-Monte-Carlo") is that it enables us to use specialized Monte-Carlo sampling procedures for each separate yield integral and thus possibly obtain lower-variance estimates, especially for fast performances, which are arguably the most interesting portion of the yield curve. For low yields, most of the points generated in the 3σ box fall outside the feasible region, making the variance of the estimate huge. Therefore, for low yields at least, we must consider alternative (b). The following section concretises the idea behind the "performance-at-a-time" Monte-Carlo procedure.

5.3 Monte-Carlo integration

The basic idea of Monte-Carlo integration is simple. Suppose that we wish to estimate the area of the region within the polytope in Figure 5.3. To do so, we construct a box large enough to contain the polytope. Then we throw darts at random in the box, and count the number of darts that fall within the polytope. The ratio of darts that fall within the polytope to the total number of darts thrown gives us an estimate of the area of the polytope. This basic scheme works in any number of dimensions. The simplicity of this scheme is not its only advantage - Monte Carlo methods are the only ones known whose accuracy is independent of dimension.

The previous paragraph dealt with estimating the volume of arbitrarily shaped objects. We are however interested in integrating a probability density function over an arbitrary polyhedron in parameter space. In other words we are interested in calculating the following integral:

$$\int\int\int_R f(z_1, z_2 \dots z_n) dz_n dz_{n-1} \dots dz_1. \tag{5.1}$$

The Monte-Carlo integration routine for this integral is inspired by the discussion for volume estimation. Here we simply enclose the region R by a bounding box C and generate points from a uniform distribution within C. If we generate n points $Z_1, Z_2, ..., Z_n$ in the region C, we calculate the integral as follows:

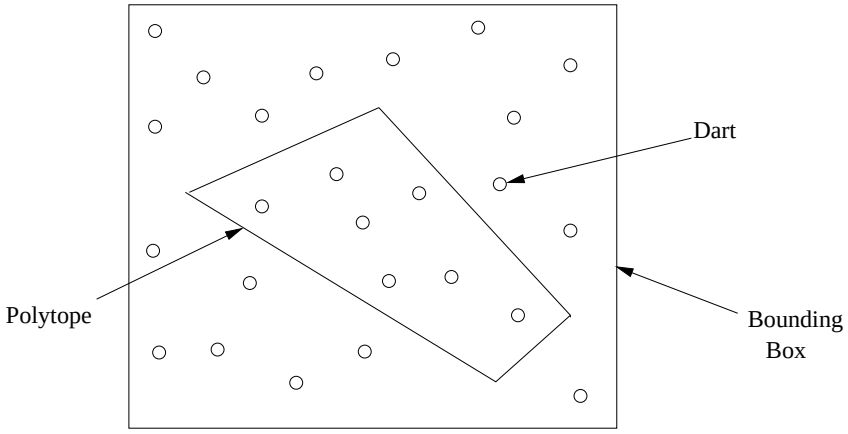


Figure 5.3: Basic idea of Monte-Carlo integration.

$$I = ((\sum_{i=1}^{n} f(Z_i)K(Z_i))/n) * Volume(C). \tag{5.2}$$

where $K(Z_i) = 1$ for $Z_i \in R, 0$ otherwise, is an indicator function. In the next section we shall investigate the integration of a simple function $f(Z) = 1$ over an arbitrary region R of n-dimensional space. The conclusions of the next section will be found to carry over to the case of integrating an arbitrary function over a complicated region in n-dimensional space.

5.4 Evaluating a simple Monte-Carlo integral: role of the bounding box

We are interested in computing the following integral by a Monte-Carlo method:

$$I = \int\int\int_R dz_n dz_{n-1} \dots dz_1. \tag{5.3}$$

Let X be a random variable that estimates the integral we are interested in. In other words, let

$$E(X) = \mu = I. \tag{5.4}$$

The accuracy of the Monte-Carlo procedure used to calculate X can be gauged from its variance, where variance is defined as:

$$\sigma^2 = E(X^2) - (E(X))^2. \tag{5.5}$$

Then we can use Chebyshev's inequality to establish our confidence in the estimate of the integral as follows:

$$Pr(\left|X - \mu\right| \geq \epsilon) \leq \frac{\sigma^2}{\epsilon^2}. \tag{5.6}$$

It is time to establish the role played by the bounding box in determining the accuracy of the Monte-Carlo procedure. In Figure 5.4, we wish to estimate the area of the shaded region via a Monte-Carlo procedure.

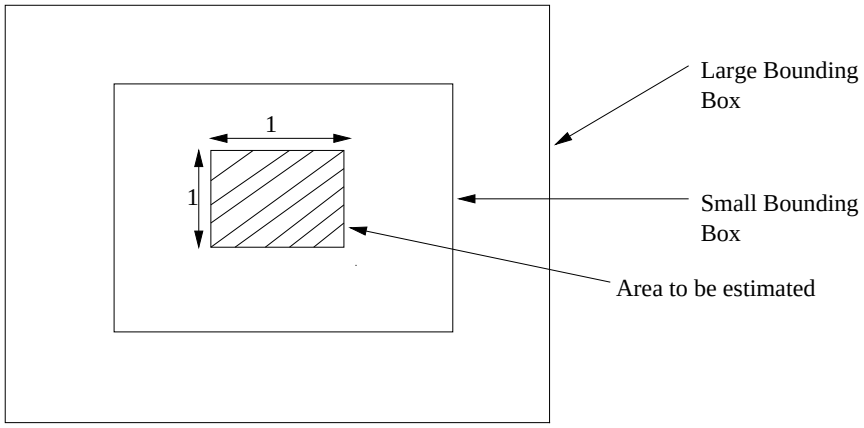


Figure 5.4: Size of bounding box matters ...

We know from inspection that this area is one, so we can accurately gauge the performance of a Monte-Carlo procedure to estimate the area. Let us sample n points uniformly from within the bounding box of side L. Let Y be the random variable denoting the number of points that fall within the shaded region of interest. It is easy to see that

$$Pr(Y = k) = \binom{n}{k} p^k (1-p)^{n-k}, \tag{5.7}$$

where $p = \frac{1}{L^2}$. In other words Y obeys a binomial distribution. Thus we know that

$$E(Y) = np = \frac{n}{L^2} \tag{5.8}$$

and

$$\begin{aligned} \sigma^2(Y) &= E(Y^2) - (E(Y))^2 \\ &= np(1-p) \\ &= \frac{n}{L^2}(1 - \frac{1}{L^2}). \end{aligned} \tag{5.9}$$

We are however interested in a random variable Z such that

$$Z = \frac{(L^2)Y}{n}. \tag{5.10}$$

It is Z that provides us with the quantity we seek, namely the area of the shaded region of Figure 5.4. It is easily seen that $E(Z) = \frac{(L^2)E(Y)}{n} = (L^2)\frac{1}{L^2} = 1$. Thus Z is an estimator of the area of the shaded region. We are interested in the quality of the estimate provided by Z. To this end, we see that the variance of Z is given by

$$\begin{aligned} \sigma^2(Z) &= E(Z^2) - (E(Z))^2 \\ &= \frac{L^4}{n^2} E(Y^2) - \frac{L^4}{n^2}(E(Y))^2 \\ &= \frac{L^4}{n^2}(E(Y^2) - (E(Y))^2 \\ &= \frac{L^4}{n^2}(\frac{n}{L^2}(1 - \frac{1}{L^2})) \\ &= \frac{(L^2 - 1)}{n}. \end{aligned} \tag{5.11}$$

In d dimensions $\sigma^2(Z)$ would be $\frac{L^d-1}{n}$. The above equation establishes that the variance of Z is directly related to the volume of the bounding box - the larger the bounding box, the greater the variance and the poorer the quality of the estimate Z. This observation appears to contradict the oft-repeated assertion about Monte-Carlo integration, namely, that it is independent of the dimension of the problem. The reason for this is that the basic probability that a sampled point belongs to the feasible region is the ratio of the size of the feasible region to the bounding box that contains it, and this probability can be made arbitrarily small by increasing the size of the bounding box. This has deleterious consequences for the variance of the procedure. Thus any "bounding-box" based Monte-Carlo method will necessarily suffer from dimensionality problems and to get around this, we must approximate the region of interest by a close-fitting bounding box. Given that our region of interest is an arbitrary polytope in parameter-space, this is not easy to do. In the next section we shall explore a fairly basic scheme to find a small bounding box. In the next chapter we shall show how the ellipsoidal method can provide a natural tight-fitting bounding box.

5.5 Finding a small bounding box

Consider the feasible region of Figure 5.5. A first-cut approach to finding a tight bounding box for this region is the smallest axis-parallel box drawn around the feasible region in this figure. Sampling within this smaller bounding box will result in a Monte-Carlo estimate that is orders of magnitude more accurate than one obtained by sampling in the larger 3σ bounding box. The only question that remains is how does one go about finding this new bounding box.

Suppose we are trying to find the boundary of the bounding box in the positive Δz_1 axis direction. For each point t on the $Deltaz_1$-axis, we can imagine constructing a hyperplane given by $\Delta z_1 = t$, and then checking to see if this hyperplane intersects the feasible region. If the hyperplane does intersect the feasible region, then we know that t does not define the rightmost bound on the feasible region in the direction of the positive Δz_1 axis. This procedure suggests the use of a linear programming technique to

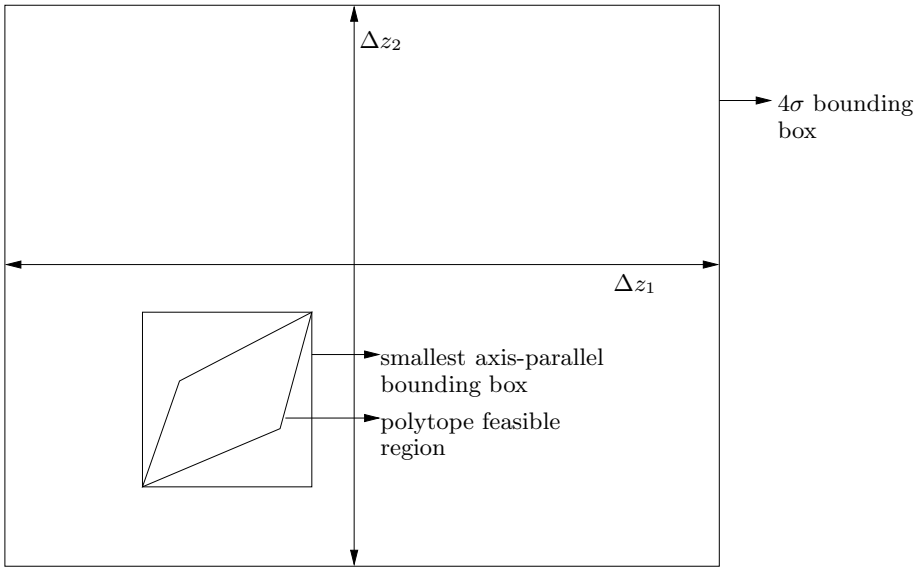


Figure 5.5: A polytope feasible region contained in an axis-parallel bounding box. The much larger 3σ bounding box is also shown.

determine the rightmost boundary in the +ve Δz-axis:

$$\begin{aligned} &\max \Delta z_1 \text{ subject to} \\ &\qquad R\Delta z \leq b, \end{aligned} \tag{5.12}$$

where the feasible region, together with the 4σ constraints is represented by $R\Delta z \leq b$. The argument above extends in the other axial directions as well, and thus we can determine the sides of the bounding box by solving 4 linear programmes. This basic approach extends to n dimensions, but we will need to solve $2n$ linear programmes in n-dimensional space.

The effectiveness of the approach in practice depends on the constraint set. In particular the approach fails to yield a good bounding box if the feasible region contains "extreme points" of the form $(4\sigma, t_1, t_2 \ldots t_{n-1})$. In other words, if it is possible for each global parameter to take on its extreme value and still have a feasible point, then the linear programmes of the type given in (5.12) will give poor results. Let us examine two different types of constraint sets as shown in Figure 5.6. Let us consider the constraint set shown in Figure 5.6a. Each column of the constraint matrix is monotone, i.e., all

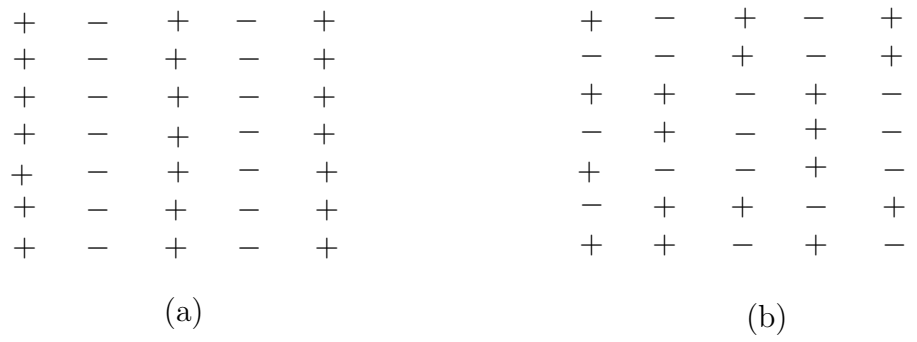


Figure 5.6: Different types of constraint sets.

elements are either positive or negative. Let us focus on generating extreme points for the first coordinate, i.e., points of the form$(4\sigma, t_1, t_2 \dots t_{n-1})$ and $(-4\sigma, t_1, t_2 \dots t_{n-1})$. Since the coefficients of the first column are all positive, it is easy to see that negative extreme points for the first coordinate can be constructed. The case for positive extreme points is a bit more complicated. Setting $\Delta z_1 = 4\sigma$ causes the left hand sides of each constraint to increase. The increase can be balanced by the other variables taking suitable values. If the signs in a particular column are all negative, then the variable corresponding to that column should be given the value 4σ. Otherwise it should be given the value -4σ. This has the effect of trying to drive the left hand side back into feasibility. Thus we may be able to construct positive extreme points as well. Note that the argument depends on the specific values of the coefficients in addition to the signs but the monotonicity of the coefficients in each column means that the assignments to the variables do not contradict the requirements of each constraint.

On the other hand, if we have a set of constraints (for each axial direction) as depicted in Figure 5.6b, then we can ensure that extreme points will not occur. Once again let us focus on generating extreme points for the first coordinate direction. Note that there are two constraints which, for all directions other than z_1, have coefficients of opposite sign (the first and the third constraint in Figure 5.6b). Now let us suppose that z_1 takes its extreme

value, and that

$$\begin{aligned} d_1 + p_1 \Delta z_1 &> t \\ d_2 + q_1 \Delta z_1 &> t. \end{aligned}$$

In order for the point to remain in the feasible region, the other terms in the respective hyperplane equations must compensate for the excess delay caused by Δz_1 taking its extreme value. If there was a coordinate (other than Δz_1) for which the coefficients in the two constraints were of the same sign then that coordinate could be made to assume an appropriate value in order to drive both constraints back into feasibility. However, since there is no such coordinate, the other coordinates cannot play the compensatory role in *both* constraints. For example, if Δz_i has a positive coefficient in the first constraint (and therefore a negative coefficient in the second constraint), it must take a negative value in the first constraint to be of any use in a compensatory role in the first constraint. But this will simultaneously drive the second constraint further into infeasibility. Thus the opposite signs limit the ability of the other coordinates to balance out the extreme behaviour of any one coordinate. This makes it unlikely that any point with one of its coordinates assuming an extreme value lies in the feasible region.

5.6 Numerical integration

We shall now investigate a deterministic method of integrating over a polytope. The one-dimensional Gaussian formulae are well known and frequently used but multi-dimensional integration via numerical methods is a tricky proposition. The straightforward extension of one-dimensional integration methods leads to the so-called product rule, wherein one applies a one-dimensional integration rule in each dimension. The downside of product rules is that they are exponential in the number of dimensions. Let us formalise the argument.

A one-dimensional integration rule has the following form:

$$\int_a^b f(x)dx = \sum_{i=1}^{i=N} w_i f(x_i), \tag{5.13}$$

where the w_is represent the weights and the x_i represent the points at which the function is sampled. The weights and points are calculated so that every function of degree no greater than m is integrated *exactly* by the formula. If one uses the well-known Gaussian quadrature than the number of points needed to integrate exactly a function of degree no greater than m is $2m - 1$. Suppose that we wish to integrate a function in two-dimensions. The integration rule in two-dimensions will have the following form:

$$\begin{aligned} &\int_{x_0}^{x_1}\int_{y_0}^{y_1} f(x,y)dydx \\ &\approx \sum_{i=1}^{i=N} w_i \int_{x_0}^{x_1} f(x,y_i)dx \\ &\approx \sum_{i=1}^{i=N}\sum_{j=1}^{j=M} w_i w_j f(x_i,y_j). \end{aligned} \tag{5.14}$$

Thus the number of points needed is the product of the number of points in each dimension, and this is what leads to the exponential complexity. It has to be remarked that there is no general product formula for integrating over an arbitrary polytope. The literature deals with integration over special regions such as a simplex, a sphere, a cube, an octahedron and so on. For reasons that will become clear in the next chapter, we shall be especially interested in integration formulae over a sphere.

Let us examine how to construct an integration formula for a polynomial on the unit sphere in 3 dimensions. An integration formula of degree m in 3 dimensions should integrate exactly a polynomial of degree no greater than m. In other words, the integral we are targeting for exact integration is of the form:

$$I = \int_{-1}^{1}\int_{-\sqrt{1-\mathrm{x}_1^2}}^{\sqrt{1-\mathrm{x}_1^2}}\int_{-\sqrt{1-\mathrm{x}_1^2-\mathrm{x}_2^2}}^{\sqrt{1-\mathrm{x}_1^2-\mathrm{x}_2^2}} x_1^\alpha x_2^\beta x_3^\gamma dx_3 dx_2 dx_1. \tag{5.15}$$

Note that the limits on the integrals are not independent of each other, and this makes the straightforward application of the product-rule concept impossible. The key step is to introduce spherical coordinates.

$$\begin{aligned} x_1 &= r\cos\theta\cos\omega \\ x_2 &= r\sin\theta\cos\omega \\ x_3 &= r\sin\omega. \end{aligned} \tag{5.16}$$

Substituting for x_1, x_2, and x_3 in the integral above we obtain

$$\begin{aligned} I &= \int_{-1}^{1}\int_{-\pi/2}^{\pi/2}\int_{-\pi/2}^{\pi/2} r^{\alpha+\beta+\gamma}\cos^{\alpha}\theta\sin^{\beta}\theta\cos^{\alpha+\beta}\omega\sin^{\gamma}\omega J d\omega d\theta dr \\ \text{where } J &= r^2 cos\omega. \end{aligned} \tag{5.17}$$

In the above equation J is the Jacobian factor. The crucial advantage of using polar coordinates is that the integrals all have independent limits, and we can then use a product-rule.

5.7 Experimental set-up

The sensitivities of the delays of the gates to the process parameters are difficult to obtain in practice. The partitioning of the die-area into sub-regions which are affected by unique process parameters as shown in Figure 4.5 appears reasonable, but once again we cannot be absolutely sure because of the unavailability of real industrial data. The discussion in a previous section on finding a small bounding box highlighted the role played by the constraints in determining a small bounding box. In particular, it was pointed out that mixed-coefficient (+ve and -ve coefficients in an arbitrary column) path sensitivity matrices enable us to find a bounding box for the feasible region much smaller than the obvious 3σ box inspired by the JPDF of process parameters. However, in reality we believe mixed coefficient constraint matrices are unlikely to occur because a particular process parameter is likely to affect each gate in the design in the same way, i.e., a positive change in the process parameter in question is likely to speed-up or slow-down all gates in the chip (albeit by different amounts), because the underlying physics that links each gate to the process parameter in question must be the same. In other words the sensitivities of the gates to a particular process parameter are likely to be

of the same sign. An interesting point to note here is that mixed-coefficient constraint matrices mean that it is highly unlikely that circuits will be manufactured that are faster than their nominal delay (delay they would have in the absence of any process variations). This is because there is likely to be no single process variation direction that speeds up *all* paths. In fact if the path vector directions are distributed uniformly over the unit sphere, any process variation direction will slow down some one or the other path thus impacting chip delay.

In light of the discussion above, we have assumed that the sensitivities of all the gates to a particular process parameter are of the same sign. This assumption leads to the interesting property that some manufactured chips will be faster than the nominal chip. The impact of this property is explored further in Chapter 6. To model locational dependence we ascribe to each gate of a particular type a sensitivity profile that is a function of the gate's position on the chip. Thus we believe that our experimental set-up corresponds to reality, although we have no means of conclusively establishing this to be the case.

Chapter 6

Ellipsoidal approximation of the feasible region

6.1 Introduction

It has already been established in the previous chapter that the original polytope representing the feasible space is a troublesome region for integration. The non-standard shape of the polytope precludes the use of effective numerical integration techniques since such techniques are only available for special geometric regions such as a sphere, a cube, a simplex and so on. This suggests that in order to use numerical integration techniques, we must *approximate* an arbitrary polytope by a nice geometric region. In this chapter, we propose the use of an ellipsoid to approximate the polytope.

We first describe mathematical programming techniques to compute the maximum volume ellipsoid that can be inscribed in the feasible region. Having computed the ellipsoid, we use it as a surrogate for the feasible region, and compute the yield integral assuming the ellipsoid is the feasible region. Some standard numerical integration routines are used for this purpose, and we can calculate the lower bound to the true curve (assuming that we can in fact integrate exactly over the ellipsoid). A section is devoted to improving this lower bound by trying to cover the feasible region with two ellipsoids instead of one. In the latter part of the chapter, we show how to use the ellipsoid to devise Monte-Carlo methods for yield integration of very low variance, especially for the faster performances. The Monte-Carlo methods target two

situations: one in which the integrand is not a standard joint probability density function, and is therefore difficult to sample from, and the other in which the integrand is a Gaussian joint probability density function and is very easy to sample from.

6.2 Ellipsoidal approximation

Consider the polytope in Figure 6.1. We approximate this polytope by the maximum volume ellipsoid that can be inscribed in it. It can be seen that the ellipsoid appears to represent the shape of the polytope reasonably well. The reader may note that this happens in more general situations as well. Ellipsoids have geometric properties that are very desirable. For example [100] states that one can find the global minimum of any quadratic over an ellipsoid in polynomial time, while the general problem of finding a global minimum of a quadratic over a polytope is NP-complete. Ellipsoidal approximations of a polytope have been used in Khachiyan's famous polynomial-time algorithm for linear programming [50]. Recent approaches to estimating the volume of convex bodies in high dimensions use an ellipsoid to approximate the volume of the convex body. Crucially for us, there are efficient ways of computing the maximum volume ellipsoid inscribed in a polytope. This fact makes an ellipsoidal approximation of the feasible region a particularly desirable one.

A number of interior-point algorithms have been proposed in the literature for finding the maximum volume ellipsoid that can be inscribed in a polytope. Nesterov and Nemirovskii [73] constructed a three-stage barrier method for finding an ϵ-optimal ellipsoid Esuch that $Vol(E) \geq Vol(E^*)e^{-\epsilon}$ where E^* is the maximum volume ellipsoid contained within the polytope P and $\epsilon \in (0, 1)$. However, the algorithm requires the knowledge of the ratio of two concentric balls, one containing P and the other contained within P. With this information, it has a complexity estimate of $O(m^{2.5}(n^2 + m) \ln \frac{mR}{\epsilon})$, where m is the number of constraints and n is the number of dimensions. The term n^2 comes from having to solve linear systems containing an $n \times n$ linear system. The algorithm of Khachiyan and Todd [51] runs in time $O(m^{3.5} \ln(\frac{mR}{\epsilon}) \ln(\frac{n \ln R}{\epsilon}))$. It uses the same barrier method as the method of Nesterov and Nemirovskii but avoids using an $n \times n$ matrix variable. After further improvements, Anstreicher [4] obtained an $O(m^{3.5} \ln(\frac{mR}{\epsilon}))$ algorithm for the problem.

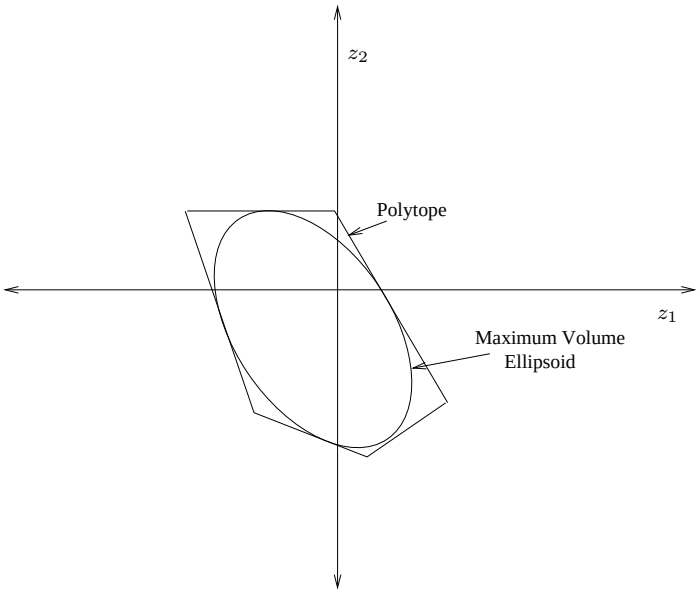


Figure 6.1: Maximum volume ellipsoid approximation to feasible polytope.

We shall review two approaches to the maximum volume ellipsoid problem. The first of these formulates the maximum volume ellipsoid problem as a MAXDET problem with linear matrix inequalities and solves it using an interior point algorithm. The second approach due to Zhang [100] is a more efficient approach and is the one we use in this thesis. However all algorithms for finding the maximum volume ellipsoid given a set of m linear constraints in n variables suffer from the fact that they are non-linear in m. This makes the ellipsoid approach impractical for situations where we have a huge number of constraints, and cannot prune them. A remarkable and highly desirable feature of using ellipsoid approaches, however, is the relatively mild dependence on the dimensionality of the problem.

6.3 MAXDET problem

Below we present a mathematical programming technique modelled on the method of [93] to compute the maximum volume ellipsoid that fits in the

polytope representing the feasible space.

The feasible space can be represented as a set of linear inequalities. Recalling (4.4), we can write:

$$F = \{z \mid R_i^T z \leq t_i, i = 1, 2, \ldots, P\}. \tag{6.1}$$

In the above equation the matrix R_{ij} is the sensitivity of the ith path to the jth global parameter, z represents the vector of global parameters, and P is the number of paths. An arbitrary ellipsoid can be expressed as a collection of points z satisfying the equation

$$E = \{z \in R^n : (z - d)^T (BB^T)^{-1}(z - d) \leq 1\}. \tag{6.2}$$

The above can be recast as

$$E = \{z \in R^n : z = By + d \mid \|y\| \leq 1\}, \tag{6.3}$$

where B is a symmetric, positive-definite matrix that linearly transforms all points in the unit sphere, and d is the centre of the ellipsoid. Note that the shape of the ellipsoid is uniquely determined by the matrix $(BB^T)^{-1}$. However this does not impose a restriction on B since any orthogonal transformation of B will do as well as the following shows: $(BQ)(Q^T B^T)^{-1} = (BB^T)^{-1}$ for any orthogonal matrix Q. Our interest is in the maximum volume ellipsoid that fits in the feasible region. To find such an ellipsoid, it is sufficient to maximise the determinant of the transformation matrix B since the volume of the ellipsoid is proportional to the determinant of the transformation matrix. The requirement that $E \subset F$ means that

$$R_i^T (By + d) \leq t_i \, \|y\| \leq 1, i = 1, 2, \ldots, P. \tag{6.4}$$

This in turn means that

$$sup_{\|y\| \leq 1}(R_i^T By + R_i^T d) \leq t_i, i = 1, 2, \ldots, P \tag{6.5}$$

or

$$\|BR_i\| + R_i^T d \leq t_i, \; i = 1, 2, \ldots, P. \tag{6.6}$$

Now that we have established the set of constraints, we turn our attention to the objective function. The function $\log detB$ can be shown to be a convex

function. Thus we establish the following convex optimisation problem to determine the maximum volume ellipsoid that fits in the feasible region:

$$\begin{aligned} &\text{maximise } \log detB \\ &\text{subject to } B = B^T > 0 \\ &\text{subject to } \|BR_i\| + R_i^T d \leq t_i,\ i = 1, 2, ..., P. \end{aligned} \tag{6.7}$$

The above formulation can be recast as a MAXDET formulation and has been the subject of intensive study in the literature [93] and [100]. We will give a brief background of this problem, since it plays a central role in this thesis. For the mathematical details, we refer the reader to the papers of [93] and [100]. The standard MAXDET formulation can be written as

$$\begin{aligned} &\text{minimise } c^T z + \log det(G(z))^{-1} \\ &\text{subject to } G(z) \succ 0,\ F(z) \succeq 0 \end{aligned} \tag{6.8}$$

where

$$\begin{aligned} G(z) &= G_0 + z_1 G_1 + \ldots + z_n G_n \\ F(z) &= F_0 + z_1 F_1 + \ldots + z_n F_n. \end{aligned} \tag{6.9}$$

Here G and F are affine functions with $G : R^n \rightarrow R^{l \times l}$, $F : R^n \rightarrow R^{m \times m}$, $G_i = G_i^T$ and $F_i = F_i^T$. The $\succ$ sign in the formulation stands for matrix inequalities: $G(z) \succ 0 \Rightarrow u^T G(z) u > 0$ $\forall$nonzero u while $\succeq$ means the following: $F(z) \succeq 0 \Rightarrow u^T F(z) u \geq 0$ $\forall u$. The maximum volume ellipsoid problem can be cast in the MAXDET form when we note that the P convex constraints in B and d are equivalent to the P LMIs:

$$\begin{bmatrix} (t_i - R_i^T d) I & BR_i \\ R_i^T B & (t_i - R_i^T d) \end{bmatrix} \geq 0, i = 1, 2, ...P. \tag{6.10}$$

In the above matrix inequality I is the identity matrix, $B = B^T \in R^{n \times n}$ and $d \in R^n$. The approach of [93] solves the MAXDET problem via an interior-point algorithm. The feature of the interior point algorithm is that it solves the problem iteratively and converges to a solution that is within

a user-specified tolerance of the optimal solution. In practice, however, we found the algorithm to be impractical for problems of high dimensionality. Although the number of iterations needed for convergence shows only a modest increase with dimensionality, the amount of time required per iteration increases dramatically and this renders the algorithm impractical. Therefore, in the next section, we turn our attention to a novel approach of [100] that substantially reduces the running time because it does not work with matrix-valued variables.

6.4 Zhang's approach

The basic convex constraints used in Zhang's approach are the same as those of [93]. The convex constraints are

$$\|BR_i\| + R_i^T d \leq t_i, i = 1, 2, ..., P. \tag{6.11}$$

We introduce the notation

$$h(B) = (\|BR_1\|, \|BR_2\|, ..., \|BR_P\|)^T, \tag{6.12}$$

where $h(B)$ is an m-dimensional vector. Requiring that all the convex constraints be satisfied is the same as asking that

$$t - Rz - h(B) \geq 0. \tag{6.13}$$

The volume of an ellipsoid is proportional to the determinant of the transformation matrix. Thus the solution of the following optimisation problem gives us the maximum volume ellipsoid contained in the given polytope P:

$$\begin{gathered} \min \ -\log\det B \text{ subject to} \\ t - Rz - h(B) \geq 0. \end{gathered} \tag{6.14}$$

where B is symmetric and positive definite. The above is a convex programming formulation for which the Karush-Kuhn-Tucker conditions which determine an optimal solution can be derived as below. First we construct the Lagrangian of the convex formulation

$$L(z, B, u) = -\log\det B - u^T(t - Rz - h(B)). \tag{6.15}$$

The vector u is a vector of Lagrange multipliers. The Karush-Kuhn-Tucker (KKT) conditions require that

$$\nabla_z L = 0, \tag{6.16}$$
$$\nabla_B L = 0. \tag{6.17}$$

We know that

$$\nabla[\log\det B] = B^{-1} \tag{6.18}$$
$$\nabla[h_i(B)] = \frac{BR_iR_i^T + R_iR_i^TB}{2h_i(B)}. \tag{6.19}$$

Thus we arrive at the following KKT conditions:

$$\begin{aligned} R^Tu &= 0, && (6.20)\\ B^{-1} - [B(R^TYR + (R^TYR)B]/2 &= 0, && (6.21)\\ v - (t - Rz - h(B)) &= 0, && (6.22)\\ Uv &= 0, && (6.23)\\ u, v &\geq 0. && (6.24) \end{aligned}$$

In the above set of equations $U = Diag(u)$ and $Y = Y(B,u) = Diag(h(B))^{-1}U$, B is symmetric and positive definite and v is a slack variable. The crucial feature of Zhang's approach that is the source of its efficiency is that it does not carry the matrix variable B. One approach to eliminate it is to observe that

$$B(y) = (R^TYR)^{-1/2}. \tag{6.25}$$

Zhang shows that this solution is unique in S_n^{++}. After further manipulations, he obtains the following formulation:

$$\begin{aligned} F(x,y,u,v) &= 0, y, u, v \geq 0 && (6.26)\\ F(x,y,u,v) &= \begin{bmatrix} R^Tu \\ Rz + h(B(y)) + v - t \\ u - Yh(B(y)) \\ Uv \end{bmatrix}. && (6.27) \end{aligned}$$

Zhang solves equations such as the above using a primal-dual algorithmic framework. We shall not go into the details of the algorithm, and instead refer the reader to the relevant paper.

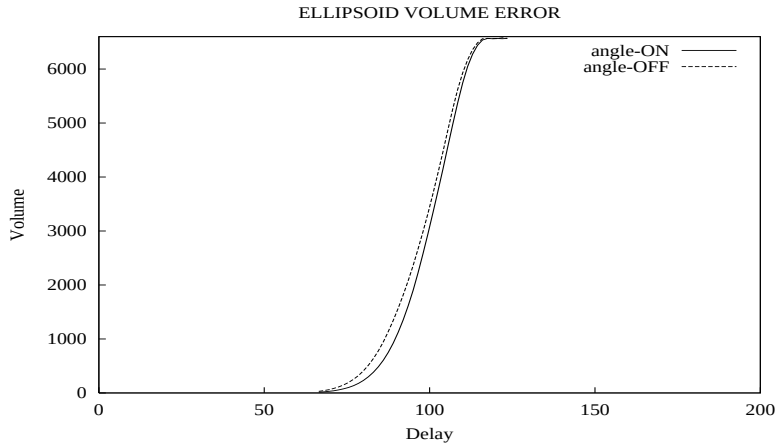


Figure 6.2: We measure the volumes of the maximum volume ellipsoid computed for the subset of C constraints resulting from path filtering ("angle-ON") and the maximum volume ellipsoid computed for C paths according to decreasing order of criticality ("angle-OFF").

6.5 Path filtering and the maximum volume ellipsoid

In Chapter 4, we remarked that there are many millions of critical paths in a moderately large circuit. The maximum volume ellipsoid algorithms described in the previous sections have at least a cubic dependence on the number of constraints in the problem. They are clearly not practical for the millions of critical paths in an average circuit. It is here that the path filtering procedure of Chapter 4 is truly invaluable; instead of finding the maximum volume ellipsoid inscribed in a polytope formed by all the m original hyperplanes (paths), we find the maximum volume ellipsoid for a carefully chosen subset C of the original set of paths. In Figure 6.2, we show how the maximum volume ellipsoid inscribed in a polytope defined by a carefully chosen subset of C constraints is closer to optimal than the maximum volume ellipsoid inscribed in a polytope defined by the top C paths in order of nominal criticality. Together with Figure 4.6, this establishes that the path filtering criterion of Chapter 4 is a useful means of reducing the number of paths in the circuit that need to be considered.

The maximum volume ellipsoid inscribed in a polytope formed by a subset

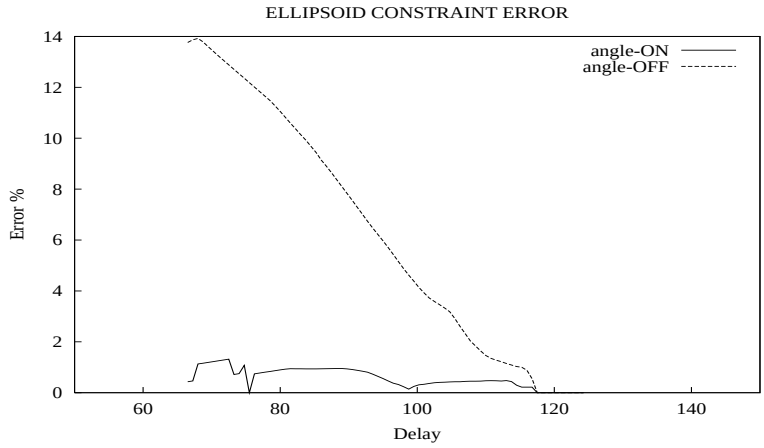


Figure 6.3: Ellipsoid error induced by path filtering. We measure the maximum fractional delay violation of any constraint in the set of paths using (a) the maximum volume ellipsoid computed by the subset of C constraints returned by path filtering ("angle-ON") and (b) the maximum volume ellipsoid computed by the top C paths according to nominal criticality ("angle-OFF").

of the original constraints is no smaller than the maximum volume ellipsoid inscribed in the polytope formed by all the constraints. This naturally leads to the question of how sub-optimal is the maximum volume ellipsoid inscribed in a polytope defined by only a subset of the constraints.

In order to measure the deviation of the approximate maximum ellipsoid from the true maximum ellipsoid, we need only to determine the number of constraints in the whole system that are violated and the extent of the violation. The rationale for this is simple: if the "approximately" maximum volume ellipsoid is feasible for *all* constraints (i.e., does not violate any constraint), then the "approximately" maximum volume ellipsoid is actually the maximum volume ellipsoid. If it violates a few constraints, then the extent of the violations determines how close the "approximately" maximum volume ellipsoid is to the true maximum. A violation of a constraint means that the ellipsoid intersects the constraint in more than one point (in two dimensions, the intersection will be in two points, but in higher dimensions the intersection will be a curve) and is therefore infeasible for the constraint. The extent of the violation can be measured by the amount that the hyperplane must

be moved so that it becomes a tangent to the ellipsoid again (and is then just feasible). This means that the paths representing violated hyperplanes must be speeded up by a small amount, so as to become feasible for the "approximate" maximum volume ellipsoid. If γ_i is the delay of the ith path, and $\delta\gamma_i$ is the amount by which it is speeded up, then the fraction $\frac{\delta\gamma_i}{\gamma_i}$ is a useful measure of the violation of the hyperplane.

For a given required performance, we can define the ellipsoid error as follows:

$$\text{Error } = \max \frac{\delta\gamma_i}{\gamma_i};\ 1 \leq i \leq P \tag{6.28}$$

It remains to calculate $\delta\gamma_i$ for each hyperplane i. Let the hyperplane equation be given by $R_i^T \Delta z \leq \eta - \gamma_i$ where η is the required performance and γ_i is the nominal delay of the ith hyperplane, and R_i is the ith row of the matrix R of equation (4.4). In z-space, the space of global process parameters, this equation can be rewritten as $R_i^T z \leq t_i$. We know that $z = By + d$ where $\|y\| = 1$. Thus the ith hyperplane in the space of global parameters can be transformed in unit sphere space as follows:

$$R_i^T By \leq t_i - R_i^T d. \tag{6.29}$$

If the distance of this hyperplane to the origin is less than one, then it means that this hyperplane in unit sphere space intersects the unit sphere in more than one point. Therefore it also intersects the ellipsoid in more than one point. In order to make the hyperplane a tangent to the hyperplane its nominal delay γ_i must be changed by an amount $\delta\gamma_i$ such that

$$\frac{\delta\gamma_i + (t_i - R_i^T d)}{\|R_i^T B\|} = 1. \tag{6.30}$$

Thus we can write

$$\begin{aligned} \delta\gamma_i &= \|R_i^T B\|\big(1 - \frac{(t_i - R_i^T d)}{\|R_i^T B\|}\big) \\ &= \|R_i^T B\|(1 - q), \end{aligned} \tag{6.31}$$

where q is the distance of the hyperplane from the origin in unit sphere space.

Note that this is a more useful measure of optimality than merely measuring the number of constraint violations. An ellipsoid that violates just one constraint by a huge margin is less preferable to one that only slightly violates a large number of constraints. We show experimental results of the ellipsoid error calculated for the same circuit that serves as a running example in this chapter and the next. Of the 5112 constraints in this circuit (assuming that there are 8 global sources of variation), 100 constraints were extracted according to the path filtering criterion of Chapter 4. The ellipsoid error obtained at each performance is plotted in Figure 6.3.

6.6 Numerical integration over the ellipsoid

Having computed the maximum volume ellipsoid that can be inscribed in the feasible region, we shall treat the ellipsoid as a surrogate for the feasible region. Integrating the joint probability density function over a given arbitrary polytope is a hard problem, but efficient numerical integration techniques can be brought to bear upon the problem of integrating the joint probability density function over an ellipsoid. First we shall transform the integral over an ellipsoid to one over the unit sphere, to make the numerical integration algorithms directly applicable. This transformation is particularly easy to do, and we can make use of the ellipsoid transformation matrix computed by the maximum volume ellipsoid (MVE) algorithm. The transformation is illustrated in Figure 6.4.

Any point z in the ellipsoid can be expressed as follows:

$$z = By + d, \|y\| = 1. \tag{6.32}$$

The yield integral can be written as

$$\begin{aligned} Y &= \int\int \ldots \int_{R^*} f(z_1, z_2 \ldots z_n) dz_n \ldots dz_1 \qquad (6.33) \\ &= \int\int \ldots \int_{R} f(B_1^T y + d_1, B_2^T y + d_2, \ldots B_n^T y + d_n)(\det B) dy_n \ldots dy_1 \end{aligned}$$

where $f(z)$ is the joint probability density function of the global parameters, R^* is the ellipsoid computed by the MVE algorithm, and R is the unit sphere. Now the yield integral is of the form directly suitable for numerical integration. In the next section we explore some standard numerical integration routines to perform integration over the unit sphere.

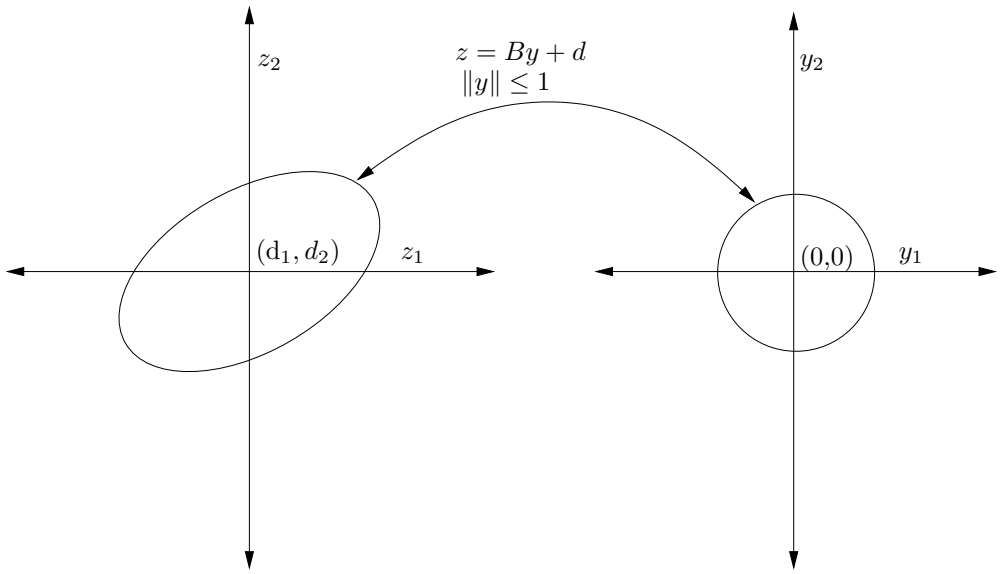


Figure 6.4: Ellipsoid is a linear transformation of the unit sphere.

6.7 Numerical integration- Stroud's formulae

Given a function $f(y_1, y_2 \dots y_n)$ over the unit sphere, Stroud's formulae to integrate the function over the unit sphere have the following form:

$$\int \int \dots \int f(y_1, y_2, \dots y_n) dy_n \dots y_1 = \sum_{i=1}^{i=N} w_i f(v_{i1}, v_{i2}, \dots v_{in}). \qquad (6.34)$$

Recall from Chapter 5 that the points v are independent of the function f. There exist formulae where some of the points v lie outside the unit sphere. In some formulae the weights of the formulae are negative. We prefer, for reasons of numerical stability to have all the points of the formulae belong to the unit sphere, and for all coefficients to be positive as far as possible. The essence of Stroud's formulae is that the weights and points are chosen such that the formulae give us the correct result when the integrand is a polynomial of degree no greater than a specified integer. For example, a degree-3 Stroud formula correctly integrates all polynomials of degree no greater than 3.

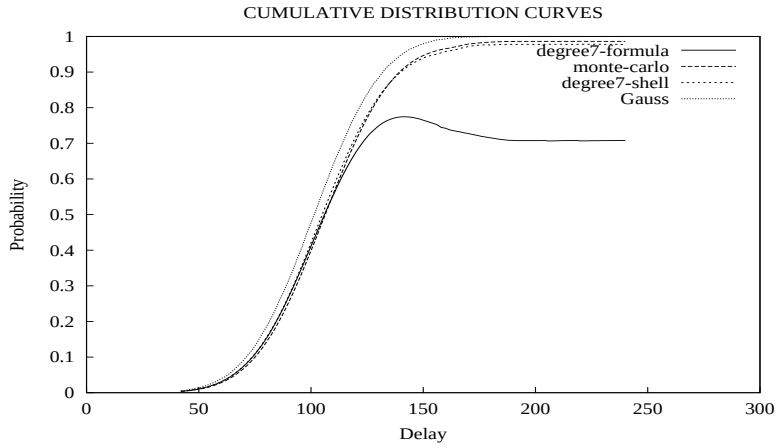


Figure 6.5: Use of a degree-7 integration formula, as well as a degree-7 shell integration formula. The curve representing the ordinary degree-7 integration formula for the whole sphere begins to fall off for the slower performances, while that for the degree-7-shell formula keeps up with the real curve for the ellipsoid integral marked "monte-carlo" over the entire range of performances. The Monte-Carlo curve (marked "Gauss") represents the integral over the entire feasible region defined by the yield polyhedron.

The following is a degree-3 Stroud formula to perform integration:

$$Y = \sum_{i=1}^{i=n} \frac{V}{2n}(f(e_i) + f(-e_i)). \tag{6.35}$$

where $e_1 = (1, 0 \ldots, 0)^T$ and V is the volume of the unit sphere in n-dimensions. We have found that degree-3 or even degree-5 formulas cannot be used for integration because they are too inaccurate. A degree-7 formula appears to be more promising. The degree-7 formula we use for integrating over the entire unit sphere is drawn from [88] and is a derivative of a degree-7 rule used for integrating over the surface of a unit sphere. The degree-7 rule to integrate over the surface of the unit sphere makes use of the following points:
(a) all points of the form $(\pm 1, 0, \ldots, 0)$, namely all on the positive and negative coordinate axes.
(b) all points of the form $(\pm\frac{1}{\sqrt{n}}, \pm\frac{1}{\sqrt{n}} \ldots \pm\frac{1}{\sqrt{n}})$, namely all 2^n points obtained

by taking all possible sign combinations of $(\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}} \ldots \frac{1}{\sqrt{n}})$
(c) all points of the form $(\pm\frac{1}{\sqrt{2}}, \pm\frac{1}{\sqrt{2}}, 0, 0 \ldots 0)$, namely all $4 * \binom{n}{2}$ points obtained by choosing non-zero values in exactly two coordinate positions, and zeroes elsewhere, and then considering all possible sign combinations of such points.

There are $2^n + 2n^2$ points in the degree-7 rule for integrating a function over the surface of a unit sphere. The transformation of this rule to integrate over the entire unit sphere uses $2^{n+1} + 4n^2$ points.

In Figure 6.5 we use the degree-7 formula above to calculate yield integrals for a whole range of performances and compare the cumulative distribution curve obtained with one obtained by Monte-Carlo simulation. We also show the curve resulting from straightforward uniform sampling over the ellipsoid. The two curves match closely for the faster performances. At higher delays, the curve corresponding to the degree-7 formula begins to fall off. We suspect that the reason for this behaviour is the sheer amount of variation in the Gaussian integrand over ellipsoids for the slower performances. A degree-7 formula proves incapable of modelling the entire variation for the slower performances. This suggests that we must try to break the feasible region into sub-regions such that we are assured that in each sub-region there is much less variation than over the entire region. Since the integrands we will be interested in have a radial symmetry (more or less Gaussian), it is natural to break the unit sphere into a series of concentric shells, apply a degree-7 formula in each concentric shell, and add up all the results to get the total yield integral. The curve resulting from this approach is marked "degree-7-shell" in Figure 6.5. This curve closely matches the Monte-Carlo curve for all performances.

Although we could successfully use a numerical integration formula to integrate a Gaussian integrand over an ellipsoid, the approach would not work at moderate to high dimensions owing to the sheer number of points needed. The number of points needed grows exponentially with the number of dimensions. Further there is no error theory as there is for Monte-Carlo integration. Nevertheless we study an approach in Chapter 7 that combines numerical integration formulae together with Monte-Carlo concepts to exploit the best features of both methods.

In Figure 6.5, we note that there is a significant gap between the Monte-Carlo

curve representing the integral over the ellipsoid, and the Monte-Carlo curve representing the entire polyhedron, marked "Gauss" in the figure. This is not surprising since the ellipsoid covers only a small part of the feasible region defined by the yield polytope. In the next section, we explore a technique to reduce this gap by trying to cover the feasible region with two ellipsoids.

6.8 Covering the feasible region by multiple ellipsoids

The ellipsoidal approximation to the interior of the polytope fails to account for the corners of the polytope. If the corners are all far away from the centre of the joint probability density function of the global parameters then we can safely assume that the probability mass contribution of the corners to the yield is not significant. In such a situation, the ellipsoidal approximation captures the bulk of the probability mass and works well. But as we move from very low yields (with the feasible region far away from the centre of the JPDF) to moderate yields (say about 0.5), the centre of the JPDF could fall just outside the feasible region, in one of the corners not covered by the ellipsoid. This situation is illustrated in Figure 6.6. In this situation, the corner concerned contributes a substantial amount to the yield, and must be covered. We shall develop a means to handle the situation described in Figure 6.6. This figure shows the centre of the JPDF to be within the feasible region, but outside the ellipsoid. The analysis given below strictly corresponds to the case when the centre of the JPDF lies outside the ellipsoid. We do not pursue the case when the centre of the JPDF enters the ellipsoid, although the analysis given below can be extended to this case as well.

Our strategy to handle this situation involves the use of two ellipsoids instead of a single ellipsoid to cover the polytope. The second ellipsoid is designed to account for the corner of Figure 6.6. In order to find this second ellipsoid, we first calculate the first ellipsoid. Then we find the point on the surface of the first ellipsoid that is closest to the centre (or mean) of the JPDF. Having found the point we draw a tangent at the point. This tangent divides the yield polyhedron into two regions - the volume of the yield polytope inside the original yield polytope but below the tangent (this polytope is already accounted for by the first ellipsoid) and the volume of the polytope above the tangent. Let the equation of the tangent be given by $a^T \Delta z + b = c$. Then

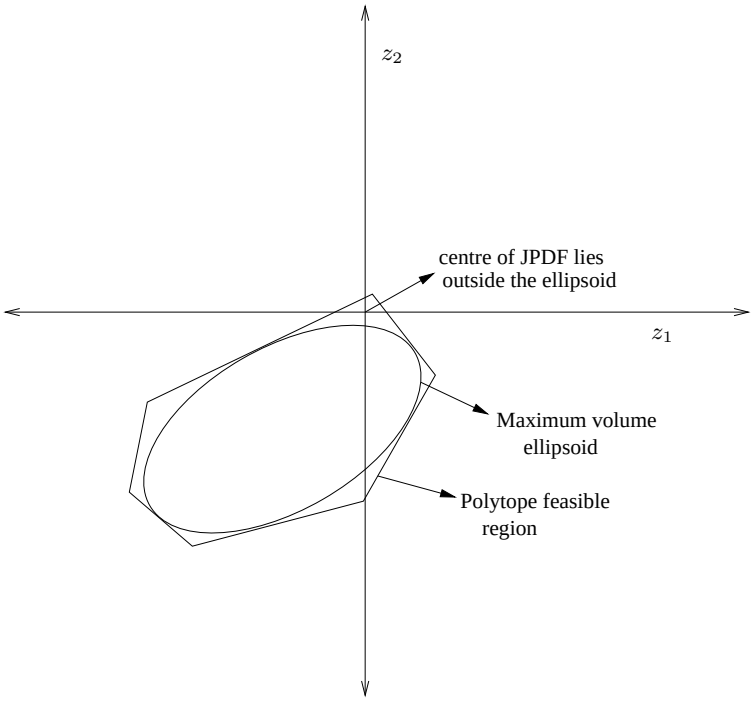


Figure 6.6: Nominal point lies in a corner of the feasible region and is not accounted for by the maximum volume inscribed ellipsoid leading to significant loss of yield.

the second region can be seen to be the solution of the following system of equations:

$$\begin{aligned} Rz &\leq [t_1\ t_2\ \cdots\ t_P]^T, \\ a^T z + b &\geq c. \end{aligned} \tag{6.36}$$

We shall now develop a technique to find the tangent in question. It turns out to be easier to find a tangent to an axis-parallel ellipsoid than to a general ellipsoid. Therefore we transform the first ellipsoid into an axis-parallel ellipsoid. Let B be the ellipsoid transformation matrix of the first ellipsoid, so that any point on it can be expressed as follows:

$$z = By + d; \|y\| = 1. \tag{6.37}$$

The ellipsoid matrix, as we have noted before is given by $(BB^T)^{-1}$. An ellipsoid of fixed orientation in space has a fixed $(BB^T)^{-1}$, but possibly many different matrices B that can serve as the ellipsoid transformation matrix. Let the matrix S be the eigenvector matrix of $(BB^T)^{-1}$, and Λ be the diagonal eigenvalue matrix such that

$$(B^{-1})^T(B^{-1}) = S\Lambda S^{-1}. \tag{6.38}$$

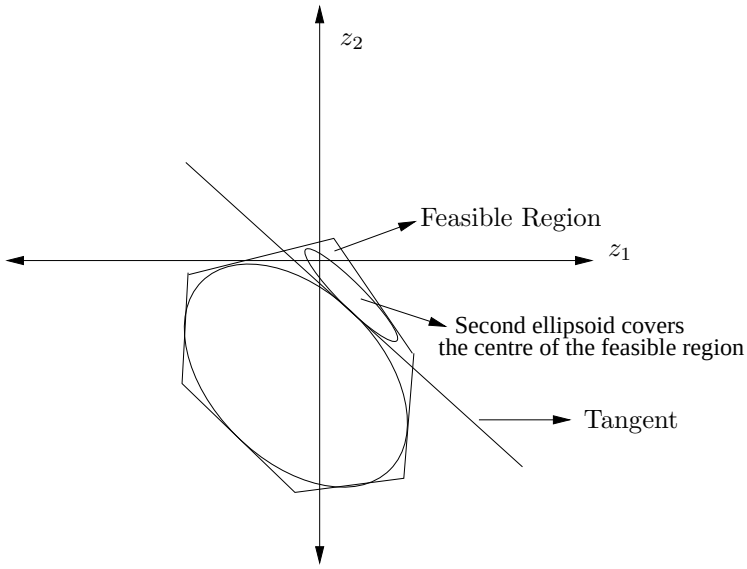


Figure 6.7: Two ellipsoids cover the feasible region.

In order to rotate the coordinate system so that the ellipsoid becomes axis-parallel, we need to perform the following transformation:

$$z = Sw + d. \tag{6.39}$$

Then from (6.37), we see that

$$\begin{aligned} S^T z &= S^T By + S^T d, \\ w &= S^T By. \end{aligned} \tag{6.40}$$

If $S^T B$ is the ellipsoid transformation matrix, then $((S^T B)^{-1})^T (S^T B)^{-1} = S^{-1}(B^{-1})^T B^{-1} S$ is the ellipsoid matrix. Using (6.38) we can see that the

ellipsoid matrix is equal to Λ. Thus the ellipsoid matrix becomes diagonal, and this corresponds to an axis-parallel ellipsoid in w-space centred at the origin.

In order to derive the tangent equation, we shall work in two-dimensions. The extension to higher dimensions shall follow from the argument given below. The equation of the axis-parallel ellipsoid in two dimensions can be written in analytic form as

$$\left(\frac{w_1}{a}\right)^2 + \left(\frac{w_2}{b}\right)^2 = 1. \tag{6.41}$$

Here w_1 and w_2 are the two variable coordinates. The tangent at a point (u_1, u_2) can be expressed analytically as follows:

$$\frac{2(u_1)}{a^2}(w_1 - u_1) + \frac{2(u_2)}{b^2}(w_2 - u_2) = 0. \tag{6.42}$$

The point (u_1, u_2) on the surface of the ellipsoid must be such that it is closest of all points on the ellipsoid to the point in the transformed space corresponding to the centre of the JPDF, which we shall refer to as (p_1, p_2). The centre of the JPDF is outside the ellipsoid in the original space, and it is outside the axis-parallel ellipsoid in the transformed space. Clearly for the point on the ellipsoid closest to (p_1, p_2), the gradient at the point (or the normal at the point) must contain the point (p_1, p_2). This is the criterion we shall use to identify the point in question. The gradient at (u_1, u_2) can be expressed vectorially as $[\frac{2(u_1)}{a^2} \ \frac{2(u_2)}{b^2}]^T$. Thus we can set up the following system of equations:

$$\begin{aligned} u_1 + t\left[\frac{u_1}{a^2}\right] &= p_1, \\ u_2 + t\left[\frac{u_2}{b^2}\right] &= p_2, \\ \left(\frac{u_1}{a}\right)^2 + \left(\frac{u_2}{b}\right)^2 &= 1. \end{aligned} \tag{6.43}$$

The first two equations in the above system express the condition that the centre of the JPDF must lie on the gradient at (u_1, u_2). The last equation simply expresses the fact that (u_1, u_2) must lie on the surface of the ellipsoid.

The above are 3 equations in 3 variables and hence can be solved. To solve the equations we express both u_1 and u_2 in terms of t and then use the third equation to get a single equation in the variable t. After some algebraic manipulation we can show:

$$\begin{aligned} u_1 &= \frac{a^2 p_1}{a^2 + t}, \\ u_2 &= \frac{b^2 p_2}{b^2 + t}. \end{aligned} \tag{6.44}$$

Substituting for u_1 and u_2 in the last equation of (6.43) we have

$$a^2(p_1)^2(b^2 + t)^2 + b^2(p_2)^2(a^2 + t)^2 - (a^2 + t)^2(b^2 + t)^2 = 0. \tag{6.45}$$

The above equation represents a polynomial of degree 4 in t which we must solve in order to find the point (u_1, u_2) on the ellipsoid which is the closest point to the centre of the JPDF. The polynomial above can have only one positive root. The easiest way to see this is to recast it in the following way:

$$\frac{a^2 p_1^2}{(a^2 + t)^2} + \frac{b^2 p_2^2}{(b^2 + t)^2} = 1. \tag{6.46}$$

Let t_0 be the smallest positive value that solves the above equation. The left-hand side of the above equation monotonically decreases with increasing t, when t is positive. Thus for all $t > t_0$ the left hand side of the above equation is less than 1, making t_0 the only positive root of the above equation. A simple geometric argument establishes that it is this positive root that determines the point on the ellipsoid closest to the centre of the JPDF. Consider a point on the ellipsoid corresponding to a negative root. Geometrically this means that starting from this point on the ellipsoid, one must move in a direction negative to that of the gradient to reach the centre of the JPDF (point A). But this means that the line joining the point on the ellipsoid with A passes through the ellipsoid, and there is another point in the ellipsoid that is closer to A than the point on the surface of the ellipsoid. Thus the point on the ellipsoid closest to the given point cannot correspond to a negative root of the polynomial above.

To find the roots of the polynomial, we resort to Newton's root finding procedure:

$$t_{n+1} = t_n - \frac{f(t_n)}{f'(t_n)}. \tag{6.47}$$

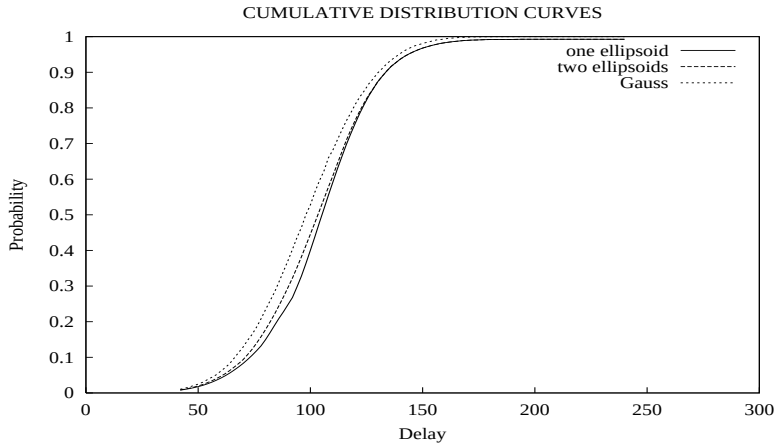


Figure 6.8: Two ellipsoids covering the feasible region. The curves have been computed for 3 global parameters. The joint probability density function was so chosen as to maximise the yield in a certain corner of the polyhedron. It is this corner that is captured by the second ellipsoid.

The starting value of t can be selected keeping in mind that it must be greater than the positive root value. Considering (6.46 we see that

$$\frac{a^2 p_1^2}{t^2} + \frac{b^2 p_2^2}{t^2} > 1, \tag{6.48}$$

leading to

$$\begin{aligned} t &< \sqrt{a^2 p_1^2 + b^2 p_2^2}, \\ t &< max(a,b) * \sqrt{p_1^2 + p_2^2}. \end{aligned} \tag{6.49}$$

The form of (6.45) suggests that in n-dimensions, we shall need to solve a degree-$2n$ polynomial. Thus the problem clearly gets more complicated in higher dimensions. Also the premise behind finding the tangent - namely, that there is a particular corner not covered by the first ellipsoid where the probability mass is high - becomes weaker in higher dimensions. This is because there is a profusion of corners in higher dimensions. As the centre of the JPDF approaches the yield body, there are bound to be several corners where the yield contribution is high. An ellipsoid will be needed to cover

each of these corners which is clearly impractical. The tangent finding procedure, however, will be shown in a subsequent section to have a significant application.

Figure 6.7 shows how two ellipsoids can be used to account for the poor approximation of a single ellipsoid. In Figure 6.8, the yield distribution curve is shown for the case of two ellipsoids. Note that for the new curve, the greatest improvement over the old curve occurs in a region of medium probability.

6.9 Revisiting performance-at-a-time Monte-Carlo

In Chapter 3, we made a remark that for a Monte-Carlo method to calculate a yield integral accurately we must find a bounding box for the feasible region that fits the region as closely as possible. It was also remarked that since for an arbitrary performance, the location of the feasible region in process parameter space is unknown, it is hard for us to construct the close-fitting bounding box needed for accurate sampling.

Fortunately there is a way out of our predicament. The ellipsoid provides us with a means to access the interior of the feasible region. The ellipsoid also closely resembles the shape of the polytope. All of this suggests that we can employ the ellipsoid to construct a bounding box around the feasible region that is better for low yields than the 4σ bounding box. In this section we formalise the arguments.

Let R be the region of integration in n-dimensional space. The integral we are interested in calculating is of the form

$$I = \int \int \ldots \int_R f(z_1, z_2, \ldots z_n) dz_1 dz_2 \ldots dz_n. \tag{6.50}$$

To evaluate the integral we first enclose the region R by a box Ω. Then we sample N points uniformly and randomly within the box Ω and compute the expression

$$Y = \frac{\sum_{i=1}^{i=N} f(Z_i)}{N} * Vol(\Omega). \tag{6.51}$$

The Z_is are independent identical random variables, sampled uniformly from the box Ω with probability $p(Z_i) = 1/Vol(\Omega)$. The Expectation of the random variable Y, $E(Y)$ provides us with an estimate of the integral since

$$\begin{aligned} E(Y) &= \frac{Vol(\Omega)}{N}(NE(f(Z_1))) \\ &= Vol(\Omega)\int\int\ldots\int_R f(z_1, z_2, \ldots z_n)p(z_1, z_2, \ldots z_n)dz_1dz_2\ldots dz_n \\ &= Vol(\Omega)\frac{1}{Vol(\Omega)}\int\int\ldots\int_R f(z_1, z_2, \ldots z_n)dz_1dz_2\ldots dz_n \\ &= \int\int\ldots\int_R f(z_1, z_2, \ldots z_n)dz_1dz_2\ldots dz_n. \end{aligned} \tag{6.52}$$

The accuracy of the estimate can be gauged from the variance of the random variable Y. The variance can be calculated as follows:

$$Var(Y) = \left(\frac{(Vol(\Omega))^2}{N}\right) * Var(f(Z_1)). \tag{6.53}$$

But $Var(Z_1)$ can be written as follows:

$$Var(f(Z_1)) = E(f^2(Z_1)) - (E(f(Z_1)))^2. \tag{6.54}$$

Noting that z is a vector and $dz = dz_1dz_2\ldots dz_n$, we have

$$E(f^2(Z_1)) = \left(\frac{1}{Vol(\Omega)}\right)\int\int\ldots\int_R f^2(z)dz \tag{6.55}$$

and

$$E^2(f(Z_1)) = \left(\frac{1}{Vol(\Omega)}\right)^2\left(\int\int\ldots\int_R f(z)dz\right)^2. \tag{6.56}$$

Putting (6.56) and (6.55) in (6.54) and using (6.53), we finally obtain

$$Var(Y) = \frac{Vol(\Omega)}{N}\int\int\ldots\int_R f^2(z)dz - \left(\frac{1}{N}\right)\left(\int\int\ldots\int_R f(z)dz\right)^2. \tag{6.57}$$

The interesting thing about (6.57) is that the variance of our estimator depends upon $Vol(\Omega)$. Thus a tight bounding box is needed to reduce the variance.

The bounding box we construct is suggested by the maximum volume ellipsoid that can be inscribed into the feasible region. Let $s_1, s_2 \ldots s_n$ denote the eigenvectors of the ellipsoid transformation matrix. They also denote the axes of the ellipsoid. The half-lengths of the axes of the ellipsoid are given by the eigenvalues of the ellipsoid transformation matrix: $\lambda_1, \lambda_2, \ldots \lambda_n$. Assuming that the maximum volume ellipsoid approximates the shape of the polytope quite well, it seems natural to construct a bounding box for the polytope such that its sides are parallel to the ellipsoid axes. In each ellipsoid axis, we need to determine how to obtain the supporting hyperplane for the polytope in the direction of the ellipsoid axis. This can be easily done via linear programming. Let us formulate the problem

$$\begin{aligned} &\text{max: } s_i^T z \text{ such that} \\ &Rz \leq [t_1\ t_2\ \cdots\ t_P]^T \end{aligned} \tag{6.58}$$

Note that the 4σ constraints are included in the above system. If d denotes the centre of the ellipsoid and z_{opt} denotes the optimal value of z produced by the linear programme then $s_i^T(z_{opt} - d)$ denotes how far one must go in the direction of s_i in order to bound the polytope.

This bounding box is shown in Figure 6.9. In practice, though we find that the gain to be had in ensuring that the ellipsoid bounding box covers the whole feasible region is minimal; in fact it could even be counterproductive for the reasons dealt with in Chapter 5 in the context of the construction of an axis-parallel bounding box. Therefore we construct an ellipsoid bounding box whose sides are parallel to the ellipsoid axes and with the same lengths as the ellipsoid axes. We lose the property that the constructed box contains the entire feasible region but we usually obtain a significant reduction in volume as compared to the 4σ-bounding box. This saves us the cost of solving $2 * n$ linear programmes to determine the actual boundaries.

Figure 6.10 shows two curves, one resulting from using the ellipsoid bounding box for sampling, and the other resulting from using the 4σ bounding box. The ellipsoid curve lies beneath the polyhedral curve, which is not surprising considering that the ellipsoid bounding box is not really a bounding box at all, and the value it computes is in the best case (zero-variance), a lower bound on the true yield. The real advantage of using the ellipsoid bounding

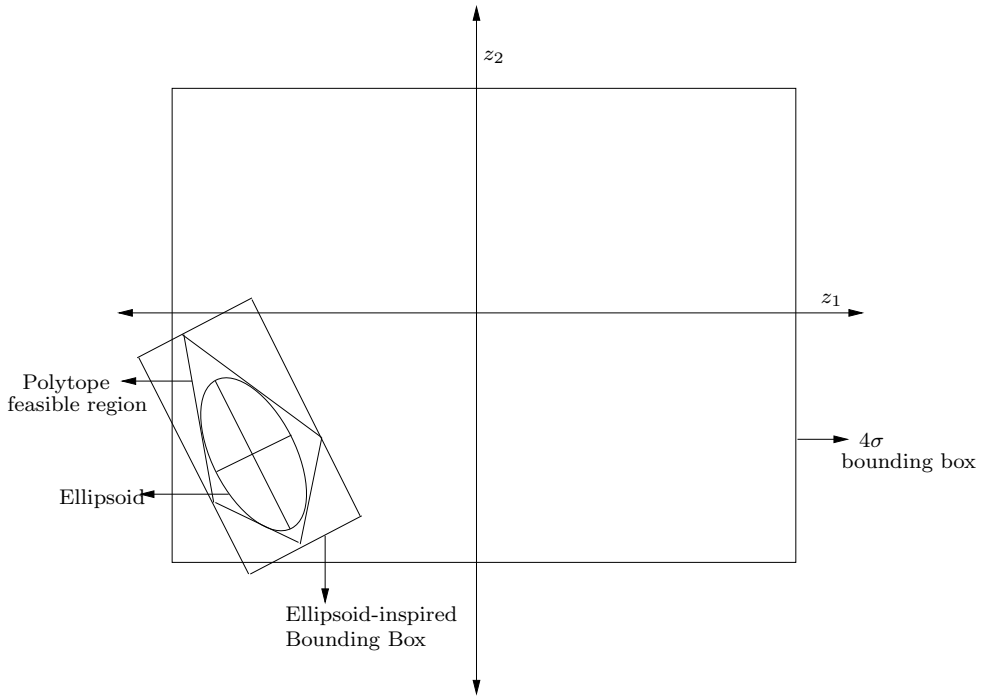


Figure 6.9: Smaller bounding box obtained from the ellipsoid.

box is brought out in Figure 6.11, which is a plot of the variance of the two methods as a function of delay.

The curves of Figure 6.11 were obtained by running each method 200 times per performance point. 10,000 samples were generated each time to obtain a yield estimate. The 200 estimates of yield at each performance point were then used to calculate the mean value of the yield predicted by each method and to calculate the sample variance at each performance point (note that (6.57) suggests that to compute real variance, we must know the result!). The mean values for each yield are plotted in Figure 6.10 which explains the relative smoothness of these curves, and the sample variance values for each method are plotted in Figure 6.11. The ratio of the variances of the two methods is an interesting measure for it gives us an idea of the relative confidence that we can have in the results produced by the two methods. If the ratio of the variances is say 10, then we will need to run the higher

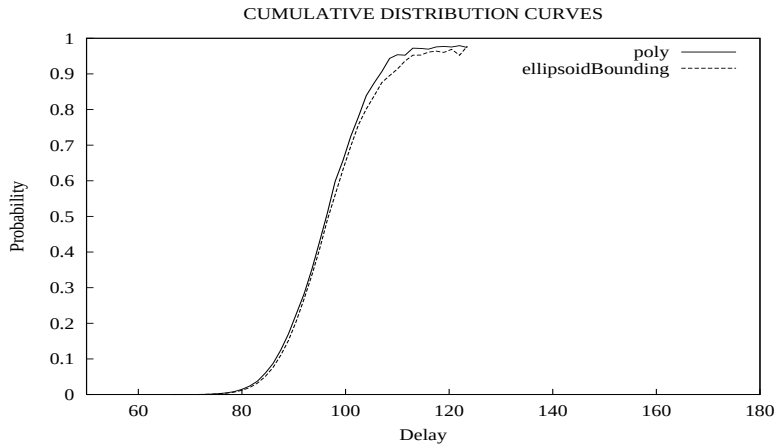


Figure 6.10: Cumulative distribution curves, one obtained by sampling over the 4σ box and the other by sampling over a smaller ellipsoid bounding box. The number of global parameters assumed was 8 and the number of hyperplanes in the feasible polyhedron was 46.

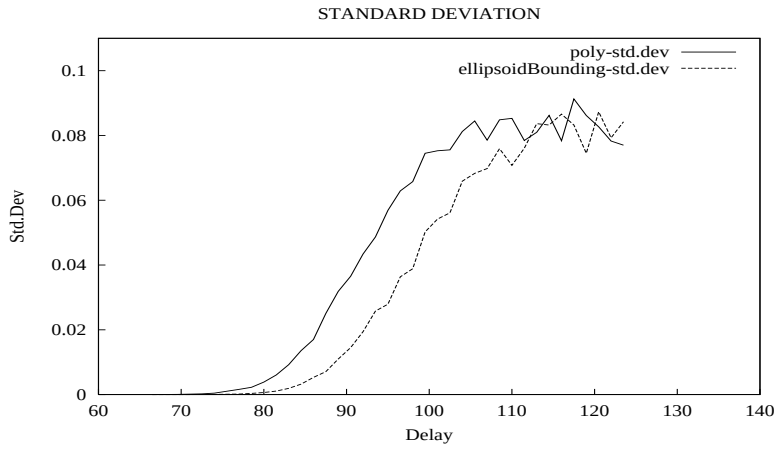


Figure 6.11: The standard deviation (*sigma*) plots of the two Monte-Carlo estimation methods of the previous figure. Sampling over the ellipsoid bounding box results in orders of magnitude less standard deviation for faster performances than sampling over the 4σ box.

variance method for 10 times the number of trials used for the lower-variance method in order to have the same confidence in the result. It can be seen from Figure 6.11 that the ratio of variances is great at very high performances (very fast circuits), and falls for less demanding circuit performances. This is not surprising since the use of a small bounding box is likely to bear the highest dividends at fast performances, when the yield feasible region is relatively small relative to the 4σ box (as depicted in Figure 6.9). As the yield feasible region gets larger and larger, the ellipsoid bounding box begins to approach the size of the original 4σ box, and the gain obtained by using this special bounding box is likely to get smaller. For the specific example circuit we used, and whose results are shown in Figure 6.11, we observed that the variance ratios declined from 100 down to 1.

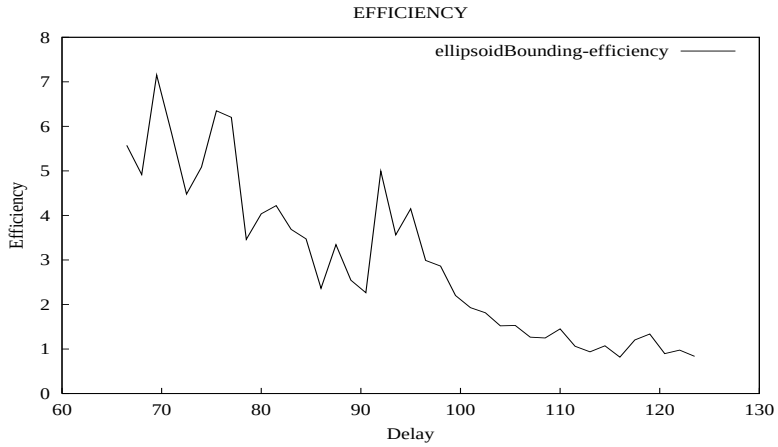


Figure 6.12: This figure shows the efficiency of using an ellipsoid-bounding box to perform sampling.The y-axis is a ratio of efficiencies.

Analysis of equation (6.57) can provide us with a clue as to what to expect from a smaller bounding box. Let us denote the actual yield by the integral $I(f)$ and the volume of the 4σ bounding box as V. Then we can rewrite (6.57) as follows:

$$Var(Y^{poly\_box}) = \frac{1}{N}(VI(f^2) - (I(f))^2). \tag{6.59}$$

Let the ellipsoid bounding box be a factor k smaller than the 4σ bounding box. The integral it computes is $I^*(f)$, which is a lower bound on the true

integral (note that if we were to actually use linear programming to compute the ellipsoid-box boundaries, the yield integral computed would be $I(f)$). We can write

$$Var(Y^{ellipsoid\_box}) = \frac{1}{N}(\frac{V}{k}I^*(f^2) - (I^*(f))^2). \tag{6.60}$$

Then the ratio of the variances can be written as

$$\begin{aligned} \frac{Var(Y^{poly\_box})}{Var(Y^{ellipsoid\_box})} &= \frac{\frac{1}{N}(VI(f^2) - (I(f))^2)}{\frac{1}{N}(\frac{V}{k}I^*(f^2) - (I^*(f))^2)} \\ &= \frac{V - \frac{(I(f))^2}{I(f^2)}}{\frac{V}{k} - \frac{(I^*(f))^2}{I^*(f^2)}} * \frac{I(f^2)}{I^*(f^2)} \\ &> k. \end{aligned} \tag{6.61}$$

The second step in the above system of equations is made possible if we can assume that $I(f) > 0$ and $I^*(f) > 0$. Since$I(f^2) > I^*(f^2)$,the third step is possible if we can assume that

$$\frac{(I(f))^2}{I(f^2)} \approx \frac{(I^*(f))^2}{I^*(f^2)}. \tag{6.62}$$

Both of these assumptions turn out to be rather mild in practice, so that as a benchmark we can assume that the gain in variance is at least a factor of k if the ellipsoid bounding box is k times smaller than the 4σ bounding box. Experimental results show that this factor k is a rather loose lower bound - the gain in variance is larger than k.

At this point, it is pertinent to situate this method of variance reduction in the context of other schemes proposed in the literature [42]. [42] discusses importance sampling, the method of control variates and stratified sampling as possible means to reduce variance. The general observation made in this paper is that in order to obtain variance reduction one has to have a good approximation of the acceptability region. In our case, this is achieved by focusing on the ellipsoid and using the fact that the ellipsoid constructed within the feasible region provides a good approximation of the feasible region. We perform a kind of importance sampling by drawing our samples

mostly from within the feasible region. In general importance sampling, sampling is performed using a distribution that peaks in the feasible region. In our case this is done by choosing the distribution

$$p(z_1, z_2, \ldots, z_n) = \frac{1}{Vol(\Omega)} \text{ for } z \in \Omega,\ 0 \text{ otherwise.} \tag{6.63}$$

where Ω is the ellipsoid-inspired bounding box, and R is the feasible region. Importance sampling is regarded by [42] to be superior to stratified sampling where we choose to divide the parameter space into non-overlapping regions, compute the yield integral components for each of these regions separately and take a weighted sum of all yield contributions according to the probability mass contained in each stratum. This conclusion is backed up by intuition as well: stratified sampling does not use information provided by the location of the feasible region unlike importance sampling; it is a general technique to reduce variance that is independent of the location, size and shape of the feasible region. Our method of importance sampling makes use of the size, shape and location of the feasible region.

Following [42], [38] let us define the efficiency

$$\epsilon = \frac{\sigma_1^2 \tau_1}{\sigma_2^2 \tau_2}. \tag{6.64}$$

Here τ_1 and τ_2 represent the computation times for N trials of uniform sampling over the whole 4σ box and uniform sampling over the smaller ellipsoid bounding box, σ_1 and σ_2 represent the variances of these two sampling procedures respectively. The logic behind the above equation is simple: if a low variance procedure has a high average time per trial value, then it may become equivalent to a high variance procedure having a low average time per trial. In other words, one can afford to run a variance procedure for the extra trials needed to make up for the gap in accuracy without incurring any increase in computation time because of the low time per trial value. The efficiency of sampling over the ellipsoid bounding box is shown in Figure 6.12. The efficiency of uniformly sampling over the ellipsoid bounding box, as compared to uniformly sampling over the whole 4σ box is largest at low values of yield, when the ellipsoid bounding box is much smaller than the whole 4σ box.

6.10 Non-uniform sampling of the space of parameters

We would now like to investigate a non-uniform sampling based Monte-Carlo integration method. Let us assume that the joint probability density function (JPDF) of global parameters is a multivariate normal density function with no correlation among its parameters. Our interest is, as before in evaluating the following yield integral:

$$I(f) = \int\int \dots \int_R f(z_1, z_2, \dots z_n) dz_1 dz_2 \dots dz_n. \tag{6.65}$$

Instead of constructing a bounding box around the feasible region and sampling *uniformly* within it, we choose instead to sample the 4σ bounding box according to the probability density function $p(z_1, z_2, \dots z_n)$. We can rewrite the above equation as follows:

$$I(f) = \int\int \dots \int_R \frac{f(z_1, z_2 \dots z_n)}{p(z_1, z_2 \dots z_n)} p(z_1, z_2 \dots z_n) dz_1 dz_2 \dots dz_n. \tag{6.66}$$

Then we can easily see that the above integral can be easily estimated by sampling the probability density function $p(z_1, z_2 \dots z_n)$ and constructing the following estimate

$$I^*(f) = \sum_{i=1}^{i=N} \frac{f(Z_i)}{p(Z_i)}, \tag{6.67}$$

where each Z_i is sampled from $p(z_1, z_2, \dots z_n)$. The variance of the random variable $\frac{f(Z)}{p(Z)}$ can be easily seen to be

$$Var(I^*(f)) = \frac{1}{N}\Big(E\Big(\frac{f^2(Z)}{p^2(Z)}\Big) - E^2\Big(\frac{f(Z)}{p(Z)}\Big)\Big). \tag{6.68}$$

As in the previous section, we note that z is a vector and let $dz = dz_1 dz_2 \dots dz_n$, to obtain

$$\begin{aligned} Var(I^*(f)) = \frac{1}{N}\Big(\int\int \dots \int_R \frac{f^2(z)}{p(z)} dz \\ - (\int\int \dots \int_R f(z) dz)^2 \Big). \end{aligned} \tag{6.69}$$

If the distribution used for importance sampling $p(z)$ is the same as $f(z)$, namely we can sample from the given joint probability density function of the parameters of variation, then the above equation reduces to

$$\begin{aligned} Var(I^*(f)) &= \frac{1}{N}\Big(\int\int\ldots\int_R f(z)dz - (\int\int\ldots\int_R f(z)dz)^2\Big) \\ &= \frac{u-u^2}{N} \\ &= \frac{u(1-u)}{N} \end{aligned} \tag{6.70}$$

where $u = I(f)$. This is the same as the original Monte-Carlo variance estimator that we derived in Chapter 2. Comparison with the variance estimator for uniform sampling shows that the difference in variance between the two estimates can be expressed as

$$D = \int\int\ldots\int_R \frac{f^2(z)}{\frac{1}{V}}dz - \int\int\ldots\int_R \frac{f^2(z)}{f(z)}dz. \tag{6.71}$$

The uniform sampling method over the ellipsoid bounding box has lower variance than the method of sampling from the JPDF if in the whole feasible region, the following holds true:

$$\frac{1}{V} > max_R(f(z)). \tag{6.72}$$

In general this inequality holds only in feasible regions far away from the centre of the JPDF. Our experimental results show that if we assume a truncated multivariate normal distribution for the JPDF of the global parameters, and sample from it, then we will outperform any uniform sampling method significantly for all but the smallest yields. The underlying message is that if we can sample from the JPDF, then we must perform Monte-Carlo integration by sampling from the JPDF, instead of sampling uniformly within a bounding box surrounding the feasible region (no matter how tight this bounding box might be). Uniform sampling must be restricted to situations where it is difficult or impossible to sample from the JPDF of global parameters. In the next section, we shall describe an importance sampling based technique that can reduce variance in case of non-uniform sampling.

6.11 Variance reduction for non-uniform importance sampling

As in the previous section, let us assume that the probability density function is a multi-variate normal density, and that the sampling technique is to sample from the JPDF and count the number of times the sample falls in the feasible region. We shall show that it is possible to decrease the variance by constructing an importance sampling density that makes use of the geometry of the feasible region. Let us recall the variance expression for importance sampling, equation (6.69):

$$Var(I^*(f)) = \frac{1}{N}\Big(\int\int \dots \int_R \frac{f^2(z)}{p(z)} dz - (\int\int \dots \int_R f(z)dz)^2 \Big). \tag{6.73}$$

The aim in importance sampling is to construct a sampling density $p(z)$ that mimics as closely as possible the behaviour of $f(z)$ in the sampling region R. The choice of $p(z) = f(z)$ proved to be a good one, as was shown in the last section. However we can do better by choosing p(z) to be a multivariate normal density of the same form as f(z), but centring it at a point close to the feasible region. We must choose this point such that $p(z) > f(z), z \in P$ since this is one way of making the first term in equation (6.69) be less than the yield value p. The technique we use is based on the idea displayed in Figure 6.13(a). Here we consider the distances of two points A and B from all points in a polytope P shown in the figure. A separating hyperplane h separates the point A from the polyhedron P, and B is obtained by dropping a perpendicular from A onto h, and taking the point of intersection as B. It can easily be seen that for any point $x \in P$, the distance d_A from point A to the point x is larger than the distance d_B from the point B to x:

$$\begin{aligned} d_A^2 &= d_B^2 + r^2 - 2 * d_B * r * \cos\theta \\ &\geq d_B^2, \text{ since } \frac{\pi}{2} \leq \theta \leq \pi. \end{aligned} \tag{6.74}$$

If we assume that A represents the centre point of the original JPDF, then we can use B to locate the importance sampler density $p(X)$. This will ensure that every point in the polyhedron is closer to the centre of the importance

sampler, which means that $p(z) > f(z), z \in P$. The only catch is that it is non-trivial to find the separating hyperplane h. Therefore we turn to a heuristic. The heuristic is to find the point on the maximum volume ellipsoid $\subset P$ that is closest to the point A which is the centre of the JPDF. We take B to be this point. The rationale behind this heuristic is supplied by the observation in the previous paragraph: B will be closer than A to all points within the ellipsoid as well as all points within the polyhedron P, which lie below the tangent drawn at B. This situation is shown in Figure 6.13(b).

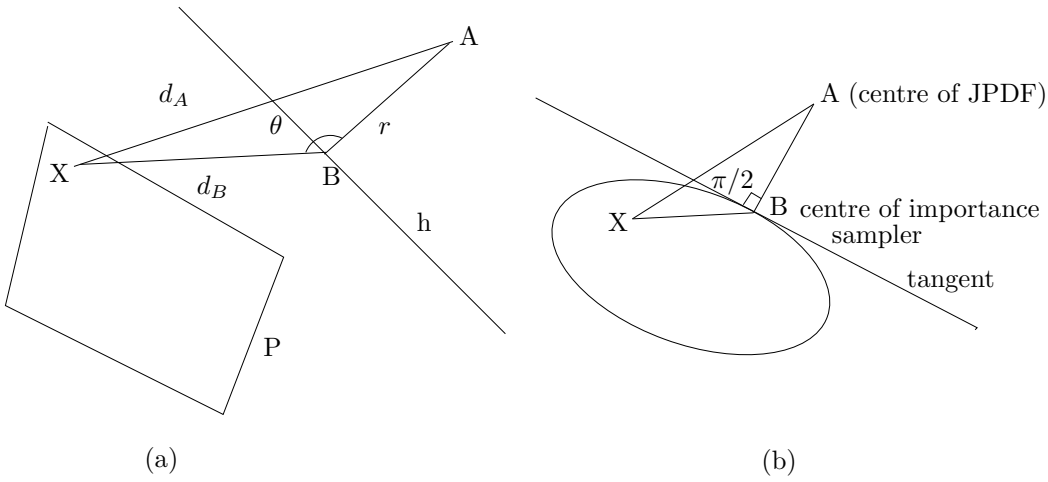


Figure 6.13: Illustration of the procedure to find the centre of an importance sampler.

The above procedure is only relevant when the nominal point of the design lies outside the feasible region. Note that both the tangent importance sampler and the JPDF sampler are estimators of the same integral, unlike in the case of the ellipsoid bounding box obtained without linear programming. When the nominal point enters the feasible region, it is no longer advantageous to place the centre of the importance sampling Gaussian density away from the nominal point.

Therefore for delay values greater than or equal to the nominal delay value, we revert to sampling from the given JPDF of global parameters. Thus in Figure 6.15 we see that a gain in variance is visible using the tangent importance sampler for all delay values less than the nominal delay; for delay

values greater than the nominal delays there is no gain since the two types of sampling become identical. Figure 6.15 suggests that there is a fair amount of gain to be had by using the tangent based importance sampler, at low values of delay. In fact, for the very low performances, we see that the ratios of variances $\frac{\sigma_2}{\sigma_1}$ is of the order of 10. This means that for these performances N trials of the tangent importance sampler would give us the same accuracy as $100N$ trials of the naive Gaussian sampling procedure. However, these gains are tempered somewhat by the fact that it takes longer to run a single trial of the tangent importance sampler procedure, since it involves computing the maximum volume ellipsoid contained in the feasible region, and thereafter finding a tangent to it to obtain the centre of the new importance sampler. As in the previous section, let us define the efficiency

$$\epsilon = \frac{\sigma_1^2 \tau_1}{\sigma_2^2 \tau_2}. \tag{6.75}$$

In this case τ_1 and τ_2 represent the computation times for N trials of the naive Gaussian sampling procedure and the tangent importance sampler procedure, σ_1 and σ_2 represent the variances of these two sampling procedures respectively.

Let us further analyse the times τ_1 and τ_2. τ_2 can be expressed as follows:

$$\tau_2 = \tau_{ellipsoid} + N * \tau_2^{trial}. \tag{6.76}$$

$\tau_{ellipsoid}$ is the fixed overhead of finding the maximum volume ellipsoid and calculating the importance sampling centre and is incurred before starting the sampling process. τ_2^{trial} is the average time/trial required for generating sample points from the importance sampler density function and checking to see if the points fall inside the feasible region. τ_1^{trial} is the average time/trial needed for generating sample points from a normal distribution and checking to see if the sample points fall within the feasible region or not. We can express τ_1 as simply

$$\tau_1 = N * \tau_1^{trial} \tag{6.77}$$

since all we need to do in this case is generate points from the JPDF of global parameters and check to see if the generated points fall within the feasible region. It might seem that $\tau_1^{trial} = \tau_2^{trial}$ but this is not true, especially at low yields. At low yields, $\tau_1^{trial} > \tau_2^{trial}$ because far more points fall within the feasible region when the points are sampled according to the importance

sampling density. Determining if a point falls within the feasible region requires checking if the point satisfies *all* the path constraints. On the other hand, when a point falls outside the feasible region relatively few constraints will need to be checked before a violation occurs. This makes the time per trial of the importance sampler greater than the time per trial of the original JPDF sampler. At higher yields though, the numbers of feasible points in the two types of sampling procedures begin to become comparable, and the gap between the times per trial also falls. To test out the efficiency of the importance sampling procedure, we performed an experiment on a circuit with 5112 paths for 8 global parameters. 100 paths were extracted from the 5002 paths according to the angle-based path selection criterion of Chapter 4.

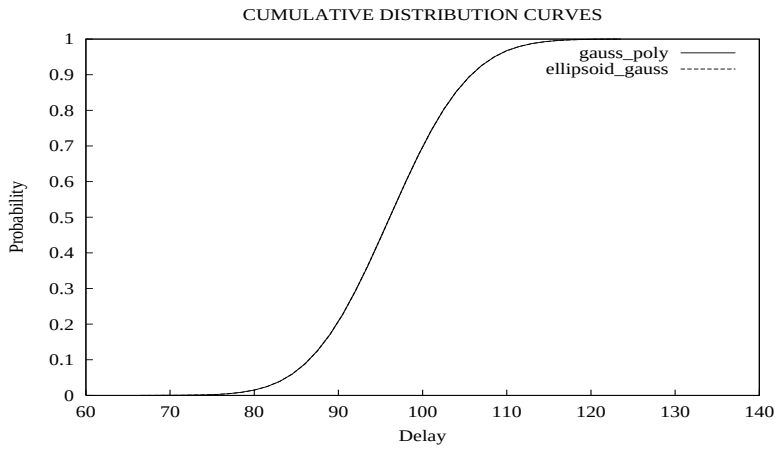


Figure 6.14: Cumulative Distribution Curves for two types of Gaussian sampling procedures. The two curves coincide for almost all values of delay. The number of trials was 10000 and the number of global parameters assumed in the system was 8.

The maximum volume ellipsoid was then inscribed in the feasible region, and the importance sampling density function determined. The cumulative distribution curves, variance curves and the efficiency of the importance sampling method are shown in Figures 6.14, 6.15 and 6.16 respectively. It is interesting to note that for very low yields (≤ 0.01) an efficiency gain of the order of 10 can be obtained. In other words, if a time t is allocated to the importance sampling procedure as well as the JPDF sampling procedure, then the importance sampling procedure can produce a yield estimate that is 10 times

as accurate as the one produced by the JPDF sampling procedure. The efficiency figure however falls rapidly as yield increases and tapers towards 1.0 as the yield values increase until the centre of the JPDF enters the feasible region. The cumulative distribution curves obtained by the tangent importance sampling method as well as ordinary Gaussian sampling methods are shown in Figure 6.14. Not surprisingly the two curves are nearly identical. The variances of the two methods are shown in Figure 6.15. The thing to note here is that the variances of both methods are significantly lower than for uniform sampling discussed previously.

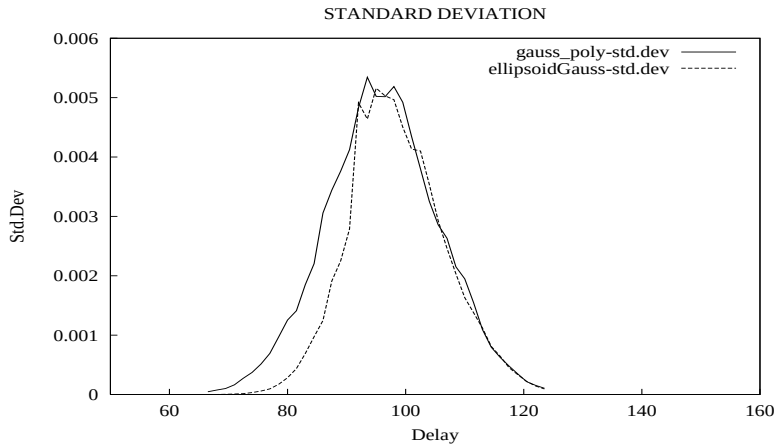


Figure 6.15: A plot of standard deviation vs delay for two Gaussian sampling methods: The "gauss _poly.std.dev" curve is obtained by sampling from the original JPDF, whereas the "ellipsoidGauss.std.dev" curve is obtained by sampling from a Gaussian density centred on the ellipsoid at a point closest to the mean of the original JPDF, as discussed in this section.

6.12 Discussion

At the end of Chapter 5, we provided details of the experimental set-up used to obtain the results of this chapter. It is pertinent to point out here that one of the methods described in this chapter depends crucially on a requirement identified at the end of Chapter 5, namely that there is a non-zero probability of manufactured circuits having a delay smaller than the nominal delay of the circuit. The method in question is the tangent importance

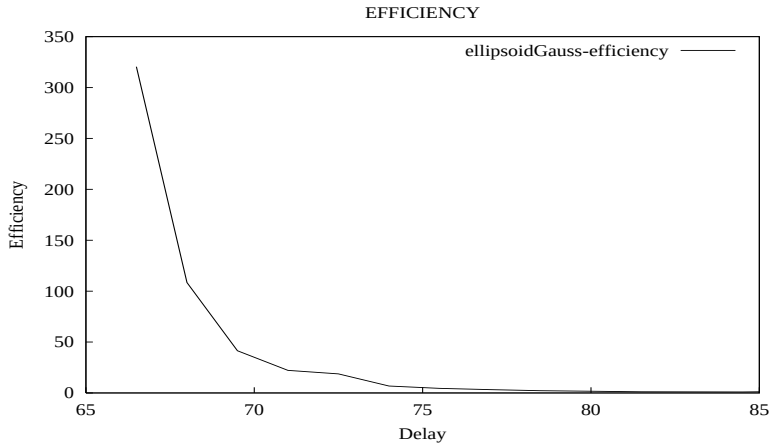


Figure 6.16: Efficiency ratio of the tangent importance sampler as a function of delay. For very small values of delay, an efficiency factor of close to 20 is obtainable i.e., the tangent importance sampler is 20 times more efficient than the naive Gaussian sampling procedure.

sampler method, which is applicable whenever the centre of the JPDF falls outside the feasible region of circuit performance. This happens if and only if the desired circuit performance (delay) is smaller than the nominal delay. When the centre of the JPDF belongs to the feasible region, it means that the required circuit performance is at least equal to the nominal delay of the circuit. The requirement that there be at least some manufactured chips that are faster than the nominal chip is not a serious limitation; indeed, even if a constraint matrix is of the mixed-coefficient kind identified in Chapter 5, it will most likely be the case that some manufactured circuits will be faster than the nominal delay.

On the other hand, the ellipsoid-box based Monte-Carlo sampling method actually benefits from having a mixed-coefficient path sensitivity matrix. A mixed-coefficient path sensitivity matrix prevents the existence of "extreme points" discussed in Chapter 5, which in turn enables the construction of tighter bounding boxes and thus reduces variance of the uniform sampling Monte-Carlo estimator.

Chapter 7

Randomised quadrature

7.1 Introduction

In the last chapter we saw that numerical quadrature is an unsatisfactory solution to computing the yield integral because of the exponential dependence on the dimensionality of the problem. Therefore we are forced to explore other alternatives. Monte-Carlo integration is the obvious recourse since Monte-Carlo methods do not depend on dimensionality and have a fairly simple error theory behind them. However Monte-Carlo methods may need a large sample size to provide real results. Researchers have proposed ways of combining the accuracy of numerical methods with the dimensionality-independence of Monte-Carlo methods. In this chapter we explore some of such techniques known as randomised quadrature.

7.2 Reformulation of yield integral

Let us reconsider the yield integral of interest to us - namely, the integral over the volume of the largest ellipsoid that fits in the feasible region.

$$\begin{aligned} Y &= \int\int \ldots \int_{R^*} f(z_1, z_2 \ldots z_n) dz_n \ldots dz_1 \qquad (7.1) \\ &= \int\int \ldots \int_{R} f(B_1^T y + d_1, B_2^T y + d_2, \ldots B_n^T y + d_n)(\det B) dy_n \ldots dy_1. \end{aligned}$$

Here R^* is the ellipsoid in the space of global parameters z and R is the unit sphere in y-space, such that $z = By + d$. One of the problems with applying a numerical integration procedure to the above integral is that at higher yields corresponding to a large feasible region the integrand shows a lot of variation. A standard low order degree integration formula is unable to capture all of the variation. We showed in the last chapter that breaking the unit sphere into a series of concentric shells and applying integration formulae to each shell helps because there is much less variation in a shell as compared to the whole unit sphere. We can take this idea further and formulate the yield integral as a spherical-radial integral. In other words we can express the yield integral as the product of a one-dimensional radial integral and a $n-1$ dimensional integral on the surface of the unit sphere. This should perform even better than the concentric shells method because the surface of the unit sphere must have even less variation than a concentric shell of arbitrarily small thickness. Below we present the transformation of the yield integral above to a spherical radial integral. The arguments follow the book of A. H. Stroud [88].

Let R_n be the n-dimensional unit sphere, and let Y_n denote its surface. Let us assume that we have an integration formula for an integral over the surface of the unit sphere:

$$\int\int\ldots\int_{Y_n} f(y_1, y_2 \ldots y_n) d\sigma = \sum_{j=1}^{j=N} B_j f(v_{j,1}, v_{j,2} \ldots v_{j,n}). \tag{7.2}$$

Note that in the above equation $d\sigma$ is a differential element on the surface of the unit sphere. We would like the points of this formula to lie on the surface Y_n. For a real number $r > 0$ let

$$rY_n = \{r\nu : \nu \in Y_n\}. \tag{7.3}$$

Let us consider the integration of a monomial term over the surface of a sphere of radius r:

$$\begin{aligned}
&\int\int\ldots\int_{rY_n} y_1^{\alpha_1} y_2^{\alpha_2} \ldots y_n^{\alpha_n} d\sigma \\
&= \int\int\ldots\int_{Y_n} r^{n-1} (ry_1)^{\alpha_1} (ry_2)^{\alpha_2} \ldots (ry_n)^{\alpha_n} d\sigma \\
&= r^{n-1+\alpha} \int\int\ldots\int_{Y_n} y_1^{\alpha_1} y_2^{\alpha_2} \ldots y_n^{\alpha_n} d\sigma.
\end{aligned} \tag{7.4}$$

The integral of a monomial over the entire unit sphere can then be written as

$$\begin{aligned} &\int\int\ldots\int_{R_n} y_1^{\alpha_1} y_2^{\alpha_2}\ldots y_n^{\alpha_n} dy_1 dy_2\ldots dy_n \\ &= \int_0^1 \big(\int\int\ldots\int_{rY_n} y_1^{\alpha_1} y_2^{\alpha_2}\ldots y_n^{\alpha_n} d\sigma\big) dr \\ &= \int_0^1 r^{n-1+\alpha} dr \int\int\ldots\int_{Y_n} y_1^{\alpha_1} y_2^{\alpha_2}\ldots y_n^{\alpha_n} d\sigma \end{aligned} \tag{7.5}$$

where $\alpha = \alpha_1 + \alpha_2 + \ldots \alpha_n$.

Suppose we have the following one-dimensional integration formula of degree d for the radial integral:

$$\int_0^1 r^{n-1} f(r) dr \approx \Sigma_{i=1}^{M} A_i f(r_i). \tag{7.6}$$

Then the points $r_i \nu_j$ and the coefficients $A_i B_j,\ i = 1, 2\ldots M, j = 1, 2\ldots N$ are an integration formula of degree d for R_n. The A_i are coefficients of the radial integration formula, while the B_j are the coefficients of the spherical integration formula. The r_i and ν_j are the points of the respective formulae.

7.3 Computing the spherical surface integral

The reformulation of the yield integral in the last section requires us to calculate a spherical integral over the surface of the unit sphere (in $n-1$ dimensions) and a radial integral in one dimension. We investigate techniques to calculate the spherical surface integral in this section. Let us denote the spherical surface integral by $I(f)$. A degree-3 formula to compute the surface integral is the following [32]:

$$I(f) \approx \left(\frac{V}{2n}\right) \sum_{i=1}^{i=n} \big(f(e_i) + f(-e_i)\big) \tag{7.7}$$

where e_i is the unit vector along the ith coordinate direction and V is the surface area of the unit sphere in n dimensions. Another degree-3 formula

that is slightly more expensive than the previous one is the following [63], [64]:

$$I(f) \approx \big(\frac{V}{2(n+1)}\big) \sum_{i=1}^{i=n+1} \big(f(u_i) + f(-u_i)\big). \tag{7.8}$$

The $n+1$ points u_i are the vertices of a regular n-simplex with the vertices located on the surface of the unit sphere. This rule can be extended to a degree-5 rule in a natural way:

$$\begin{aligned} I(f) \approx V\Bigg(& \frac{(7-n)n}{2(n+1)^2(n+2)} \sum_{i=1}^{i=n+1} \big(f(u_i) + f(-u_i)\big) + \\ & \big(\frac{2(n-1)^2}{n(n+1)^2(n+2)}\big) \sum_{i=1}^{n(n+1)/2} \big(f(v_i) + f(-v_i)\big)\Bigg). \end{aligned} \tag{7.9}$$

The u_i are as in the degree-3 formula while the v_i are obtained by taking the midpoint of the line segment joining every pair of the vertices u_i and projecting the midpoint onto the unit sphere. The total number of points in the formula is $(n+1)(n+2)$.

If we rotate the points of a given integration formula, we obtain points for another integration formula that is valid for the function f. To see this let $h(y) = f(Qy)$ where Q is a rotation matrix. Consider the spherical integral over the unit sphere R

$$I(h) = \int_R h(y)dy. \tag{7.10}$$

This integral can be transformed as follows:

$$\begin{aligned} I(h) &= \int_R h(y)dy \\ &= \int_R f(Qy)dy \\ &= \int_R f(u)\det(Q^{-1})du \\ &= \int_R f(u)du \\ &= I(f). \end{aligned} \tag{7.11}$$

In the above set of equations we have made use of the transformation $u = Qy$. Now applying the integration formula to $h(y)$ we see that

$$\begin{aligned} h(y) &= \sum_{i=1}^{i=N} w_j h(y_j) \\ &= \sum_{i=1}^{i=N} w_j f(Qy_j). \end{aligned} \tag{7.12}$$

Therefore, if the points of the original integration formula are rotated (orthogonal transformation), then the new points constitute another integration formula of the same degree.

7.4 Randomising quadrature rules

The last section showed us how to obtain another integration formula of a given degree given the points of one integration formula. Each integration formula provides us with an estimate of the integral in question. This suggests that if we were to choose "random" orthogonal transformations, and apply the transformation to the points of a given integration formula, then we have a basis for randomising the original integration formula. The question of course is how to select the random orthogonal transformation Q. Let $S(Q, f)$ be an integration rule of degree $2n + 1$ such that:

$$S(Q, f) = \sum_{i=1}^{i=N} w_i f(y_i). \tag{7.13}$$

Before we present the algorithm for generating a random orthogonal matrix, we must discuss the notion of a QR factorisation. Every matrix X can be expressed as a product of an orthogonal matrix Q and an upper triangular matrix R.

We state the following theorem along the lines of Theorem 1 in [32].

Theorem 6.1: Let S be an integration rule of degree $2n+1$ and Q be an $n \times n$ random orthogonal matrix of the Haar distribution (generated by performing

a QR factorisation of a matrix X whose entries are samples of independent zero mean, unit variance random variables). Then

$$S(Q, f) = \int \int \dots \int_{Y_n} f(u)du \tag{7.14}$$

is exact for all functions f of degree less than equal to $2n + 1$. Further

$$E(S(Q, f)) = \int \int \dots \int_{Y_n} f(u)du \tag{7.15}$$

for any integrable f of any degree. The Expectation in 7.15 is over the space of all random matrices Q.

Proof: This theorem is stated without proof in the work of [32]. We provide an informal proof for the interested reader in the Appendix.

7.5 Generating random orthogonal matrices

Most of the methods for generating a random orthogonal matrix of the Haar distribution that is needed for the theorem above use Heiberger's method [86]. The method consists of first generating an $n \times n$ matrix X with independent entries $x_{ij} = Normal(0, 1)$. Then a QR factorisation of X is computed, $X = QR$. This provides a random matrix Q of the required distribution (the reader may read an informal proof of Theorem 1 in Appendix A to see why this is the case). The basic method of QR factorisation has been refined in the work of [86].

The basic Stewart algorithm is to construct Q as a product of reflectors:

$$Q = (I - \beta x_1 x_1^T)(I - \beta x_2 x_2^T) \dots (I - \beta x_{n-1} x_{n-1}^T). \tag{7.16}$$

In the above equation x_k is of the form $x_k = [0 \dots 0, *, \dots *]$. We use a variation in Stewart's algorithm which combines the generation of the matrix Q with the transformation of the points of the integration formula. Below we present the pseudo-code for the algorithm taken from [33]:

```
Input n
x = 0
Initialise matrix V with the vertices of the n-simplex.
for k = n-1 to 1 do{ \\
  for i = k, i <= n; i++){
    generate random x_{i} = Normal(0,1);
  }
  s = ||x||;
  x[k] = x[k] - s;
  beta = 1/(x[k]s);
  V = V + (beta)x*(Transpose(x)*V);
}
```

7.6 Results

We shall now present results obtained for the same circuit used to generate the curves of Chapter 6. As for the 4σ and ellipsoid-bounding box methods, we show both the cumulative distribution function curves and the variance curves. The cumulative distribution curve was generated using 10,000 points. As can be seen, the yield value returned by integration on the sphere is a lower bound on the true yield. The ellipsoid is a tighter lower bound for 6 dimensions than it is for 8 dimensions, since there are more uncovered corners in higher dimensions. In Figures 7.2 and 7.5, the variance of a Monte-Carlo procedure that uniformly samples over the ellipsoid is shown. The curve clearly shows that uniform sampling is much worse than either Gaussian sampling over the whole polytope, or randomised quadrature. The real value of randomised quadrature is brought out in Figure 7.3 and 7.6. At low yields, the efficiency is in the order of 1000s. Further, the ellipsoid is a tighter bound at low yields (fast performances) than it is at high yields. Thus randomised quadrature is an invaluable technique for quickly calculating low yields.

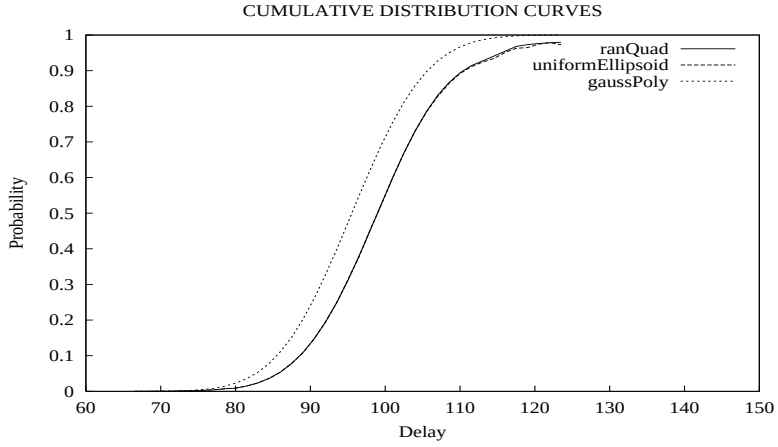


Figure 7.1: This figure shows the cumulative distribution curve obtained by performing spherical-radial integration (denoted by "ranQuad") for 6 global parameters. The spherical-integration method provides a lower bound to the true yield represented by the "gaussPoly" curve.

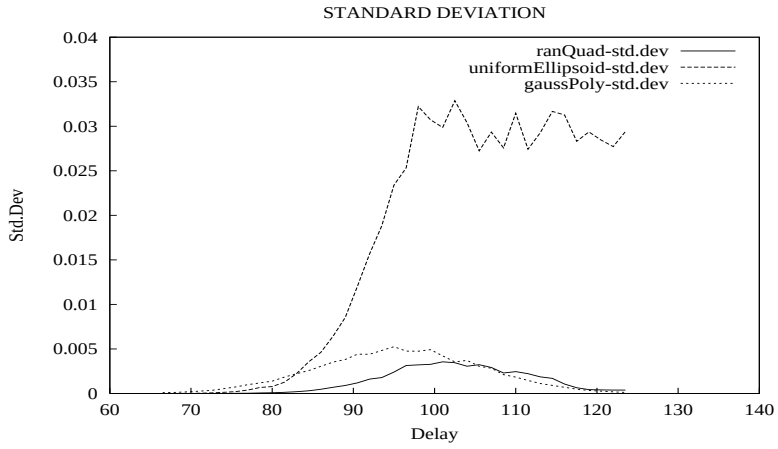


Figure 7.2: This curve represents the standard deviation obtained with the randomised quadrature procedure. 100 random matrices were used for spherical integration and 5 shells were employed for radial integration. The variance of the spherical-radial integration method ("ranQuad") is lower than either sampling uniformly over the ellipsoid ("uniformEllipsoid") or using Gaussian samplingover the yield polytope ("gaussPoly").

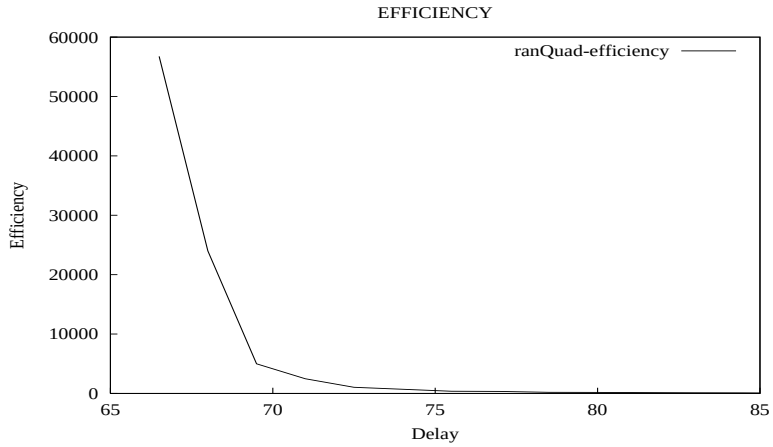


Figure 7.3: This curve represents the efficiency of the randomised quadrature procedure relative to Gaussian sampling. For very fast circuits, the efficiency is in the tens of thousands.

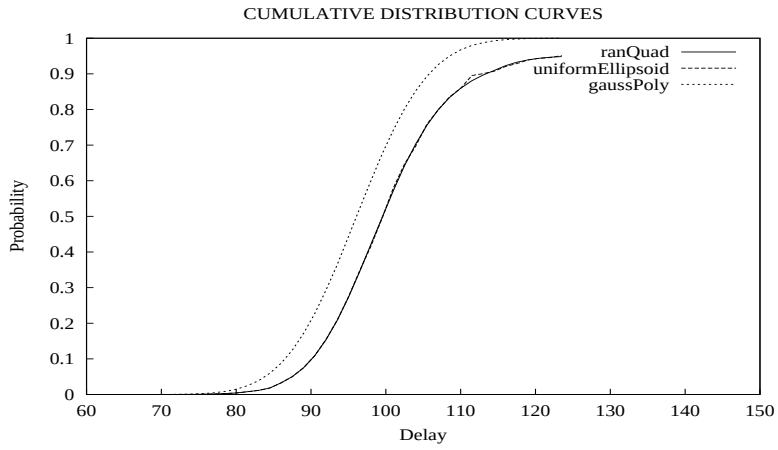


Figure 7.4: Cumulative distribution curves for 8 global parameters. The randomised quadrature lower bound is weaker than for 6 dimensions, because there are more uncovered corners in higher dimensions.

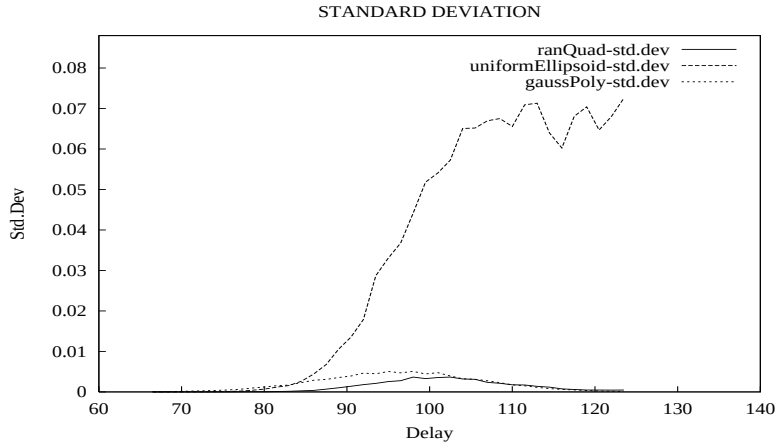


Figure 7.5: Standard Deviation σ Vs Delay for the randomised quadrature method at 8 dimensions. The standard deviation of the randomised quadrature procedure is much lower than that of Gaussian sampling. Both randomised quadrature and Gaussian sampling are much superior to sampling uniformly over the ellipsoid.

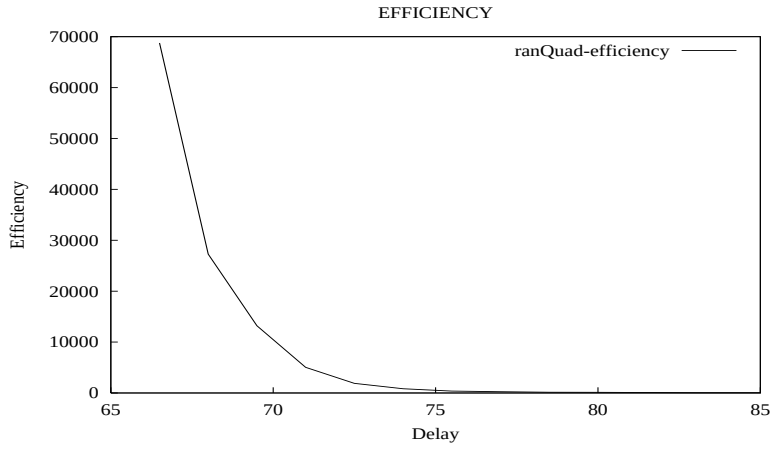


Figure 7.6: Efficiency Vs Delay for the randomised quadrature method at 8 dimensions. The efficiency of the randomised quadrature method is orders of magnitude greater than the efficiency of the tangent importance sampler of Chapter 6, especially for very fast circuits, but it only computes a lower bound on the true yield.

Chapter 8

Yield optimisation

8.1 Introduction

Once we are able to accurately describe the feasible region for circuit performance, and characterise the probability distribution of the process parameters we can try to determine how to increase yield subject to certain constraints. The constraint could be requiring that the probability distribution function must remain fixed, in which case we must move the boundaries of the feasible region. Conversely, we may require that the feasible region remain fixed in space and modify the probability distribution function to increase yield. In this chapter we focus on the former approach i.e., moving the feasible region boundaries while holding the probability distribution fixed. The movement of the feasible region boundaries is achieved tuning the circuit transistor sizes or sensitivities to process parameters. This chapter is organised as follows: first we will describe previous work in yield optimisation including the method of yield gradients and geometric methods such as the minimax method. Then we will show how our ellipsoid framework can be harnessed to give us yield improvement directions. In the last section we will show how to tune the circuit to increase the yield.

8.2 Background

Yield optimisation is a natural extension of the yield estimation exercise. Consequently it has been the subject of intensive research over several decades especially in the context of analog circuits. A compact tutorial of research and methods in this area may be found in [22]. There are many different flavours of yield optimisation. In [22] it is noted that the performance of VLSI circuits is affected by electrical device parameters such as threshold voltage and sheet resistance, and layout geometry parameters such as device sizes. The JPDF that models all these parameters is likely to be complicated. It is far more tractable to model the JPDF in process parameter space. Process parameters include variations of temperatures, or impurity diffusivities, and it is thought that these parameters are at least to first-order independent. Thus the JPDF of process disturbances is likely to be much easier to model than the JPDF of circuit parameters. In yield process optimisation, attempts are made to influence the JPDF of process parameters and move the JPDF closer to the feasible region of circuit operation. However at the circuit level we assume that the JPDF of process parameters is fixed and we modify the feasible region boundaries in order to increase yield.

A taxonomy of yield optimisation methods is provided in [22]. A yield maximisation method that explicitly makes use of knowledge of the JPDF to evaluate the multi-dimensional yield integral and employs the yield integral itself as an optimisation objective is called a *direct* optimisation method. If the yield optimisation procedure makes use of an explicit approximation to the feasible region, it is called an *explicit* method; otherwise it is known as an *implicit* method. Direct methods can be seen intuitively to give the best possible results since they handle the yield integral itself as their objective. However evaluating the yield integral is a computationally expensive process if quadrature methods are used as these methods explode in time complexity with increasing dimensionality. On the other hand, Monte-Carlo methods can be used to approximate yield but they require a lot of simulations at very low yields which is relevant for high performance circuits. Yield optimisation would require, in addition to the computation of yield, the computation of *yield gradients* as well if mathematical programming techniques are applied. Direct methods require a thorough knowledge of the JPDF when in reality, very little is known of the JPDF except perhaps that it is unimodal. An example of a direct approach to yield optimisation is the boundary integral

formulation of [28], [29]. Here the yield integral is formulated as a surface integral using Stoke's theorem and expressions are derived for the yield gradients which are then used in optimisation.

The computational and modelling requirements of direct methods have led researchers to consider indirect methods where yield itself is not an optimisation objective. Design centring is the most well-known indirect approach as it seeks to place the nominal design vector of the given design at the "centre" of the feasible region of design parameters, as this will implicitly increase the manufacturability of the circuit, and thus yield. [24] shows that if the feasible region can be assumed to be convex, and the JPDF is unimodal, then design centring is equivalent to inscribing the largest JPDF-norm body into the feasible region. In contrast, in [41], the acceptability region boundaries are moved away as far as possible from the design centre in order to increase yield. Design centring based approaches have been described in the works of [5], [35], [12] and [95]. In fact the ellipsoidal method of design centring presented by [95] bears some similarity with our own approach.

Implicit methods of yield optimisation do not attempt to construct an approximation of the feasible region. For example in [84], parametric sampling is used as the basis for design centring. The feasible region is usually known as the solution of a set of analytical equations, and implicit methods are content with merely being able to sample the solution space and thus determine if the performance is acceptable. On the other hand, explicit methods use an explicit approximation of the feasible region. One well-known approach to approximating the feasible region is the simplicial approximation proposed by [24]. As an aside, [25] describes how to calculate yield using a simplicial approximation of the feasible region.

Some yield optimisation methods work in the so-called *input space* [27]. The input space includes design parameters (such as the nominal delays of the gates) and operating conditions. Output spaces, on the other hand, include such parameters of circuit performance such as propagation delay and power consumption. For the purpose of yield estimation, input space corresponds to parameter space approaches considered in this thesis, while output space corresponds to performance space methods that were briefly described in Chapter 4.

Deterministic approaches using geometrical arguments have been proposed by several researchers, among whom are [7], [11], [60], [35] and have been

noted to be computationally expensive as the number of parameters (dimensions) increases. On the other hand, statistical techniques for yield optimisation have also been considered. These techniques are not afflicted by the curse of dimensionality and use sampling techniques. Examples of statistical methods for yield optimisation are the papers of [17], [23], [77], [57]. A good place to find a summary of statistical techniques for yield optimisation is [21]. Gate-sizing using statistical models must also be considered within the domain of yield optimisation. An approach of this kind is described in [43]. [13] considers false-path analysis in conjunction with gate-sizing.

An interesting approach of [89] considers the impact of design uncertainty on IC design. This is distinct from the statistical variations in manufacturing parameters, and arises due to the parallel development of designs and processes.

8.3 Our approach

In the context of previous research we will propose an approach that does not make use of the JPDF information (and is therefore indirect) but makes use of an approximation of the feasible region (explicit). The central feature of our approach is to approximate the feasible region by the maximum volume ellipsoid contained in it. It is pertinent to mention that the problem we handle is easier than some of the previously considered problems in that the feasible region is taken to be both convex and specifiable as the solution space of a set of linear inequalities. The constructed ellipsoid is taken to be the approximation of the feasible region, and the axes of the ellipsoid provide the directions in which to nudge the acceptability region boundaries. Let us begin by describing the yield optimisation problem mathematically so as to frame it in a formal setting.

The circuit performance is expressed as a function of *designable* parameters and *process* parameters which are not under the control of the designer. The designable parameters include the lengths and widths of individual transistors whereas the process parameters include the thickness of gate oxide. To provide a mathematical formulation of the yield optimisation problem we shall use the terminology of [22]. Let the vector p represent the designable parameters and let ξ represent the random vector that characterises process deviations. Let the circuit performance of interest (in our case) delay be

represented by ϕ. Then we have

$$\phi = \phi(p, \xi). \tag{8.1}$$

Let us denote the feasible region in process disturbance space for a given nominal vector $p = p_0$ by $F_\xi(p_0)$. Mathematically this feasible region can be expressed as

$$F_\xi(p_0) = \{\xi | \phi^L \leq \phi(p_0, \xi) \leq \phi^U\}. \tag{8.2}$$

The feasible region simply represents the region in process disturbance space such that if the process disturbance vector ξ were to belong to this region, then for the given nominal parameters p_0 the circuit will still have an acceptable performance. It is crucial to note that the shape of the feasible region, and its location in process disturbance space depends on the (designable) nominal parameter vector p_0. Manipulating the nominal parameter vector will cause the boundaries of the feasible region to move and is thus our means of optimising yield. The parametric yield can be expressed as

$$Y = Prob(\xi \in F_\xi(p^0)) = \int_{F_\xi(p^0)} f_\xi(\xi) d\xi. \tag{8.3}$$

The optimisation problem is therefore

$$max_{p_0}\Big[Y = Prob(\xi \in F_\xi(p_0)) = \int_{F_\xi(p_0)} f_\xi(\xi) d\xi\Big]. \tag{8.4}$$

In the above two equations $f_\xi(\xi)$ is the joint probability density function of the process disturbances. In the next section we shall delve into the mechanics of modifying the acceptability region boundaries so as to increase the yield.

8.4 Acceptability region modification

Yield optimisation is much more tractable when the acceptability region boundaries are given by hyperplanes and not curved boundaries. Curved boundaries require us to compute yield gradients as in the approach of [28], which is a time-consuming process. The closest relative to the method we propose is the so-called "minimax method" [41]. Let us describe this method first.

8.5 Minimax method

This technique was pioneered by Texas instruments researchers and a description of it can be found in [41]. The basis of the technique is to move the hyperplanes in order to increase yield. The movement of hyperplanes to increase yield means that the nominal delays of gates in the circuit will need to be changed. Therefore the procedure can be seen as a "tuning" procedure to improve yield. Let the designable parameters of the circuit be represented in the vector p as in the previous section. The designable parameters can be taken to be the lengths of individual transistors in the circuit.

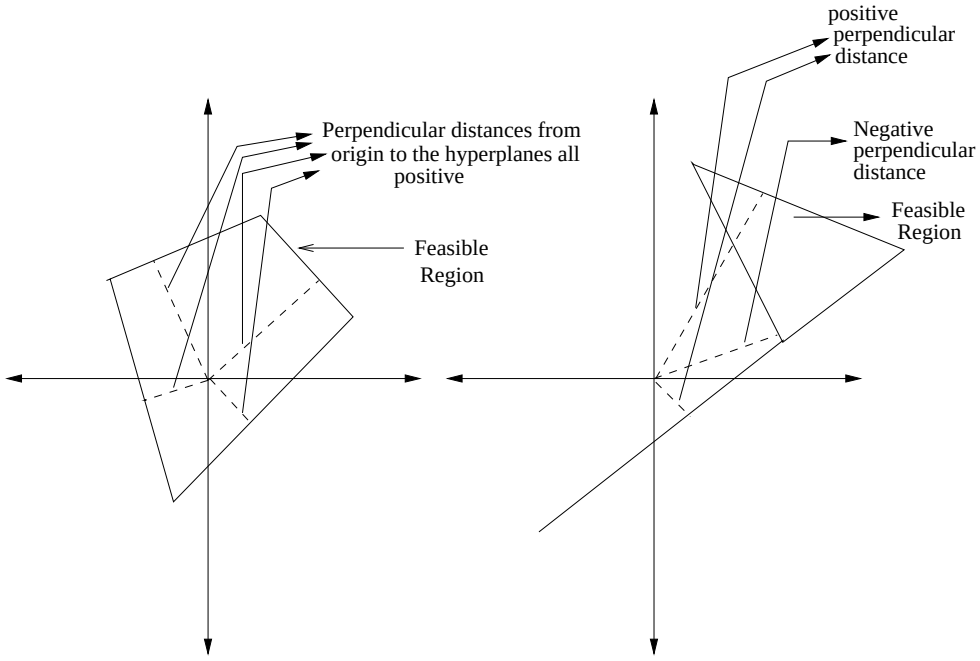


Figure 8.1: Positive and negative distances from the origin to the feasible region hyperplanes in the minimax method.

The basic idea of the minimax method of yield optimisation is to maximise the minimum distance from the mean point of the joint probability density function of global parameters to the acceptability region boundaries (in our case hyperplanes). The rationale behind this is that the boundaries will move

so that the new feasible region will include regions of greater probability mass thus increasing the yield. The minimax method uses an iterative approach to solve the problem wherein in a given iteration, the distance functions to the feasible region boundaries are linearized, and a linear programming problem formulated to maximise the minimum distance. Let us reconsider equation (4.3):

$$\gamma + R\Delta z \leq [\eta\ \eta\ \cdots\ \eta]^T . \tag{8.5}$$

The vector γ is the vector of nominal path delays. There are P rows in the above matrix system where the ith row represents the constraint that the nominal delay of path i plus the perturbation of the delay due to the variations of the global parameters be less than η. The mean point of the JPDF corresponds to the origin in the hyperplane system defining (8.5) and the distance from the origin to the jth hyperplane is given by

$$m_j = \frac{\eta - \gamma_j}{\|R_j\|} \tag{8.6}$$

where $R_j^T \Delta z + \gamma_j \leq \eta$ is the equation of the jth hyperplane. If the origin falls within the feasible region then all the distances to the hyperplanes will be positive. If the origin lies outside the feasible region, then at least one distance to some hyperplane is negative. Figure 8.1 demonstrates the situation when the origin is inside the feasible region, as well as when the origin is outside the feasible region. We are now ready to formulate the linear programming problem. Instead of working with distances m_j we prefer to work with their negatives $\nu_j = -m_j$. Then instead of maximising the minimum m_j we minimise the maximum ν_j. We must now develop constraints for the problem. Note that the nominal delay of each path as well as the hyperplane coefficients depend on the vector of designable parameters p. To create linear constraints we must linearise the distance functions. If the distance between the origin and hyperplane j is ν_j then the change in the distance caused by a small change in the design parameter vector Δp is given by $\nabla_p \nu_j \Delta p$. This allows us to pose the following linear programme assuming that we allow only small changes in the design parameter vector [41]:

$\min_{\Delta p^k} r^k$
subject to
$\nu_j^k + \nabla_p \nu_j^k . \Delta p^k \leq r^k, j = 1 \ldots P$
$-\lambda^k \leq \Delta p^k \leq \lambda^k$

$$B_l \leq p^k + \Delta p^k \leq B_u.$$

The first P equations above state that the negative distances of all the hyperplanes from the origin after they are moved due to a change in the design parameter vector of magnitude Δp^k, are no greater than r^k. Note that k denotes the iteration number as we solve the above linear programme repeatedly until we are satisfied with the answer. Thus the above is a minimax problem. The magnitude of the change in the design parameter vector is bounded by the next set of equations (this bound is what allows the linearisation of constraints) while the last set of equations make sure that the design parameter vector lies between user-specified upper and lower bounds. In order to make the above LP functional we must specify the gradient of the distance function with respect to the vector of designable parameters:

$$\nabla_p \nu_j = \frac{\nabla_p \gamma_j}{\|R_j\|} + \frac{\eta - \gamma_j}{\|R_j\|^3} R_j^T \frac{\partial R_j^T}{\partial p}. \tag{8.7}$$

Let us now turn to the actual iteration control algorithm to solve the LPs. First we fix λ and then decide whether to accept or reject the new iterate Δw^k. The decision to accept or reject is based on checking if the actual increase in the minimum distance compares favourably with the predicted increase in the minimum distance. The two distances can be defined as follows:

$$\begin{aligned} \Delta r_a &= \max_j\{\nu_j^k\} - \max_j\{\nu_j^{k+1}\}, \\ \Delta r_p &= \max_j\{\nu_j^k\} - r^k. \end{aligned} \tag{8.8}$$

For a given choice Δw_k, we can solve the linear programme to determine the actual improvement in the minimum distance Δr_a. This improvement can be compared to the predicted (or desired) change Δr_p in order to accept or reject the choice Δp_k. If the choice is accepted (when $\frac{\Delta r_a}{\Delta r_p} > 10^{-3}$) then $p_{k+1} = p_k + \Delta p_k$. Otherwise $p_{k+1} = p_k$. The step-length bound for the next iteration, λ^{k+1} is then adjusted to be equal to $\frac{\Delta r_a}{\Delta r_p}\|\Delta p_k\|_\infty$. The iterations are continued until the user is satisfied with the solution.

The main drawback of the minimax solution is that increasing the minimum distance from the mean point of the JPDF to the hyperplanes may not increase the yield always. This is because distances greater than the minimum

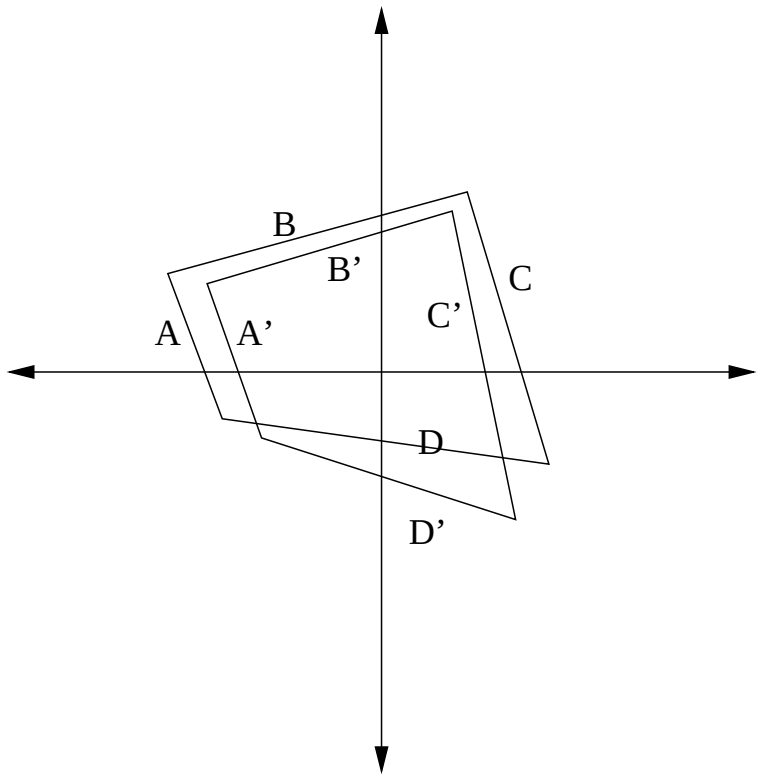


Figure 8.2: minimax method in action; hyperplanes A,B,C,D move to A',B',C',D'. The minimum distance from the origin to the hyperplanes constituting the original feasible region, given by the hyperplane boundaries A,B,C and D, is the distance to the hyperplane D. After optimisation D moves to D' such that the minimum distance to D' is greater than the distance to D but the distances to the other hyperplanes reduce in such a way as to reduce the size of the feasible region.

distance may be brought down by the optimisation procedure as shown in Figure 8.2. Further in the design vector, every transistor length may have to be included which will make the design vector huge for a circuit block consisting of thousands of transistors. In the next section we develop a method that does not depend on the dimensionality of the vector of designable parameters, but instead depends on the dimensionality of the process parameters.

8.6 Ellipsoidal method for yield optimisation

In this section we will develop an approach that is complementary to the minimax approach of the previous section, although the basic methodology for yield optimisation remains the same; that is, we seek to increase yield by moving the hyperplane boundaries. However, we will avoid the use of gradients and design vectors. Instead we shall first determine how much to move individual hyperplanes in order to increase the yield by a specified amount, and then tune the driving capacitances of circuit gates in order to enforce the changes in the hyperplane positions.

Our starting point is to recall that the ellipsoid transformation matrix B can be expressed as

$$B = S\Lambda S^{-1}. \tag{8.9}$$

In the above equation S is an $n \times n$ matrix whose columns represent the eigenvectors of the matrix B. Λ is a diagonal matrix whose entries are the eigenvalues of the matrix B. Next we note that

$$\begin{aligned} \det B &= \det(S\Lambda S^{-1}) \\ &= (\det S)(\det \Lambda)(\det S^{-1}) \\ &= \det \Lambda \\ &= \prod_{i=1}^{i=n} \lambda_i. \end{aligned} \tag{8.10}$$

Thus we note that the determinant of B can be expressed as the product of its eigenvalues. We would like to determine the change in $\det B$ if we were to change an eigenvalue (say λ_i by an amount $\Delta\lambda_i$). Clearly

$$\begin{aligned} \Delta \det B &= \frac{\partial \det B}{\partial \lambda_i} \Delta\lambda_i \\ &= \prod_{j=1, j\neq i}^{j=n} \lambda_j \Delta\lambda_i. \end{aligned} \tag{8.11}$$

The form of (8.11) shows that the largest increase in the volume of the determinant of B and hence the volume of the ellipsoid occurs when the

smallest eigenvalue is changed. Therefore our strategy for yield improvement is to increase the smallest eigenvalue. This will cause the length of the smallest axis of the ellipsoid to increase, and would require the readjustment of some hyperplanes. We illustrate this in Figure 8.3.

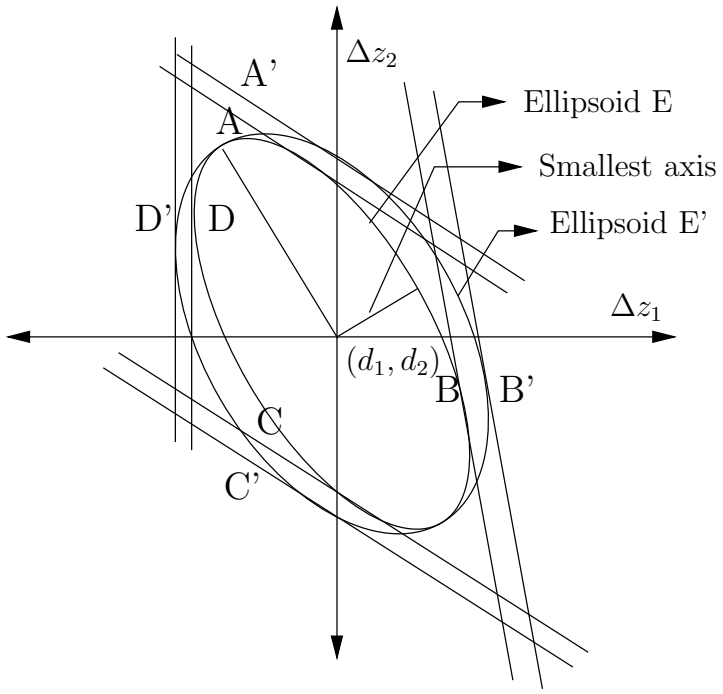


Figure 8.3: Strategy for yield optimisation. The ellipsoid E is expanded in the direction of its minor axis to get the ellipsoid E'. The hyperplanes are moved to accommodate the expanded ellipsoid.

Note that the above statement has a caveat - namely, that we have disregarded the presence of the JPDF. It is conceivable that the ellipsoid could be so oriented with regard to the JPDF that there is greater real yield increase in a direction other than the eigenvector corresponding to the smallest eigenvalue. In such cases, we might resort to picking the ellipsoid axis whose axial endpoint is closest to the point on the ellipsoid that is closest to the centre of the JPDF. The point on the ellipsoid closest to the centre of the JPDF can be computed via the tangent computation method of Chapter 6.

We shall now discuss how to accommodate the expanded ellipsoid by moving

some hyperplanes. First we must transform the hyperplane and ellipsoid system to the unit sphere space. Following (4.3) let the ith hyperplane be represented by

$$\gamma_i + R_i^T \Delta z \ \leq \eta. \tag{8.12}$$

Noting that $\Delta z = By + d$ for y in the unit sphere, this equation gets transformed to

$$\begin{aligned} \gamma_i + R_i^T (By + d) &\leq \eta, \\ R_i^T By + (\gamma_i + R_i^T d) &\leq \eta. \end{aligned} \tag{8.13}$$

Due to an expansion of the ellipsoid, this hyperplane might have to be moved. We make the assumption that for small changes in the eigenvalue, the hyperplane coefficients will remain unchanged. We shall later remove this assumption. Therefore the hyperplane will move parallel to itself(if it has to move at all). We must determine conditions under which a hyperplane must move and the amount of the movement. To do this we must first quantify the change in the ellipsoid transformation matrix due to a slight change in an eigenvalue of the ellipsoid transformation matrix. Let us assume without loss of generality that the first eigenvalue in Λ is the smallest eigenvalue (the first column of S is the eigenvector corresponding to the smallest eigenvalue). Then the new ellipsoid transformation matrix becomes $B + \delta B$ where

$$\begin{aligned} B + \delta B &= S(\Lambda + \delta\Lambda)S^{-1} \\ &= S\Lambda S^{-1} + S\delta\Lambda S^{-1} \\ &= B + S\delta\Lambda S^{-1}, \end{aligned} \tag{8.14}$$

and

$$\delta\Lambda = \begin{bmatrix} \delta\lambda_1 & 0 & \dots \\ 0 & 0 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}.$$

Further we can easily see that

$$\delta B = \delta\lambda_1 s_1 s_1^T \tag{8.15}$$

so that

$$B + \delta B = B + \delta\lambda_1 s_1 s_1^T. \tag{8.16}$$

In unit sphere space, every ellipsoid is transformed to the unit sphere. The fact that expansion of the ellipsoid requires some hyperplanes to move is manifested in the unit sphere space in terms of those hyperplanes intersecting the unit sphere in more than one point. In other words, the distance from the origin to the hyperplanes in question becomes less than one. Therefore the condition for testing if a hyperplane in unit sphere space intersects the unit sphere in more than one point is to check if its distance from the origin is less than one:

$$\frac{\eta - (\gamma_i + R_i^T d)}{\|R_i^T (B + \delta B)\|} \leq 1. \tag{8.17}$$

In case the above inequality is satisfied by the ith hyperplane then the amount $\|R_i^T(B+\delta B)\|(1 - \frac{\eta-(\gamma_i+R_i^T d)}{\|R_i^T(B+\delta B)\|})$ needs to be added to the right-hand side of the hyperplane equation so that it becomes a tangent to the unit sphere. The new hyperplane equation becomes:

$$R_i^T(B + \delta B)y \leq \|R_i^T(B + \delta B)\|. \tag{8.18}$$

In original (ellipsoid) space, where $\Delta z = (B + \delta B)y + d$ this new equation becomes:

$$R_i^T z - R_i^T d \leq \|R_i^T(B + \delta B)\|. \tag{8.19}$$

In ellipsoid space, it turns out that the new hyperplane equation (8.19) is obtained from the old equation by adding a term $\|R_i^T(B+\delta B)\|(1-\frac{\eta-(\gamma_i+R_i^T d)}{\|R_i^T(B+\delta B)\|})$ to the right-hand side of the old hyperplane equation. Of course, the shifting of the hyperplane is only necessary provided the following condition holds true:

$$\frac{\eta - (\gamma_i + R_i^T d)}{\|R_i^T(B + \delta\lambda_1 s_1 {s_1}^T)\|}) \leq 1. \tag{8.20}$$

The entire process of calculating hyperplane movements is shown in Figure 8.4.

We can determine if any hyperplane has to be moved by checking if the condition in (8.20) is satisfied for that hyperplane. Moving a given hyperplane means decreasing the nominal delay of the path corresponding to the hyperplane. For each path we are therefore able to determine if the path should be tuned or left alone. In order to complete the tuning we must devise a

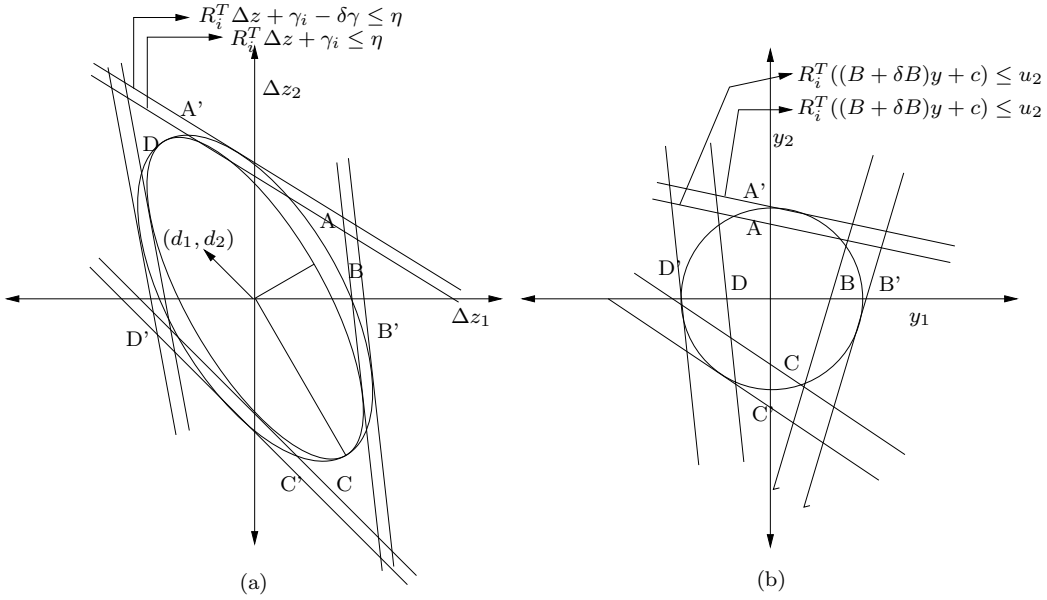


Figure 8.4: The tuning process is illustrated in the figure above. The hyperplanes A,B,C and D enclose the feasible region before yield optimisation. After optimisation, in which the smallest ellipsoid axis's length is increased, the hyperplanes move to A', B', C' and D' respectively. The amount of movement is calculated in unit sphere space.

strategy to actually change nominal gate delays and enforce the required change in the nominal delay of a path. It is pertinent at this stage to point out the differences between the minimax method and our own method, since both are geometric methods. The minimax method works explicitly with the design vector (whose size is the number of gates in the logic block being optimised), whereas in our case the design vector is implicit. The minimax method needs derivatives of the nominal delay of a path with respect to the relevant design variables in order to compute gradients. A solution to the minimax problem will automatically provide a means to actually enforce the solution since we explicitly compute the new design vector. In our case, we determine first how much to change the nominal delay of a path and then proceed to tweak individual gate elements to achieve the change. The minimax formulation is more general in that it can handle a situation where the sensitivity coefficients of a path to the global parameters is a function

of the design variables. We assume that the sensitivity coefficients do not depend on the design variables if the changes in the eigenvalue (and therefore nominal delays) is not too large. Our method explicitly increases the yield volume in that all the points of the old polytope belong to the new polytope (one obtained by moving some hyperplanes). This is in contrast with the minimax technique where maximising the minimum distance will cause the minimum distance to the hyperplanes to increase but could also reduce distances greater than the minimum.

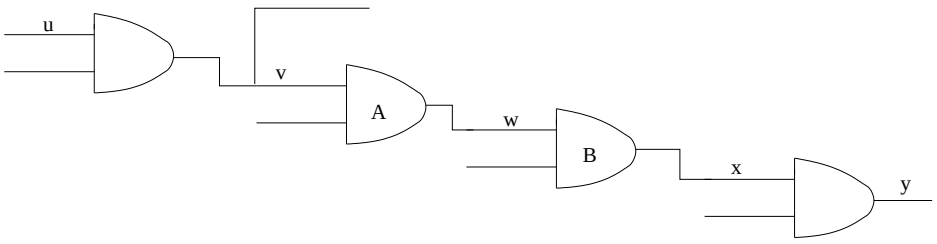


Figure 8.5: A path in a circuit to be optimised, marked as $u - v - w - x - y$.

Consider a path of the kind in Figure 8.5. The outputs of some of the gates have a fanout greater than one; we do not show the off-path gates. Let us focus on the gates marked (A) and (B) in the figure and assume that one of the gates (say A) is faster than the other gate, say (B). Let C_{in}^A and C_{out}^A represent the input and output capacitances of gate A. Let C_{in}^B and C_{out}^B be the input and output capacitances of gate B. The delay of gate A is proportional to the ratio of its output capacitance to its input capacitance:

$$d_A = k\frac{C_{out}^A}{C_{in}^A} \tag{8.21}$$

and

$$d_B = k\frac{C_{out}^B}{C_{in}^B}. \tag{8.22}$$

This follows the logical effort modelling of gate delay if we ignore the parasitic component of delay. Next we shall see that if we were to make the faster gate (A) slightly slower, and the slower gate (B) slightly faster we shall accomplish a net reduction in delay, thus speeding up the path. First we observe that

since gate (A) feeds only gate (B), $C_{in}^{B} = C_{out}^{A}$. Let the output capacitance of A be changed by a small amount δC. Then we have

$$\delta d_A = \frac{\delta C}{C_{in}^{A}}. \tag{8.23}$$

Further

$$\begin{aligned} \delta d_B &= \frac{C_{out}^{B}}{C_{out}^{A} + \delta C} - \frac{C_{out}^{B}}{C_{out}^{A}} \\ &= \frac{C_{out}^{B}}{C_{out}^{A}}(1 + \frac{\delta C}{C_{out}^{A}})^{-1} - \frac{C_{out}^{B}}{C_{out}^{A}} \\ &= \frac{C_{out}^{B}}{C_{out}^{A}}(1 - \frac{\delta C}{C_{out}^{A}}) - \frac{C_{out}^{B}}{C_{out}^{A}} \\ &= \frac{-\delta C C_{out}^{B}}{(C_{out}^{A})^2}. \end{aligned} \tag{8.24}$$

We have made use of the approximation $(1 + x)^{-1} \approx (1 - x)$ in the above simplification. In order to ensure that the path is speeded up we have to make sure that

$$\delta d_A + \delta d_B < 0. \tag{8.25}$$

This means

$$\frac{\delta C}{C_{in}^{A}} - \frac{\delta C C_{out}^{B}}{(C_{out}^{A})^2} < 0. \tag{8.26}$$

Thus if

$$\frac{C_{out}^{A}}{C_{in}^{A}} < \frac{C_{out}^{B}}{C_{out}^{A}} \tag{8.27}$$

we must change the output capacitance of gate A by a positive amount $\delta C > 0$ in order to reduce the nominal delay of the path. Thus if gate A is faster than gate B, we must increase the delay of gate A and correspondingly reduce the delay of gate B in order to make sure that the net delay of the path is reduced. On the other hand, if

$$\frac{C_{out}^{A}}{C_{in}^{A}} > \frac{C_{out}^{B}}{C_{out}^{A}} \tag{8.28}$$

then $\delta C < 0$ in order to reduce the nominal delay of the path. The upshot of this derivation is that if there is a delay differential between gates A and B, we can speed up the path by reducing the delay differential between the two gates. The exact amount δC can be directly calculated from the change in the constant term of the hyperplane corresponding to the path in question.

Finally, let us end this section by trying to relax the constraint that the hyperplanes corresponding to paths can only be moved by changing their nominal delays. Recalling (8.20) we see that one way to increase the distance of a hyperplane from the origin and thereby make it a tangent to the expanded ellipsoid is to decrease the value of the denominator $\|R_i^T(B+\delta B)\|$. This can be accomplished by changing R_i.

It must be remarked that changing nominal delays to increase yield at a given performance without tinkering with the sensitivity coefficients of the paths, will also increase yield at any other performance. If a hyperplane $\gamma_i + R_i^T \Delta z$ is tuned to $\gamma_i^{'} + R_i^T \Delta z$ with $\gamma_i^{'} < \gamma_i$, then the latter path will always be faster than the pre-tuning path. The performance value η does not enter the equation here. On the other hand, if we were to also allow changes in the sensitivity signature of a path so that R_i becomes $R_i^{'}$, then we can no longer guarantee that tuning performed for a given performance to increase yield will also increase yield at any other performance. This is because there are some performances for which the pre-tuning path is less critical than the post-tuning path, and can therefore be less yield-limiting than the post-tuning path. Let us now consider changing the sensitivity vector P for an arbitrary cell, and evaluate its effect in pushing the hyperplane boundaries outwards. Every path in the circuit that passes through this particular cell will be affected. There may potentially be thousands of paths passing through this cell. We can use the path filtering technique of Chapter 4 to obtain a small set M of critical paths that represent the entire angular spectrum of the paths that pass through the cell in question. Let us now evaluate the effect of the change in the cell sensitivity vector on each of the paths $R_i \in M$.

Let us first rewrite $B + \delta B$ as $S\Omega S^{-1}$ where S is the same eigenvector matrix as for B, and the diagonal matrix $\Omega = \Lambda + \Delta\Lambda$. Further, let the vector $S^{-1}R_i$ be represented by the vector Q_i. Since S^{-1} is a rotation matrix the norm of Q_i will be the same as R_i. Let $Q_i = (q_1^i, q_2^i, \ldots q_n^i)^T$. Let the sensitivity vector change of the cell in question be written as $\Delta P = (\Delta p_1, \Delta p_2, \ldots \Delta p_n)^T$. Let $S^{-1}\Delta P = \Delta V = (\Delta v_1, \Delta v_2, \ldots, \Delta v_n)$. Then the denominator of (8.20)

becomes:

$$\|(R_i + \Delta P)^T (B + \delta B)\| = \sum_{j=1}^{n} (q_j^i + \Delta v_j)^2 \omega_j^2. \tag{8.29}$$

The Δv_js must be so chosen as to reduce the sum on the right as this will reduce the norm and the denominator of (8.20). One way to do this is to let $-2q_j^i \leq \Delta v_j \leq 0$ for each Δv_j if $q_j^i > 0$ and $0 \leq \Delta v_j \leq -2q_j^i$ if $q_j^i < 0$. Such constraints can be written for all the other paths in M as well. Taken together, these constraints may yield only the zero-vector for ΔV and therefore $\Delta P = S\Delta V$, especially if there is a lot of angular variability in the paths passing through the cell in question. In case the constraints turn out to yield the zero-vector for ΔP, we must seek recourse to other means of reducing the norm. In case of tuning the nominal delays, special care has to be taken in the choice of the cell so as to ensure that all paths passing through that cell are not slowed down. If we choose to not change the nominal delay of the selected cell, and only tune the sensitivity coefficients, then there is considerably more leeway in the choice of a cell than in the case of nominal delay tuning to increase yield.

Although the choice of the cell in the above derivation was described as arbitrary, this need not be the case. Some cells are obviously more important than other cells, since a large number of yield-critical paths may pass through them. The following procedure on identifying promising cells suggests itself. For each path whose sensitivity vector is along the most promising ellipsoid axis of improvement, identify the list of cells that lie along this path. Augment the weight of each cell along the given path by the amount that the path must be speeded up in order to become a tangent to the expanded ellipsoid. When this exercise is performed for all the yield-critical paths, the cells with the larger weights will be more important from a yield-enhancement point of view, than those with smaller weights. The methods outlined in this chapter can be used to tune for yield given the list of yield-critical cells, or an existing tuner can make use of the cell weights as it tunes for some other objective, such as low power.

Chapter 9

Conclusions and future work

In this thesis we examine different aspects of the statistical timing problem for combinational circuits. The need for statistical timing analysis is easily established given the growing prominence of process variations in modern semiconductor processes. Once the need for statistical timing is accepted, there remains the question of establishing how the process variations impact the delays of the individual gate elements. As may be imagined the complexity of process variations has meant that there are no standard statistical models that can be used for statistical timing.

Two different models are examined: in the first, the delays of the gate elements are assumed to be completely independent of each other. This model is shown to be similar to computational model of the statistical PERT problem, which falls within the domain of operations research. Although the gate delays in this model are independent, path sharing introduces correlations between paths. The super gate approach of [14] is used to take care of the correlation introduced by path sharing. Although deterministic lower and upper bounds are constructed for the cumulative distribution curve of circuit delay, the problem is computationally very hard for all but the simplest circuits with a modest amount of reconvergent fanout.

It is reasonable to expect that process variations affect different gates on a die in similar ways. One may imagine that there are a set of global parameters such as L_{eff}, T_{ox} that can be assumed to be random variables. Different gate delays may then be assumed to have different *sensitivities* to these global parameters. Thus a first order model can be constructed where individual

gate delays vary linearly with small changes in the values of the "global" process parameters. Requiring that the circuit operates at a delay no greater than η then means that the "global" process parameters must take their values in a "feasible region" defined by hyperplanes that represent paths in the circuit. Thus the yield computation problem is reduced to one of computing the volume of a polytope in n-dimensions where n is the number of global parameters in the circuit. Two avenues of future work readily suggest themselves. They will be described in the next two sections.

9.1 Improving model sophistication

Assuming that the delays of the gates are not independent, but depend on the same set of global parameters, is more realistic than assuming complete independence. However, even this model may not be realistic. A recent paper, [94] reflects on the possibility that the gate delays may depend on a common set of global parameters as is assumed in this thesis, but each gate delay may also have a small independent component. This model cannot easily be adapted to the methods proposed in this thesis. A straightforward extension of the linear delay model assumed in this thesis would introduce a random variable for each gate delay to represent the independent component of the delay, in addition to random variables representing global parameters. The problem with this approach is the explosion in the dimensionality of the model - the success of the methods we examine in this thesis is contingent upon the number of parameters (dimensionality) not exceeding a few tens. If each gate delay is modeled by a separate random variable, then the dimensionality would shoot up into the thousands which would make our methods unviable. In such a huge space, it may well be that Monte-Carlo methods for yield estimation are the only viable ones. As shown in Chapter 6, the choice of sampling function is crucial to the variance of the Monte-Carlo yield estimate. We believe that the tangent importance sampler of Chapter 6 can be adapted to the situation where the gate delays have an independent component in addition to the global process parameters.

Table 9.1: Summary of methods considered in this thesis.

Method	Model Sophistication	Computational Complexity	Error theory	Tuning information
Discretisation of gate delay distributions	Crude-gate delays independent	exponential in number of supergate fanout points	reasonable; increasing number of impulses reduces error	Not available
uniform sampling over polytope	gate delays correlated; path sharing taken into account	linear in number of paths and dimensions	high variance; gets worse for slow performances and high dimensions	Not available
uniform sampling over ellipsoid bounding box	gate delays correlated; path sharing taken into account	linear in number of paths and dimensions	high variance; better than uniform sampling over polytope; gets worse for slow performances and high dimensions	Available; axes of ellipsoid determine yield improvement direction
Gaussian sampling from JPDF mean	gate delays correlated; path sharing taken into account	linear in number of paths and dimensions	low variance; better than uniform sampling; standard deviation curve is bell-shaped; dimension independent	Not available
Gaussian sampling from tangent point	gate delays correlated; path sharing taken into account	non-linear in number of paths and dimensions	very low variance; better than Gaussian sampling for very fast performances; gets worse at slower performances. Otherwise, Gaussian-like	Available; axes of ellipsoid determine yield improvement direction
randomised quadrature	gate delays correlated; path sharing taken into account	non-linear in number of paths; quadratic in number of dimensions	lowest variance of all methods for very fast performances; gets worse at slower performances. Computes lower bound on true yield.	Available; axes of ellipsoid determine yield improvement direction

9.2 Relaxing the linearity assumption

The gate delays in this thesis are assumed to vary linearly with respect to the changes in the global parameters. This assumption becomes less true when the changes in these global parameters are not small. In such situations one must assume a non-linear model of gate delay dependence. As a direct consequence, the yield body will no longer be a feasible region bounded by hyperplanes, but a more complicated region possibly bounded by curved surfaces, as shown in the figure below. Depending on the non-linear model, it is possible that the feasible region may not even be convex. The maximum

volume ellipsoid method can only handle convex regions specifiable by linear matrix inequalities. For the more complicated regions, one may have to resort to a linear feasible region approximation. The feasible region may be thought to be the intersection of a number of half-spaces, just like in the model proposed in this thesis. However the half-spaces will not represent "paths" in the circuit as in the present model. Instead they will merely be the tangent planes at designated points on the surface of the feasible region as shown in Figure 9.1. This technique is a form of simplicial approximation that has been explored in the context of design centring by [24]. One way to arrive at a linear approximation of the feasible region containing only a few half-spaces is to first choose a a lot of points at random on the surface of the feasible region, and find the tangent planes at each of these points. One may then apply the path filtering criterion of Chapter 4, to obtain a feasible region described by only a subset of the original constraints. Then the methods outlined in this thesis become applicable.

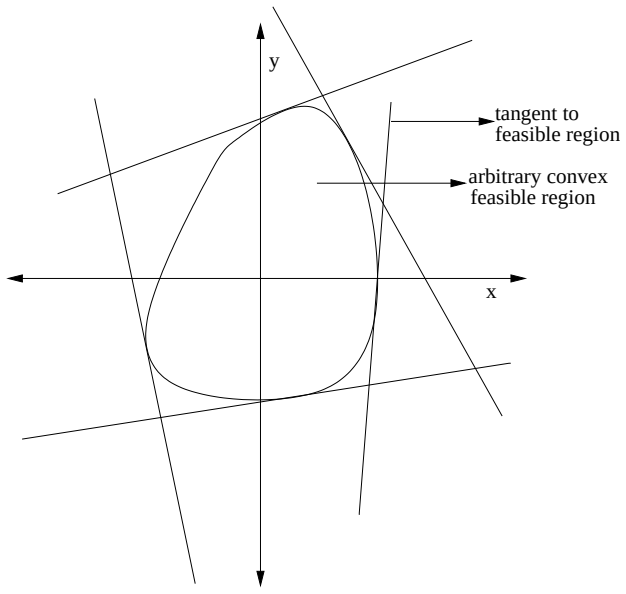


Figure 9.1: Linearisation of a an arbitrary convex feasible region.

9.3 Tackling high yield situations

A glance at the Table 9.1 shows that while we have good methods for evaluating very low yields (corresponding to fast performances), we do not have a good method for evaluating high yield situations (corresponding to slow performances). We shall briefly explore how to achieve variance reduction for high yield situations.

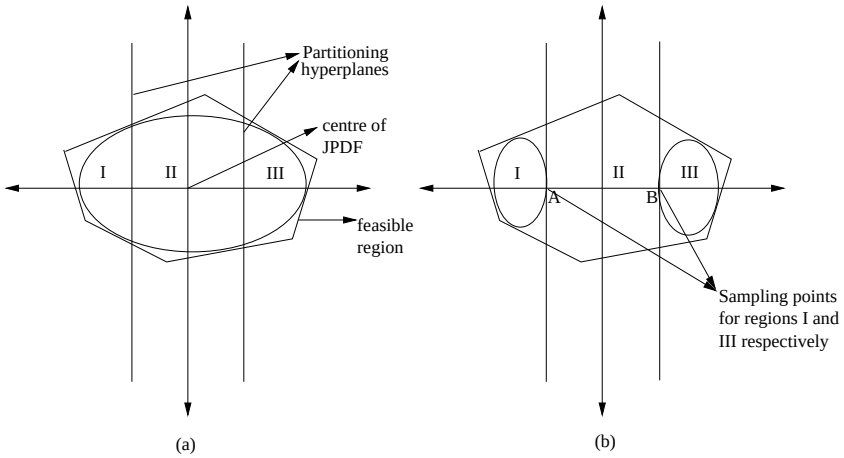


Figure 9.2: Evaluating the yield integral for a situation where the nominal point of the JPDF is deep inside the feasible region.

It was noted in Chapter 6 that the low-variance method of sampling from the point on the ellipsoid that is closest to the joint probability density function of global parameters would not work for situations where the nominal point of the JPDF is located deep inside the feasible region. In this section, we outline a possible line of attack to handle high yield situations. The proposed scheme involves partitioning the feasible region at slow performances into several regions. We then calculate the yield as the sum of the contributions of each individual region. The partitioning hyperplanes can be found by first finding the maximum volume ellipsoid that can be inscribed in the feasible region, and then picking the partitioning hyperplanes so as to (a) be perpendicular to the major axis of the ellipsoid, and (b) cut the positive and negative major axes in half, as shown in Figure 9.2a. The nominal point of the JPDF will fall in the "central" region. For this "central" region, the JPDF itself should be

used for sampling. The nominal point of the JPDF lies outside each extreme region. Therefore this situation is similar to the low-yield feasible region explored in Chapter 6, and as in Chapter 6, we can find a maximum volume ellipsoid that can be inscribed in each extreme region and for each region, sample from the point on the ellipsoid that is closest to the nominal point of the JPDF.

APPENDIX

We provide an informal proof of the theorem that is the basis for chapter 7. The theorem is restated here for convenience:

Theorem 6.1: Let S be an integration rule of degree $2n+1$ and Q be an $n \times n$ random orthogonal matrix (generated by performing a QR factorisation of a matrix X whose entries are samples of independent zero mean, unit variance random variables). Then

$$S(Q, f) = \int \int \ldots \int_{Y_n} f(u)du \tag{A-1}$$

is exact for all functions f of degree less than equal to $2n + 1$. Further

$$E(S(Q, f)) = \int \int \ldots \int_{Y_n} f(u)du \tag{A-2}$$

for any integrable f. The Expectation is over the space of all random matrices Q. Y_n is the surface of the unit sphere.

Proof: We shall establish the theorem for the two dimensional case. Then we shall consider 3 and higher dimensions:

(a)**2-dimensions**:

Let us consider the process of QR factorization given by the equation

$$X = QR \tag{A-3}$$

where X is an $n \times n$ matrix of independent, zero mean, unit variance normal random variables, Q is an $n \times n$ orthogonal matrix, and R is an upper triangular matrix. We shall show that the columns of Q are identically distributed and that the distribution is a uniform distribution over the 2-dimensional unit sphere - in other words, each column of Q is a random vector with equal probability of pointing in any direction. Given this fact, we can then easily establish that the expectation is equal to the surface integral as desired by the statement of the theorem.

Let us consider the two columns of the 2×2 orthogonal matrix Q. They can be written as follows:

$$\begin{aligned} Q_1 &= \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \\ Q_2 &= \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix}. \end{aligned} \tag{A-4}$$

The fact that the columns of Q are random variables means that the angles θ and ϕ are also random variables. We will show that these angles are identically but *not* independently distributed. In order to demonstrate this, let us look at the process of QR factorisation:

$$\begin{aligned} X &= QR \\ \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix} &= \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{bmatrix}. \end{aligned} \tag{A-5}$$

From the above equation, we can write:

$$\begin{aligned} Q_1 &= \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \\ &= \begin{bmatrix} \frac{x_{11}}{r_{11}} \\ \frac{x_{21}}{r_{11}} \end{bmatrix}. \end{aligned} \tag{A-6}$$

The above equation in turn leads to the following:

$$\tan\theta = \frac{x_{21}}{x_{11}} \tag{A-7}$$

Note that x_{11} and x_{21} are independent zero mean, unit variance random variables. It can be shown (see [78]) that the random variable representing the arc-tangent of the ratio of two independent zero-mean, same variance random variables has a uniform distribution over the range $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$. Thus , θ is a uniform random variable where

$$p(\theta) = \frac{1}{\pi}, \theta \in [0, \pi]\,. \tag{A-8}$$

We have therefore established that the first column of Q points in a uniformly random direction. Since the second column is orthogonal to the first we have the following equation:

$$\begin{aligned} (Q_1)^T Q_2 &= 0 \\ \cos\theta\cos\phi + \sin\theta\sin\phi &= 0 \\ \cot\theta &= -\tan\phi. \end{aligned} \tag{A-9}$$

Now recalling that $\tan\theta$ is simply a ratio of two independent $N(0,1)$ random variables, we observe from the above equation that $\tan\phi$ is also a ratio of

two independent $N(0,1)$ random variables, and thus, using arguments as in the case of θ, we have:

$$p(\phi) = \frac{1}{\pi}, \phi \in [0, \pi]\,. \tag{A-10}$$

Note that the orthogonality restriction imposes the constraint that θ and ϕ are not independent of each other. With the probability distribution of θ and ϕ established we can now establish the fact that randomising the integration rule gives us the integral we are trying to calculate. The expectation over all random matrices can be written as follows:

$$\begin{aligned}
&= \sum_{i=1}^{i=4} w_i E(f(Qz_i)) \\
&= w_1 E(f(Q_1)) + w_2 E(f(-Q_1)) + w_1 E(f(Q_2)) + w_2 E(f(-Q_2)) \\
&= \frac{\pi}{2}(E(f(Q_1)) + E(f(-Q_1)) + E(f(Q_2)) + E(f(-Q_2))) \\
&= \pi((E(f(Q_1)) + E(f(-Q_1))).
\end{aligned} \tag{A-11}$$

In the above sequence of equations we have made use of the fact that all the coefficients of the basic simplex rule (with each z_i being a unit vector in a coordinate direction) we are using are equal ($= \frac{\text{Surface Area}}{2n} = \frac{2\pi}{4}$), and the fact that the expectation over the second column of the matrix Q is the same as the expectation over the first column, so that it suffices to consider just the first column and double the result.

Expanding the last line of (A-11) we have

$$\begin{aligned}
Exp &= \pi((E(f(Q_1)) + E(f(-Q_1))) \\
&= \pi\Big(\int_0^{\pi} f(\cos\theta, \sin\theta) p(\theta) d\theta + \int_0^{\pi} f(-\cos\theta, -\sin\theta) p(\theta) d\theta\Big) \\
&= \int_0^{\pi} f(\cos\theta, \sin\theta) d\theta + \int_0^{\pi} f(-\cos\theta, -\sin\theta) d\theta.
\end{aligned} \tag{A-12}$$

In (A-12), we have substituted for $p(\theta)$ from (A-8). Making a change of variables in the last line of (A-12) (i.e., substituting $\alpha = \theta + \pi$ in the second integral), we have

$$Exp = \int_0^{\pi} f(\cos\theta, sin\theta) d\alpha + \int_{\pi}^{2\pi} f(\cos\alpha, \sin\alpha) d\alpha. \tag{A-13}$$

The surface integral can be written as

$$\begin{aligned} S &= \int_0^{2\pi} f(u)d\sigma \\ &= \int_0^{2\pi} f(\cos\alpha, \sin\alpha)d\alpha. \end{aligned} \tag{A-14}$$

In (A-14), we have made the substitution $u_1 = \cos\alpha, u_2 = \sin\alpha$ with $d\sigma = d\alpha$. Comparing (A-13) with (A-14), we see that they are the same.

(b): **n-dimensions,** $n \geq 3$: As might be expected, the n-dimensional (for $n \geq 3$) case is much more complicated. We shall show that the first column of Q in 3-dimensions is uniformly distributed over the surface of the unit sphere in 3-dimensions. A similar argument will establish that the first column of Q in n-dimensions is uniformly distributed over the surface of the unit n-sphere. Then we shall show the other columns of Q are distributed according as the first column of Q. Thus, to calculate the expectation, it will suffice to consider the first column of Q and multiply the result by n.

Let us write the first column of Q in 3-dimensions as follows:

$$\begin{aligned} Q_1 &= \begin{bmatrix} \sin\theta_3 \\ \sin\theta_2\cos\theta_3 \\ \cos\theta_2\cos\theta_3 \end{bmatrix} \\ &= \begin{bmatrix} \frac{x_{11}}{r_{11}} \\ \frac{x_{21}}{r_{11}} \\ \frac{x_{31}}{r_{11}} \end{bmatrix}. \end{aligned} \tag{A-15}$$

The above equation leads us to form the following two equations:

$$\begin{aligned} \tan\theta_2 &= \frac{x_{21}}{x_{31}}, \\ \tan\theta_3 &= \frac{x_{11}}{\sqrt{x_{21}^2 + x_{31}^2}}. \end{aligned} \tag{A-16}$$

For reasons that will be clear shortly, let us define an auxiliary variable $\theta_1 = x_{11}$. The joint probability density function of $\theta_1, \theta_2, \theta_3$ can be computed as follows:

$$p(\theta_1, \theta_2, \theta_3) = \frac{f(x_{11}, x_{21}, x_{31})}{J\left(\frac{\theta_1,\theta_2,\theta_3}{x_{11},x_{21},x_{31}}\right)}. \tag{A-17}$$

The auxiliary variable θ_1 was defined so that we could define the Jacobian of the transformation from the space of x_{11}, x_{21}, x_{31} to the space $\theta_1, \theta_2, \theta_3$. We can write the Jacobian as follows:

$$\begin{aligned} J\left(\frac{\theta_1,\theta_2,\theta_3}{x_{11},x_{21},x_{31}}\right) &= \begin{vmatrix} \frac{\partial\theta_1}{\partial x_{11}} & \frac{\partial\theta_1}{\partial x_{21}} & \frac{\partial\theta_1}{\partial x_{31}} \\ \frac{\partial\theta_2}{\partial x_{11}} & \frac{\partial\theta_2}{\partial x_{21}} & \frac{\partial\theta_2}{\partial x_{31}} \\ \frac{\partial\theta_3}{\partial x_{11}} & \frac{\partial\theta_3}{\partial x_{21}} & \frac{\partial\theta_3}{\partial x_{31}} \end{vmatrix} \\ &= \begin{vmatrix} 1 & 0 & 0 \\ * & \frac{x_{31}}{x_{21}^2+x_{31}^2} & -\frac{x_{21}}{x_{21}^2+x_{31}^2} \\ * & -\frac{x_{11}x_{21}}{\sqrt{(x_{21}^2+x_{31}^2)(x_{11}^2+x_{21}^2+x_{31}^2)}} & -\frac{x_{11}x_{31}}{\sqrt{(x_{21}^2+x_{31}^2)(x_{11}^2+x_{21}^2+x_{31}^2)}} \end{vmatrix} \\ &= \frac{\sin^3\theta_3}{\theta_1^2\cos\theta_3}. \end{aligned} \tag{A-18}$$

Substituting for the Jacobian in (A-17) the probability density function $p(\theta_1, \theta_2, \theta_3)$ can be written as follows:

$$p(\theta_1,\theta_2,\theta_3) = f\left(\theta_1, \frac{\theta_1\sin\theta_2}{\tan\theta_3}, \frac{\theta_1\cos\theta_2}{\tan\theta_3}\right)\frac{\theta_1^2\cos\theta_3}{sin^3\theta_3}. \tag{A-19}$$

Now x_{11}, x_{21}, x_{31} are independent zero-mean, unit variance random variables. Therefore their joint density function can be expressed as

$$f_{x_{11},x_{21},x_{31}}(x_{11},x_{21},x_{31}) = \frac{1}{(2\pi)^{3/2}}e^{\frac{-(x_{11}^2+x_{21}^2+x_{31}^2)}{2}}. \tag{A-20}$$

Therefore we have

$$\begin{aligned} p(\theta_1,\theta_2,\theta_3) &= f(\theta_1, \frac{\theta_1\sin\theta_2}{\tan\theta_3}, \frac{\theta_1\cos\theta_2}{\tan\theta_3})\frac{\theta_1^2\cos\theta_3}{sin^3\theta_3} \\ &= \frac{1}{(2\pi)^{3/2}}\frac{\theta_1^2\cos\theta_3}{sin^3\theta_3}e^{-\frac{\theta_1^2\sec^2\theta_3}{2\tan^2\theta_3}}. \end{aligned} \tag{A-21}$$

In order to determine the density $p(\theta_2, \theta_3)$ which is the density we are interested in, we must integrate out the auxiliary random variable, θ_1. Doing so

gives us

$$\begin{aligned} p(\theta_2, \theta_3) &= \int_{-\infty}^{\infty} p(\theta_1, \theta_2, \theta_3) d\theta_1 \\ &= \int_{-\infty}^{\infty} \frac{1}{(2\pi)^{3/2}} \frac{\theta_1^2 \cos\theta_3}{sin^3\theta_3} e^{-\frac{\theta_1^2 \sec^2\theta_3}{2\tan^2\theta_3}} d\theta_1. \end{aligned} \tag{A-22}$$

The above integral is of the form $\int_{-\infty}^{\infty} p\theta^2 e^{-q\theta^2} d\theta$ where $p = \frac{1}{(2\pi)^{3/2}} \frac{\cos\theta_3}{\sin^3\theta_3}$ and $q = \frac{\sec^2\theta_3}{\tan^2\theta_3}$. We know that

$$\int_{-\infty}^{\infty} e^{\frac{-\theta^2}{2}} d\theta = \sqrt{2\pi}. \tag{A-23}$$

Thus we can show that (A-22) amounts to

$$\begin{aligned} p(\theta_2, \theta_3) &= \frac{p}{q^{3/2}} \sqrt{2\pi} \\ &= \frac{1}{2\pi} \cos\theta_3. \end{aligned} \tag{A-24}$$

Thus the first column is distributed according to the distribution given by the above equation. Note that "uniform distribution" as defined above is slightly counter-intuitive in that there is a dependence on θ_3. This is because as θ_3 increases from 0 to $\pi/2$, the strips of the surface of the sphere corresponding to θ_3 keep shrinking in area, requiring the offset of $\cos\theta_3$ in the probability density function.

The first and second columns of Q are orthogonal, and that the second column Q^2 can be obtained as follows:

$$Q_2 = \frac{1}{r}(X_2 - \frac{X_1^T X_2}{X_1^T X_1} X_1), \tag{A-25}$$

where r in the above equation is the length of the vector$X_2 - \frac{X_1^T X_2}{X_1^T X_1} X_1$. In other words, the second column of the orthogonal matrix is obtained by subtracting from the first column of X, the projection of the first column of X on the second column of X, and normalising it to 1.. The third column of Q is obtained by subtracting from X_3 its projection onto the plane containing X_1

and X_2. The distributions of the second and third column can be computed using the same principles as for the first column, namely by defining auxiliary variables and computing the relevant Jacobians, and finally integrating out the auxiliary variables to get the required joint probability density function of the angles of the column vector in question.

However, the fact that the matrix Q arising out of the QR factorisation of a matrix X whose entries are N(0,1) random variables, is a uniform random orthogonal matrix, is a result of the theory of compact topological groups. The argument below is adapted from [86] and is included here only for completeness. The group of random orthogonal matrices can be shown to be a compact topological group [86]. Every compact topological group has a unique normalised left-invariant measure μ [37] such that

$$\begin{aligned} \mu(G) &= 1, \\ \mu(HG) &= \mu(G), \end{aligned} \tag{A-26}$$

where G denotes the group of all orthogonal matrices of order n. The measure μ is a group-theoretic analogue of the uniform probability density function over real numbers. Given that $X = QR$, pre-multiplying both sides of the equation by the orthogonal matrix H gives $HX =$ HQR. Since X is a matrix of N(0,1) random variables, HX is also N(0,1). This can be seen as follows (for n=2):

$$\begin{aligned} Y &= HX, \\ \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} &= \begin{vmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{vmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \\ f_Y(y_1, y_2) &= \frac{f_X(x_1, x_2)}{J\left(\frac{y_1, y_2}{x_1, x_2}\right)}, \\ f_Y(y_1, y_2) &= \frac{f_X(h_{22}y_1 - h_{12}y_2, -h_{21}y_1 + h_{11}y_2)}{det(H)}, \\ f_Y(y_1, y_2 &= \frac{1}{2\pi} e^{\frac{-1}{2}((h_{22}y_1 - h_{12}y_2)^2 + (-h_{21}y_1 + h_{11}y_2)^2)}, \\ f_Y(y_1, y_2) &= \frac{1}{2\pi} e^{\frac{-1}{2}(h_{22}^2 y_1^2 + h_{12}^2 y_2^2 + h_{21}^2 y_1^2 + h_{11}^2 y_2)}, \\ f_Y(y_1, y_2) &= f_{y_1}(y_1) f_{y_2}(y_2). \end{aligned} \tag{A-27}$$

In the above sequence of equations the probability density of the variables y_1 and y_2 is first expressed in terms of the probability density function of the variables x_1 and x_2 via a Jacobian transformation. The joint probability density function of the variables y_1 and y_2 can expressed in separable form in the last equation, since the rows of H are independent causing the cross-terms to disappear. Finally each $y_i = \Sigma_{i=1}^{i=2} h_{ij} x_j$ is a N(0,1) variable since the norm of each row of H is equal to 1.

HQ is an orthogonal matrix just like Q, and is obtained by a QR factorisation of a N(0,1) matrix HX. Therefore HQ is distributed according to the same distribution as Q. Thus the distribution of Q is invariant under left translations, and hence Q must be distributed according to the unique left-invariant measure μ called the Haar measure. The unique normalised left-invariant measure of (A-26) can also be shown to be right-invariant [37] [86]. In particular, this means that QP has the same distribution as Q where P is a permutation matrix (which is orthogonal) that interchanges columns 1 and 2 of Q. Thus the first column of QP is uniformly distributed over the surface of the unit sphere, since the first column of Q is uniformly distributed. But the first column of QP is the second column of Q. Thus the first and second columns of Q are identically distributed. Since P can be any permutation matrix, all columns of Q are identically distributed.

We will now show that the expectation over the random matrices is equal to the surface integral as required by the statement of the theorem. The uniform density of the column vectors in cartesian coordinates translates to

$$p(x, y, z) = \frac{1}{4\pi}. \tag{A-28}$$

The expectation over all random matrices can be written as follows:

$$\begin{aligned} Exp &= \sum_{i=1}^{i=6} w_i E(f(Qz_i)) \\ &= w_1 E(f(Q^1)) + w_2 E(f(-Q^1)) + w_3 E(f(Q^2)) + \\ &\quad w_4 E(f(-Q^2)) + w_5 E(f(-Q^3)) + w_6 E(f(-Q^3)) \\ &= \frac{\pi}{2}(E(f(Q^1)) + E(f(-Q^1)) + E(f(Q^2)) \\ &\quad + E(f(-Q^2)) + E(f(Q^3)) + E(f(-Q^3))) \\ &= 3\big(\frac{4\pi}{6}\big)((E(f(Q^1)) + E(f(-Q^1))). \end{aligned} \tag{A-29}$$

In the above sequence of equations we have made use of the fact that all the coefficients of the basic simplex rule we are using are equal ($= \frac{\text{Surface Area}}{2n} = \frac{4\pi}{6}$), and the fact that the expectation over the second and third columns of the matrix Q is the same as the expectation over the first column, so that it suffices to consider just the first column and triple the result.

Finally we have

$$\begin{aligned} E(f(Q^1)) &= \int\int\int_R f(x,y,z)p(x,y,z)dxdydz \\ &= \frac{1}{4\pi}\int\int\int_R f(x,y,z)dxdydz, \\ E(f(-Q^1)) &= \int\int\int_R f(-x,-y,-z)p(x,y,z)dxdydz \\ &= \frac{1}{4\pi}\int\int\int_R f(u,v,w)dudvdw. \end{aligned} \tag{A-30}$$

From (A-29) and (A-30) we can see that the expectation over all random matrices is equal to the surface integral.

Bibliography

[1] E. Acar, S. Nassif, Y. Liu, L.T. Pileggi, Assessment of true worst case circuit performance under interconnect parameter variations, *IEEE International Symposium of Quality Electronic Design*, pages 431-436, March 2001.

[2] S.A. Aftab, M.A. Styblinski, A new efficient approach to statistical delay modeling of CMOS digital combinational circuits, *IEEE/ACM International Conference on Computer-Aided Design*, pages 200-203, November 1994.

[3] A. Agarwal, D. Blaauw, V. Zolotov, S. Vrudhula, Statistical timing analysis using bounds, *Design Automation and Test in Europe Conference and Exhibition*, pages 62-67, March 2003.

[4] K. Anstreicher, Improved complexity for maximum volume inscribed ellipsoids, manuscript 2001.

[5] K.J. Antreich and R.K. Koblitz, Design centering by yield prediction, *IEEE Transactions on Circuits and Systems*, Volume CAS-29, No. 2, pages 88-95,February 1982.

[6] F. Baccelli, A.J-Marie, Z. Liu, A survey on solution methods for task graph models, *Second QMIPS Workshop*, Go"tz Herzog, Rettelbach (Re'd), Arbeitsberichte der IMMD, 26, 14, Erlangen, March 1993.

[7] J.W. Bandler and H.L. Abdel-Malek Optimal centering, tolerancing and yield determination via updated approximations and cuts, *IEEE Transactions on Circuits and Systems*, Volume CAS-25, No. 10, pages 853-871 , October 1978.

[8] M. Berkelaar, Statistical timing analysis, a linear time method, *ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, pages 15-24, December 1997.

[9] D.S. Boning and S.R. Nassif, Models of process variations in device and interconnect, *Design of High Performance Microprocessor Circuits*, Eds: A. Chandrakasan, W. Bowhill and F. Fox, pages 98-115, 2000, IEEE Press.

[10] R.B. Brashear, N. Menezes, C. Oh, L.T. Pillage, M.R. Mercer, Predicting circuit performance using circuit-level statistical timing analysis, *Design Automation and Test in Europe*, pages 332-337, February 1994.

[11] R.K. Brayton, S.W. Director and G.D. Hachtel, Yield maximization and worst-case design with arbitrary statistical distributions, *IEEE Transactions on Circuits and Systems*, Vol. CAS-27, No. 9, pages 756-764, September 1980.

[12] R. K. Brayton, G.D. Hachtel and A.L. Sangiovanni-Vincentelli, A survey of optimization techniques for integrated-circuit design, *Proceedings of the IEEE*, Volume 69, pages 1334-1362, 1981.

[13] G. Chen, H. Onodera, and K. Tamaru, Timing and power optimization by gate sizing considering false path, *6th Great Lakes Symposium on VLSI*, pages 154-159, March 1996.

[14] D.I. Cheng, M.M. Sadowska, K.T. Cheng, Speeding up power estimation by topological analysis, *IEEE Custom Integrated Circuits Conference*, pages 623-626, 1995.

[15] B. Choi and D.M.H. Walker, Timing analysis of combinational circuits including capacitive coupling and statistical process variation, *18th IEEE VLSI Test Symposium*, pages 49-54, April 2000.

[16] J. Cohen and T. Hickey, Two algorithms for determining volumes of convex polyhedra, *Journal of the ACM*, vol. 26, pages 401-414, July 1979.

[17] P.Cox, P. Yang, S.S. Mahant-Shetti, and P. Chatterjee, Statistical modeling for efficient parametric yield estimation of MOS VLSI circuits,

IEEE Transactions on Electron Devices, Vol ED-32, pages 471-478, February 1985.

[18] S. Devadas, Statistical timing analysis of combinational circuits, *IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pages 38-43, October 1992.

[19] M.E. Dyer and A.M. Frieze, On the complexity of computing the volume of a polyhedron, *SIAM Journal on Computing*, Vol. 17, No. 5, pages 967-974, 1988.

[20] A. Dharchoudhury and S.M. Kang, Worst-case analysis and optimization of VLSI circuit performances, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 14, No. 4, pages 481-492, April 1995.

[21] S.W. Director and W. Maly, eds, Statistical approach to VLSI, vol. 8 of *Advances in CAD for VLSI*, North-Holland 1994.

[22] S.W. Director, P. Feldmann, and K. Krishna, Optimization of parametric yield: a tutorial, *Custom and Integrated Circuits Conference*, pages 3.1.1-3.1.8, May 1992.

[23] S.W. Director, P. Feldmann, and K. Krishna, Statistical Integrated Circuit Design, *IEEE Journal on Solid-State Circuits*, Vol. 28, No. 3, pages 193-202, March 1993.

[24] S.W. Director and G.D. Hachtel, The simplicial approach to design centering, *IEEE Transactions on Circuits and Systems*, CAS-24, pages 363-372, July 1977.

[25] S.W. Director, G.D. Hachtel and L.M. Vidigal, Computationally efficient yield estimation procedures based on simplicial approximation, *IEEE Transactions on Circuits and Systems*, Vol. CAS-25, pages 121-130, March 1978.

[26] B. Dodin, Bounding the project completion time distribution in PERT networks, *Operations Research*, 33(4), pages 862-881, 1985.

[27] S.G. Duvall, Statistical circuit modeling and optimization, *5th International Workshop on Statistical Metrology*, pages 56-63, June 2000.

[28] P. Feldmann and S.W. Director, Accurate and efficient evaluation of circuit yield and yield gradients, *International Conference on Computer-Aided Design*, pages 120-123, November 1990.

[29] P. Feldmann and S.W. Director, Integrated circuit quality optimization using surface integrals, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 12, No. 12, pages 1868-1879, December 1993.

[30] A. Gattiker, S. Nassif, R. Dinakar, and C. Long, Timing yield estimation from static timing analysis, *International Symposium on Quality Electronic Design*, pages 437-442, March 2001 IEEE.

[31] A. Genz, Fully symmetric interpolatory rules for multiple integrals over hyper-spherical surfaces, *Journal of Computers and Applied Mathematics*, Vol. 157, pages 187-195, 2003.

[32] A. Genz and J. Monahan, Stochastic Integration Rules for Infinite Regions, *SIAM Journal on Scientific Computing*, Vol. 19, pages 426-439, 1998.

[33] A. Genz and J. Monahan, A stochastic algorithm for high dimensional integrals over unbounded regions with Gaussian weight, *Journal of Computers and Applied Mathematics*, Vol. 112, pages 71-81, 1999.

[34] D.S. Gibson, R. Poddar, G.S. May, Statistically based parametric yield prediction for integrated circuits, *IEEE Transactions on Semiconductor Manufacturing*, Vol. 10, No. 4, pages 445-458, November 1999.

[35] H.E. Graeb, C.U. Wiser, and K.J. Antreich, Circuit analysis and optimization driven by worst case distances, *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, Vol. 13, No. 1, pages 57-71, January 1994.

[36] J.N. Hagstrom, Computational complexity of PERT problems, *Networks*, Vol. 18, pages 139-147, 1988.

[37] P.R. Halmos, *Measure Theory*, Van Nostrand, 1976.

[38] J.M. Hammersley and D.C. Handscomb, *Monte-Carlo Methods*, Metheun and Co. limited, London, 1964.

[39] R.B. Hitchcock, Sr., Timing verification and the timing analysis program, *19th Design Automation Conference*, pages 594-604, 1982.

[40] R.B. Hitchcock, G.L.Smith and D.D. Cheng, Timing analysis of computer hardware, *IBM Journal of Research and Development*, pages 100-105, January 1982.

[41] D.E. Hocevar, P.F. Cox and P. Yang, Parametric yield optimization for MOS circuit blocks, *IEEE Transactions on Computer-Aided Design*, Vol. 7, No. 6, pages 645-658, June 1988.

[42] D.E. Hocevar, M.R. Lightner and T.N. Trick, A study of variance reduction techniques for estimating circuit yields, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-2, No.3, pages 180-191, July 1983.

[43] E.T.A.F. Jacobs, and M.R.C.M. Berkelaar, Gate sizing using a statistical delay model, *Design Automation And Test in Europe*, pages 283-291, March 2000.

[44] J.A.G. Jess, K. Kalafala, S.R. Naidu. R.H.J.M. Otten and C. Visweswariah, Statistical timing for parametric yield prediction of digital integrated circuits, *Design Automation Conference*, pages 932-937, June 2003.

[45] X. Jiang and S. Horiguchi, Statistical skew modeling for general clock distribution networks in presence of process variations, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 9, No. 5, pages 704-717, October 2001.

[46] B. Johnson, T. Quarles, A.R. Newton. D.O.Pederson, and A.Sangiovanni-Vincentelli, *SPICE3 Version 3f User's Manual*, University of California, Berkeley, 1992.

[47] H.F. Jyu, S. Malik, S. Devadas and K.Keutzer, Statistical timing analysis of combinational logic circuits, *IEEE Transactions on Very Large Scale Integration*, Vol. 1, No. 2, pages 126-137, June 1993.

[48] H.F. Jyu and S. Malik, Statistical timing optimization of combinational logic circuits, *IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pages 77-80, October 1993.

[49] J. Kamburowski, Bounding the distribution of project duration in PERT networks, *Operations Research Letters*, 12(1):pages 17-22, 1992.

[50] L. Khachiyan, A polynomial algorithm in linear programming, *Doklady Akademi Nauk SSSR*, 211:pages 1093-1096, 1979.

[51] L. Khachiyan and M. Todd, On the complexity of approximating the maximal inscribed ellipsoid for a polytope, *Mathematical Programming*, 61:pages 137-159, 1993.

[52] G.B. Kleindorfer, Bounding distributions for stochastic acyclic networks, *Operations Research*, 19:pages 1586-1601, 1971.

[53] R.B.Lin and M.C Wu, A new statistical approach to timing analysis of VLSI circuits, *11th International Conference on VLSI Design*, pages 507-513, January 1998.

[54] J. -J. Liou, A. Krstic L.-C Wang, K.-T. Cheng, False-path-aware statistical timing analysis and efficient path selection for delay testing and timing validation, *Proceedings of the Design Automation Conference*, pages 566-569, June 2002.

[55] J.-J. Liou, K.-T. Cheng, S. Kundu, A. Krstic, Fast statistical timing analysis by probabilistic event propagation, *Design Automation Conference*, pages 661-666, June 2001.

[56] Y. Liu, S.R. Nassif, L.T. Pileggi, A.J. Strojwas, Impact of interconnect variations on the clock skew of a gigahertz microprocessor, *Design Automation Conference*, pages 168-171, June 2000.

[57] P.C.K. Liu and K.C. Li, A circuit design methodology based on statistical tolerance optimization, *International Conference on Circuits and Systems*, pages 753-756, June 1991.

[58] A. Ludwig, R.H. Mohring, and F. Stork, A computational study on bounding the makespan distribution in stochastic project networks, *Technical Report No. 609/1998*, Department of Mathematics, Technical University of Berlin.

[59] E. Malavasi, S. Zanella, J. Uschersohn, M. Misheloff, C. Guardiani, Impact analysis of process variability on digital circuits with performance

limited yield, *IEEE International Workshop on Statistical Methodology*, pages 60-63, Kyoto ,Japan, June 2001.

[60] H.A.-Malek, and J.W. Bandler, Yield optimization for arbitrary statistical distributions, part 1- theory part 2 - implementation, *IEEE Transactions on Circuits and Systems*, Vol. CAS-27, No. 4, pages 245-262, April 1980.

[61] W. Maly, A.J. Strojwas and S.W. Director, VLSI yield prediction and estimation: a unified framework, *IEEE Transactions on Computer-Aided Design*, Vol. CAD-5, No. 1, pages 114-130, January 1986.

[62] M. Miyama, S. Kamohara, K. Okuyama, Y. Oji, Parametric yield enhancement system via circuit level device optimization using statistical circuit simulation, *Symposium on VLSI Circuits*, pages 163-166, June 2001.

[63] I.P. Mysovskikh, The approximation of multiple integrals by using interpolatory cubature formulae, in *Qualitative Approximation* (R.A. DeVore, and K.Scherer, eds.) (New York) pp 217-243, Academic Press, 1980.

[64] I.P. Mysovskikh, *Interpolatory Cubature Formulas*, Moscow-Leningrad, Izd 'Nauka', 1981(Russian text).

[65] A. Nadas, Probabilistic PERT, *IBM Journal of Research and Development*, Vol. 23, No. 3, pages 339-347, May 1979.

[66] S.R. Naidu, An impulse-train approach to statistical timing analysis, *Workshop Notes of the International Workshop on Logic Synthesis*, June 12-15, 2001, Granlibakken, CA.

[67] S.R. Naidu, Timing yield calculation using an impulse-train approach, *Proceedings of the Asia-South Pacific Design Automation Conference/International Conference on VLSI Design*, pages 219-224, January 2002.

[68] A. Nardi, A. Neviani, E. Zanoni, M. Quarantelli, and C. Guardiani, Impact of unrealistic worst-case modelling on the performance of VLSI circuits in deep submicron CMOS technologies, *IEEE Transactions on*

Semiconductor Manufacturing, Vol. 12, No. 4, pages 396-402, November 1999.

[69] S. R. Nassif, Within-chip variability analysis, *Proceedings of IEDM*, pages 283-286, December 1998.

[70] S.R. Nassif, Modeling and forecasting of manufacturing variations, *Asia South-Pacific Design Automation Conference*, pages 145-149, January 2001.

[71] S.R. Nassif, Design for variability in DSM technologies, *International Symposium on Quality Electronic Design*, pages 451-454, March 2000.

[72] S.R. Nassif, A. Strojwas, and S. Director, FABRICS-II: A statistically based IC fabrication process simulator, *IEEE Transactions on Computer-Aided Design*, Vol. CAD-3, No. 1, pages 40-46, January 1984.

[73] Y.E. Nesterov and A.S. Nemirovskii, *Interior Point Methods in Convex Programming - Theory and Applications*, The Society for Industrial and Applied Mathematics, Philadelphia 1991.

[74] M. Orshansky, J.C. Chen, and C. Hu, Direct sampling methodology for statistical analysis of scaled CMOS technologies, *IEEE Transactions on Semiconductor Manufacturing*, Vol. 12, No. 4, pages 403-408, November 1999.

[75] M. Orshansky, K. Keutzer, A general probabilistic framework for worst-case timing analysis, *Design Automation Conference*, pages 556-561, June 2002.

[76] M. Orshansky, L. Milor, P.Chen , K.Keutzer and C.Hu, Impact of spatial intra-chip gate length variability on the performance of high-Speed digital circuits, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 21, No. 5, pages 544-553, May 2002.

[77] S.W. Pan, and Y.H. Hu, PYFS - a statistical optimization method for integrated circuit yield enhancement, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Volume 2, Issue 2, pages 296-309, February 1993.

[78] A. Papoulis, *Probability and Statistics*, Prentice-Hall International, Englewood Cliffs, 1990.

[79] J.A. Power, B. Donnellan, A. Mathewson, and W.A. Lane, Relating statistical MOSFET model parameter variabilities to IC manufacturing process fluctuations enabling realistic worst case design, *IEEE Transactions on Semiconductor Manufacturing*, Vol. 7, No. 3, pages 306-318, August 1994.

[80] L. Scheffer, Cadence Design Systems, San Jose, California, private communication, March 2004.

[81] S.C. Seth, L. Pan, V.D. Agrawal, PREDICT: probabilistic estimation of digital circuit testability, *Proceedings of the International Symposium on Fault Tolerant Computing*, pages 220-225, June 1985.

[82] A.W. Shogan, Bounding distributions for a stochastic PERT network, *Networks*, 7:pages 359-381, 1977.

[83] K. Singhal and V. Visvanathan, Statistical device models from worst case files and electrical test data, *IEEE Transactions on Semiconductor Manufacturing*, Vol. 12, No. 4, pages 470-484, November 1999.

[84] K. Singhal and J.F. Pinel, Statistical design centering and tolerance using parametric sampling, *IEEE Transactions on Circuits and Systems*, vol. CAS-28, No.7 , pages 692-701, August 1985.

[85] M. Sivaraman and A.J. Strojwas, Delay fault coverage: A realistic metric and an estimation technique for distributed path delay faults, *International Conference on Computer-Aided Design*, pages 494-501, 1996.

[86] G. Stewart, The efficient generation of random orthogonal matrices with an application to condition estimators, *SIAM Journal on Numerical Analysis*, 17(3):pages 403-409, 1980.

[87] D. Stoyan, *Comparison Methods for Queues and other Stochastic Models*, Wiley, New York, 1984.

[88] A.H. Stroud, *Approximate Calculation of Multiple Integrals*, Prentice-Hall Inc, Englewood Cliffs, NJ, 1971.

[89] M.A. Styblinski, P. Gager and J. Lei, A symbolic fuzzy number approach to the propagation of uncertain statistical information in IC Design, *International Symposium on Circuits and Systems*, Volume 3, pages 1664-1667, June 1997.

[90] N. Telang, and J.M. Higman, Statistical modeling techniques: FPV vs BPV, *IEEE 2001 International Conference on Microelectronic Test Structures*, Vol 14, pages 71-75, March 2001.

[91] S. Tsukiyama, M. Tanaka, and M. Fukui, Techniques to remove false paths in statistical static timing analysis, *4th International Conference on ASIC*, pages 39-44, October 2001.

[92] S. Tsukiyama, M. Tanaka, M. Fukui, A statistical static timing analysis considering correlations between delays, *Proceedings of the Asia South Pacific Design Automation Conference*, pages 353-358, January 2001.

[93] L. Vandenberghe, S. Boyd and S.P. Wu, Determinant maximization with linear inequality constraints, *SIAM Journal on Matrix Analysis and Applications*, 19(2):pages 499-533, 1998.

[94] C. Visweswariah, IBM T.J. Watson Research Center, Yorktown Heights, New York, private communication, March 2004.

[95] J.M. Wojciechowski and J. Vlach, Ellipsoidal method for Design Centering and Yield Estimation, *IEEE Transactions on Computer-Aided Design of ICs and Systems*, Vol. 12, No. 10, pages 1570-1579, 1993.

[96] P. Yang, D.E. Hocevar, P.F. Cox, C. Machala, and P.K. Chatterjee, An integrated and efficient approach to MOS VLSI statistical circuit design, *IEEE Transactions on Computer-Aided Design of ICs and Systems*, Vol. CAD-5, pages 5-14, January 1986.

[97] H.C. Yen, S. Ghanta and H.C.Du, A path selection algorithm for timing analysis, *25th ACM/IEEE Design Automation Conference*, pages 720-723, June 1988.

[98] S.H.C. Yen, D.C. Du, and S. Ghanta, Efficient Algorithms for extracting the K most critical paths in timing analysis, *26th ACM/IEEE Design Automation Conference*, pages 649-654, June 1989.

[99] S. Zanella, A. Neviani, B. Franzimi, C. Guardiani, Statistical timing models of digital IP libraries, *5th International Workshop on Statistical Metrology*, pages 76-79, June 2000.

[100] Y. Zhang, On numerical solution of the maximum volume ellipsoid problem, *CAAM Technical Report TR01-15*, Department of Computational and Applied Mechanics, Rice University, August 2001.

Index

Biography

Srinath R. Naidu was born on 2nd June 1974, in Bangalore, India.

He completed his schooling in Bangalore, India in 1992 before starting his undergraduate education in Computer Engineering at the Banaras Hindu University, Varanasi. After receiving his undergraduate degree with honours in 1996, he proceeded to the Indian Institute of Science, Bangalore to pursue a master's degree in Computer Science. He graduated in 1998, with a dissertation entitled "Polynomial-time testable Combinational Circuits". Then he worked briefly for Synopsys(India).

From September 28, 1999 to January 24, 2004, Srinath was employed as an AiO at the Department of Electrical Eindhoven University of Technology. Upon finishing his AiO contract, Srinath has started working for Magma Design Automation, Netherlands. He hopes to be able to defend his thesis on July 13, 2004.