EFFICIENT MONTE CARLO BASED METHODS FOR VARIABILITY AWARE ANALYSIS AND OPTIMIZATION OF DIGITAL CIRCUITS

by

Vineeth Thazhathu Veetil

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering)
in The University of Michigan
2010

Doctoral Committee:

Associate Professor Dennis Michael Sylvester, Chair
Professor David Blaauw
Professor Marios C. Papaefthymiou
Professor Romesh Saigal

ACKNOWLEDGEMENTS

I am deeply indebted to Prof. Dennis Sylvester and Prof. David Blaauw for their excellent guidance throughout my PhD. Their insights and timely suggestions on numerous occasions have steered many of these projects in the right direction. I have truly learned a lot working with them. I would also like to thank Prof Marios Papaefthymiou for several discussions regarding the project and otherwise.

I am also very thankful to my family - father, mother, Anoop, Deepa and Smitha for their support throughout. Of course, Sharon for her encouragement and company in difficult times. I am grateful to my labmate Yung-Hsu Chang who played a major role in the completion of the project on acceleration of statistical timing analysis using Graphics Processing Units. Also thanks to Kaviraj Chopra, Sanjay Pant, Ravikishore Gandikota, Vivek Joshi, Cheng Zhuo, Mingoo Seok, Ashish Srivastava and Shantanu Gupta for their help and inputs on various occasions. Special thanks to Brian Cline, Matt Fojtik and Dave Fick for always being available to help with computing infrastructure issues.

I also gratefully acknowledge Nvidia Corporation for their donation of the Tesla GPU sytem.

TABLE OF CONTENTS

Acknowledgements ... ii

List of Tables ... vi

List of Figures ... vii

CHAPTER I: Introduction ... 1

I.1. Timing Analysis and Optimization ... 3

I.1.1. Variability Aware Timing Analysis ... 4

I.1.2. Variability Aware Optimization ... 7

I.1.3. Gate Delay Modeling ... 8

I.2. Standby Power: Leakage Analysis ... 10

I.3. Overview of Monte Carlo Variance Reduction ... 12

I.3.1. Quasi Monte Carlo ... 13

I.3.2. Stratified sampling ... 15

I.3.3. Latin Hypercube Sampling ... 16

I.4. Thesis Overview and Key Contributions ... 17

CHAPTER II: Efficient Monte Carlo based Incremental Statistical Timing Analysis ... 20

II.1. Introduction ... 20

II.2. Smart sampling based on timing criticality ... 23

II.2.1. Process variation model ... 23

II.2.2. Stratification+Hybrid Quasi Monte Carlo(SH-QMC) ... 24

II.2.3. Timing criticality Pcrit ... 26

II.3. Pruning based algorithm for timing analysis ... 27

II.4. Incremental Evaluation of a Percentile Delay ... 29

II.4.1. Algorithm.......30
II.4.2. Computing circuit arrival time bound for samples.......32
II.5. Results.......34
II.6. Summary.......41
CHAPTER III: Efficient Smart Monte Carlo based SSTA on Graphics Processing Units with Improved Resource Utilization.......43
III.1. Introduction.......43
III.2. CUDA Platform.......46
III.3. Smart Sampling based SSTA: SH-QMC.......47
III.4. Monte Carlo based Statistical Timing Analysis on GPUs.......48
III.5. Enhanced resource utilization for implementation of SH-QMC on GPU 48
III.6. Critical Graph Analysis for MC SSTA.......54
III.6.1. Nominal STA based Critical Graph Identification.......54
III.7. Results.......55
III.8. Conclusions.......62
CHAPTER IV: A Lower Bound Computation Method for Evaluation of Statistical Design Techniques.......64
IV.1. Introduction.......64
IV.2. Exact Deterministic Optimization.......67
IV.3. Lower Bound For Design With Statistical Timing Yield Constraint.......71
IV.4. Sample Level Optimization in Parallel (SLOP).......75
IV.5. Results.......77
IV.6. Conclusions.......81
CHAPTER V: Efficient Smart Sampling based Full-Chip Leakage Analysis for Intra-Die Variation Considering State Dependence.......82
V.1. Introduction.......82
V.2. Smart Sampling for Leakage Analysis.......84
V.2.1. Quasi Monte Carlo.......85
V.3. Leakage Analysis for Inter-Die Variation with Smart Sampling.......86

V.3.1. Process Variation Model 86
V.3.2. Traditional Leakage Analysis Flow for Inter-Die Variation 87
V.3.3. Proposed Leakage Analysis Flow with Smart Sampling 88
V.4. Leakage Analysis for Total Variation with Smart Sampling 89
V.5. Results 95
V.6. Conclusions 100
CHAPTER VI:Fast And Accurate Waveform Analysis With Current Source Models 102
VI.1. Introduction 102
VI.2. Precharacterization 104
VI.2.1. Bicubic Spline based DC Current Source Model 104
VI.2.2. Modeling Parasitics 106
VI.3. Weibull-based Runtime Engine 108
VI.3.1. Basic Concept and Flow 108
VI.3.2. Enhancing Accuracy 111
VI.4. Results 113
VI.5. Conclusions 115
CHAPTER VII:Conclusions and Future Work 118
VII.1. Smart Monte Carlo SSTA : SH-QMC 118
VII.2. Acceleration of SH-QMC on Graphics Processing Units (GPUs) 119
VII.3. Comparison of Statistical Design Optimization Techniques 119
VII.4. Smart Sampling based approach for full-chip leakage analysis 120
VII.5. Future Work 121
Bibliography 124

LIST OF TABLES

Table II.1 Comparison of random sampling, LHS based and SH-QMC approaches based on sample size. The last two columns show the speedup of LHS and SH-QMC respectively, over random sampling. 34
Table II.2 Runtime comparison of SHQMC with SSTA. AT = circuit delay 36
Table II.3 Comparison of three SH-QMC-based approaches with no pruning, single stage pruning and double stage pruning on benchmark circuits. 39
Table II.4 Performance of incremental evaluation of 95th and 99th percentile delay with gate size change for SH-QMC with 80 samples. AT=Arrival Time 42
Table III.1 Comparison of runtime for SH-QMC (192 samples) vs. random sampling based MC SSTA on GPU. SH-QMC is implemented with (a) sample level parallelism or SP (b) sample + gate parallelism or SGP. (c) SGP + efficient shared memory usage or SGP+S.Mem. 56
Table III.2 Comparison of runtime for SH-QMC (192 samples) on GPU vs. CPU. and single STA on CPU. The CPU is a 3.16GHz Intel Xeon processor. 59
Table III.3 Quality of results for the critical graph analysis technique. 60
Table III.4 Runtime improvement from graph reduction combined with the proposed technique. 61
Table IV.1 Comparison of Burns, RGP and SLOP approaches against the lower bound for area at benchmark circuits. RGP is implemented on a CPU. Burns and SLOP are implemented on a CPU with a GPU co-processor, to utilize the parallelism available in the algorithms 80
Table V.1 Comparison of proposed approach with Golden (Monte Carlo 20,000 samples) for benchmarks. * indicates that state probability is considered for instances in the circuit. 98
Table V.2 Comparison of proposed approach with Wilkinson's based approach. * indicates that state probability information is considered for instances in the circuit. 100
Table VI.1 Comparison of stagewise timing analysis for a bicubic spline fit (spline) vs. a fourth order polynomial fit (Poly) for standard cells in an industrial 90nm library. 106
Table VI.2 Error statistics compared to SPICE of delay and slew for proposed and traditional techniques for various benchmark circuits. 113

LIST OF FIGURES

Figure I.1 Gordon Moore’s original graph from 1965. Source: http://www.intel.com/technology/mooreslaw/ 1

Figure I.2 Resolution enhancements in photolithography have stalled due to difficulties associated with EUV (Extreme Ultravioloet lithography). This is a source of variation as transistor geometries shrink at advanced technology nodes. Source: Mark Bohr, Intel 2

Figure I.3 Smart sampling techniques for SSTA can be parallelized on GPUs to achieve significant speed ups in statistical timing analysis 6

Figure I.4 Quasi random and pseudo random sequences. 13

Figure I.5 Stratification of a 2D space. Variable X is divided into 4 bins, thus dividing the sample space into 4 strata. 15

Figure I.6 Latin Hypercube Sampling (a) Divide each variable in 8 equal probability bins and sample in bins. (b) Combine randomly to form 8 triplets 16

Figure II.1 Ordering variables using timing criticality. 23

Figure II.2 Stratified Latin Hypercube Sampling (a) Ordering variables based on timing criticality. (b) One of 16 strata in the sample space. (c) QMC triplets and LHS pairs. (d) These are combined to obtain final samples. .. 24

Figure II.3 Slack distribution of gates ‘g1’ and ‘g2’ obtained by evaluation of minimal SH-QMC set. The threshold percentile for each distribution is plotted as a dotted line. ‘g2’ is pruned in this case as the criterion of positive slack percentile is satisfied. 29

Figure II.4 (a) Samples are visited in decreasing order of circuit arrival time, starting from the xth percentile (tx). Samples with delta crossing tx are selected, others pruned. (b) Recomputation of circuit arrival time is performed at the selected sample and tx is updated. 30

Figure II.5 Error comparison of random sampling, LHS and SH-QMC for a VGA circuit (90831 gates) w.r.t. golden of MC count 40,000. 34

Figure II.6 Performance comparison of traditional SSTA with multi-threaded SH-QMC for VGA circuit (90831 gates) as function of number of grids in process variation model. 37

Figure II.7 Comparison of 99th percentile error of SH-QMC vs. traditional SSTA w.r.t golden of 40,000 MC count for USB circuit. 38

Figure II.8 Arrival time distribution of SH-QMC (96 samples) and traditional SSTA w.r.t golden(40,000 MC) for USB circuit. 39

Figure II.9 Comparison of the 95th percentile error in s for single stage and two stage pruning for a large benchmark circuit. 40
Figure II.10 Comparison of runtime vs. pruning parameter for single stage and two stage pruning for a large benchmark circuit. 40
Figure III.1 Gate scheduling. Gates in a sample with no dependence are computed in parallel. In graph shown, gates g1,g2,g3 have no dependence and can be assigned to the same level. However, g3 is a 2-input gate which if assigned to the same level as g1 and g2, increases the computational steps in the level. Therefore, g3 is assigned to the next level along with gate g4. 50
Figure III.2 Algorithm 1 and Algorithm 2 for gate scheduling. 51
Figure III.3 Summary of proposed approaches to improve resource utilization. Concurrent computation on gates in the same level and use of shared memory are illustrated. 53
Figure III.4 Illustration of graph reduction. Slacks for nodes are indicated next to corresponding gates. Gates with slacks higher than a threshold of 0.3 at output node are removed to obtain the reduced graph in the example. 55
Figure III.5 Improvement in runtime due to the techniques proposed for improved resource utilization. 58
Figure III.6 Comparison of runtime for SH-QMC performed with successive gate sizing (100,000 sizing steps) on 4 GPU cards for Ether circuit. Runtime for an implementation utilizing sample level parallelism (SP) is compared with the proposed approach (SGP+S.Mem). 60
Figure III.7 SH-QMC with 192 samples on GPU is compared with SH-QMC on CPU and STA on CPU, on a logarithmic scale, for Ethernet circuit with 57327 gates. SH-QMC on multi-GPU is faster than STA on CPU. 61
Figure III.8 Speedup of SH-QMC algorithm implemented on multi-GPU for a USB circuit (14,503 gates) with smart scheduling algorithm Algorithm 2. X axis indicates the maximum number of gates computed in parallel. A discontinuity in resource requirements above a parallelism of 64 leads to the discontinuity in the graph. 62
Figure IV.1 Illustration of Theorem 1. Ai represents the optimal cost solution at sample si. Design D meets timing constraint T for the 75th percentile of the given sample set. The cost of this design is A(D) and exceeds Ai, i=1,…,3 as D meets the timing constraint at samples si, i=1,…,3. Therefore, A(D) exceeds the 75th percentile of the distribution Ai, i=1,…,4 for the given sample set. 72
Figure IV.2 SLOP overview: Each virtual die i is optimized in parallel. The optimization consists of an S-Phase and an HPS-Phase. In the S-Phase, the virtual die is optimized to meet the timing constraint T using a greedy approach. In the subsequent HPS-Phase, the xth percentile sample of the design i optimized in S Phase, called HPS(i), is selected. If T is already met at HPS(i), optimization terminates. Else, the design is optimized further, this time with sample HPS(i) constrained to meet a timing T. The best among parallel solutions is selected. 77

Figure IV.3 Comparison of sizing curves for Burns, RGP and SLOP against the lower bound for area computed at varying timing constraints for the 99% timing yield 80
Figure V.1 Traditional and proposed leakage analysis flow for global variation with multiple sources. 84
Figure V.2 Quasi random and pseudo random sequences. 87
Figure V.3 Proposed leakage analysis flow for within-die variation. 88
Figure V.4 Reusing samples for local distribution computation. Inter-die samples weigh the samples in total variation space according to local probability distribution. 90
Figure V.5 (a) Total (Inter+Intra die) distribution and local pdf at an inter-die sample. (b) QMC based samples are generated according to total variation. (c) For computing mean of local pdf the samples generated in (b) are weighed according to the ratio of the probabilities in the two distribution functions. 94
Figure V.6 Comparison of sigma of leakage distribution without considering spatial correlation with that of a grid-based spatial correlation model for VGA circuit (43214 gates) 95
Figure V.7 Comparison of error in estimating s of leakage distribution for inter-die variation using QMC vs. random sampling for VGA circuit(43214 gates).. 96
Figure V.8 Comparison of accuracy of proposed approach with random sampling based approach vs. runtime. The circuit considered is Chip1 with 200,000 gates. 97
Figure V.9 Total leakage distribution considering intra-die variation for Chip1 (200,000 gates). Proposed approach is compared with the analytical approach based on [7] and the golden. The distribution due to inter-die variation is also plotted. The analysis considers state dependence of leakage for the instances in the circuit. 101
Figure VI.1 Polynomial-based fitting model error in current relative to SPICE as a function of input and output voltages. (b) Spline-based fitting model error in current relative to SPICE. 105
Figure VI.2 Schematic of the proposed modified Blade-based model 109
Figure VI.3 Typical waveforms for source and load currents as in the proposed model for an output falling case. The difference is the error function (referred to as f (t)). 110
Figure VI.4 (a)% error in slew (b) Absolute error in gate delay (c) Absolute error in arrival time. 116
Figure VI.5 Error histograms for slew estimations in two large ISCAS85 circuits given a primary input excitation 117

CHAPTER I

INTRODUCTION

The number of transistors that can be packed together in semiconductor chips have increased exponentially over the past 50 years. This behavior was first predicted famously by Gordon Moore in 1965 as illustrated in Figure I.1. Moore's law predicts that the number of transistors in semiconductor chips doubles every two years. The trend in the semiconductor industry has been more or less consistent with this prediction. Modern chips contain up to a billion transistors. The dense packing of transistors in a chip is enabled by advances in process technology with the transistor device geometry shrinking at every new technology node. The spatial resolution of transistors can be as low as 32nm in the latest technologies. Thus today's technology enables man to produce powerful computing devices at massive scales by being to able to control the behavior of matter at the level of just a few atoms.

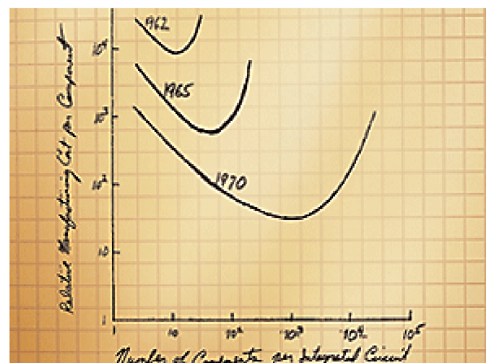


Figure I.1. Gordon Moore's original graph from 1965.
Source: http://www.intel.com/technology/mooreslaw/

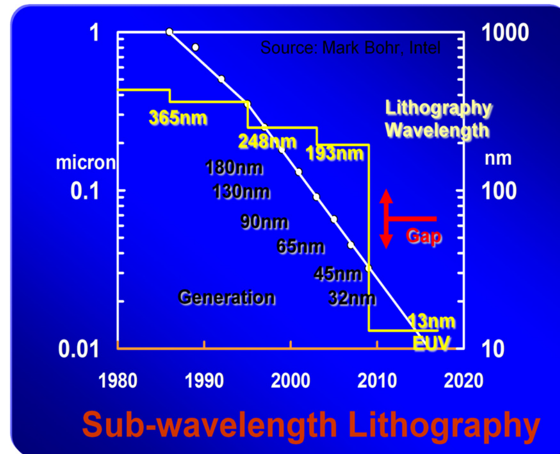


Figure I.2. Resolution enhancements in photolithography have stalled due to difficulties associated with EUV (Extreme Ultravioloet lithography). This is a source of variation as transistor geometries shrink at advanced technology nodes. Source: Mark Bohr, Intel

At the same time, these advances in technology are accompanied by novel challenges which are threatening to limit the pace at which we make progress in accordance with the Moore's law. Earlier transistors were much larger and their physical dimensions (e.g. length) and other properties were easier to control. However with shrinking sizes it has become increasingly difficult to control these properties/parameters. Termed process variation, this phenomenon implies that the behavior of a chip has an uncertainty related to the exact manufacturing process and is only known to within bounds during the design phase of the chip. This means that meeting the performance specification becomes more challenging. The complexity of process variations and the uncertainty is increasing with technology scaling . There are different sources for this process variation. Lithography is the process of using light to transfer a geometric pattern to the silicon substrate or thin films on the substrate. The resolution related to the wavelength of light leads to lithographic variation as illustrated in Figure I.2. Other sources of variation are related to the process of doping or intentional addition of impurities to silicon to achieve desired properties. The

final geometric pattern achieved on silicon or the layout affects the distribution of stress in silicon which leads to layout dependent stress variation in certain silicon technologies called strained silicon technologies. It is important to study and model the effect of these sources of variation to enable better knowledge of chip properties in the design phase [1-3]. Timing verification/analysis and power analysis are two key steps in assessing the quality of an integrated circuit design. In the optimization step, insights from timing analysis are used to tune the design to achieve, for example minimum area in silicon while meeting target timing constraints. These are briefly described in the following sections along with the impact of process variations on the accuracy of these analyses.

I.1 Timing Analysis and Optimization

The aim of timing verification of a chip is to make sure that a chip operates at a specified clock frequency with a desired yield under the specified range of operating conditions. A simple definition of yield is the fraction of chips manufactured which meet the performance specifications. Timing verification involves timing analysis of the network of logic gates in the circuit or the netlist (a description of a circuit in terms of electrical connections of gates). Timing analysis can be static or dynamic. Dynamic timing analysis involves propagation of vectors at the input ports of the network to the output ports and computing the timing behavior of the circuit. Static timing analysis does not consider individual input vector patterns and is therefore more conservative in the timing behavior estimate. However, due to the complexity of usage the standard methods used for processors and ASICs (Application Specific Integrated Circuits) do not involve dynamic timing analysis. Hence we only consider static timing analysis in this work.

The basic steps involved in a static timing analysis on a netlist are summarized here [4-5]

- Output pin timing constraints are determined
- Input pin arrival time is determined.
- Clocks in the netlist are identified.
- Timing constraints are generated at all circuit pins
- Minimum/maximum rising and falling signal timing are propagated from inputs to outputs. Signals may also be propagated from outputs to inputs for certain analyses.
- The delay of each circuit element (transistors, gates) are estimated in the above step.
- Considering clock uncertainty, check if all paths meet the timing constraints.
- If there are violations, these have to be fixed through changes in the netlist. Static timing analysis is performed on the modified netlist.

I.1.1 Variability Aware Timing Analysis

With increasing process parameter variations there was a need to incorporate uncertainty into timing analysis. A traditional conservative approach is to perform static timing analysis at multiple process conditions or "corners" for a given circuit. A process corner is a set of values assigned to all circuit parameters with the hope that some combination of the values assigned will elicit worst case performance for the circuit. Process corners are generated by assuming upper and lower bounds for each process parameter independently. The process corners of specific interest include the best, nominal and worst case corner.

For example, the worst case corner is the process corner where the maximum operating frequency of the circuit is the lowest. To find the best and worst corners, all the process corners need to be evaluated. However as the number of process variation sources increases, the number of candidate corners increases exponentially which makes this approach expensive. An alternate approach is to assume worst case behavior for each device which leads to large guard bands and loss of performance. Therefore, rather than using simple corner models, modern CAD tools are moving towards a more probabilistic view of circuit timing behavior. In replacing corner models, there are two primary approaches that incorporate process parameter uncertainty in timing analysis. The first is to perform statistical static timing analysis (SSTA) by modeling gate delay as a function of process parameters and propagating these distribution functions to compute the distribution of circuit delay [6-15]. We refer to these approaches as traditional SSTA. In traditional SSTA it has proven challenging to efficiently model factors such as “skewness” in the arrival time distribution. Skewness is a measure of deviation from an assumed standard fitting function, and can be attributed to factors including non-linear dependence of the gate delays on process variation. Also, a number of modeling issues are still in early stages of development, such as combined analysis of large interconnect structures driven by non-linear drivers, coupling events, and modeling of transparent latches. While some progress has been made in addressing these issues [6-15], it is expected that a fully mature traditional SSTA tool capable of performing timing sign-off may not be widely available in the near future. The second approach is Monte Carlo based SSTA, which involves selection of samples of the process variation space to obtain statistical distributions of circuit timing behavior. The application of Monte Carlo (MC) for statistical timing was dis-

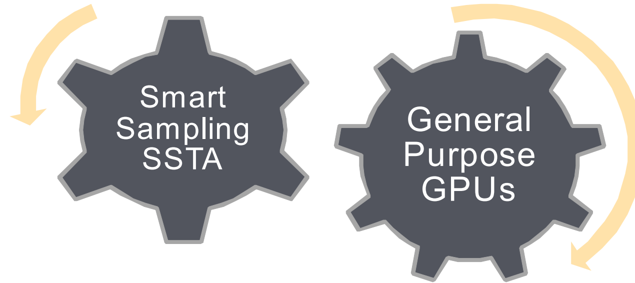


Figure I.3. Smart sampling techniques for SSTA can be parallelized on GPUs to achieve significant speed ups in statistical timing analysis

cussed in [16], where it was shown that Monte Carlo based SSTA is accurate even in scenarios with high dimensionality and non-standard distributions in the process variation space, where traditional SSTA has difficulties. However, there are two main difficulties with this approach. First, the standard MC approach of random selection of samples in the process variation space requires too many samples for sufficient accuracy, resulting in high runtime cost. Second, there is no work to show the applicability of MC based SSTA for incremental statistical timing analysis. We address both concerns in Chapter II.

As important as reducing the sample size for Monte Carlo based SSTA is to exploit the trivial parallelism in the algorithm by performing computations on parallel machines. Fortunately, recent years have seen the rapid scaling of throughput-optimized processors, such as Graphics Processing Units (GPUs). Modern GPUs deliver over 1 TeraFlops of computational power with more than 100 GB/second of memory bandwidth while conventional processors face difficulties with frequency scaling and are increasingly incorporating multiple cores on a chip to keep up with Moore's law. However, to exploit the benefits of throughput-optimized processors such as GPUs, applications need to be rede-

signed to achieve performance and efficiency. We present techniques to speed up statistical timing analysis on throughput processors in Chapter III. Our proposed smart sampling technique, Stratification + Hybrid Quasi Monte Carlo (SH-QMC), is implemented on a GPU based on NVIDIA CUDA architecture. We show that although this application is based on MC analysis with straightforward parallelism available, achieving performance and efficiency on the GPU requires exposing more parallelism and finding locality in computations. This is in contrast with random sampling based algorithms which are inefficient in terms of sample size but can keep resources utilized on a GPU. Results presented provide a compelling case for the adoption of GPUs for SSTA.

I.1.2 Variability Aware Optimization

Next, we focus on optimizing a given circuit design to minimize cost metrics such as area on silicon or power consumed by the chip while meeting a specified timing constraint. Optimization takes advantage of the fact that the same logic function can be performed by different implementations of logic gates which trade-off performance for area of power consumed. For example, standard cell libraries are available from vendors, where there is a choice of different gate sizes for a given logic gate. A higher gate size usually means that the gate incurs lower delay to propagate the logic, however with an area penalty or increase in power consumption. Timing optimization can be performed by choosing the appropriate gate type to perform each logic function such that timing constraints are met. However, variability in timing makes this more challenging as the timing constraints in the optimization problem are now probabilistic functions rather than deterministic. As discussed in the case of timing analysis, it is possible to select among differ-

ent process corners (generated by assuming upper and lower bounds for each process parameter independently) and optimize only at the worst case corner. However, increase in variability in the nanometer era has contributed to pessimistic guardbands for circuit design techniques that optimize at worst-case process corners. Smart deterministic approaches have been proposed that employ statistical timing analysis to reduce pessimism in the guardbands while retaining the deterministic nature of the algorithms. In other words, the optimization itself has deterministic objective function and constraints. However, the result obtained from the optimization is analyzed using an SSTA tool to check if the statistical objective is also met. If not, deterministic optimization continues till the statistical objective is attained. Other statistical optimization techniques focus on optimization algorithms which directly work with statistical objective and constraints, where clearly the computational cost is higher. It is not clear how much improvement can be gained using the latter set of approaches over smart deterministic approaches. This work presents a new lower bound to evaluate these statistical optimization techniques, drawing inspiration from recent advances in sampling based SSTA. We also compare several statistical design optimization approaches, including one proposed in this work called SLOP, against the computed lower bound. We show that the existing optimization methods have nearly exhausted the obtainable improvement from being statistically aware and mostly provide trade-offs in runtime speed.

I.1.3 Gate Delay Modeling

An important step in timing verification is to calculate the delay of each circuit element. Though we do not provide a detailed treatment of variability effects on gate delay

modeling in this work, we propose a technique to achieve increased accuracy in gate delay modeling compared to conventional techniques. This technique is not currently integrated with the rest of the methods developed in this work for timing analysis and optimization. However, with additional research to incorporate variation effects, this can potentially increase the efficiency of these other techniques further. Hence, we present a discussion of this approach in Chapter VI.

Gate delay depends on several variables, including input signal transition time and the characteristics of the load driven by the gate. Gates drive other gates and their input pin capacitance adds to the capacitive load at the driver gate. Gates may also drive long global wires on the chip with large capacitive and resistive components. Modeling of gate delays has become an important challenge in recent technologies. Traditional standard cell libraries (libraries with pre-designed logic gates to achieve different functionalities and drive strengths) have modeled logic gates as voltage sources based on a Thevenin model. Timing libraries provide data for each logic gate where the delay characteristics are precharacterized as a function of input signal transition time and a simple lumped output capacitive load model (without considering distributed capacitive or resistive effects). The output load of a gate is approximated to a single capacitance to make use of the information in the timing library. This information is used to model the gate itself as a voltage source with a resistance in series, using an iterative approach. The respective parameters for the voltage source and resistor are dependent on the lumped capacitance model. These models are inadequate to capture the timing behavior in modern nanometer scale CMOS. The lumping of load capacitances into an effective capacitance leads to

errors in timing analysis. Also, signal integrity issues require a level of accuracy in waveform shapes which cannot be achieved with these models.

Recently current source models (CSMs) have become popular for use in standard cell characterization and static timing analysis. However, there has not been any detailed study of what aspects of the gate parasitics and DC current source behavior should be modeled for sufficient accuracy, and there have been no results reported incorporating a CSM with the above complexity into a timing analysis flow with reasonable runtime. This work addresses these two limitations by investigating complexity/accuracy trade-offs in CSMs in Chapter VI. We then present a novel technique to perform fast, accurate waveform analysis using current source models. Timing analysis results on benchmark circuits show significantly reduced errors compared to a traditional Thevenin-based flow.

I.2 Standby Power: Leakage Analysis

Ideally the power consumption of chips with scaling should not be a significant problem. However, in reality the trends in increase in power consumption by digital circuits is alarming. This can be attributed to the fact that the while device geometry has been scaling consistently, the corresponding reduction in supply voltage has not been consistent. High power consumption increases the need for cooling of the chip and beyond a limit this can be very expensive. The power consumption of a chip can be broadly classified as dynamic power consumption and static or standby power consumption. Dynamic power consumption is due to charging and discharging of the nodes in the circuit. Static power consumption occurs when there is no switching activity in the circuit. The main component of static power is leakage power. Power analysis of a design involves estimating the dynamic and

static power consumption of a circuit. The static power consumption of a circuit increases exponentially with leakage current and this is a major concern with process scaling. In this research, we focus on the static component of power consumption.

As in the case of timing analysis, increasing process variation with scaling adds complexity to static power or leakage analysis. A promising solution is to perform statistical analysis of leakage and use this to guide leakage optimization and design changes. Analogous to the case of timing analysis, current approaches to calculate full-chip leakage power can be classified into two main categories. The first category of methods are analytical in nature. These attempt to model full chip leakage using a standard distribution, most commonly a lognormal distribution. The moments of this distribution are computed by matching moments with an expression involving summation of leakage distributions at the gate level [24-27]. In [24] a lognormal distribution is used to approximate the leakage current of each gate and the total leakage is obtained by summing the log normals. A low rank quadratic approximation to capture non-lognormal leakage distributions is proposed in [25]. It is noted that a 20% error is observed when modeling leakage distributions as purely lognormal using a linear approximation. The authors in [26] attempt to capture high level characteristics of a candidate chip design for early mode leakage estimation. In [27] the authors propose a systematic characterization of leakage related parameter variations. A quadratic model of the logarithm of leakage current is also proposed. Traditionally these approaches have provided the desired accuracy. However they make assumptions about either the nature of the statistical distribution of process variation parameters or the nature of the dependence of standard cell leakage on the underlying variables for handling process variation. The process variation parameters are assumed to have a standard distribu-

tion, most commonly Gaussian, or the logarithm of standard cell leakage is assumed to be a linear or quadratic sum of the variables modeling process variation. It is not clear that these assumptions will still hold true considering secondary effects in process variation and a growing number of variation sources at technology nodes below 45nm.

The second category of methods fall into the classification of Monte Carlo based techniques involving selection of samples in the process variation space and using these samples to compute leakage distribution. Monte Carlo techniques can handle non-standard distribution of process parameters and lookup tables for dependence of standard cell leakage on process variables. Therefore they do not require simplifying assumptions about the dependence of leakage on process parameters or the nature of process parameter distribution, making them highly scalable. Also the inherent parallelism in evaluating Monte Carlo samples make these techniques amenable to multi-core and Graphics Processing Unit (GPU) computing. However Monte Carlo techniques typically require a large sample size rendering them expensive. There is a need for smart selection of samples to reduce the number of samples that require evaluation without compromising accuracy. In [28] the author describes such techniques, known as variance reduction techniques. These techniques need to be tailored to the system under consideration for efficient reduction in sample size. In the context of integrated circuits it has been shown that a suitable choice of these techniques can lead to significant sample size reduction for statistical timing analysis[29].

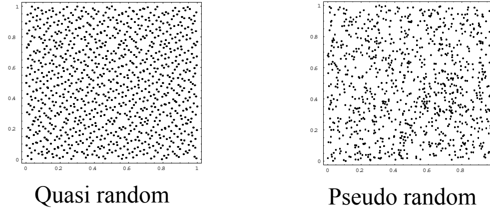


Figure I.4. Quasi random and pseudo random sequences.

I.3 Overview of Monte Carlo Variance Reduction

Since the major contributions in this work are based upon a Monte Carlo sampling perspective of the process parameter variation space, an overview of Monte Carlo sampling and variance reduction approaches is in place here. The standard Monte Carlo method addresses the problem of approximating the integral of a function *f(x)* over the s-dimensional hypercube $C^s = [0, 1)^s$, where *x* represents a point in an s-dimensional space. The MC estimate of the integral $\bar{f}$ is given by the arithmetic mean of f_i, which are values of the function *f(x)* evaluated at *n* samples distributed throughout the hypercube.

MC based statistical timing involves selecting samples of the process variation space to obtain statistical distributions of circuit delay. This is mapped to the standard mathematical problem of MC, which is to estimate the integral of a function, using samples in its domain. There are standard techniques for variance reduction of MC, which include Quasi Monte Carlo techniques, Latin Hypercube sampling, stratified sampling, importance sampling and control variates. In this section, we briefly discuss their applicability to digital circuit analysis.

I.3.1 Quasi Monte Carlo

The Koksma-Hlawka inequality relates the error bound of a method to numerically estimate an integral using a sequence of samples, to a mathematical measure of uniformity for the distribution of the points, called "discrepancy" [34]. This inequality suggests that we should use a sequence with the smallest possible discrepancy to evaluate the function in order to achieve the smallest possible error bound. Such sequences constructed to reduce discrepancy are called Low Discrepancy Sequences (LDSs). Quasi monte carlo techniques are characterized by their use of LDSs to generate samples. LDSs are deterministic sequences, in other words there is no randomness in their generation. Intuitively, these sequences are well dispersed through the domain of the function, minimizing any gaps and/or clustering of points. Figure I.4 illustrates that quasi random sequences generate samples with lower discrepancy compared to pseudo random sequences (sequences with properties similar to "truly" random sequences). Sobol[35], Faure and Niederreiter[33] are LDSs that have been studied extensively. In this work, we consider Sobol sequences, which are known to be simple to construct and more resistant to the *pattern dependency* issue (mentioned below), compared to the other sequences. Interested readers can refer to [35] for a construction of the Sobol sequence, and [36] for an implementation.

In the context of statistical timing analysis, Quasi Monte Carlo techniques have been studied in [33]. The author notes that LDSs are imperfect and as the number of dimensions in the problem increases, there is degraded uniformity. This effect is especially significant among the higher coordinates of LDSs, which show undesirable patterns as opposed to the low discrepancy pattern in Figure I.4. This phenomenon is referred to as *pattern dependency*. The author suggests that in timing analysis the lower coordinates of Sobol

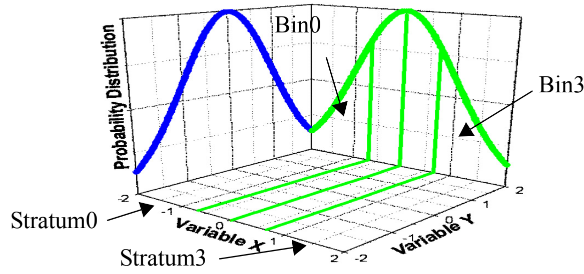


Figure I.5. Stratification of a 2D space. Variable X is divided into 4 bins, thus dividing the sample space into 4 strata.

sequences, which have no significant pattern dependencies, be assigned to the important variables in the sampling procedure. Therefore, a concept of criticality of variables in timing analysis needs to be defined, which can be used to sort the variables in the order of their decreasing importance.The coordinates of the Sobol sequence can then be assigned to variables in this order. We present a technique for ordering the variables based on their criticality to circuit delay in the statistical timing framework.

A related point is that Sobol sequences are not accurate beyond a certain number of dimensions. Hence, in this work, we use Quasi Monte Carlo techniques in conjunction with stratified sampling and Latin Hypercube Sampling (LHS). The next two subsections provide a brief overview of stratified sampling and LHS.

I.3.2 Stratified sampling

Stratified sampling is a technique to partition the sample space into mutually exclusive strata, and then sample using any of the known variance reduction techniques within each [28]. The stratification method in this work is illustrated for a 2D example in Figure I.5, where random variable X is divided into 4 equal probability bins (X is equally likely to fall in any of the 4 bins), whereas random variable Y is not binned. This method is adopted when X is critical to the function value to be estimated, whereas Y is not. In this way, the

2D space is partitioned into 4 strata as shown in the figure. Throughout the work, we use 'bin' to refer to regions in individual variables, and 'strata' to refer to partitions in the *n*D space, where *n* is the dimensionality. In general in multidimensional space, 1 or more variables are binned, and the permutations of bins across variables define strata. In the case of timing analysis, the timing behavior of the circuit is more sensitive to the critical variables by selection and these variables are binned. Therefore within strata the timing behavior exhibits lower variation and is easier to estimate. The technique leads to accuracy with few samples, however cannot be used over very large dimensions since the number of strata increases exponentially.

I.3.3 Latin Hypercube Sampling

Latin Hypercube sampling is a technique in variance reduction which deals with multidimensional systems [37]. This technique tries to sample each variable involved uniformly by dividing the variable into equal probability bins. The samples from bins in variables are combined across dimensions to obtain faster convergence than random sampling. This is in contrast with taking all permutations of the bins across variables to define strata, and then sampling within each stratum as in stratified sampling described

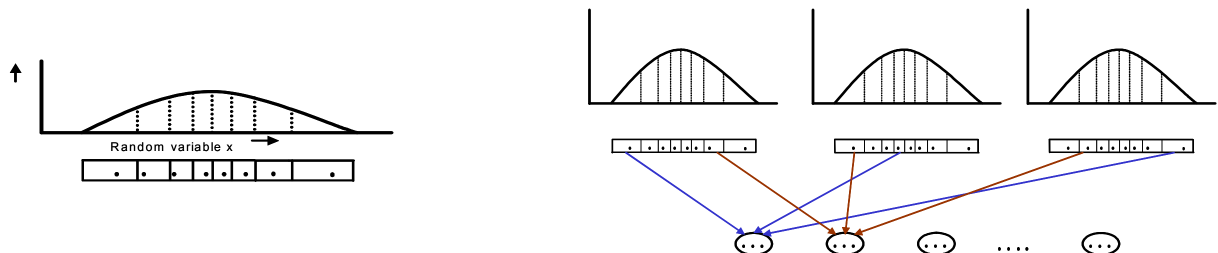


Figure I.6. Latin Hypercube Sampling (a) Divide each variable in 8 equal probability bins and sample in bins. (b) Combine randomly to form 8 triplets

above. This means that LHS can deal with large dimensions, however with a moderate rate of convergence compared to full stratification.

The LHS procedure is illustrated in Figure I.6. Each random variable is divided into equal probability bins. One sample is generated within each bin. Such samples are combined across variables to obtain Latin Hypercube samples. This is the procedure to obtain k samples, where k is the number of bins per variable. To obtain mk number of samples, we repeat the LHS procedure m times.

Two other techniques that have been studied for application to integrated circuit yield estimation are importance sampling and control variates. In general, these methods require more detailed information about the circuit. For literature in statistics about the method, refer to [28]. More work is required to establish the effectiveness of these approaches for use in the modern integrated circuit design process.

I.4 Thesis Overview and Key Contributions

- We describe methods to improve the efficiency of Monte Carlo-based statistical static timing analysis. We propose a Stratification + Hybrid Quasi Monte Carlo (SH-QMC) approach to reduce the number of samples required for Monte Carlo based SSTA. Our simulations on benchmark circuits up to 90K gates show that the proposed method requires 23.8X fewer samples on average to achieve comparable accuracy in timing estimation as a random sampling approach. Results on benchmark circuits also show that when SH-QMC is performed with multiple parallel threads on a quad core processor, the approach is faster than traditional SSTA with comparable accuracy. SH-QMC scales better than traditional SSTA with circuit size. When the proposed SH-

QMC technique is extended to include a graph pruning method the runtime is further reduced by 48% on average for the benchmark circuits considered. We also propose an incremental approach to recompute a percentile delay metric after ECO. The results show that on average only 1.4% and 0.7% of original samples need to be evaluated for exact recomputation of the 95th percentile and 99th percentile delays, after sample size reduction using SH-QMC.

- We illustrate possibilities to exploit the parallelism in the SH-QMC algorithm with the implementation of the algorithm on a Graphics Processing Unit (GPU). We show that although straightforward parallelism is available, achieving performance and efficiency on the GPU requires exposing more parallelism and finding locality in computations. This is in contrast with random sampling based algorithms which are inefficient in terms of sample size but can keep resources utilized on a GPU. We show that SH-QMC implemented on a Multi GPU is twice as fast as a single STA on a CPU for benchmark circuits considered. In terms of an efficiency metric, which measures the ability to convert a reduction in sample size to a corresponding reduction in runtime w.r.t a random sampling approach, we achieve 73.9% efficiency with the proposed approaches compared to 4.3% for an implementation involving performing computations on smart samples in parallel. Another contribution of the work is a critical graph analysis technique to improve the efficiency of Monte Carlo based SSTA, leading to 2-9X further speedup.

- We propose a technique to compute a lower bound for the minimum possible area that can be achieved for a design while meeting a particular timing yield, which is the percentage of die that meeting a specified timing constraint. We then compare several

statistical design optimization approaches, including one proposed in this paper called SLOP, against the computed lower bound. We show that even the simplest statistical optimization approaches produce area results which are, on average, within 9.6% of the lower bound while the best ones performed only marginally better, reaching within 3.7% of the bound. This demonstrates that the proposed bound is a close bound.

- Leakage power minimization is critical to semiconductor design in nanoscale CMOS. On the other hand increasing variability with scaling adds complexity to the leakage analysis problem. In this work we seek to achieve tractability in Monte Carlo-based statistical leakage analysis. A novel approach for fast and accurate statistical leakage analysis considering inter-die and intra-die components is proposed. We show that the optimal way to select samples, to capture intra-die variation accurately, is according to the probability distribution function of total process variation. Intelligent selection of samples is performed using a Quasi Monte Carlo technique. Results are presented for benchmarks with sizes varying from approximately 5,000 to 200,000 gates. The largest benchmark with 198461 gates is evaluated in 3 minutes with the proposed approach compared to 23 hours for random sampling with comparable accuracy. Compared to a conventional analytical approach using Wilkinson's approximation, the proposed technique offers superior accuracy while maintaining efficiency. State dependence and multiple sources of variation are considered and the approach is scalable with number of process parameter variables for standard cell characterization cost. We also show reduction in sample size to meet target accuracy for computing leakage distribution due to the inter-die component only when compared to random selection of samples.

CHAPTER II

EFFICIENT MONTE CARLO BASED INCREMENTAL STATISTICAL TIMING ANALYSIS

II.1 Introduction

Process parameter variations have taken on increasing importance in nanometer-scale CMOS. Rather than using simple corner models that capture worst-case behavior at the device level (and lead to large guard bands), modern CAD tools are moving towards a more probabilistic view of circuit timing behavior. Two different approaches exist to capture the timing behavior - (1) analytical approaches which propagate standard distribution functions through the circuit, and (2) Monte Carlo (MC) based approaches which analyze samples of the circuit in the process variation space. Previous work in these areas were overviewed in Chapter I.

Standard techniques to reduce the sample size for MC based approaches exist in statistics literature and are called variance reduction techniques. The application of these techniques for parametric yield estimation has been analyzed in literature [30-33]. In [30], a Latin Hypercube approach for parametric yield estimation is proposed. In [31], mixture importance sampling for statistical SRAM design and analysis is proposed. The approach in [32] uses the control variates technique in conjunction with importance sampling for

timing yield estimation. However, while several approaches are reviewed, no results are presented. In [33], the authors propose to use Quasi Monte Carlo Analysis for yield estimation. However, it is not clear how this approach can be extended to systems with large number of dimensions (variables) which is often the case with process variation. Also, these approaches do not focus on the specific problem of using MC as an alternative to traditional SSTA for timing analysis. Variance reduction relies heavily on information about the system [28], hence it is important to adapt it specifically to timing analysis. To the best of our knowledge this work is the first to directly study variance reduction aimed at improving the efficiency of MC-based SSTA with an accurate process variation model considering intra-die variation with spatial correlation [7] and uncorrelated random variation.

ECO(Engineering Change Order) and synthesis tools require incremental timing analysis techniques for fast recomputation of circuit delay with small changes in the design. To meet time to market, designers need tools capable of performing fast incremental timing analysis, and such tools need to incorporate process variations. While incremental techniques for traditional SSTA exist in literature [6], the lack of such techniques has been a major drawback for MC based approaches to SSTA. We address the specific problem of recomputing a percentile delay metric after incremental circuit sizing. To the best of our knowledge, this work is the first to address incremental timing analysis in MC based SSTA.

This work has three main contributions. First, we introduce a new approach for variance reduction in MC based SSTA, Stratified Sampling + Hybrid Quasi Monte Carlo (SH-QMC). In SH-QMC, we propose to use circuit timing criticality information for

sample size reduction. We use information about the criticality of variables to the circuit delay to order them. For the most critical variables, we then employ techniques that achieve high accuracy with few samples. For the less critical variables, we use techniques that are effective for problems of higher dimensionality. The proposed approach is implemented and tested on benchmark circuits with sizes up to 90,000 gates, and compared to a random sampling approach for selecting samples in the process variation space. In general SH-QMC shows large speedups relative to the random sampling approach: 23.8X on average and up to 44X on the benchmarks studied. Our results also show that the number of samples required does not increase with the number of gates in the circuit. Additionally, when SH-QMC is implemented with multiple threads on a quad core processor, it is faster than traditional SSTA for comparable accuracy. We also observe that the performance of SH-QMC scales better than traditional SSTA with circuit size.

Second, we propose an extension to SH-QMC to consider a graph pruning method. In this method we use the information obtained from the evaluation of a few SH-QMC samples to reduce the circuit graph size. This enables fast evaluation of the remaining samples leading to up to 84% additional reduction in runtime.

Third, we propose a technique to recompute a percentile delay metric after incremental circuit sizing, where individual gates are resized. In this technique, we use information local to the resized gate to prune out most of the samples, leaving only a few samples to be reevaluated. Our results for the incremental computation of the 95th percentile and 99th percentile delays of benchmark circuits show that on average only 1.4% and 0.7% of original samples need to be evaluated for exact recomputation, even after sample size reduction using SH-QMC.

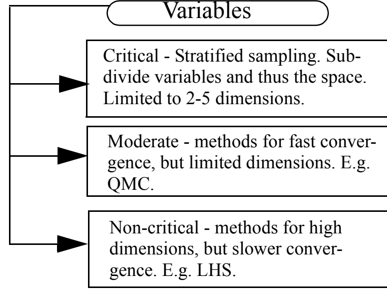


Figure II.1. Ordering variables using timing criticality.

This chapter is organized as follows. Section II.2 presents our work on variance reduction for MC based SSTA and proposes a graph pruning method to improve the efficiency of SH-QMC. In Section II.4, we propose our approach to incremental statistical timing analysis. We present detailed results in Section II.5 and conclude with Section II.6.

II.2 Smart sampling based on timing criticality

In this section, we first describe our process variation model and then go on to discuss our smart sampling approach.

II.2.1 Process variation model

Our process variation model is based on [7] which takes into account intra-die spatially correlated variation [17-22] by partitioning the die into $n * n$ grids and assuming identical parameter variations within a grid. Therefore, each source of variation is represented by a set of random variables for all grids. For example, transistor gate length variation is represented by a set of random variables for all grids and the set is of multivariate normal distribution with a covariance matrix R_{Lg}. Principal component analysis [23] is performed on these correlated random variables to obtain a set of principal components. Similarly, principal components are obtained for other sources of variation.

Let $p_i : i=1,...,m$ be the principal components of all global sources of variation. In addition to these global sources of variation, we have an independent random variable Δr to account for random variation at the gate level. The delay for a gate is expressed as a linear combination of principal components of p_i's and Δr:

$$d = d_0 + k_1 \times p_1 + \ldots + k_m \times p_m + k_{m+1} \times \Delta r$$

where d_0 is the gate delay mean, k_i: $i=1,...,m$ are the coefficients for the principal components. p_i's and Δr are independent unit normal random variables after suitably scaling their coefficients.

II.2.2 Stratification+Hybrid Quasi Monte Carlo(SH-QMC)

In our smart sampling approach SH-QMC, we propose to use circuit timing criticality information to reduce the sample size for MC based statistical timing analysis. In the previous subsection, we have defined the variables representing process parameter variation. In our proposed approach, we order these variables based on their criticality to the circuit delay using a timing criticality parameter P_{crit} defined in the next subsection. We then apply Quasi Monte Carlo (QMC), stratified sampling and LHS to variables based

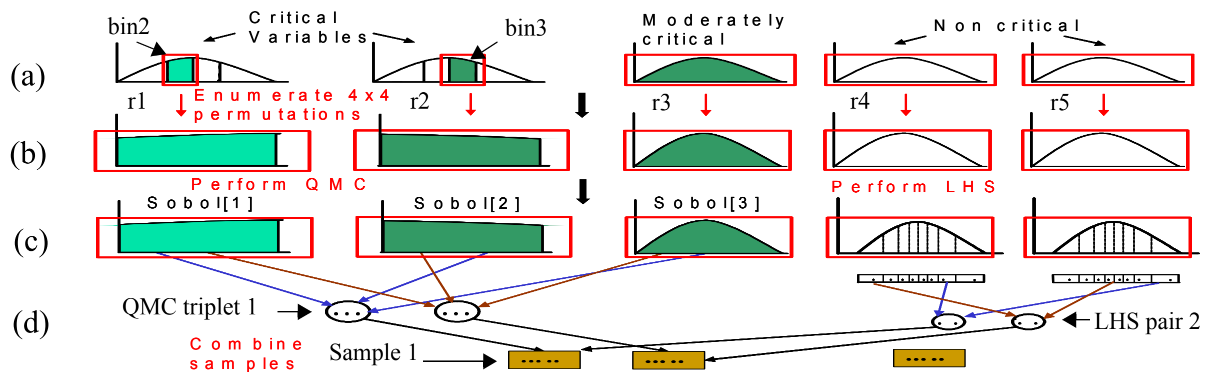


Figure II.2. Stratified Latin Hypercube Sampling (a) Ordering variables based on timing criticality. (b) One of 16 strata in the sample space. (c) QMC triplets and LHS pairs. (d) These are combined to obtain final samples.

on their convergence property and the ability to handle multiple variables (dimensions) as illustrated in Figure II.1. The topmost critical variables guide the stratified sampling approach, which leads to faster convergence. Only the top 2-5 variables are used to guide stratification since the number of strata increases exponentially with the number of variables. QMC method is then employed on the topmost to moderately critical variables for its fast convergence properties. However, QMC can exhibit pattern dependencies with large number of variables, so only a limited number of variables are sampled using QMC. On the non-critical variables, we use Latin Hypercube Sampling which is applicable for large number of variables, but has slower convergence to an accurate result.

The method is illustrated in Figure II.2 using a 5 variable example. As mentioned before, variables are ordered as critical, moderately critical and non-critical. The two most critical variables $r1$ and $r2$ are divided into 4 bins each (Figure II.2a). A stratum is defined as a set of points in the 5D space restricted to one bin each in $r1$ and $r2$, but unrestricted in $r3$, $r4$ and $r5$. The total number of strata is 16, arising from 4 by 4 permutations of the bins. Figure II.2b illustrates one particular stratum which we use to explain the remaining steps. In this stratum, points are restricted to bin 2 in $r1$ and bin 3 in $r2$. As shown in Figure II.2c, QMC method based on Sobol sequence is used to sample $r1$, $r2$ and $r3$ in the stratum and LHS is applied to $r4$ and $r5$. Note that since we are only sampling within the stratum, samples of r1 and r2 are restricted to the respective bins. QMC generates triplets as shown in the figure. For performing LHS, $r4$ and $r5$ are divided into 8 bins each and one value is selected from each bin as in Figure II.2c. 8 LHS pairs are generated by randomly picking from $r4$ and $r5$ in one step of LHS. Two LHS pairs are shown in Figure II.2d. Next, the LHS pairs are combined with the QMC triplets to generate our final

samples. The procedure is repeated: LHS pairs are generated again in $r4$ and $r5$, and QMC triplets are generated in the other 3 variables. These are then combined as before. After generating the samples in this stratum, we move to the next stratum and repeat our steps. In this manner, we generate samples in all 16 strata.

Among the variables on which QMC is employed, the lower coordinates of LDSs are assigned to the more critical variables. The order of criticality here is again decided using the parameter P_{crit}.

II.2.3 Timing criticality P_{crit}

To order the principal components, we employ a timing criticality metric P_{crit}. To compute P_{crit}, we perform static timing analysis on the nominal circuit to identify critical paths within a slack of $s\%$ of worst-case arrival time, where s is a parameter. This STA run is performed under nominal process conditions. Now, each grid is assigned a weight equal to the number of gates falling in any of the potential critical paths. Let w^i_g be the weight of the ith grid. The weight of the jth principal component is given by

$$w_j = \sum_i \left(w_g^i \times k_{ij} \right)$$

where k_{ij} is the coefficient of the jth principal component in the ith grid variation. This empirical technique leads to fast computation of P_{crit} with sufficient accuracy to guide our proposed SH-QMC.

II.3 Pruning based algorithm for timing analysis

In Section II.2.2 above we discussed our Stratification + Hybrid Quasi Monte Carlo (SH-QMC) approach to reduce the sample size using timing criticality information. In this section we propose a technique to further improve the performance of SH-QMC through graph reduction. As explained before, timing criticality information for SH-QMC is obtained by performing STA on the nominal circuit. Clearly as we evaluate the samples for SH-QMC we extract more information about the statistical behavior of the circuit with each additional sample evaluated. The algorithm presented here has two stages. In the first stage of the algorithm, the basic idea is to use slack information for every node in the circuit graph obtained at the nominal sample to identify non-critical nodes. The corresponding gates are excluded from consideration for statistical analysis based on SH-QMC. In the second stage we make use of information obtained from evaluation of successive SH-QMC samples to find bounds on the statistical behavior of timing slack at each node. When enough information is gathered to tag a certain node as having negligible statistical probability to fall in the critical path(s) for any sample, the node is pruned or eliminated from consideration for the SH-QMC samples yet to be evaluated.

The algorithm is explained in more detail here. As explained, the first stage involves eliminating nodes based on nominal STA. Nodes with positive slack exceeding a threshold value are pruned. This threshold value is chosen to be 10% of the nominal case worst arrival time. The choice is such that the likelihood of a path with greater than the threshold slack being critical is very low. The second stage of pruning is performed once a subset of SH-QMC samples is evaluated. The samples generated by SH-QMC can be partitioned into sets such that within a set the samples are complementary in their coverage of the

process variation space. These sets are referred to as minimal SH-QMC sets in the rest of the chapter. As a result when all samples within any of these sets are evaluated the statistics of the circuit arrival time are estimated more accurately than a set of the same sample size with elements selected at random from the SH-QMC set. As mentioned before in Section II.2.2, SH-QMC combines stratified sampling, Quasi Monte Carlo sampling (QMC) and Latin Hypercube Sampling (LHS). First the sample space is partitioned into strata. Next QMC and LHS are applied in combination within each stratum. A minimal SH-QMC set consists of equal-sized subsets from each stratum. As mentioned before, within a stratum a subset of the variables is sampled using LHS. The sample size of the subset corresponding to a stratum is defined by the number of bins in the variables sampled using LHS. The samples in the subset are selected such that each variable sampled using LHS has exactly one value per LHS bin. This is the same as selecting an LHS set (consisting of one value per bin for each variable in the set). Note that the variables sampled with QMC do not play a role in the selection of samples as QMC samples have no granularity restrictions from the way they are constructed. For example, suppose the LHS technique used divides each variable into 20 bins and there are 4 strata in the process variation space. Then a minimal SH-QMC set has 80 samples and each subset has 20 samples. Intuitively a minimal SH-QMC set has all LHS bins and strata covered which leads to the lower error for delay statistics computation for the sample size. With this backdrop the second stage of pruning can be explained. Exactly one minimal SH-QMC subset is selected from the SH-QMC set and the elements evaluated. At every circuit node we thus have a slack distribution obtained from the minimal subset. Given the slack distributions, the procedure to prune nodes is explained using Figure II.3. The slack

distributions for gates ‘g1’ and ‘g2’ are illustrated in the figure. Note that slack by definition cannot be negative once the circuit is fixed. The dotted line indicates a low percentile of the slack distribution for each gate. A gate is pruned if the dotted line coincides with zero, that is the low percentile value is zero. The problem now is to determine the optimal percentile point of slack distribution for pruning. Lower percentile points are expected to be accurate but cost runtime. We evaluate the trade-off with varying percentile point and present our results.

II.4 Incremental Evaluation of a Percentile Delay

ECO and synthesis tools require efficient incremental timing analysis techniques for fast recomputation of circuit delay with small changes in the design, while also accounting for process variation. In MC based SSTA there is a lack of incremental capability to date. In this section, we present an approach for the incremental evaluation of a specific percentile delay of a circuit with a small change in circuit sizing. We illustrate the approach for the case of single gate sizing in this work. However, the approach can be extended to the case of simultaneous multiple gate sizing. The key intuition is that if the

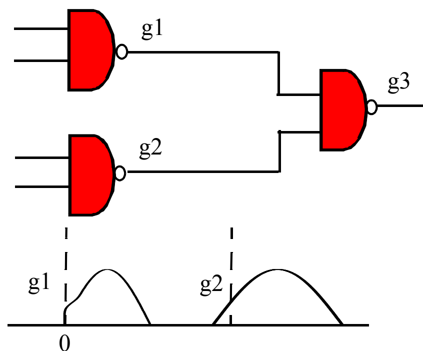


Figure II.3. Slack distribution of gates ‘g1’ and ‘g2’ obtained by evaluation of minimal SH-QMC set. The threshold percentile for each distribution is plotted as a dotted line. ‘g2’ is pruned in this case as the criterion of positive slack percentile is satisfied.

samples for SH-QMC on circuit *C* are reused for *C'* (*C* with gate *g* sized), then most samples need not be reevaluated to recompute the xth percentile delay; only those samples that have a circuit arrival time 'close' enough to the xth percentile delay of *C* need to be reevaluated. An upperbound on change in circuit arrival time of a sample from *C* to *C'* can be determined from a local bound computation involving only a few gates connected to the gate *g* being resized. This bound can be used to prune out a majority of the samples, leaving us with a few that need to be reevaluated. Further speedup can be achieved with established techniques for incremental STA on the samples selected for reevaluation.

II.4.1 Algorithm

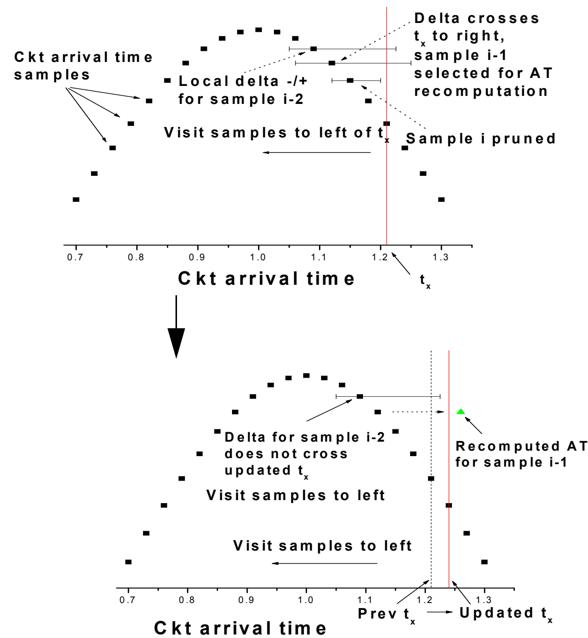


Figure II.4. (a) Samples are visited in decreasing order of circuit arrival time, starting from the xth percentile (t_x). Samples with delta crossing t_x are selected, others pruned. (b) Recomputation of circuit arrival time is performed at the selected sample and t_x is updated.

We perform timing analysis on an original circuit *C* using our SH-QMC approach and store the samples for the process variation space and the corresponding circuit arrival time in memory. Our approach for the recomputation of a specific percentile delay using the stored samples is illustrated in Figure II.4. For each sample, a bound on change in circuit arrival time from *C* to *C'* (*C* with gate *g* sized) is obtained as explained in Section II.4.2. Each sample has a positive bound and negative bound for either direction of change. The samples are sorted in the order of increasing circuit arrival time for *C*. In Figure II.4a, the samples are represented by points on the circuit arrival time distribution curve. They are visited in the decreasing order of arrival time starting from the xth percentile value t_x. A sample *k* is selected for reevaluation if its arrival time for circuit *C* and the positive bound for *k* add up to exceed t_x. For example, in Figure II.4a, sample *i* is pruned out since its positive bound is not large enough to cross t_x. However, sample *i*-1 is reevaluated as it has a large enough upper bound to cross t_x. As illustrated in Figure II.4b, the arrival time for *i*-1 is recomputed. Sample *i*-1 is updated with this value of arrival time which shifts t_x to the right. Next sample *i*-2 is reevaluated, however the arrival time value obtained is less than t_x, so t_x does not change. Sample *i*-2 is also updated with the recomputed arrival time value. After considering all samples to the left of t_x, we visit the samples to the right. The criterion for reevaluating a sample here is that its arrival time for *C* and the negative bound for the sample should add up to less than t_x. After this step, we repeat the procedure and visit samples to the left of the updated t_x. Samples reevaluated earlier are not visited again. The termination criterion is that there are no samples to the left or right of t_x which satisfy the criterion for reevaluation. The final value of t_x is the xth percentile delay of *C'*.

The justification for reuse of samples is that our metric to guide SH-QMC P_{crit} (Section II.2.3) is measured at the grid level in our process variation model, so within reasonable ECO changes the timing criticality of the circuit does not change to significantly alter our metric P_{crit}. In particular, we are only concerned about the relative ordering of variables based on P_{crit}. Therefore with single gate sizing, the samples are still accurate. For cases where there is significant design change, SH-QMC is performed again to generate new samples. As mentioned the samples for C are stored in memory. Our results on the benchmarks studied demonstrate that the number of samples for SH-QMC that gives sufficient accuracy is 80 for the largest circuits. Therefore, we need to store 80 samples for each gate. In general, if the number of samples required is much higher, the memory overhead could be significant. Section II.2.1 defined the variables to model process variation, which are the principal components for all sources of variation and an independent random component at the gate level. Now, it is enough to store samples for these components, as the device parameters can be retrieved using the values of components. Storing samples for the principal components incurs negligible memory overhead. In the case of the independent random component, instead of storing all samples of the component for all the gates, we store the initial 'seed' value for the pseudorandom number generator. Note that for STA, gate delays are propagated in the topological order. This offset in the topological order along with the 'seed' value is provided to the pseudorandom number generator which reproduces the random numbers while incremental analysis is performed.

II.4.2 Computing circuit arrival time bound for samples

We compute the maximum possible increase and decrease in the circuit arrival time for each sample of circuit *C* using local gate delay change information when gate *g* is sized. Define sets *Fi(g)* of fanin gates of *g*, *FoFi(g)* of fanouts of gates in *Fi(g)* and *Fo(g)* of fanout gates of *g*. We select subpaths that are candidates for obtaining the bounds in circuit arrival time and evaluate the change in delay of these subpaths when *g* is sized. Every subpath starting from an input pin of a gate in *Fi(g)* and ending in an output pin of a gate in either *Fo(g)* or *FoFi(g)* is a candidate for this evaluation. Some such subpaths could have more than one gate in *Fi(g)*. We assume that delay change is significant only in the gates in the three sets defined above, therefore only these gates affect the change in subpath delay. Now, we obtain bounds for circuit arrival time change for a sample *S* as follows. Let *P(g)* be the set of all candidate subpaths. $t_S(p)$ and $t'_S(p)$ are delays for subpath *p* in sample *S* before and after sizing gate *g*, respectively. Then the negative and positive bounds are given by:

$$delta_neg(g,S) = \mathrm{in}\{t_S'(p) - t_S(p) \forall p \in P(g), 0$$

$$delta_pos(g,S) = \max\{t_S'(p) - t_S(p) \forall p \in P(g), 0\} \quad .$$

In other words, we find the maximum and minimum values of the change in delay of candidate subpaths. As gate delay change is assumed to be significant only in the local subcircuit (set of gates belonging to *Fi(g), Fo(g)* and *FoFi(g))*, the computational overhead is low. In our algorithm in Section II.4.1, we only need either of *delta_neg* or *delta_pos* for most samples. A *delta_neg* or *delta_pos* computation for a sample involves gate delay computation and propagation in the local subcircuit twice, one each before and

Table II.1. Comparison of random sampling, LHS based and SH-QMC approaches based on sample size. The last two columns show the speedup of LHS and SH-QMC respectively, over random sampling.

Circuit	No of gates	RS count	LHS count	SH-QMC count	LHS speedup	SH-QMC speedup
C432	256	1120	1120	240	1	4.7
C499	544	1760	1360	40	1.29	44
C880	500	1760	1440	80	1.22	22
C1908	603	1440	960	80	1.50	18
C2670	780	1600	1200	80	1.33	20
C3540	1163	2320	1440	160	1.61	14.5
C5315	1692	2160	1120	80	1.93	27
C6288	3834	1840	880	80	2.09	23
C7552	2152	3040	1280	80	2.38	38
VD1	14503	1360	800	80	1.70	17
VD2	34082	2000	880	80	2.27	25
USB	32898	2240	1200	80	1.87	28
ETHER	57327	2080	1600	80	1.30	26
VGA	90831	2000	800	80	2.50	25

after gate sizing. Therefore, the cost of arrival time bound computation across all the samples for the percentile delay recomputation is approximately twice that of performing Monte Carlo analysis on the local subcircuit with smart samples. The runtime for this is negligible compared to that of a single STA run for most practical circuits.

II.5 Results

Our simulation results are based on a 90nm industrial technology library. In our implementation we only consider channel length variation as a source of process variation

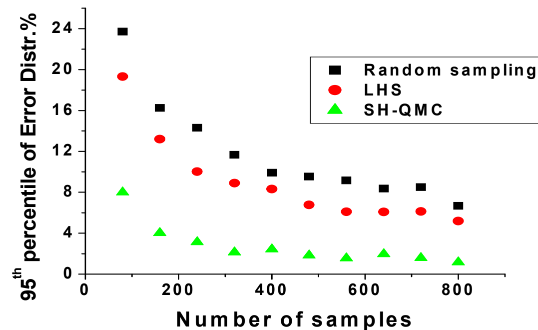


Figure II.5. Error comparison of random sampling, LHS and SH-QMC for a VGA circuit (90831 gates) w.r.t. golden of MC count 40,000.

for simplicity. However, this is not a limitation of our approach. The inter-die spatially correlated intra-die and uncorrelated random components of channel length variation are considered. The overall standard deviation is 10% of nominal channel length. This amount of process variation increases absolute variability, but more importantly serves to highlight the accuracy comparison of the techniques considered. The number of grids in the spatial correlation model for individual circuits is varied linearly with post-placement area starting from 2 by 2 for the smallest circuit to 16 by 16 for the largest circuit. This corresponds to a grid area of approximately 40μm by 40μm for all the circuits. We compare our proposed SH-QMC approach with random sampling and LHS based techniques. Simulations are performed on ISCAS85 benchmark circuits [38], and 5 large circuits. These are Viterbi Decoder 1(VD1), Viterbi Decoder 2(VD2), USB2.0 Core (USB), Ethernet MAC Core (ETHER) and VGA Controller Core (VGA), with gate counts varying from approximately 15,000 to 90,000. We perform synthesis and APR on all the circuits using commercial tools.

Our comparisons are based on the error in estimating statistical moments of arrival time distribution for a given method w.r.t the moments from a golden of 40,000 Monte Carlo runs. Consider for example a given trial MC_1 of size 100 samples. This gives a circuit arrival time distribution. From this, moments μ_1 and σ_1 (mean arrival time and standard deviation in arrival time) are obtained and error (magnitude of deviation from the golden) calculated for both. From repeated trials (each of 100 samples in this example), we get 2 distributions for error. The nature of the error distributions show the efficiency of the technique. For example, as we increase the number of samples from 100 to 200 in the

Table II.2. Runtime comparison of SHQMC with SSTA. AT = circuit delay

Circuit	No of gates	Mean AT Error(%)		σ AT Error (%)		SSTA Run-time(s)	SH-QMC Runtime(s)	
		SSTA	SH-QMC	SSTA	SH-QMC		Multi thread	Single thread
VD1	14503	1.56	0.08	2.43	1.80	0.92	0.83	2.9
VD2	34082	1.66	0.34	2.37	2.12	3.79	2.42	8.7
USB	32898	1.36	0.53	3.48	1.85	4.37	4.22	14.2
ETHER	57327	0.35	0.05	1.8	2.3	8.18	6.2	19.9
VGA	90831	0.40	0.08	0.03	1.80	9.93	6.85	22.1

above example and repeat the experiments, the error distribution is expected to get tighter and closer to zero. In particular, the 95th percentile of the error gets closer to zero and we use this value as a criterion to compare different techniques. The minimum number of samples required by a technique such that the 95th percentile of error distribution is less than 5% for both mean arrival time and standard deviation of arrival time is our performance metric for the technique.

Table II.1 compares the number of samples required for random sampling, an LHS-based technique and our proposed SH-QMC approach. The proposed approach achieves on an average 23.8X reduction (lowest 4.7X up to 44X) in number of samples w.r.t random sampling, whereas LHS achieves a modest improvement of 1.7X on average (lowest 1X up to 2.5X). The improvements are consistent across the benchmark circuits studied. In Section II.2.3, we mention that critical paths are identified within a slack of *s%* for computing timing criticality P_{crit}. We investigated the sensitivity of the results to the parameter *s* and found that varying *s* from 1-5% showed no changes in the number of samples required to meet the stated accuracy objective, indicating that the proposed technique is stable with respect to this parameter. Figure II.5 visually presents the 95th percentile of error of random sampling, LHS and SH-QMC for our largest circuit VGA (90831 gates) w.r.t. the golden model. Though we have two error distributions

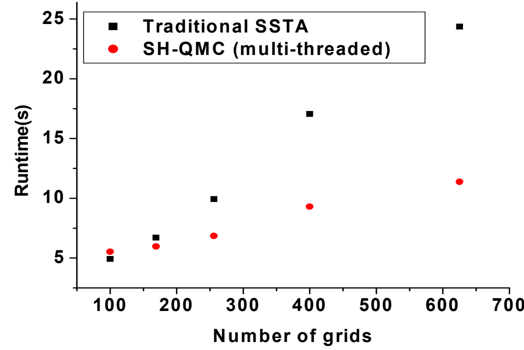


Figure II.6. Performance comparison of traditional SSTA with multi-threaded SH-QMC for VGA circuit (90831 gates) as function of number of grids in process variation model.

(corresponding to mean and standard deviation of arrival time), our simulations show that the error in estimating standard deviation always dominates the error in mean. The error plotted in Figure II.5 is therefore for the standard deviation of arrival time.

Table II.2 compares the runtime of SH-QMC and traditional SSTA. For both mean and standard deviation of arrival time the error for SH-QMC in the table is the average absolute deviation from their values in the golden model; for traditional SSTA this is the error w.r.t the golden. The golden model is MC with 40,000 samples. One drawback of Monte Carlo techniques in general is that every time an experiment is performed, the error w.r.t golden is different. This means that the error in one particular MC experiment is sometimes higher than the average value mentioned. However, the 95th percentile of the absolute error distribution is still less than 5% for all the circuits in the table. This translates to an error of 3-7ps in absolute time for different circuits, which is a reasonable target for the given process technology. All our simulations were performed on a single Quad Core processor. For SH-QMC, we perform two different experiments, in one we spawn 4 threads to use the parallelism in the Quad Core machine, and in the other we run a single thread on the machine. The former uses the parallelism in sample evaluation,

which is straightforward in MC methods but not true of traditional SSTA. Parallelizing traditional SSTA is non-trivial and would incur runtime cost. We consider circuits with more than 10,000 gates for meaningful runtime comparisons. SH-QMC with multi-threading performs better than traditional SSTA in runtime. Also, further speedup in SH-QMC can be achieved in a straightforward manner using parallel processing on more than one processor. Figure II.6 compares the performance of traditional SSTA with SH-QMC for the VGA circuit as a function of number of grids in the process variation model. This illustrates that SH-QMC scales better than traditional SSTA. Figure II.7 is a typical case comparison of efficiency in estimating a high percentile statistic in arrival time distribution obtained from our approach w.r.t a traditional SSTA approach. The error in estimating the 99th percentile arrival time for SH-QMC is better than traditional SSTA at more than 72 samples for the USB2.0 Core circuit (32898 gates) considered. In general, our approach estimates the 99th percentile arrival time better than traditional SSTA for all benchmark circuits studied at a low number of samples. Figure II.8 compares the probability distribution curve of arrival time of the USB circuit for SH-QMC (96 samples) and a traditional SSTA approach, w.r.t the golden. Our technique captures the mean arrival

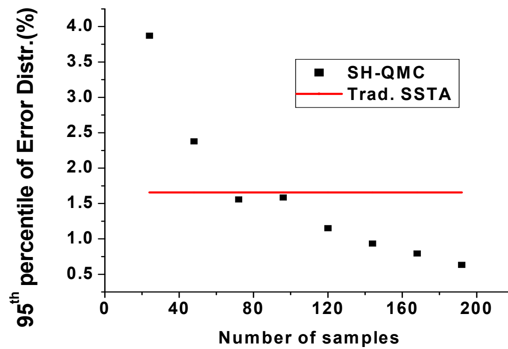


Figure II.7. Comparison of 99th percentile error of SH-QMC vs. traditional SSTA w.r.t golden of 40,000 MC count for USB circuit.

Table II.3. Comparison of three SH-QMC-based approaches with no pruning, single stage pruning and double stage pruning on benchmark circuits.

Circuit	No of gates	No pruning	Single stage		Two stage	
		Runtime(s)	% gates pruned	Runtime (s)	% gates pruned	Runtime(s)
VD1	14503	2.9	52.4	1.76	54.1	1.58
VD2	34082	8.7	28.4	8.67	29.9	8.08
USB	32898	14.2	68.3	5.71	75.4	5.60
ETHER	57327	19.9	98.2	6.34	98.3	3.10
VGA	90831	22.1	66	12.6	68	13.0

time (marked with vertical lines) and the overall shape of the distribution better than the traditional SSTA approach.

The results of the proposed graph pruning approaches for SH-QMC for the benchmark circuit VD1 are presented in Figures 9 and 10. As explained in Section II.3 the pruning criterion is that a low percentile of the slack distribution be zero. In our implementation we approximate the criterion assuming a normal distribution for slack. The cutoff percentile is determined in terms of the pruning parameter k defined to be such that if $\mu - k\sigma$ of the gate slack distribution is non-negative, then the gate is pruned. The plots in Figures 9 and 10 plot the error and runtime of the pruning approach while varying the pruning parameter. As the pruning parameter increases the pruning criterion gets more restrictive which leads to higher accuracy while costing runtime. The error metric is the

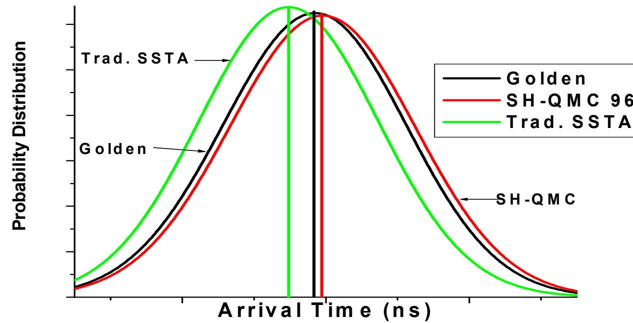


Figure II.8. Arrival time distribution of SH-QMC (96 samples) and traditional SSTA w.r.t golden(40,000 MC) for USB circuit.

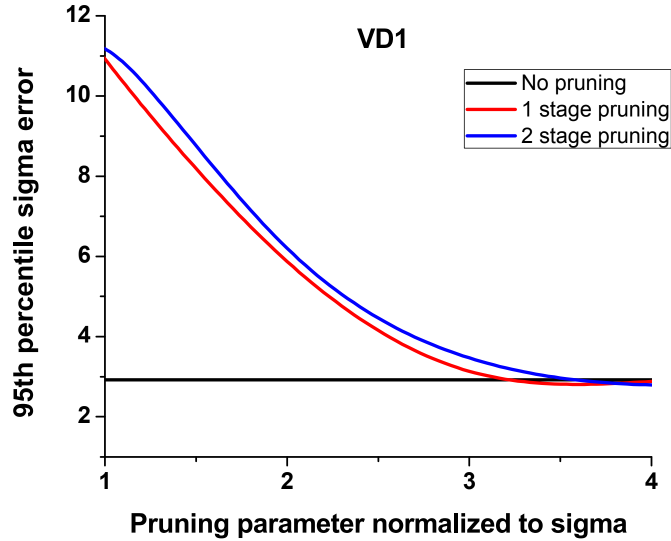


Figure II.9. Comparison of the 95th percentile error in σ for single stage and two stage pruning for a large benchmark circuit.

95th percentile error in estimating the standard deviation in arrival time compared to a golden Monte Carlo analysis with 40,000 samples. The error is compared with respect to the SH-QMC approach without graph pruning. For pruning parameter k exceeding 3.5, the errors are comparable for both single stage and two stage pruning approaches with respect to the analysis performed without graph pruning. The runtime of both the approaches are better than the case of no pruning since non-critical gates have been pruned from

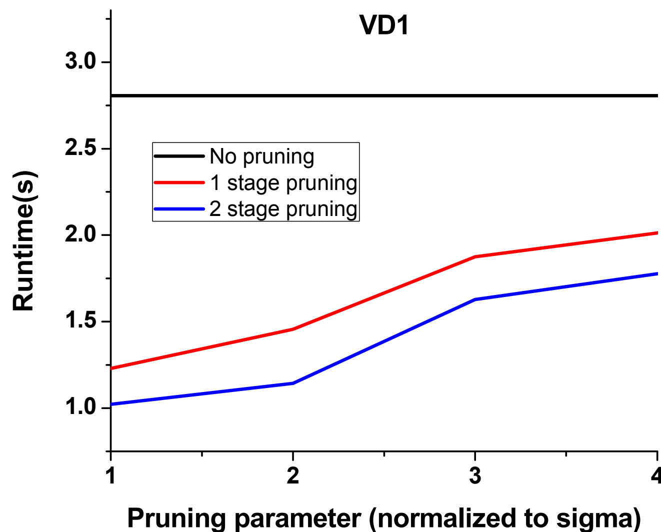


Figure II.10. Comparison of runtime vs. pruning parameter for single stage and two stage pruning for a large benchmark circuit.

consideration after evaluating a few samples. Two stage pruning is faster than single stage pruning. This is because highly non-critical gates are removed before the first SH-QMC sample is evaluated in the two stage approach. In general the error of both single and two stage approaches are comparable to the case of no pruning for all the benchmark circuits for pruning parameter k exceeding 4. Table II.3 compares the three SH-QMC based approaches with no pruning, single stage pruning and two stage pruning. The pruning parameter k is fixed at 4 where the errors of all three approaches are comparable for the benchmark circuits. Single stage pruning has up to 68% lower runtime compared to the no pruning case whereas two stage pruning has up to 84% lower runtime. Single stage pruning performs is faster by 42% and two stage pruning by 48% on average.

Table II.4 presents our results for the incremental evaluation of the 95th percentile and 99th percentile delay after a gate size change using our approach in Section II.4. In our experiments, we select 100 gates at random for a given circuit. Each gate is sized up individually and the percentile delays recomputed. Our simulations show that on average only 1.4% and 0.7% of samples need to be reevaluated for exact recomputation of the 95th percentile and 99th percentile delays after performing SH-QMC.

II.6 Summary

This chapter presents a Stratification + Hybrid Quasi Monte Carlo (SH-QMC) approach to improve the efficiency of MC based statistical static timing analysis. The proposed approach uses easily computable timing criticality information, and achieves on average 23.8X and up to 44X reduction in the number of samples required for timing

Table II.4. Performance of incremental evaluation of 95th and 99th percentile delay with gate size change for SH-QMC with 80 samples. AT=Arrival Time

Circuit	No of gates	Avg. Incremental evaluations per gate		Avg. Incremental evaluations per gate/sample size(%)	
		95th percentile AT	99th percentile AT	95th percentile AT	99th percentile AT
VD1	14503	1.515	0.51	1.89	0.64
VD2	34082	0.54	0.515	0.68	0.64
USB	32898	1.625	0.57	2.03	0.71
ETHER	57327	0.96	0.535	1.20	0.67
VGA	90831	0.84	0.505	1.05	0.63

estimation compared to a random sampling approach. With multithreading on a quad core processor for SH-QMC, the approach is faster than traditional SSTA for comparable accuracy. Also, further speedup of SH-QMC is straightforward using parallel processing across machines. In addition, SH-QMC scales better than traditional SSTA with circuit size. Our approach estimates the 99th percentile arrival time better than traditional SSTA for benchmark circuits studied using only a low number of samples. We also propose an extension to SH-QMC to consider graph pruning based on information obtained from sample evaluation. This additionally reduces the runtime for SH-QMC by 48% on average. We proposed an incremental approach to recompute a percentile delay metric after ECO. The results show that on average only 1.4% and 0.7% of original samples need to be evaluated for exact recomputation of the 95th percentile and 99th percentile delays after ECO.

CHAPTER III

EFFICIENT SMART MONTE CARLO BASED SSTA ON GRAPHICS PROCESSING UNITS WITH IMPROVED RESOURCE UTILIZATION

III.1 Introduction

Recent strides have been made in the development of throughput-optimized processors, such as Graphics Processing Units (GPUs). Throughput processors recognize two crucial aspects of machine organization which are *parallel execution* and *hierarchical memory organization*. To increase performance in throughput processors, applications will need to expose parallelism while finding locality in their computations to overcome restrictions arising from communication bandwidth bottlenecks. In this work we show the importance of these two aspects for improving performance and efficiency in the context of statistical timing analysis by drawing inferences from the implementation on a specific GPU architecture.

We focus on a Monte Carlo based SSTA (MC SSTA) approach to statistical timing analysis, which involves analyzing samples of the process variation space to obtain statistical distributions of circuit timing behavior. MC techniques are “embarrassingly parallel” and have inherent advantages over traditional SSTA in exposing parallelism for performance improvements in throughput processors. However, the large number of

samples required in a standard MC approach of random selection of samples leads to high runtimes.

An effective solution to address the high runtime cost of MC SSTA is to use techniques to reduce the sample size. Some such techniques have been discussed in Chapter II. As mentioned, sample size reduction is achieved using a combination of standard techniques in statistics called variance reduction techniques [28] and the use of circuit specific information.

In this work we draw upon these advancements in MC SSTA. However, with a reduced sample size the objective of exposing parallelism for performance on throughput processors poses new challenges. In the case of GPUs, the level of parallelism required in the application is much higher than the units of parallelism to hide bottlenecks including memory access time. In a random sampling based approach the sample size is in the range of tens of thousands, about two orders of magnitude higher than the units of parallelism available in the GPU hardware. Therefore, this enables high utilization of resources on the GPU simply by performing computations on the samples in parallel. An implementation of random sampling MC SSTA on GPUs was explored in [58], where it is illustrated that such an implementation is sufficient for adequate resource utilization. However smart sampling algorithms can achieve accurate results with a sample size that is typically in the range of 100-200, which is the same order of magnitude as the hardware parallelism available on a GPU. This reduction in sample size cannot be translated to a corresponding reduction in runtime for a GPU with such a straightforward implementation. In addition to enabling fast statistical timing analysis of chips with millions of gates, this additional improvement opens up possibilities for using SSTA in a design optimization loop.

The main contribution of this work is to illustrate performance and efficiency improvements in the context of smart sampling based MC SSTA by recognizing the aspects of *parallel execution* and *hierarchical memory organization* in throughput processors. This translates to the following key ideas leading to the implementation.

Expose more parallelism. In the context of smart sampling based MC SSTA, gates in a circuit that do not depend on each other for input data given the computations already performed can be analyzed in parallel, leading to data parallelism or *gate parallelism*. We propose a smart scheduling algorithm for allocation of gates to parallel threads to make use of this parallelism. We show that exposing gate parallelism is crucial to achieving *parallel execution* on GPUs in the context of smart sampling based MC SSTA.

Find locality in computations. Finding locality in computations is critical to avoid restrictions arising from communication bandwidth bottlenecks. We lump together computations that are manageable within the fast local memory to avoid bottlenecks from accessing slow global memory.

We attempt to illustrate these general principles for throughput processors through an implementation of the smart sampling based MC SSTA technique called SH-QMC (Stratified Hybrid + Quasi Monte Carlo), which was proposed in [29], on Nvidia's CUDA-based GPU platform. Though the implementation itself is specific to the platform, this serves to illustrate the effectiveness of these concepts. The algorithm in [29] achieves a significant reduction in the number of samples needed to achieve accurate timing results while also considering a detailed process variation model incorporating within die variation. We compare the proposed implementation of SH-QMC with a straightforward sample level parallelism approach. Average speedups over random sampling MC SSTA

improve from 11.2X to 192.5X for the two implementations of SH-QMC on benchmark circuits ranging from 15,000 to 60,000 gates. When the GPU system is compared with a CPU an average speedup of 153X is achieved. The average runtimes normalized to a single STA on a CPU is 0.46, pointing to the result that *smart sampling based MC SSTA on a GPU is faster than a single STA on a CPU*.

A second contribution of this work is a critical graph analysis technique to speed up MC SSTA. Nominal STA is used to identify gates with very low probabilities of falling on critical paths under process variation, and are pruned from further consideration, without impacting the accuracy of statistical timing analysis. This enables fast evaluation of circuit samples leading to a 6.8X runtime reduction for the benchmark circuits considered.

The chapter is organized as follows. Section 2 describes the important relevant hardware and software features in GPUs. Section 3 briefly discusses the SH-QMC algorithm for smart sampling based MC SSTA. Section 4 describes the implementation of MC SSTA on GPUs and proposes techniques to achieve resource utilization when mapping SH-QMC onto GPUs. Section 5 discusses the critical graph analysis technique. Section 6 presents results and the paper concludes in Section 7.

III.2 CUDA Platform

NVIDIA CUDA (Compute Unified Device Architecture) is a general purpose parallel computing architecture that is easily programmable and exhibits good performance in scientific applications [59]. The CUDA architecture is built around multiprocessors, each consisting of several scalar processor (SP) cores. From a software perspective, threads are the basic unit for parallel computation and the code they execute is called the kernel. A

thread block, also referred to as a block, contains a batch of threads. Threads in the same block can efficiently share information through shared memory and run on the same multiprocessor. Within a block, 32 consecutive threads are grouped into a *warp*. All threads in a warp follow the exact same sequence of instructions. CUDA threads have accesses to multiple memory spaces during their execution. Global memory has the largest size but also exhibits long access times compared to other on-chip memory. The CUDA warp consists of two half-warps of 16 threads each. If all 16 threads of a half-warp access consecutive words from global memory, the overhead is significantly lower than when non consecutive words are accessed [60]. Other types of fast on-chip memory in the targeted CUDA architecture include *register memory* and *shared memory*. Shared memory can be shared within a block and is significantly faster than global memory.

III.3 Smart Sampling based SSTA: SH-QMC

We propose to implement the smart sampling based MC SSTA approach SH-QMC (Stratification + Hybrid Quasi Monte Carlo), proposed in Chapter II, on GPUs. This algorithm significantly speeds MC based SSTA using sample size reduction. In this technique, circuit timing criticality information is used for intelligent selection of samples. It is shown that 100-200 samples are sufficient for accurate statistical timing analysis. The process variation model is based on [7], which considers intra-die spatially correlated variation by partitioning the die into $n * n$ grids and assuming identical parameter variations within a grid. SH-QMC uses a combination of standard techniques and circuit timing criticality information to reduce sample size for MC based analysis (variance reduction techniques). The variance reduction techniques employed are Quasi Monte

Carlo (QMC), stratified sampling, and Latin Hypercube Sampling (LHS). These techniques are employed on variables based on their convergence properties and the ability to handle multiple variables. A detailed analysis of the algorithm is presented by in Chapter II.

III.4 Monte Carlo based Statistical Timing Analysis on GPUs

This section describes techniques for efficient implementation of MC SSTA on GPUs. In a random sampling based MC approach, samples of the chip are generated using process variation information. These samples have no data dependence on each other and therefore are directly amenable to parallelism. This is referred to as sample parallelism. Each thread is dedicated to the computation of one sample (representing one virtually fabricated die) of the circuit. Gates are visited in the topological order in the circuit by a thread and delay computations are performed. For the computation of process variation samples we use a Mersenne Twister based random number generator [58]. A detailed discussion is omitted for brevity.

III.5 Enhanced resource utilization for implementation of SH-QMC on GPU

Sample parallelism is sufficient to keep resources utilized on GPUs when employing random sampling [58]. However the sample size in SH-QMC is typically only 100-200, which is comparable to the number of streaming processors available in GPUs. A straightforward implementation in the spirit of the approach in [58] leads to under-utilization of resources. In this section, we describe techniques which adhere to the two

key ideas for performance and efficiency in throughput processors introduced in Section 1. With this improved resource utilization, performing SSTA repeatedly in a design optimization loop with hundreds of thousands of iterations becomes a possibility for moderately sized circuits.

a. *Parallelism - exposing gate parallelism*

For gates with no data dependence gate delay calculations can be performed in parallel. To leverage this parallelism we propose static scheduling of gates in the circuit using a scheduling algorithm prior to performing statistical timing analysis. A schedule table assigns gates to levels such that gates at a given level are assigned to parallel threads only after computations on previous levels have been completed (Figure III.1). All threads working on gates in the same sample are grouped together within a CUDA block. A block consists of threads that can efficiently communicate with each other using shared memory. Threads working on different samples of the circuit are grouped into different blocks, allowing flexible use of memory and resources. For assigning gates to the schedule table, we propose two algorithms - *Algorithm 1* and *Algorithm 2*. The pseudocode for both algorithms is presented in Figure III.2. In Algorithm 1, the gates are sorted in topological order. A gate is ready to be scheduled when all its fanin gates have been scheduled in previous levels of the schedule table. All ready gates are assigned to levels such that the number of gates per level does not exceed the key parameter *MaxPerLevel*. Algorithm 2 performs smart scheduling of gates to reduce the total number of computation steps. In this case gates that are ready to be scheduled are preferentially grouped together in a level based on three criteria:

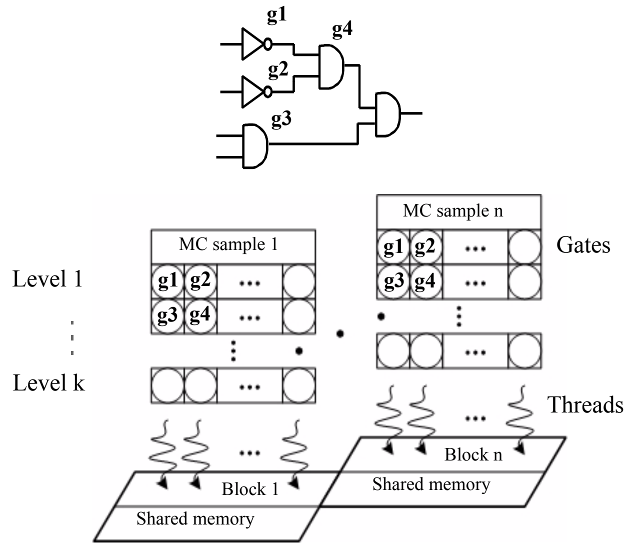


Figure III.1. Gate scheduling. Gates in a sample with no dependence are computed in parallel. In graph shown, gates g1,g2,g3 have no dependence and can be assigned to the same level. However, g3 is a 2-input gate which if assigned to the same level as g1 and g2, increases the computational steps in the level. Therefore, g3 is assigned to the next level along with gate g4.

Fanout count: Gates with large fanouts are assigned a higher preference for scheduling. This allows more freedom for gate choices in subsequent levels where more gates are likely to have their fanin gates already scheduled.

Global memory access: Gates with common fanin gates are grouped together to avoid redundant fetching of delay data from slow global memory.

Pin count: Gates with the same or similar number of inputs are assigned to the same level. Gates with higher input pin counts involve more delay computation steps. Since all threads within a CUDA warp are forced to perform the same number of computations, all threads in a warp complete at the same time as the thread for the gate with the largest pincount. Therefore, grouping together gates with lower input counts in the same level leads to speed up.

```
Scheduling Algorithm 1
Topologically sort gates in the circuit
queue ReadyGate
Level = 0
while all gates are scheduled
{
   for all gates g not scheduled, if g is ready
         Add g to queue ReadyGate
   while ( size of Schedule[Level] < MaxPerLevel .AND. ReadyGate is not empty)
            Add to list Schedule[Level] ( ReadyGate.pop() )
    Level ++
}
return schedule
```

```
Scheduling Algorithm 2
Topologically sort gates in the circuit
queue ReadyGate
Level = 0
while all gates are scheduled {
   for all gates g not scheduled, if g is ready
         Add g to queue ReadyGate
   faninlist = {}
   faninM = 1
   while size of Schedule[Level] < MaxPerLevel .AND. ReadyGate is not empty {
          Find g in ReadyGate to maximize Weight(g,faninlist,faninM)
          Add g to Schedule[Level]
          Remove g from ReadyGate, Add fanin gates of g to faninlist
          faninM = max( fanincount(g), faninM )
    }
    Level ++
}
return Schedule
```

```
Weight(g,faninlist,faninM)
Fanout = fanoutcount(g)
ExtraFanin = number of fanin gates of g not in faninlist
FaninDiff   = max( fanincount(g) - faninM, 0 )
Weight =  Fanout - CoeffMem*ExtraFanin - CoeffDelayStep*FaninDiff
```

Figure III.2. *Algorithm 1* and *Algorithm 2* for gate scheduling.

Given the list of ready gates, a gate *g* is selected for scheduling to the next level such that a linear sum of costs based on the above three criteria is maximized. *ExtraFanin* is the number of fanin gates of *g* that are not already fanins for other gates in the current level. The algorithm tries to select gates with low values of *ExtraFanin* to minimize the total memory accesses from global memory required to perform computations on gates in the current level. *FaninDiff* is positive if the new gate selected has more pins than the gates in the current level. A maximum of *MaxPerlLevel* gates are selected per level and the list of ready gates is updated before gates are allocated to the next level in the schedule table.

b. *Localizing computations in shared memory*

Since global memory has much higher latency and lower bandwidth than on-chip memory, global memory accesses should be minimized. Shared memory is a fast on-chip memory resource and therefore ideal for storing all intermediate information. However, the shared memory size for each multiprocessor is small (16KB in typical CUDA architectures), which is small compared to global memory. The maximum size of shared memory allocated to a block is no larger than 16KB, which is not sufficient to store all intermediate information in practical sized circuits. Hence global memory is used to store the delay information. To minimize access of this data from global memory, we propose a technique to localize computations such that they are manageable within the limits of shared memory. As mentioned in Section 4.2a, the circuit is scheduled into multiple levels in a schedule table to expose gate parallelism. We group N levels into one entity or subcircuit, where the parameter N is a function of the shared memory size. Before gates in the first level of the subcircuit are scheduled, input data for this subcircuit is loaded into the shared memory. When gates in subsequent levels require input data already accessed or computed by gates in the previous levels within the same subcircuit, this is accessed from the shared memory. This minimizes access of data from global memory. In addition, while computations on gates in the current subcircuit are being performed, the algorithm fetches data required for the next subcircuit (defined by the next N levels) if not an output of the current computation. This allows overlapping of memory access steps and arithmetic computations so that global memory latency is effectively hidden.

Figure III.3 summarizes these techniques. Computations for one circuit sample are illustrated in the figure. The schedule table resides in global memory. Gates in the current

level of the schedule table are assigned to different threads. Process variation samples for the gates are computed in parallel and gate delay computations are performed. The input delay information required for the computations is accessed from *current_subckt* in shared memory. Here *N* levels in the schedule table are grouped together. The *current_subckt* in shared memory consists of all input information required by gates in *N* levels including the current level. This data was previously loaded into the shared memory from global memory. While arithmetic computations are performed one level at a time on the set of *N* levels, delay information for the next set of levels *i+N* to *i+2*N-1* is loaded from global memory to shared memory as illustrated. The table *next_subckt* stores this data in shared memory. This allows better hiding of latency for the global memory access within each

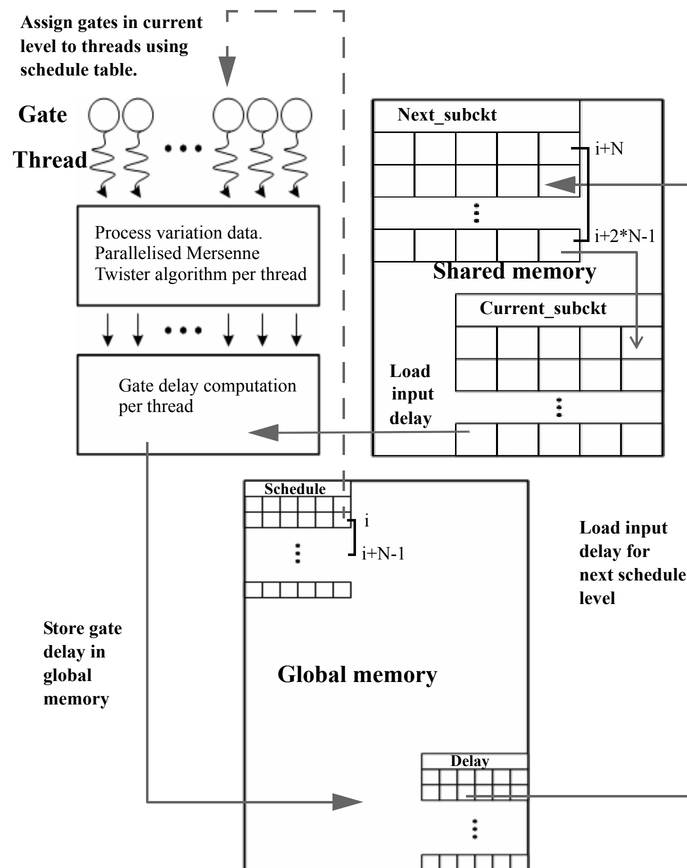


Figure III.3. Summary of proposed approaches to improve resource utilization. Concurrent computation on gates in the same level and use of shared memory are illustrated.

thread. Also, access of data for *N* levels at a time avoids repeated accesses from global memory. The output information from the current delay computations are stored in the delay table in global memory and also updated appropriately in the table *next_subckt*.

III.6 Critical Graph Analysis for MC SSTA

In this section we propose a technique to improve the performance of SSTA through critical graph analysis. The basic idea is to identify critical paths in the graph by performing heuristics. In other words, gates which are expected to have a negligible effect in determining the worst case arrival time of the circuit can be pruned or avoided from consideration in subsequent analyses, leading to speedups in the overall statistical analysis. In the context of variability, criticality is statistical. The challenge here is to assign probability values to gates/paths in the circuit based on a measure of criticality. In [61], the authors propose an algorithm to compute criticality probability of gates in the circuit. This algorithm computes criticality accurately, however it can potentially add a significant runtime overhead to the SSTA. It may be noted that the proposed critical graph analysis technique only requires that all sufficiently critical gates be selected for accuracy in subsequent SSTA. The exact values for criticality probability are not required in the further analysis. Therefore, we propose a simpler technique for critical graph identification. We propose that slack information obtained from STA performed at the nominal process corner be used to identify the critical graph. The timing overhead for this technique is significantly lower.

III.6.1 Nominal STA based Critical Graph Identification

This technique uses information obtained from timing analysis of the circuit at the nominal process corner. The example in Figure III.4 illustrates the technique. Nominal STA is performed and slack information is obtained at all gates in the circuit. Gates with significant slack, in this case higher than a threshold value of 0.3 are excluded from consideration when applying MC based SSTA. The reduced graph size will allow the runtime-dominant MC STA runs to be reduced roughly linearly with circuit size. The threshold slack is defined as *s%* of the worst arrival time at the nominal sample, where *s* is the pruning parameter. For instance, s is 30% in the above example if circuit delay is 1 unit.

III.7 Results

We implement the proposed approach on an Nvidia Tesla S1070 GPU with a 3.16 GHz Intel Xeon-based Linux machine serving as the host. The GPU system has 4 GPU cards totalling 960 streaming processor cores [62]. Results in this section are based on a 65nm commercial technology library. In our implementation we only consider channel length variation as a source of process variation for simplicity, however other sources can

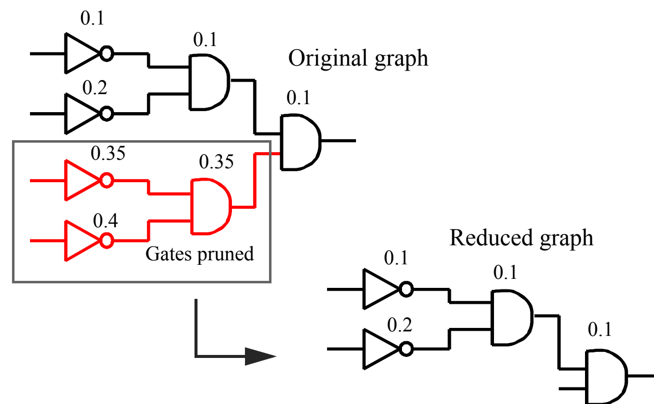


Figure III.4. Illustration of graph reduction. Slacks for nodes are indicated next to corresponding gates. Gates with slacks higher than a threshold of 0.3 at output node are removed to obtain the reduced graph in the example.

Table III.1. Comparison of runtime for SH-QMC (192 samples) vs. random sampling based MC SSTA on GPU. SH-QMC is implemented with (a) sample level parallelism or SP (b) sample + gate parallelism or SGP. (c) SGP + efficient shared memory usage or SGP+S.Mem.

Circuit	# of Gates	RS 50k (ms)		SH-QMC on 1 GPU (ms)/Speedup w.r.t RS on 1 GPU			SH-QMC on 4 GPU (ms)/Speedup w.r.t RS on 4 GPU		
		1 GPU	4 GPU	SP	SGP	SGP+S.Mem	SP	SGP	SGP+S.Mem
VD1	**14503**	**4630**	**1620**	155/30X	32/147X	**28/164X**	148/11X	15/109X	**13/123X**
VD2	**34082**	**10870**	**3810**	366/30X	70/155X	**64/170X**	349/11X	33/116X	**30/128X**
USB	**32898**	**11360**	**4050**	364/31X	71/160X	**65/175X**	344/12X	17/232X	**16/256X**
Ether	**57327**	**19370**	**6810**	634/31X	119/163X	**106/182X**	600/11X	29/233X	**26/263X**

be readily implemented. The inter-die, spatially correlated intra-die and uncorrelated random components of channel length variation are considered. The overall standard deviation is 10% of nominal channel length. Simulations are performed on four large circuits, Viterbi Decoder 1 (VD1), Viterbi Decoder 2 (VD2), USB 2.0 Core (USB), and an Ethernet MAC Core (ETHER), with gate counts varying from approximately 15,000 to 60,000. We perform synthesis and APR on all the circuits using commercial tools.

Table III.1 compares the implementation of a random sampling based MC SSTA approach with the SH-QMC approach, both on the Tesla S1070 GPU system. The results indicate that a straightforward implementation of SH-QMC exploiting sample level parallelism does not lead to speedups corresponding to the reduced sample size w.r.t a random sampling based approach, whereas much higher speedups are obtained using the proposed techniques to improve resource utilization for smart sampling techniques. The sample size used in the random sampling based approach is 50,000. The number of samples used in the SH-QMC approach is 192 (the exact number is related to the granularity of sample size in the SH-QMC approach based on [29]). The SH-QMC approach is implemented in three variants:

1) SP: Only sample level parallelism is considered. This is based on the implementation in [58]. An active thread is dedicated to computations on a single sample;

2) SGP: Multiple threads perform computations on a sample by exploiting gate parallelism;

3) SGP+S.Mem. In this case shared memory is utilized and prefetching of data is performed.

We define the *efficiency* of the SH-QMC implementation as the runtime per sample of the random sampling approach divided by the runtime per sample for the SH-QMC approach. If the reduction in sample size for SH-QMC w.r.t random sampling could be translated into a corresponding reduction in runtime by the same factor then the efficiency is 100% according to the definition. In the GPU implementation we show results using both a single GPU card and four GPU cards in the S1070 GPU system. For the single GPU implementation, the speedup of the implementation compared to a random sampling approach increases from 30.5X for sample parallelism (SP) to 172.7X for SGP+S.Mem. This demonstrates that exposing gate parallelism and exploiting shared memory greatly increases resource utilization in the GPU for smart sampling based MC SSTA. The efficiency metric for the implementation increases from 11.7% for SP to 66.3% for SGP+S.Mem. When all cards in the multi-GPU system are used, we achieve 192.5X speedup on average for SGP+S.Mem compared to 11.2X for SP. In this case the runtime improvement is more pronounced compared to the case of a single GPU, since more resources are available per sample, leading to even lower resource utilization without the proposed techniques. The efficiency metric increases from 4.3% for SP to 73.9% for SGP+S.Mem for SH-QMC. Figure III.5 illustrates the trend in performance improvement

versus sample size for SP and SGP+S.Mem. As the sample size decreases the performance improvement with resource utilization increases significantly, underlining the synergy of these techniques with smart sampling based MC techniques.

Table III.2 compares the efficiency of SH-QMC with 192 samples on a CPU versus a GPU system. The results are also shown for a single STA run on the CPU. On average a 48X speedup is achieved for a single GPU card over a quad core CPU. The average speedup is 153X when the multi-GPU system is compared with the quad-core CPU. The runtimes normalized to that of a single STA on a CPU are 1.26X and 0.46X, respectively, for a single GPU and multi-GPU. Thus MC SSTA runtime on a GPU is comparable to that of a single STA run on a CPU.

As shown in Table III.1, the runtime for SH-QMC on the Ether circuit (57K gates) is only 600ms on a GPU even with a simple implementation using sample parallelism. Extrapolating from this data, this means that we can perform statistical analysis on large designs with millions of gates with low runtime. In addition, for circuits of similar sizes, the additional 20X improvement with the proposed approaches opens up possibilities for

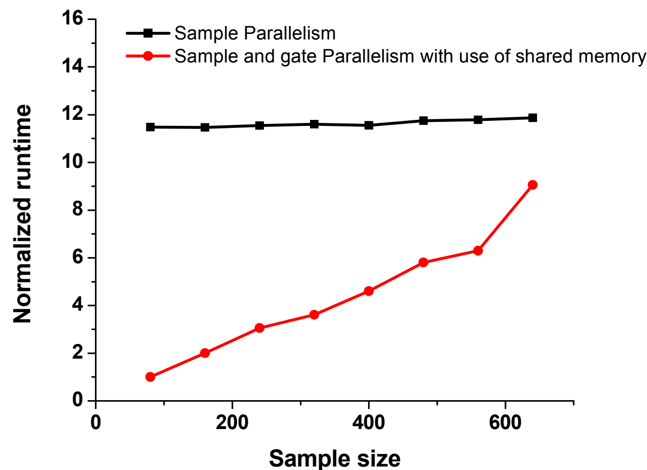


Figure III.5. Improvement in runtime due to the techniques proposed for improved resource utilization.

Table III.2. Comparison of runtime for SH-QMC (192 samples) on GPU vs. CPU. and single STA on CPU. The CPU is a 3.16GHz Intel Xeon processor.

Circuit	Single STA on CPU	SH-QMC on quad core CPU	SH-QMC on GPU 1 card/Speed up w.r.t. quad core	SH-QMC on GPU 4 cards/ Speed up w.r.t quad core CPU	Runtime of SH-QMC on Tesla GPU norm. to CPU STA	
					1 card	4 cards
VD1	20ms	1.1s	28ms/39X	13ms/85X	1.4	0.65
VD2	50ms	2.9s	64ms/45X	30ms/97X	1.28	0.6
USB	50ms	3.2s	65ms/49X	16ms/200X	1.3	0.32
Ether	100ms	6.0s	106ms/57X	26ms/231X	1.06	0.26

the use of SSTA in a design optimization setting where the analysis can be performed repeatedly in a loop for better quality of results. To illustrate the basic idea we demonstrate a simple experiment where SH-QMC is performed in tandem with gate sizing. Here, we select gates randomly for sizing, and perform SH-QMC after every sizing step. This is repeated for 100,000 sizing steps and the runtime is reported in Figure III.6. The proposed approach for implementation of SH-QMC (SGP+S.Mem) is compared with the sample level parallelism based implementation (SP) for the benchmark circuits studied. For the largest circuit with 57K gates, the runtime is reduced from 16.7 hours to 42 minutes with the proposed approach. In a similar spirit, we compare the runtime of SH-QMC on both CPU and GPU with STA on a CPU, in Figure III.7. The analyses are performed in a loop involving 100,000 sizing iterations for the Ether circuit (57K gates). The runtime for the cases of SH-QMC on a CPU and STA on a CPU are 166 hours and 2.7 hours respectively, compared to 42 minutes for the proposed SH-QMC implementation on GPU.

Figure III.8 illustrates the runtime improvement versus the degree of gate parallelism when using scheduling algorithm 2 from Figure III.2. The improvement in runtime

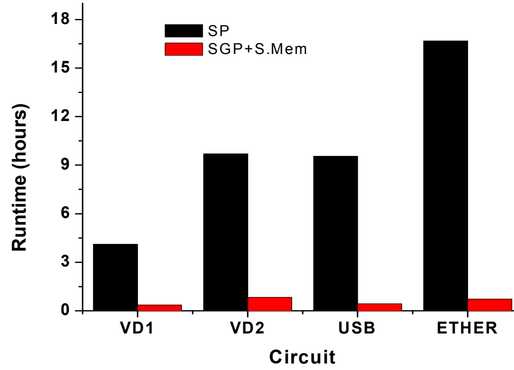


Figure III.6. Comparison of runtime for SH-QMC performed with successive gate sizing (100,000 sizing steps) on 4 GPU cards for Ether circuit. Runtime for an implementation utilizing sample level parallelism (SP) is compared with the proposed approach (SGP+S.Mem).

Table III.3. Quality of results for the critical graph analysis technique.

Circuit	% gates pruned	% of samples with zero error	Error in 99th percentile AT
VD1	65.4	99.74	0.018
VD2	43.2	99.60	0.015
USB	81.7	98.60	0.080
Ether	89.3	98.82	0.045

saturates around a gate parallelism of about 200. The discontinuity in the graph at a gate parallelism of 64 is because an additional warp is required in the block to accommodate a new thread in this case (64 is a multiple of 32, the warp size in CUDA). Beyond this point, each block requires more registers (for the new warp) and the number of registers required per multiprocessor exceeds the capacity. Therefore, the number of blocks that can be active per multiprocessor is reduced. This leads to the runtime overhead.

We evaluate the accuracy of the critical graph identification approach in Table III.3. The second column shows the percentage of gates pruned from consideration after critical graph analysis. We perform experiments on 80,000 samples where the possible error arising from critical graph analysis is computed at each sample. The third column shows

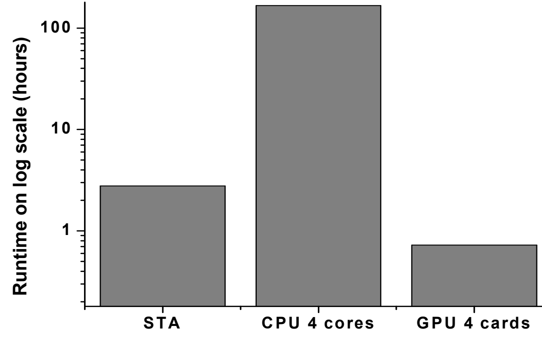


Figure III.7. SH-QMC with 192 samples on GPU is compared with SH-QMC on CPU and STA on CPU, on a logarithmic scale, for Ethernet circuit with 57327 gates. SH-QMC on multi-GPU is faster than STA on CPU.

the percentage of samples which incur absolutely no error. The error in computation of the 99th percentile of the circuit delay distribution (using 80,000 samples) is shown in the fourth column. We see that across all the benchmarks studied, more than 98.6% of samples incur absolutely no error. Also, the error in computation of the 99th percentile of worst arrival time is negligible. Table III.4 illustrates the results from graph reduction for the benchmarks studied. The point on the sizing curve such that the hardware intensity (defined as the magnitude of the ratio of percentage change in power to percentage change in timing constraint) is 1 is selected for analysis in the table. On average a 6.8X speedup is achieved through the new pruning approach on the GPU implementation of SH-QMC, which is orthogonal to the speedups described above.

Table III.4. Runtime improvement from graph reduction combined with the proposed technique.

Circuit	# of Gates	% gates pruned	S.G.P+ S.Mem (ms)	S.G.P.+ S.Mem+ G.Red (ms)	Speedup due to graph reduction
VD1	14503	65.4	13.1	4.5	2.9
VD2	34082	43.2	29.7	16.9	1.8
USB	32898	81.7	15.8	2.9	5.5
Ether	57327	89.3	25.9	2.7	9.4

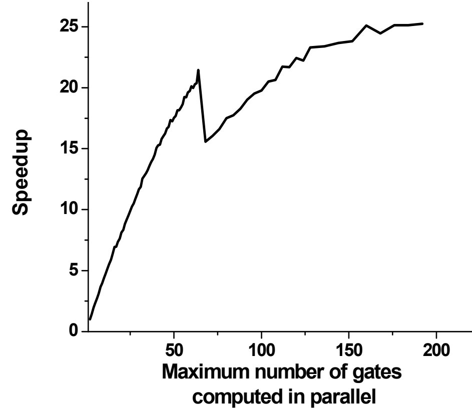


Figure III.8. Speedup of SH-QMC algorithm implemented on multi-GPU for a USB circuit (14,503 gates) with smart scheduling algorithm Algorithm 2. X axis indicates the maximum number of gates computed in parallel. A discontinuity in resource requirements above a parallelism of 64 leads to the discontinuity in the graph.

III.8 Conclusions

We present an implementation of smart sampling based MC SSTA on a GPU system. We show that a straightforward implementation of smart sampling that exposes only sample parallelism under-utilizes resources in the GPU, in contrast to random sampling based MC SSTA approaches where this type of parallelism is sufficient. We propose several techniques to achieve high resource utilization for the case of smart sampling based MC SSTA, particularly gate parallelism and enhanced use of shared memory. While sample parallelism leads to only 11.2X speedups using 192 samples compared to random sampling with 50,000 samples, our techniques lead to 192.5X speedups for the same comparison. In terms of an efficiency metric, the proposed techniques achieve an efficiency of 73.9% for smart sampling MC SSTA compared to a modest 4.3% for sample parallelism. Most significantly, MC SSTA runtime on a multi-GPU is shown to be over twice as fast as a single STA run on a CPU. This work also proposes a critical graph

analysis technique to further speedup MC SSTA, achieving a 2-9X speedup on several benchmarks.

CHAPTER IV

A LOWER BOUND COMPUTATION METHOD FOR EVALUATION OF STATISTICAL DESIGN TECHNIQUES

IV.1 Introduction

Recent research has paid significant attention to statistical design techniques, where the focus is on optimizing designs to increase their robustness under variability while minimizing any increase in cost (e.g., area or power). Optimization at worst-case corners leads to pessimism and large guardbands. Therefore, smart deterministic algorithms were introduced where the key observation is that the pessimism incurred by worst-case corner approaches is mainly due to an inability to set appropriate guardbands (i.e., timing constraints) for optimization in a deterministic setting, rather than the quality of the algorithm itself [66-69]. Therefore, it is sufficient to augment a deterministic algorithm with statistical analysis to minimize pessimism in the timing constraint used for optimization. The authors in [69] summarize these approaches and propose a technique to capture WID variation effects without explicitly modeling them in the optimization procedure. Here, a circuit-level guardband for the target timing constraint is used to capture WID (within-die) variation effects. A conventional transistor sizing algorithm similar to [70] which captures only D2D effects, is then used to minimize cost to meet this guardbanded target. The appropriate guardband is found by sweeping its value and

repeating the conventional optimization followed by SSTA at each point, until timing yield is met.

Other techniques have focused on explicitly capturing statistical sensitivities at the device/gate level or introducing spatial correlation-aware margins at the device level to reduce the pessimism found in worst-case corner optimization [8,71-74]. The authors in [8] propose a yield optimization technique using a gradient-based non-linear optimizer. An efficient heuristic is proposed to compute yield gradient. In [71], the robust statistical optimization problem is formulated as a second order conic program (SOCP). A linear relationship between delay and parameters affecting variability is assumed. A piecewise linear gate delay model as a function of gate size is constructed to enable the problem formulation as an SOCP. In robust geometric programming (RGP) [72], a worst-case corner approach is used to incorporate process variation effects. A Geometric Programming model is used to capture the delay of gates as an analytical function of design parameters and parameters representing parasitic effects. The effect of variations is included by adding appropriate margins to the delay constraints at the output pin of each logic gate.

These algorithms provide reasonable solutions to the problem at hand, which is to arrive at a design that meets performance requirements with sufficient confidence given process variability.

However it is not clear the scope of improvement possible by using statistical optimization approaches rather than smart deterministic algorithms such as [69]. Given the additional runtime costs of the fully statistical approaches, it would benefit designers

and future researchers to know the potential improvements available to statistical techniques.

The major contribution of this work is a lower bound computation method to compare the Quality of Results (QoR) of statistical design techniques under process variations. This method takes its inspiration from recent developments in statistical timing analysis where a sample-level view of the process variation space is taken [29,33,56]. We show that the lowest cost for any design to meet a specified timing yield objective is bounded by a theoretical limit. This limit is related to the exact solution obtained if different samples in the process variation space were to be optimized deterministically. Given a set of samples in the variation space, the optimal design to meet a deterministic timing T while fixing the process parameters at each sample can be obtained using an exact optimization technique. We show that for a large enough sample size, the xth percentile of the cost (area, power) of the designs obtained by optimizing each individual sample to meet timing constraint T is a lower bound for any design that meets T with a specified timing yield of x%.

Second, in the same spirit as the lower bound computation, we propose a statistical design technique that draws upon a sample level perspective of the variation space - Sample Level Optimization in Parallel (SLOP). As the name indicates, different samples in the process variation space are optimized in parallel using two phases. In the first phase, a straightforward deterministic optimization is performed for each sample. These results are then fine tuned in the second phase by shifting the focus of the algorithm to the optimization of an intelligently selected high percentile sample taken from phase one, such that the timing yield is met. Among the solutions thus obtained at each sample, the

lowest cost solution is selected. The technique is highly amenable to parallelism on multi-core machines and GPUs.

We compare the results obtained from SLOP and two other techniques proposed in literature, Burns [69], and Robust Geometric Programming or RGP [72], against the lower bound computed using the proposed technique. We show that the solutions obtained from SLOP and RGP are close to the theoretical limit for the cost. Further improvements possible to solutions computed by the Burns, SLOP and RGP methods are at most 9.6%, 7.5% and 3.7% respectively, on average for the benchmark circuits studied. The improvement in quality of solution for SLOP and RGP comes at the cost of an average of 7.4× and 41.5× increase in runtime, respectively compared to the Burns approach. Therefore, we conclude that smart deterministic approaches are sufficiently accurate while incurring low runtime cost. At the same time both Burns and SLOP also offer possibilities for massive parallelization on GPUs and multiprocessor systems.

The rest of the paper is organized as follows. Section 2 discusses previous work on exact optimization of a design at a fixed process corner. Section 3 explains the proposed technique to obtain a theoretical limit for the cost of a design to meet a given timing yield. Section 4 describes the proposed Sample Level Optimization in Parallel (SLOP) approach. Section 5 presents results and conclusions are presented in Section 6.

IV.2 Exact Deterministic Optimization

Given the growing parallelism available in modern CPUs and GPUs, there is interest in using sample-based approaches to perform statistical timing and power analysis and optimization. When examining specific samples in the process variation space these

approaches rely on finding the optimal solution for that given sample (die). This section discusses the existing literature on exact optimization of a design at a given point in process space. It is theoretically possible to obtain cost lower bounds at a process corner using branch and bound or simulated annealing techniques. However, in practice, these techniques exhibit very high runtimes. Other techniques to obtain the optimal design at a process corner make assumptions regarding the gate delay model. In [75], the authors propose an approach for exact transistor sizing. The method involves two phases. In the first phase, called the D-phase, incremental changes in delay are assigned to each node of the circuit graph. This is formulated as the dual of a min cost network flow problem and is solved exactly. In the second phase, or the W-phase, feasible transistor sizes are calculated to incorporate the node delay changes assigned in the previous phase with minimal increase in cost. This phase assumes that the transistor delay is expressible as a sum of simple monotonic functionals. A simple monotonic function applied to this case has the property that it is monotonically decreasing in one transistor parameter and montonically increasing in all other transistor parameters. Reference [76] discusses a Geometric Programming approach to solve circuit design optimization problems. Geometric Programming (GP) models can address a wide variety of integrated circuit design problems [76]. Also, commercial solvers are available that can handle large-scale GPs efficiently [77]. Therefore, we focus on a GP-based approach to obtain the cost lower bound for a design.

In a GP model, the objective function and the constraint functions are expressed as a general class of functions called posynomial functions. A posynomial function is a sum of monomials.

A GP can be expressed in the following form

$$
\begin{aligned}
&\textit{minimize } f_0(x) \\
&\textit{subject to :} \\
&f_i(x) \le 1 \qquad\qquad i=1,\dots,m \\
&g_i(x) = 1 \qquad\qquad i=1,\dots,p
\end{aligned}
$$

Here $f_i(x)$ are posynomial functions, $g_i(x)$ are monomial functions, and x_i are optimization variables.

Generalized Geometric Programs (GGPs) [76] extend the formulation to a more general class of functions called generalized posynomials. A generalized posynomial is any function that is expressible using the operations of summation, multiplication, positive (fractional) powers, and maximum of posynomials. An example of a generalized posynomials is illustrated below :

$$h_1(x) = f_1(x)^{a_1} f_2(x)^{a_2} \qquad\qquad a_1, a_2 > 0$$

where $f_1(x), f_2(x)$ are posynomials and $h_1(x)$ is a generalized posynomial.

The formulation of an optimization problem as a GGP requires modeling the objective and constraint functions as generalized posynomials. Techniques to approximate practical functions using generalized posynomials are discussed in [76]. In this work, we employ a Max-Monomial fitting approach for this purpose. A max-monomial function has the form

$$h(x) = \max_{k=1,\dots K} f_k(x)$$

where fk(x), k=1,…,K are monomials. A heuristic algorithm for finding the max-monomial approximation to a data set for a fixed K is provided in [76]. The algorithm selects a subset of the data for each monomial where the monomial is the maximum compared to the other monomials. The monomial fit for this subset of data is then improved using a simple least squares linear fit after performing logarithmic operations on both sides.

With the max-monomial representation for delay, the optimization problem at a process corner can be formulated as a GGP:

$$\text{minimize} \sum_{i=1}^{N} x_i$$

s.t.:

$$\begin{aligned} & AT_j + d_{i,j} \le AT_i \\ & d_{i,j} = h(x_i, s_j, p_{i,j}, \vec{r_i}) \end{aligned} \quad \forall j : e(j,i) \in E$$

$$AT_i \le T$$

where is a max-monomial function. xi is the size of the gate at node i, s_j is the slew at node j and $p_{i,j}$ represents the parasitic values at nodes i and j. r_i is the vector of process parameter values at the sample for the gate at node i. E is the set of edges in the circuit graph. T is the specified timing constraint, AT_i is the arrival time at node i, and $d_{i,j}$ is the delay of edge e(j,i).

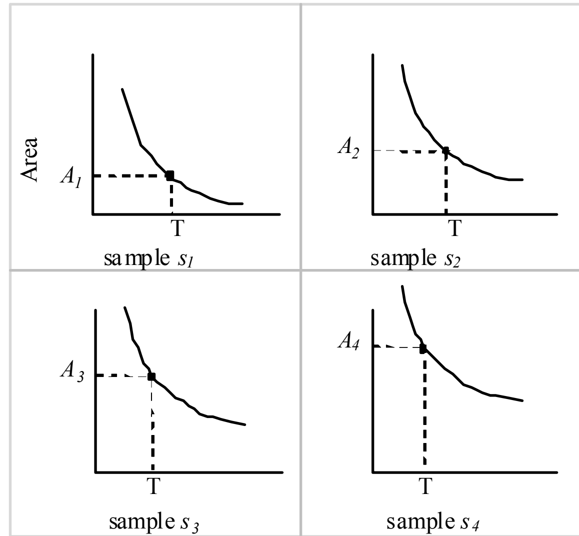
IV.3 Lower Bound For Design With Statistical Timing Yield Constraint

This section describes a new technique to obtain a lower bound for the typical case cost of a design while meeting a statistical timing constraint T with yield x. The lower bound is computed using results from independent exact optimization of samples in the process variation space. The exact optimization technique used at each sample was described in Section 2. We consider the distribution of costs obtained by exact optimization of each sample in a sample set that is sufficiently large and adequately captures the distribution of process parameters. We show that the cost of any design that meets the constraint T with timing yield x is higher than the xth percentile of the cost distribution.

Figure 1 illustrates the approach for a sample set S = {s_1, s_2, s_3, s_4}. Each of the samples in S are optimized to meet a timing constraint T with resulting (provably minimal) costs A_1, A_2, A_3, A_4, respectively. D is any design that meets the timing constraint T with respect to the sample set S with a yield of x=75% and area *A(D)*. Note that D meets the constraint T at samples s_1, s_2 and s_3. Therefore, *A(D)* must be higher than A_1, A_2 and A_3. However, the same cannot be said of A_4. Since the 75th percentile of the distribution A_i, i = 1...4 is at least equal to max(A_1, A_2, A_3), *A(D)* must be at least equal to the 75th percentile of distribution A_i with respect to set S. For a sample set S that captures the process parameter distribution accurately, this means that the area of any design D that meets constraint T with timing yield x is lower bounded by this value, subject to an error

related to estimation of timing yield using the sample set. A more rigorous approach considering this error is presented below.

Let S be a set of samples in the process variation space. $A_i(T)$ denotes the lowest



(a) Optimal area (exact) vs. timing constraint at samples in set S.

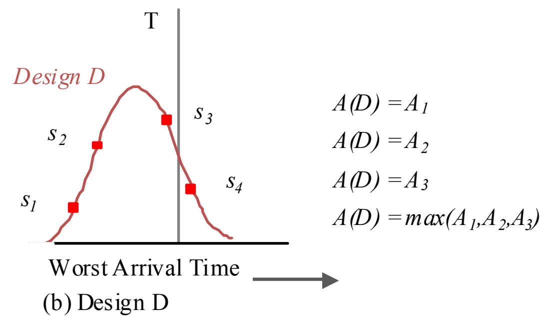


Figure IV.1. Illustration of Theorem 1. A_i represents the optimal cost solution at sample s_i. Design D meets timing constraint T for the 75th percentile of the given sample set. The cost of this design is *A(D)* and exceeds A_i, i=1,…,3 as D meets the timing constraint at samples s_i, i=1,…,3. Therefore, *A(D)* exceeds the 75th percentile of the distribution A_i, i=1,…,4 for the given sample set.

typical case cost for a design to meet timing constraint T at sample . Let $A_S^x(T)$ denote the xth percentile of the distribution $A_i(T)$ with respect to sample set S for constraint T.

n(S) denotes the number of elements in S. $t_S^x(D)$ is the xth percentile of the worst circuit delay for design D w.r.t. sample set S.

Theorem 1. Given a design D, the following is true:

$$t_S^x(D) \le T \quad \Rightarrow \quad A_S^x(T) < A(D)$$

In other words, $A_S^x(T)$ is a lower bound for the typical case cost of a design to meet T for the xth percentile of the worst arrival time distribution with respect to sample set S.

Proof: Let A(D) denote the nominal cost of design D. If D meets the timing constraint T w.r.t sample, then

$$A_i(T) \le A(D)$$

Let D satisfy T for the xth percentile of the worst arrival time distribution with respect to sample set S or $t_S^x(D) \le T$. In other words, D meets timing constraint T for at least x% of the samples in S. It follows from (2) that

$$\exists \mathrm{C} \subset \mathrm{S}: \quad A_i(T) \le A(D) \quad \forall s_i \in C$$

$$n(C) \ge n(S) * \frac{x}{100}$$

i.e., $A_i(T) \le A(D)$ holds for at least x% of the samples in S.

This in turn imposes the following constraint on the xth percentile of the distribution with respect to sample set S.

$$A_S^x(T) \leq A(D)$$

Theorem 2. Let S_n be a set of n samples in the process parameter space such that $\left|t_{S_n}^x(D') - t^x(D')\right| < \varepsilon$ for any design D' in the design space, where $t^x(D')$ is the xth percentile worst circuit delay of D'. For a given design D which satisfies $t^x(D) \leq T$, $A_{S_n}^x(T+\varepsilon) \leq A(D)$

Proof:

Given $\quad \left|t_{S_n}^x(D) - t^x(D)\right| < \varepsilon$

$$t^x(D) \leq T \;\Rightarrow\; t_{S_n}^x(D) < T + \varepsilon$$

$$\therefore A_{S_n}^x(T+\varepsilon) \leq A(D) \qquad \text{(Theorem 1)}$$

Note that the proof for Theorem 1 assumes $n(S) * \frac{x}{100}$ is an integer. This assumption can be removed easily. The proof is omitted for brevity.

Theorem 2 suggests a technique to obtain the lower bound on the cost of a design to meet a statistical timing yield constraint. This is summarized below for the case of x% timing yield at T.

Note that as the sample size $n \rightarrow \infty$, $\varepsilon \rightarrow 0$ and a closer lower bound is obtained. Also, smart sampling techniques have been proposed in literature to obtain moments and percentile values of the circuit delay distribution with a low sample size, and the error incurred in estimation of the moments and percentile values using some of these

Algorithm 1
Generate set of samples S_n in the variation space according to the process parameter distribution. Estimate error bound ε for computation of the xth percentile of worst circuit delay using set S_n.
Obtain cost $A_i(T+\varepsilon)$ for the design optimized to minimize typical case cost while meeting constraint T at each corner/sample $s_i \in S_n$
Obtain the xth percentile of the cost distribution $A^x_{S_n}(T+\varepsilon)$

techniques have been discussed [29,33,56]. These techniques can be used to perform efficient computation of the lower bound with few samples.

IV.4 Sample Level Optimization in Parallel (SLOP)

This section proposes a Sample Level Optimization in Parallel (SLOP) technique for statistical circuit optimization. The optimization problem addressed here is to find the minimum cost solution for a design to meet a given timing constraint at the xth percentile of its worst case circuit delay distribution considering process variations. SLOP takes a sample level view of the process variation space. First, samples are generated that are essentially virtually fabricated dies. Within each virtual die, the process parameters are fixed and deterministic optimization is then performed. The optimization steps are based on a greedy strategy such as in [70] where gates in the critical path are selected and sized iteratively. The selection criterion is based on a metric that estimates the gain in circuit speed for a unit upsizing of the gate. Each virtual die can be optimized in parallel. Once

the optimization steps are complete, the design with the best cost among the solutions from different virtual dies is selected.

Figure 2 illustrates the approach. SLOP consists of two phases, S-phase and HPS-phase. Note that SLOP progresses for each sample in parallel, and hence operates on a single virtual die.

- S-phase or Sample Phase. In this phase, a virtual die is optimized using a greedy strategy to meet the timing constraint T. The result is a set of gate sizes that will meet the performance constraint for this specific point in the process variation space (i.e., sample).

- HPS-phase or High Percentile Sample Phase. SSTA is performed on the design returned from the first phase and the xth percentile sample is selected. It is not surprising that the design from the S-phase will not satisfy the timing constraint T for the entire process variation space. This phase seeks to zoom in on the portion of the process variation space that poses the greatest difficulties for the design from the S-phase, and re-optimize. Monte Carlo Sampling based SSTA is employed to select the xth percentile sample. Analytical SSTA techniques cannot be used here as they do not provide information at the sample/virtual die level as required by SLOP. We use a smart sampling based SSTA approach proposed in [29] called SH-QMC for this purpose. SH-QMC provides good accuracy in computing a high percentile of the worst arrival time with a small numbers of samples (100-200). The selected sample is then optimized to meet the constraint T using a similar greedy approach as in the S-phase. The results from different optimization runs (i.e., the parallel virtual die) are compared

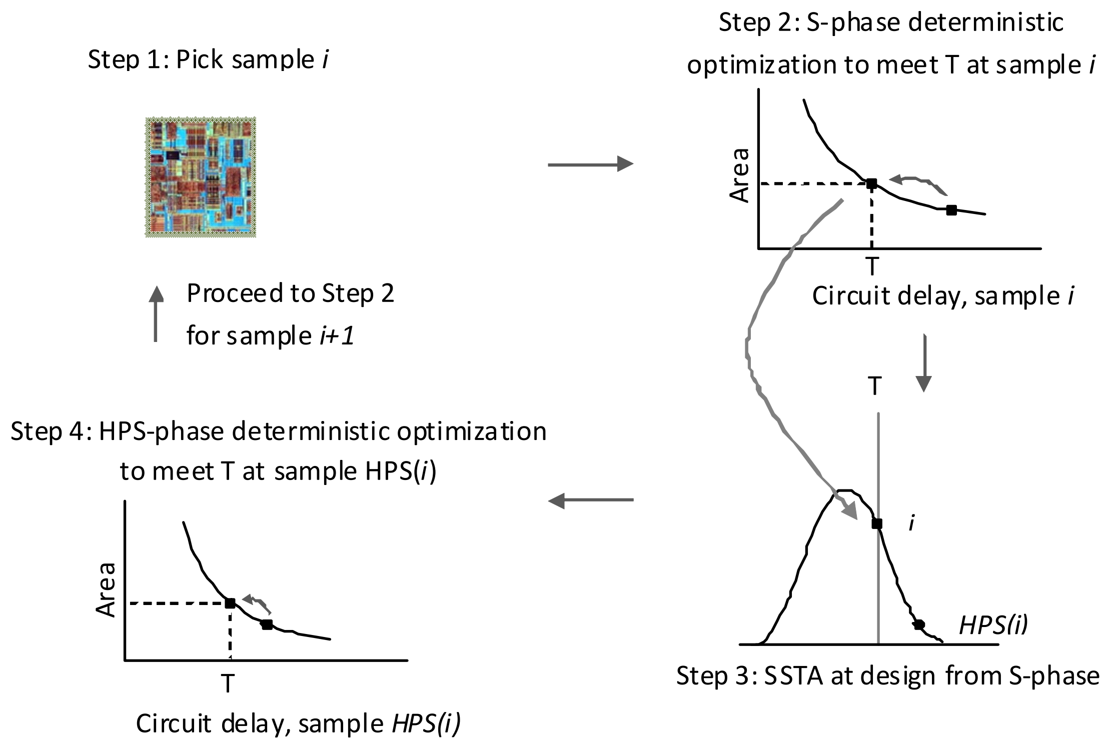


Figure IV.2. SLOP overview: Each virtual die i is optimized in parallel. The optimization consists of an S-Phase and an HPS-Phase. In the S-Phase, the virtual die is optimized to meet the timing constraint T using a greedy approach. In the subsequent HPS-Phase, the xth percentile sample of the design i optimized in S Phase, called HPS(i), is selected. If T is already met at HPS(i), optimization terminates. Else, the design is optimized further, this time with sample HPS(i) constrained to meet a timing T. The best among parallel solutions is selected.

and the design meeting T for the xth percentile worst case delay with minimum cost is chosen as the final solution.

IV.5 Results

Simulation results in this paper are based on a 65nm industrial technology library. The implementation considers channel length, oxide thickness, and threshold voltage variations as process parameters. Inter-die, spatially correlated intra-die and uncorrelated random components of variation are considered for each parameter. The relative amounts of process parameter variation among die-to-die, spatially correlated, and random sources have been reported in the literature [78-80]. An increase in systematic die-to-die

component of variation accompanied by an increase in random WID variation has been reported in [78] for a 45nm technology compared to 90nm. Systematic component of frequency variation is estimated at 52% in [79] for a 65nm technology node. In our implementation, the standard deviation in channel length variation is 5%. The standard deviation for oxide thickness is 1.3%. The contribution of D2D components of channel length and oxide thickness are fixed at 50% while dividing the random and spatially correlated WID components equally. The threshold voltage variation is modeled based on [80], where a Pelgrom model is used to compute the random component of threshold voltage variation. The number of grids in the spatial correlation model for individual circuits is varied linearly with post-placement area starting from 2 × 2 for the smallest circuit to 16 × 16 for the largest circuit. This corresponds to a grid area of approximately 40μm × 40μm for all the circuits.

Simulations are performed on ISCAS85 benchmark circuits [38] and three additional large benchmark circuits: Viterbi Decoder 1 (VD1), Viterbi Decoder 2 (VD2), and USB 2.0 Core (USB) with gate counts ranging from approximately 15,000 to 35,000. We perform synthesis and APR on all the benchmarks using commercial tools. Simulations were performed on a 16-core 2.0GHz AMD machine with 32GB RAM.

Table 1 shows the lower bound in area for benchmark circuits to meet a specified timing constraint with a timing yield of 99%. As noted in Section 2 the lower bound computation is constrained by the ability to obtain a perfect model of gate delay as a function of gate size, parasitics, and process variation effects. Hence, we model gate delays in the standard cell library with the approach described in Section 2 that uses the max-monomial fitting algorithm. We use 11 monomial terms in our implementation.

Further increases in the number of monomials are limited by the physical memory constraints on the server system for the largest benchmark circuits studied. For the purpose of lower bound computation, we use this gate delay model in the results for the various approaches considered. This enables us to make conclusions about the robustness of each of these techniques in comparison with the exact lower bound. Timing constraints were set such that the hardware intensity of each benchmark circuit is 1.0. The hardware intensity of a design is defined as the magnitude of the ratio of percentage change in cost to percentage change in timing constraint. The solutions obtained from three different approaches - Burns [69], Robust Geometric Programming (RGP) and the proposed SLOP technique are compared to the lower bound. The runtime values reported for Burns and SLOP are for implementation on a CPU using a GPU as a co-processor to exploit the parallelism available in these algorithms. The GPU is an Nvidia Tesla S1080 system with 4 cards. On average 12.8× and 6.4× improvements are obtained through such parallelism compared to a purely CPU-based implementation for Burns and SLOP, respectively. RGP is implemented on a CPU since no straightforward source of parallelism is available in the algorithm.

Table 1 indicates that the area of solutions obtained by Burns, RGP and SLOP are on average 9.6%, 3.7% and 7.5% higher, respectively, than the lower bound. This shows that the sub-optimality in the results obtained from these methods are low compared to the absolute best solution possible. Figure 3 shows sizing curves for the different optimization approaches as well as the lower bound computed using the proposed technique. The figure illustrates that the room for further improvement of results beyond smart deterministic approaches is low.

Table IV.1. Comparison of Burns, RGP and SLOP approaches against the lower bound for area at benchmark circuits. RGP is implemented on a CPU. Burns and SLOP are implemented on a CPU with a GPU co-processor, to utilize the parallelism available in the algorithms

Circuit	# gates	Area	Area/Runtime(s)			Area Lower Bound	Δ Area w.r.t. Lower Bound (%)			
		Worst Corner	Burns	RGP	SLOP		W.C	Burns	RGP	SLOP
C432	256	687.6	647.6/0.3	614/2.2	635.1/0.4	588.1	16.9	10.1	4.4	8.0
C499	544	1234.7	1168.2/0.4	1126/5.8	1163.8/1.2	1065.3	15.9	9.7	5.7	9.2
C880	500	1648.2	1558.3/0.7	1506/10.3	1541.4/1.5	1448.8	13.8	7.6	3.9	6.4
C1908	603	1654.3	1555.9/0.5	1492/6.6	1533.8/1.4	1454.1	13.8	7.0	2.6	5.5
C2670	780	2217.6	2095.7/0.5	1952/9.2	2084.5/2.1	1891.0	17.3	9.3	3.2	10.2
C3540	1163	3653.2	3477.1/1.4	3269/19	3422.1/6.5	3169.7	15.3	9.7	3.1	8.0
C5315	1692	5595.6	5236.1/1.7	4980/31	5206.5/9.7	4871.6	14.9	7.5	2.2	6.9
C6288	3834	8923	8447.1/6.1	7847/75	8024/32.1	7639.0	16.8	10.6	2.7	5.0
C7552	2152	6128.9	5833.7/3.8	5513/38	5785.9/23.6	5329.1	15.0	9.5	3.5	8.6
VD1	14503	45780.6	37302/16.9	35593/310	36933/129.7	34598.5	32.3	7.8	2.9	6.7
USB	32898	107131	90866/83.1	79856/4950	84291/658.4	74275.0	44.2	22.3	7.5	13.5
VD2	34082	97959	81308/108	79932/3846	80265/794.7	78305.7	25.1	3.8	2.1	2.5

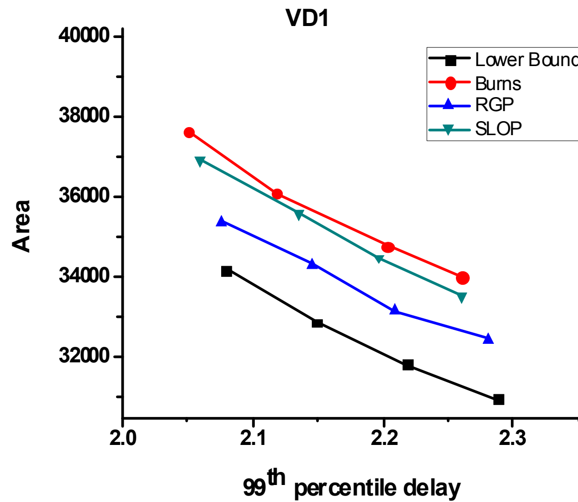


Figure IV.3. Comparison of sizing curves for Burns, RGP and SLOP against the lower bound for area computed at varying timing constraints for the 99% timing yield

IV.6 Conclusions

Worst-corner based deterministic approaches lead to pessimistic design. A significant amount of this pessimism can be reduced using statistical guidance for setting guardbands for the timing constraint. This paper proposes a lower bound computation method to evaluate statistical design optimization techniques. A statistical optimization technique that takes a sample level view of the process variation space, called Sample Level Optimization in Parallel (SLOP) is also proposed. Results from the lower bound computation method are compared against the solutions obtained from SLOP, and two previously proposed approaches - a smart deterministic approach (Burns) and a robust statistical optimization technique (RGP). Results indicate that any statistically aware technique has area within 10% of the lower bound on average. Burns achieves the lowest runtime. More statistically aware techniques (SLOP, RGP) do obtain lower area solutions; however the additional improvement is only 5.9% on average (for RGP). The results from RGP are within 3.7% of the lower bound, with additional runtime cost of 41.5X compared to Burns. SLOP has higher area compared to RGP by 3.8% on average, however is faster than RGP by 5.6X.

CHAPTER V

EFFICIENT SMART SAMPLING BASED FULL-CHIP LEAKAGE ANALYSIS FOR INTRA-DIE VARIATION CONSIDERING STATE DEPENDENCE

V.1 Introduction

As circuit design moves to smaller technology nodes the standby power dissipation of devices has become an important concern. In addition, variability leads to significant variation in the standby power which adds complexity to the problem. We discussed existing approaches to analyze standby power in Chapter I. We study the applicability of smart sampling based Monte Carlo techniques for leakage analysis in this work.

There are two main contributions in this work. To the best of our knowledge this work is the first to study sample size reduction for statistical leakage analysis using a Monte Carlo based approach. We consider intra-die variation, state dependence and multiple sources of process variation. Second, we address the issue of standard cell characterization, which is largely ignored in literature. Statistical circuit leakage analysis involves characterization of standard cells at grid points in the process variation space. This is illustrated in the schematic for a traditional flow in Figure V.1. Although characterization is only performed once in the design flow for a library the number of grid points grows exponentially with

the number of process variation parameters. There is a need to select samples to reduce characterization cost while meeting target accuracy in leakage analysis.

We first consider the problem of leakage analysis for the case of inter-die variation involving multiple process variation parameters. For this we propose to use a Quasi Monte Carlo technique [35] for selecting samples in the process variation space. We show that for a large benchmark circuit there is significant reduction in sample size to meet target accuracy when compared to a random selection of samples for computing leakage distribution. Standard cell characterization needs to be performed only at these samples which reduces the cost of standard cell leakage characterization. Next we propose a solution for the case of the total leakage distribution considering inter-die and intra-die components which is the major contribution of this paper. We recognize that this problem can be formulated as selecting samples for inter-die variation and computing the local distributions at each of these samples due to intra-die variation. Computation of the moments of the local distribution requires additional samples in the neighborhood of each inter-die sample. The number of these additional samples can be prohibitively high. We propose techniques for efficient selection of the samples. The key ideas are as follows. First we show that the optimal way to select samples to compute local distributions accurately is to select samples according to the probability distribution function of total process variation. Second, the selection of samples is performed intelligently by using the Quasi Monte Carlo technique. Experiments are performed on benchmark circuits synthesized in a 45nm commercial technology. State dependence information is also considered. We compare our technique with 3 approaches 1) random sampling, 2) a technique referred to as *Method1,* and 3) a traditional analytical approach based on [24]. *Method1* involves smart selection

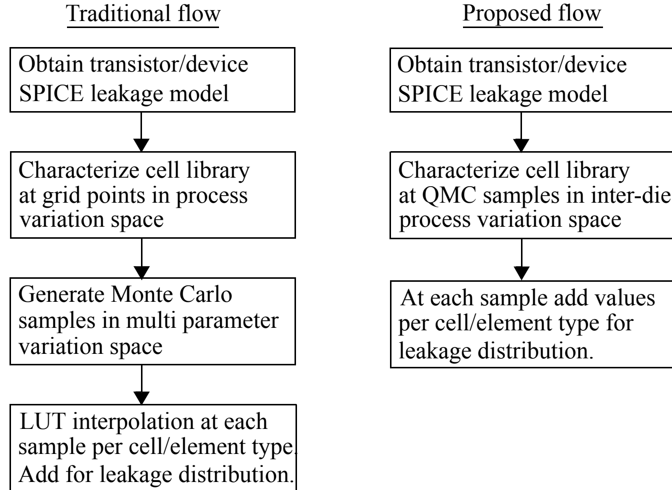


Figure V.1. Traditional and proposed leakage analysis flow for global variation with multiple sources.

of inter-die samples but no intelligence or reuse of samples for intra-die variation. For the largest benchmark considered with 198461 gates, the proposed approach requires 3 minutes whereas random sampling and *Method1* complete the task in 23 hours and 18.4 hours, respectively. We also achieve accurate results for estimation of m, s, and the 95th percentile of chip leakage distribution for all benchmarks considered with low runtime.

The paper is organized as follows. Section V.2 describes Quasi Monte Carlo approach, which is a standard technique to reduce sample size for Monte Carlo analysis. Section V.3 proposes a leakage analysis technique for the case of inter-die variation using a Quasi Monte Carlo technique. Section V.4 addresses leakage analysis for total leakage analysis involving inter-die and intra-die variation using smart samples. Results and conclusions are presented in Sections III.5 and III.6 respectively.

V.2 Smart Sampling for Leakage Analysis

Monte Carlo-based leakage analysis involves selecting samples in the process variation space to obtain a statistical distribution of circuit leakage. This is mapped to the standard

mathematical problem of Monte Carlo (MC), which is to estimate the integral of a function using samples in its domain. There are standard techniques for variance reduction of MC, including Quasi Monte Carlo techniques. These techniques are detailed in [28].

V.2.1 Quasi Monte Carlo

The standard Monte Carlo (MC) method addresses the problem of approximating the integral of a function *f(x)* over the s-dimensional hypercube $C^s = [0, 1)^s$ where x represents a point in an s-dimensional space. The MC estimate of the integral is given by the arithmetic mean of f_i which are values of the function *f(x)* evaluated at n samples distributed throughout the hypercube. The error bound of a method to numerically estimate an integral using a sequence of samples is mathematically related to a measure of uniformity for the distribution of the points called “discrepancy”. A sequence with the smallest possible discrepancy has the property that when used to evaluate the mean it achieves the smallest possible error bound. Sequences constructed to reduce discrepancy are called Low Discrepancy Sequences (LDSs). Quasi Monte Carlo techniques are characterized by their use of LDSs to generate samples. LDSs are deterministic sequences, i.e., there is no randomness in their generation. Intuitively these sequences are well dispersed through the domain of the function, minimizing any gaps or clustering of points. Sobol, Faure, and Niederreiter are LDSs that have been studied extensively. In this work we consider Sobol sequences, which are known to be simple to construct. Interested readers can refer to [35] for a construction of the Sobol sequence. In the context of circuits Quasi Monte Carlo techniques have been studied for statistical timing analysis [29] where results indicate that the techniques are a good fit and are amenable to multi-core and GPU computing. This

work is the first to study the application of Quasi Monte Carlo (QMC) techniques for statistical leakage analysis.

V.3 Leakage Analysis for Inter-Die Variation with Smart Sampling

In this section we first describe the steps in an industrial leakage analysis flow. A typical industrial flow circuit leakage analysis involves characterization of a standard cell library and computation of circuit leakage using the characterized data as explained in Section 3.2. Further we introduce our approach to estimation of statistical leakage due to inter-die parameter variation to achieve tractability for multiple sources of process variation.

V.3.1 Process Variation Model

Process variation parameters such as critical dimension (CD) and oxide thickness exhibit correlations. To account for correlations between parameters principal component analysis (PCA) is performed. Critical dimension, threshold voltage and oxide thickness are thus expressed as linear combinations of principal components. For process technology nodes 45nm and below some foundries provide such statistical information with principal component analysis. Now process variation models with inter-die and intra-die components are widely used in the literature [7]. Each process variation parameter has a global or inter-die component, which is modeled by a single random variable for a parameter in a die. Intra-die components account for spatial correlation within the die and uncorrelated random variation per device. In this model the die is partitioned into $n * n$ grids and

identical parameter variations are assumed within a grid. Therefore, each source of variation is represented by a set of random variables, one for each panel in the grid. For example, transistor gate length variation is represented by a set of random variables for all grids and the set is of multivariate normal distribution with covariance matrix R_{Lg}. As mentioned above the process variation parameters have been resolved into principal components. It follows that each component is represented by a set of random variables for all grids. Principal component analysis (PCA) is again performed on these spatially correlated variables. In addition an independent random variable accounts for random variation at the device level for components resulting from PCA on process variation parameters.

V.3.2 Traditional Leakage Analysis Flow for Inter-Die Variation

The standard cell library is characterized for leakage information at grid points in the process variation space. To include state dependence information, standard cells are characterized at the grid points for each input state. If state dependence is not considered then an average of the leakages for all input states is computed.

In a traditional Monte Carlo-based leakage analysis flow (to account for inter-die parameter variation) process parameter variables or their principal components are sampled. As only global variation is considered the same sample set is assigned to every element type in the standard cell library. The leakage value per element type in the library is

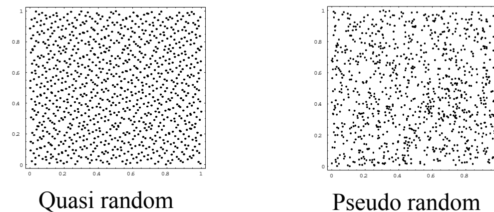


Figure V.2. Quasi random and pseudo random sequences.

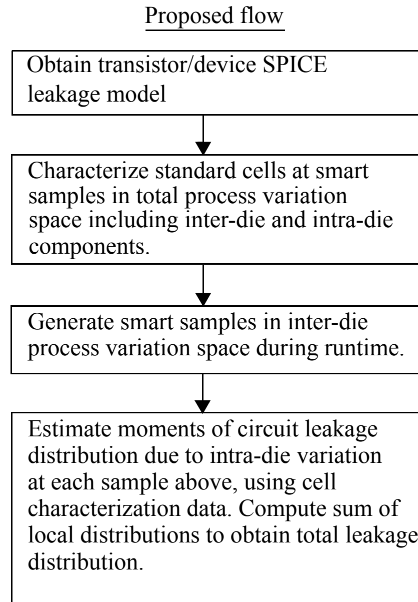


Figure V.3. Proposed leakage analysis flow for within-die variation.

obtained by interpolation in the leakage lookup table for the element type. The circuit leakage is obtained by adding up the leakage value obtained for each element type after weighting by the number of occurrences of the element type in the circuit.

The above approach does not consider state dependence of standard cell leakage. To enable leakage calculation to account for state dependence, the standard cell characterization data must have leakage information for every cell state as mentioned above. In addition, at the circuit level state probability information is required for every instance of each element type in the circuit. Various approaches exist in the literature to arrive at an estimate of state probability for each instance. For a detailed discussion on this topic refer to [39].

V.3.3 Proposed Leakage Analysis Flow with Smart Sampling

We propose to use Quasi Monte Carlo based sampling for standard cell library characterization and runtime leakage analysis. In a traditional flow standard cells are characterized at discrete grid points in the space of random variables to model process variation as explained in Section V.3.2. In the proposed approach the characterization is performed at samples generated using a Quasi Monte Carlo (QMC) based approach. In particular we use Sobol sequences in the QMC approach in this work. QMC samples refer to Sobol samples in the rest of the paper. The same process variation samples are used for characterization of all element types in the standard cell library and their states.

The proposed approach differs from a traditional flow during runtime in that samples are not generated at this stage. The inter-die samples are precomputed during cell library characterization. A given inter-die sample is assigned to every element type in the library as before and the circuit leakage is obtained by adding up the leakage values from element types as in the traditional flow. It follows that there is now no need for interpolation in the look-up table from cell characterization. The leakage values are readily available in the tables without need for interpolation. The traditional and proposed flows are illustrated in Figure V.1.

V.4 Leakage Analysis for Total Variation with Smart Sampling

This section proposes an algorithm for estimating full-chip leakage considering inter-die and intra-die components of variation. In sub-45 nm technologies secondary effects in process variation are important and the number of significant sources of process variation is increasing. Existing approaches to calculate full-chip leakage power make simplifying assumptions about either the nature of statistical distribution of process variation parame-

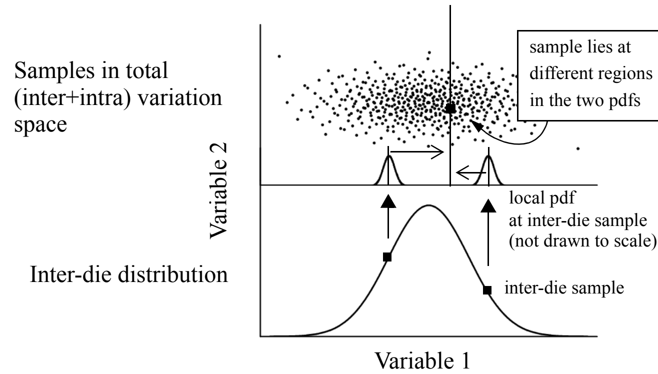


Figure V.4. Reusing samples for local distribution computation. Inter-die samples weigh the samples in total variation space according to local probability distribution.

ters or the nature of dependence of the standard cell leakage on these parameters. The parameters are assumed to have a standard distribution or the logarithm of standard cell leakage is assumed to be a linear or quadratic sum of the parameters. Combined with a growing number of process variation sources this is a limitation on the accuracy. Monte Carlo based methods on the other hand are expensive when handling intra-die variation. The proposed approach can efficiently handle any non-standard distribution of variables or dependence of full-chip leakage on these variables.

A schematic of the proposed approach for total variation is illustrated in Figure V.3. Process variation consists of inter-die and intra-die components. In Section V.3 we discussed generation of samples in the space of inter-die variation distributed according to the joint probability distribution of the variables involved. We apply intra-die variation to such a sample around the nominal and obtain a local leakage distribution for the circuit. The sum of these distributions from all samples should give the total leakage distribution. From a sampling perspective this translates to generating more samples distributed according to the intra-die distribution around each inter-die sample. In this way the problem of total leakage variation can be formulated as a two-level sampling problem, the first level corre-

sponds to inter-die variation and the second level corresponds to intra-die variation at each of the samples in the first level. Using Quasi Monte Carlo sampling, accurate results for inter-die variation can be achieved with few samples as explained in Section V.3. However even for a low number of first level, or inter-die, samples the total number of samples in the second level can be prohibitively high. The idea here is that if the second level samples are chosen optimally such that either the entire set or a subset can be used for computation at every inter-die sample the number of samples can be minimized. A uniform sampling approach in a bounded space enveloping the inter-die samples may be tried. However while this considers outliers in the inter-die distribution this does not weigh samples close to the nominal adequately. The problem is to arrive at a pdf which is optimal for all samples.

Consider the first level or inter-die samples in the process variation space. The problem is to find a pdf for optimality in computation at every inter-die sample. Such a pdf is obtained by summation of the pdfs of local distributions at the inter-die samples. Now we have the surprisingly simple result that if the number of inter-die samples is large enough the summation of the pdfs converges to the pdf for the distribution obtained for total variation with inter-die and intra-die components. The proof has been omitted for brevity. Our experiments indicate that if the inter-die samples are chosen according to a Sobol sequence and the sample size is large enough (typically more than 100) this is indeed true. Therefore we select the second level samples according to the pdf for total variation. To minimize the number of second level samples we use Sobol sequences to sample in this space.

For the case of no spatial correlation the idea is illustrated in Figure V.4 where two samples are shown on the inter-die distribution. The second level samples are chosen to be Quasi Monte Carlo based samples in the total process variation space. One such sample in Figure V.4 lies in different regions of the pdf for the two inter-die samples. Therefore the first level samples assign different weights to the leakage values obtained at a particular second level sample. The characterization step needs to compute leakages for standard cells at the second level samples only. The procedure to reuse samples is illustrated in Figure V.5 for the case of a 2D process variation space. Figure V.5a shows the total process variation distribution along with the local distribution at an inter-die sample *S*. Figure V.5b shows the second level samples generated in the total distribution space x_i : *i=1...N.* These samples are reused for computation of moments of local distribution at S as in Figure V.5c. In particular the mean of local distribution at S for the circuit, $\overline{L}(S)$ is given by

$$\overline{L}(S) = \sum_{i=1}^{N} \frac{L(x_i) \times JpdfIntra(x_i - S)}{JpdfTotal(x_i)} \tag{1}$$

where *JpdfIntra* is the probability distribution for intra-die variation and *JpdfTotal* is the probability distribution for total variation. Similarly higher moments for the local distribution can be computed. The total leakage distribution is a sum of local leakage distributions and is computed using $\overline{L}(S)$ and the higher moments obtained for all samples. In the case of spatially correlated intra-die variation, the sample for one variable is not a single value but a set of values corresponding to grids in the spatial correlation model. This means that each element of vector x_i in (1) is not a scalar but a vector with correlated elements. The number of elements in this vector is equal to the number of grids. The functions $JpdfIntra(x_i - S)$ and $JpdfTotal(x_i)$ are modified to include the spatial correlation. We explore spa-

tial correlation later and show that this level of modeling process variation is not needed for large circuit and full-chip leakage analysis.

Now the local distribution corresponding to one sample can be approximated using Central Limit Theorem abbreviated as CLT [40]. If spatial correlation is not considered then this local distribution has contribution from sum of identical independent random variables from instances of a given element type in the cell library. If there are enough instances the local distribution approaches a normal distribution. Also for a large number of instances the variance of this distribution approaches zero according to CLT. This means that the local distribution approaches a single number which is the mean of the distribution. In the presence of spatial correlation as long as there are sufficient independent regions in a die, i.e., the circuit is large enough the Central Limit Theorem can be applied [41] as if all intra-die variation was uncorrelated. A reduction in variance of the local distribution translates to a reduction in the number of second level or additional samples for a target accuracy. For large circuit blocks and chips the problem essentially is to compute only the mean of the local distribution at each inter-die sample. For circuits where spatial correlation has a significant effect on leakage distribution, the technique can still be

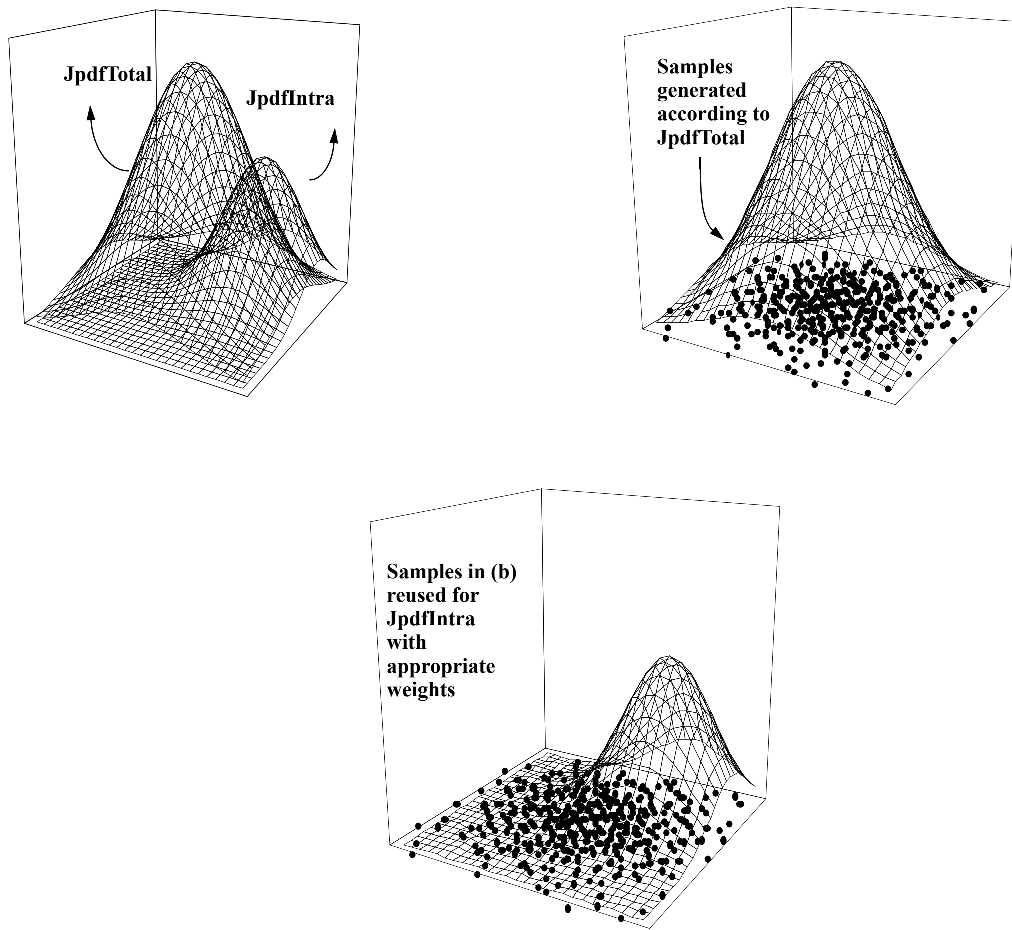


Figure V.5. (a) Total (Inter+Intra die) distribution and local pdf at an inter-die sample. (b) QMC based samples are generated according to total variation. (c) For computing mean of local pdf the samples generated in (b) are weighed according to the ratio of the probabilities in the two distribution functions.

applied. The local distribution within a grid panel has contribution from the sum of identical independent random variables from instances of a given element type in the cell library. Therefore the local distribution within each grid panel approaches a normal distribution with number of instances in the panel, which reduces the number of additional samples, with spatial correlation considered, to capture the local distribution.

V.5 Results

Our simulation results are based on a 45nm commercial technology. Principal component analysis is used to obtain principal components for the correlated process variation parameters including CD, oxide thickness and threshold voltage. Simulations are performed on industrial circuits with sizes ranging from approximately 5000 to 200,000 gates. In our implementation, we only consider inter-die variation and uncorrelated intra-die variation. Spatially correlated intra-die variation is not implemented. In the presence of spatial correlation as long as there are sufficient independent regions in a die, i.e., the circuit is large enough, the Central Limit Theorem can be applied as explained in [41] and therefore the results are accurate for large circuit blocks and chips. This is illustrated in Figure V.6 for a benchmark circuit with approximately 43,000 gates. The standard deviation of the leakage distribution without considering spatial correlation is compared to the case where a grid-based spatial correlation model is considered. The total standard deviation for intra-die variation is the same in both cases. The assumption of no spatial correlation accurately estimates standard deviation for number of grid panels above 256,

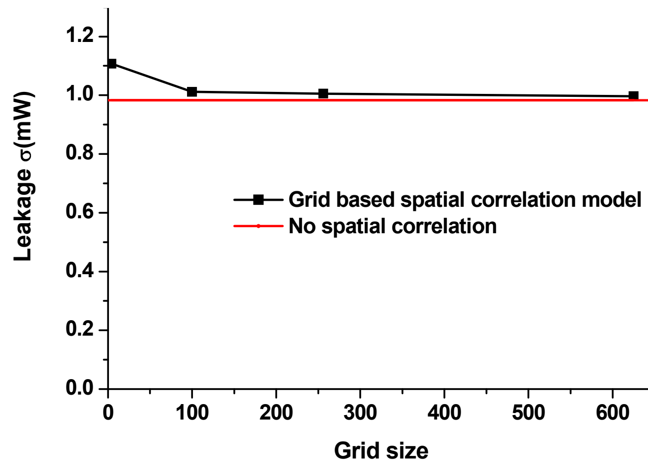


Figure V.6. Comparison of sigma of leakage distribution without considering spatial correlation with that of a grid-based spatial correlation model for VGA circuit (43214 gates)

supporting the argument in [41]. Therefore, spatial correlation is not a limitation for circuit blocks and chips with practical sizes for the current implementation, which is our focus in this work. The modification in the algorithm for the case of smaller circuits is discussed in Section V.4.

Figure V.7 shows the result for our proposed approach for inter-die parameter variation using smart samples. The smart samples are obtained from a Sobol sequence. The error in estimating s of leakage distribution for inter-die variation using smart samples is compared with a random sampling based approach for a VGA circuit with approximately 43,000 gates. The golden value is obtained from a Monte Carlo simulation with 20,000 samples. We compare the minimum sample size required to achieve target accuracy of 3% error in estimating s for both methods. The proposed approach requires 9.3X fewer samples compared to random sampling. In a typical industrial flow the standard cells are characterized at grid points in the process variation space. With 7 grid points chosen for each of the three principal components in our implementation the number of points to be char-

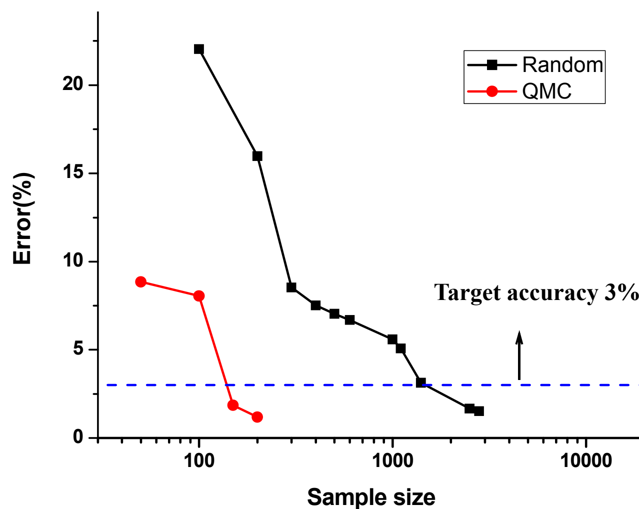


Figure V.7. Comparison of error in estimating σ of leakage distribution for inter-die variation using QMC vs. random sampling for VGA circuit(43214 gates).

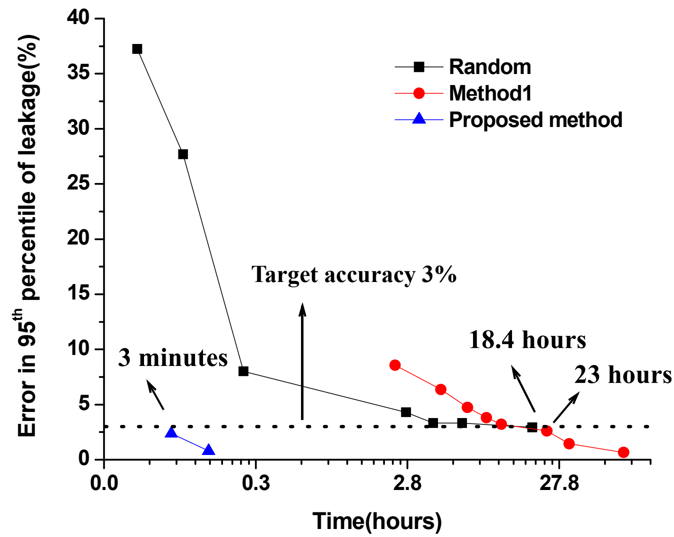


Figure V.8. Comparison of accuracy of proposed approach with random sampling based approach vs. runtime. The circuit considered is *Chip1* with 200,000 gates.

acterized is 343 in a traditional flow whereas the proposed approach requires only 150 from Figure V.7, a 56% reduction in standard cell characterization overhead.

We now present our results for total process variation considering both inter-die and intra-die components of variation. Table V.1 shows results comparing the proposed approach with 20,000 Monte Carlo runs on benchmark circuits. The metrics compared are mean m, sigma s, and the 95th percentile of the circuit leakage distribution. The errors in estimating these metrics for the largest benchmark circuit *Chip1* are less than 3%. The errors in estimating the metrics are less than 3.6% for all the benchmark circuits. Note that there is higher accuracy for the largest benchmark studied. The proposed approach has a runtime of less than 3 minutes for the largest benchmark, which illustrates the runtime efficiency. The larger runtime for *Chip1*, even accounting for the larger circuit size, is attributed to the fact that state probability information is only considered for this circuit. State probability consideration for each instance adds significant cost to the computation.

Figure V.8 plots the accuracy against runtime of the proposed approach and a random sampling approach. We also compare this with the result for another smart sampling based technique called *Method1*. As explained in Section V.4 the proposed approach first generates inter-die samples using smart sampling. In the next step a smart selection of samples in the total variation space is coupled with reuse of these samples to compute the mean of local leakage distributions at inter-die samples. In *Method1* inter-die samples are generated using a Sobol sequence as in the proposed approach. However a random sampling based Monte Carlo analysis is performed at each inter-die sample to obtain the local distribution. In other words there is no intelligence or reuse of samples in total variation space, however as inter-die samples are generated using smart samples this method is expected to be faster than random sampling. Figure V.8 shows that the proposed approach has a runtime of less than 3 minutes to achieve target accuracy for the largest benchmark whereas *Method1* has a runtime of 18.4 hours. This result illustrates the advantage of smart sampling and reuse of the additional samples in the total variation space. The random sampling approach has a runtime of 23 hours. It may be noted that the slope of the curve for *Method1* is steep in the beginning compared to the rest of the curve. This is because in *Method1* the number of inter-die samples is increased in the beginning till the inter-die

Table V.1. Comparison of proposed approach with Golden (Monte Carlo 20,000 samples) for benchmarks. * indicates that state probability is considered for instances in the circuit.

		Golden (Monte Carlo 20k samples)				Proposed approach				Error (%)			
	Gate count	μ (mW)	σ (mW)	95th percentile (mW)	Runtime	μ (mW)	σ (mW)	95th percentile (mW)	Runtime (s)	μ	σ	95th percentile	Speed up
VD1	5536	0.51	0.18	0.85	1.7 hours	0.51	0.17	0.87	1.77 s	0.03	3.55	2.21	3405
VD2	13258	1.21	0.42	1.99	4.8 hours	1.20	0.40	2.04	1.83 s	0.30	2.61	2.51	9495
USB	15946	1.11	0.36	1.79	7.4 hours	1.11	0.36	1.85	1.95 s	0.01	1.97	3.35	13738
ETHER	23939	1.40	0.46	2.26	10.2 hours	1.40	0.45	2.33	2.09 s	0.06	1.99	3.10	17633
VGA	43214	2.85	0.98	4.71	15.6 hours	2.84	0.96	4.85	2.02 s	0.49	2.31	2.97	27778
*Chip1	198461	10.63	2.67	15.59	19.2 days	10.64	2.63	15.96	278 s	0.10	1.71	2.37	5969

component of variation is captured accurately. After that only the number of random samples to capture the local distribution is increased while keeping the number of inter-die samples constant, hence the decrease in slope. The slow convergence of random sampling to capture local distribution is the reason for comparable runtimes of *Method1* and random sampling.

Table V.2 compares the proposed approach with an analytical approach to compute leakage distribution based on [24]. In [24] the authors approximate the logarithm of gate leakage as a linear expression involving process variation variables. Wilkinson's approximation is used to compute sum of lognormals to obtain circuit leakage as a lognormal expression. From Table V.2 the maximum error in estimating m is 3.7% for the analytical approach compared to 0.5% for the proposed approach. Similarly the maximum error in estimating s is 6.1% for the analytical approach compared to 3.6% for the proposed approach. It may also be noted that the proposed approach incurs less error as circuit size increases but no such trend is observed for the analytical approach. For the largest benchmark *Chip1* state dependence has been implemented for both methods. The errors in estimating m and s are significantly lower for the proposed approach in this case as illustrated. As mentioned before the runtime for *Chip1* is significantly higher compared to other circuits, even accounting for circuit size because state probability information of instances is considered in this circuit. In the case of the analytical approach the increase in time cost is much higher because the dependence on number of states is quadratic.

Figure V.9 compares the total leakage distribution of the largest benchmark circuit with 200,000 gates for the proposed approach with the golden and the analytical approach based on [24]. The leakage variation considering only inter-die variation is also plotted.

This analysis considers state probability information for instances in the circuit. The state probability information is extracted using a commercial tool. We see that the distribution curve is captured with accuracy by the proposed approach whereas there is significant error with the analytical approach.

V.6 Conclusions

Monte Carlo-based techniques are promising for statistical leakage analysis because of the generality and scalability of the approach even when complex relations exist between leakage and process parameters. This work addresses the problem of reducing the sample size for Monte Carlo based leakage analysis. For a large benchmark circuit the sample size is reduced by 9.3X compared to a random sampling approach to achieve target accuracy. The standard cell characterization cost is also reduced by 56%. We also propose a solution to estimate the total leakage distribution considering inter-die and intra-die components. A novel technique involving smart sampling combined with reuse of samples is introduced to address this issue. The proposed approach is compared with random sampling, *Method1* where samples are not reused, and an analytical approach. For the largest benchmark con-

Table V.2. Comparison of proposed approach with Wilkinson's based approach. * indicates that state probability information is considered for instances in the circuit.

Circuit	Gate Count	Proposed approach			Wilkinson's approach		
		% Error		Run-time(s)	% Error		Run-time(s)
		μ	σ		μ	σ	
VD1	5536	0.03	3.55	1.77	3.43	4.81	0.16
VD2	13258	0.30	2.61	1.83	3.16	1.80	0.16
USB	15946	0.01	1.97	1.95	3.62	6.13	0.19
ETHER	23939	0.06	1.99	2.09	3.69	0.53	0.20
VGA	43214	0.49	2.31	2.02	3.03	1.37	0.20
*Chip1	198461	0.10	1.71	278	3.08	5.46	3094

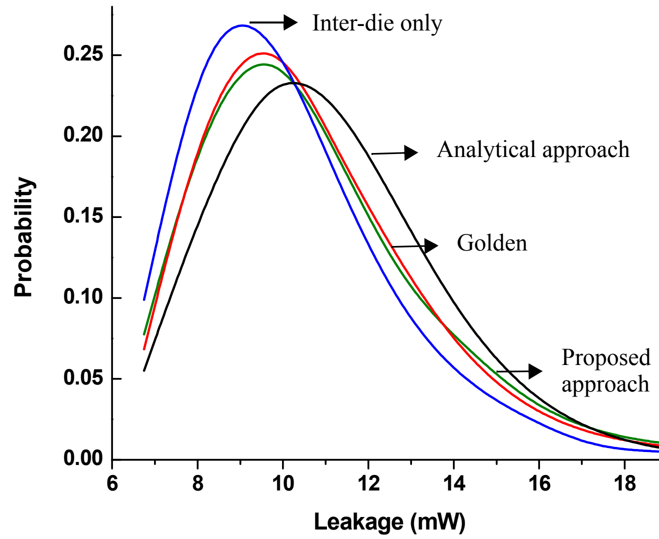


Figure V.9. Total leakage distribution considering intra-die variation for *Chip1* (200,000 gates). Proposed approach is compared with the analytical approach based on [24] and the golden. The distribution due to inter-die variation is also plotted. The analysis considers state dependence of leakage for the instances in the circuit.

sidered the proposed approach performs the computation in 3 minutes whereas the random sampling approach and *Method1* complete the task in 23 hours and 18.4 hours, respectively. The analytical approach has up to 3.7% and 6.1% in approximating m and s compared to 0.5% and 3.6% for the proposed approach. In addition the characterization cost for the total leakage distribution is scalable with respect to the number of process variation variables since Quasi Monte Carlo sample size increases moderately with the number of variables whereas in a traditional grid-based characterization approach the cost grows exponentially with the number of process variation variables.

CHAPTER VI

FAST AND ACCURATE WAVEFORM ANALYSIS WITH CURRENT SOURCE MODELS

VI.1 Introduction

Recent research has focused on new approaches which attempt to keep the gate delay model interconnect insensitive for capturing complex loads. A model called 'Blade' was proposed by Croix and Wong [42]. This models the DC current characteristic of the gate as current source and the parasitics as a single output capacitance. The single output capacitance does not capture non-linearity. Also, the approach in runtime is to perform numerical integration, which is expensive. In [43], Keller et al propose to use a current source model for the drivers for crosstalk induced delay change analysis. In [44], the authors propose linear and nonlinear driver models for timing and noise analysis. The work points out how basis functions can be effectively used to propagate voltage waveforms. In [48], the authors proposed to characterize the driver's linear and saturation region operations individually with Linear Time Varying (LTV) models such that they are interconnect insensitive. The work in [42-44] have been followed up in [45-47]. The models proposed in these are called Current Source Models (CSMs). In [45], the authors map the time shift parameter proposed in [42] to an RC ladder, and try to model non linearity in capacitance. In [46], the authors proposed a multi-port current source model to

deal with multiple switching effects. This approach, while accurate, adds more complexity to the model and it is unclear how much the computational efficiency will be in the runtime engine for timing analysis. An interesting approach towards statistical analysis using current source models considering process variations has been proposed by [47]. These approaches are interesting, yet they fall short of giving us an efficient runtime engine. In particular, there is no work showing that the parasitic model and the DC current source model in CSMs can be incorporated in an efficient runtime engine.

In this work, we focus on CSMs. We attempt to address the specific issues to be dealt with in implementing a practical timing analysis approach with an efficient runtime engine, based on current source models. Our contributions are twofold. First, we model the DC current behavior and the transient behavior for optimality in the accuracy vs. runtime trade-off. We make observations regarding efficient ways to capture the DC current source model. We propose a Bicubic Spline based DC Current Source Model, and show that this is highly accurate, as well as amenable to fast runtime analysis. For modeling the transient, we show that a simple parasitic capacitance model with a time shift parameter is sufficient to make an accurate model. Second, we propose a solution for fast and accurate run time waveform analysis utilizing the current source model. Our work is the missing link between the fact that Weibull functions represent waveforms effectively [49], and that CSMs have the potential to be the future in timing analysis. Specifically, we propagate voltage waveforms as Weibull functions and exploit the properties of our current source model to efficiently solve for Weibull parameters at every gate. Additionally, the method can be extended to more elaborate load models for the future, and for the case of noise analysis, as well as to model process variations. Timing

analysis results on benchmark circuits show significantly reduced errors (and error spreads) compared to a traditional Thevenin-based flow. In terms of $\mu+\sigma$ percentile, we gain by 20-150% in slew and up to 220% in delay through this approach.

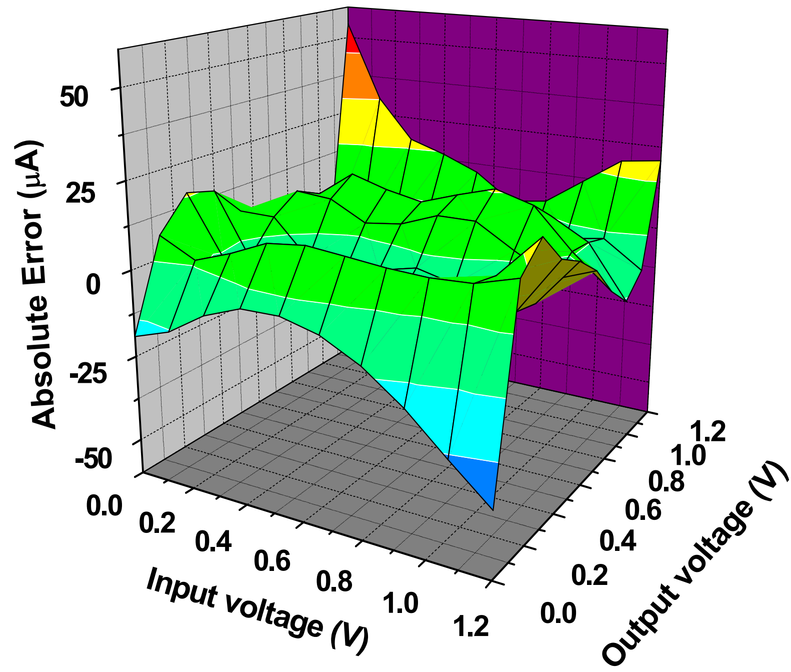
We present our work in the following sections. Section 2 deals with our approach towards precharacterization of gates in a CSM. In Section 3, we describe a technique for fast and accurate waveform analysis during runtime. Section 4 shows results on benchmark circuits while Section 5 concludes the work.

VI.2 Precharacterization

The precharacterization step in CSMs for a given process/voltage/temperature (PVT) corner involves two steps as mentioned before. First the DC current sourced by the gate is modeled as a function of active input pin and output pin voltages. The transient waveform also depends on the parasitic capacitance. In the second step, this parasitic behavior is modeled with a capacitance-based or charge-based model. Here we describe our approach to precharacterization and show how our Bicubic Spline based DC Current Source Model is efficient for incorporating in a runtime engine. We also discuss our approach to modeling parasitic behavior.

VI.2.1 Bicubic Spline based DC Current Source Model

For a given process/voltage/temperature (PVT) corner, DC supplies are attached to input and output pins and swept from $0-\Delta V$ to $Vdd+\Delta V$. A 2-D table of output current versus input and output voltages is obtained [42]. In our case, ΔV is 0.1V. Now, an accurate and efficient model of output DC current is extracted from the data. We propose a



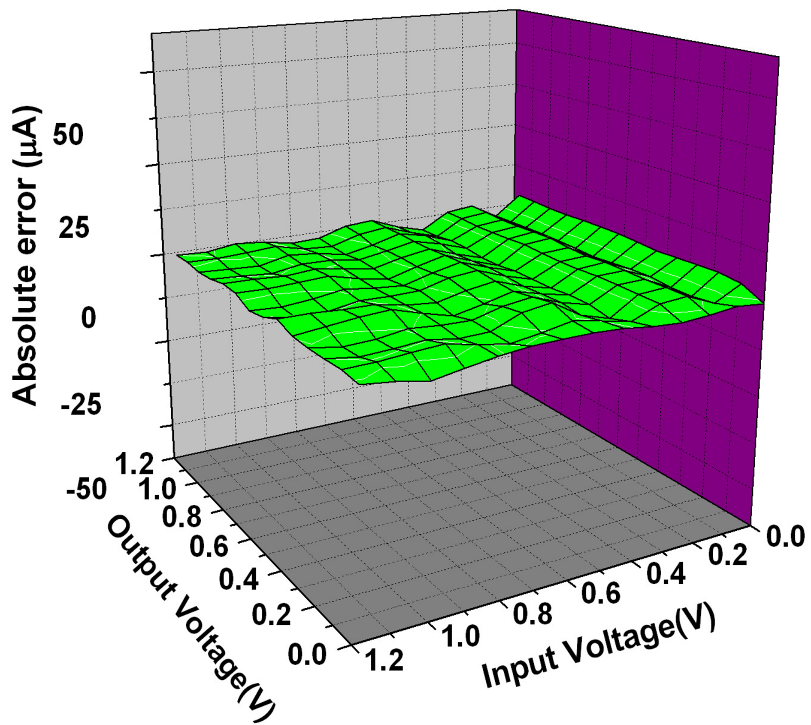


Figure VI.1. Polynomial-based fitting model error in current relative to SPICE as a function of input and output voltages. (b) Spline-based fitting model error in current relative to SPICE.

Bicubic spline model. We compare a fourth order polynomial fit in two variables to the bicubic spline fit [51] for the data, and find that the bicubic spline fit is stable with higher accuracy. Figs. 1(a) & (b) show typical error plots for the two models. Note that the fourth-order polynomial fit is unstable at the steady-state points (0, 1.2) and (1.2, 0) because of sharp trends in the region. Also, the peak error magnitude is an order lower for bicubic spline. Table 1 shows data for the stagewise timing analysis performance of standard cells in an industrial 90nm library for the two approaches, where it is clear that the proposed approach has much higher accuracy. From our experiments, we also observe that a bicubic spline with 3 by 3 internal knots is comparable in accuracy to 4 by 4 internal knots for this purpose, meaning 3 by 3 internal knots is sufficient.Bicubic splines are continuous up to the second derivative. This is of important consequence in implementing a slew of approaches to solving non-linear equations in a runtime engine. We exploit this property in our runtime engine in section 3.

VI.2.2 Modeling Parasitics

Table VI.1. Comparison of stagewise timing analysis for a bicubic spline fit (spline) vs. a fourth order polynomial fit (Poly) for standard cells in an industrial 90nm library.

	10-30% (% error)		50% (% error)		20-80% (%error)	
	Spline	Poly	Spline	Poly	Spline	Poly
Avg	-0.5	2.8	-0.8	-1.5	0.6	2.6
Stdev	0.9	2.7	0.3	1.0	0.4	1.7

We calibrate two capacitance values each for rise and fall transitions. One each for 10-50% (C10-50) and 50-90% (C50-90) output voltage transition regions. For a given gate for a rise/fall transition, we calibrate the capacitance parameters for minimum error across

a set of input slew rate and output load capacitance combinations. These input slew rates range from fast to slow (F04 to 8 F04) transitions. The maximum output load capacitance is such that the output slew does not exceed 8 F04 for any input slew. The array of capacitance values is distributed uniformly below this value.

At a given combination of input slew s and output load capacitance Cload, we run SPICE for the gate. Next we compute the 10-50% parameter C10-50(s, Cload) (this corresponds to 10-50% output voltage for output rise transition and 90-50% output voltage for output fall transition). For this, we simulate our bicubic spline based current source model for this input slew s and output load capacitance Cload with an additional capacitance C at the output. We sweep the value of C. The approach in simulation is to use numerical integration. As precharacterization is a one time effort, the time cost of numerical integration does not matter. The value of C which minimizes the error in 10-50% output transition time compared to SPICE is C10-50(s, Cload). Now we sweep s and Cload. C10-50 is obtained using a weighted average of C10-50(s, Cload). The weights take into account that relative error in C10-50 to total output load is higher at smaller Cload values.

The procedure to obtain C50-90 is similar. The calibrated C10-50 is used till the 50% output voltage transition in the simulation of the gate model. The 50-90% parameter C50-90(s, Cload) minimizes error with respect to SPICE and C50-90 is the weighted average of these values.

We also calibrate a constant time shift parameter for combinational library cells. The procedure is similar as above; here the error in 50% delay is minimized with respect to SPICE using a time shift. Again, the calibrated C10-50 is used till the 50% output voltage

transition in the simulation of the gate model. We find it worth mentioning that the transient model in current source models is a set of calibrating parameters, and does not correspond to the actual parasitic capacitance values. Hence, the actual parasitic model may be complex, but the output voltage curve is observed to respond smoothly in spite of this. This is the basis of our choice of a simple transient model.

VI.3 Weibull-based Runtime Engine

This section presents a novel method to perform timing analysis for a circuit. Our method exploits the fact that the bicubic spline based DC current source model obeys smoothness properties, and therefore lends itself to various simple mathematical analyses. It has been noted in [49] that the cumulative distribution function of a Weibull function is very efficient in capturing waveform shape. This, coupled with the Bicubic Spline based DC current source model, enables a simple and fast yet accurate method to propagate waveforms as Weibull-based functions.

VI.3.1 Basic Concept and Flow

For simplicity, we consider here the simplest three-parameter Weibull function. CDF of Weibull functions can be written as follows:

$$W(a,b,t0) = 1 - \exp(-((t - t0)/b)^{a} \qquad (1)$$

Refer to Fig 2. Let the rising input waveform to a gate be represented by Let output waveform Vc1 be of the form Note that this is for an output falling transition of a gate;

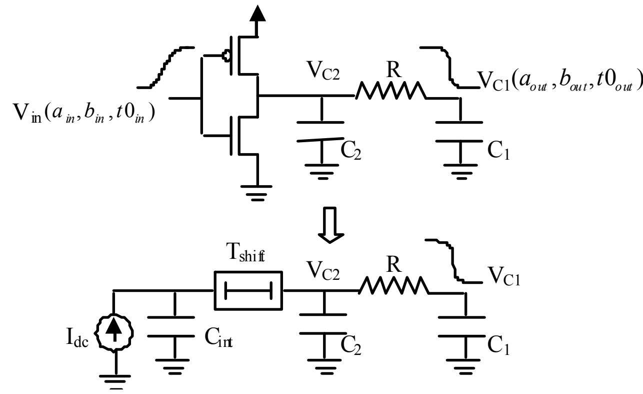


Figure VI.2. Schematic of the proposed modified Blade-based model

for the output rising case, the forms are interchanged. Also, let the bicubic spline model of Idc be as follows (refer Fig 2):

$$I_{dc} = \sum_{i=0}^{3} \sum_{j=0}^{3} \alpha_{ij} V_{in}^{i} V_{c2}^{j} \qquad (2)$$

where are coefficients of a piecewise bicubic polynomial.

Now, consider our model of a library cell loaded with a π load, as in the schematic in Fig 2. The KCL equation for current in this situation can be written as

$$I_{dc} + I_{load} = 0$$

Our aim is to come up with parameters to minimize the error function given by

$$\begin{aligned} f(t) &= I_{dc} + I_{load} \\ &= \left(\sum_{i=0}^{3} \sum_{j=0}^{3} \alpha_{ij} V_{in}^{i} V_{c2}^{j}\right) + \left(\partial(p_1 W_{c1})/\partial t + \partial^2(p_2 W_{c1})/\partial t^2\right) \end{aligned} \qquad (3)a,b$$

where

$$V_{c2} = V_{dd}(1 - W_{c1} - RC_1.\partial W_{c1}/\partial t)$$
$$V_{c1} = V_{dd}.(1 - W_{c1}(a_{out}, b_{out}, t0_{out}))$$
$$C_2^{'} = C_2 + C_{int}$$
$$p_1 = V_{dd}.(C_1 + C_2^{'})$$
$$p_2 = V_{dd}.R.C_1.C_2^{'}$$

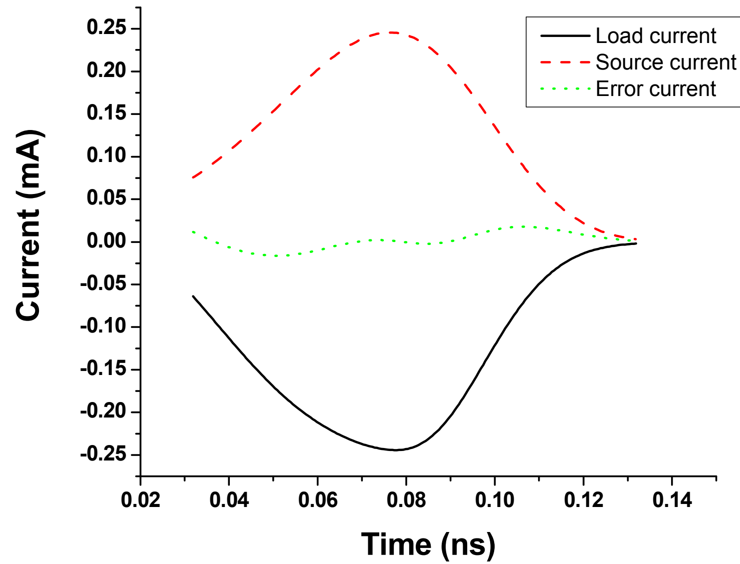


Figure VI.3. Typical waveforms for source and load currents as in the proposed model for an output falling case. The difference is the error function (referred to as f (t)).

The first term in eqn 3(b) refers to the current sourced by the DC current model (refer to as source current), and the sum of second and third is the current flowing into modeled and real loads (refer to as load current) (Fig 3). Therefore, the problem can be formulated as solving for parameters (a_{out},b_{out},$t0_{out}$) such that the error in f(t) is minimized for all t. A least square approximation by integrating $f(t)^2$ for all t may be tried. However, the function is not explicitly integrable. Also, the large number of parameters makes a look-up table form for the integration infeasible - it will include parameters a_{in}, $b_{in,}$ a_{out}, b_{out} and

$t0_{out}$, along with the time limits of the integration, say t1 and t2 (since are coefficients of a piecewise bicubic, this involves piecewise integration). However, the function is continuous and differentiable with respect to all three parameters; therefore an iterative method may be adopted to solve a system of non-linear equations. We considered several methods including Newton Raphson and Conjugate Gradient based steepest descent method. We intend to solve the following system of non-linear equations by Newton-Raphson iterations (where f(t) is the function as above).

$$
\begin{aligned}
f(t_1, a_{out}, b_{out}, t0_{out}) &= 0 \\
f(t_2, a_{out}, b_{out}, t0_{out}) &= 0 \\
f(t_3, a_{out}, b_{out}, t0_{out}) &= 0
\end{aligned}
\qquad (4)
$$

The time points (t1, t2, t3) are chosen to be the 20, 50 and 80% transition points of VC1. In our case, Newton Raphson is observed to converge faster, typically in 3-4 iterations. Steepest descent methods have a disadvantage because the parameters (a_{out},b_{out},$t0_{out}$) we search for are not homogenous quantities. For starting values of a_{out}, b_{out} and $t0_{out}$ - we need the following fitting coefficients per gate:

$$
\begin{aligned}
t_{d_out} &= a_1 + a_2 * tr_in + a_3 * cap \\
t_{tr_out} &= a_1 + a_2 * tr_in + a_3 * cap
\end{aligned}
\qquad (5)
$$

where t_{d_out}, t_{tr_out} are output delay and slew respectively, cap is the gate load cap, tr_in is the input slew. This can either be taken from the vendor device datasheet or extracted during device characterization.

VI.3.2 Enhancing Accuracy

It is possible to improve the accuracy by using basic understanding of the current flow in a gate. It is observed that though the error function (when seen as a function of t) at time points of 50% and 80% transition points in output voltage (corresponding to equations 4(ii) and 4(iii)), are smooth in the neighbourhood of t, the error function near the 20% transition point (equation 4(i)) can have local fluctuations. For an improved solution, therefore, it is desirable to fit the early part of the transition (corresponding to the 20% point) with more points. Note that the above procedure in section 3 A basically seeks to obtain a charge flow waveform by matching its derivative, i.e., current at t1, t2 and t3. Near the 20% point t1, it helps to obtain an approximation for the average current flow in the neighborhood of t1 in t, and use this quantity directly as the error to be minimized, instead of current at just t1. For this, we derive an approximation for the total charge flow between two time points t1,0 and t1,2 in the neighborhood. Equation (6) below computes this error charge err_c(t1,0,t1,2). We then divide it by the time interval. 6(b) means that we resort to a simple quadratic interpolation for Idc using calculated values at time points $t_{1,0}$, $t_{1,1}$ and $t_{1,2}$ near the 20% transition region. We have chosen (10%, 15%, 20%) of output voltage transition for this purpose. Now, since comparing one charge quantity and two current quantities (at 50% and 80% points of the transition) in a system of equations creates difficulties in convergence, we normalize this charge term with the time interval over which the approximation is considered. Thus, effectively the first quantity becomes an average current as in eqn 6(d). This is used in place of f(t1) in eqn (4).

$$err_c(t_{1,0},t_{1,2}) = \int_{t_{1,0}}^{t_{1,2}} f(t)dt = \int_{t_{1,0}}^{t_{1,2}} (I_{dc} + I_{load})dt$$

$$I_{dc} = \sum_{i=0}^{2} I_{dc}(t_{1,i}) * \frac{(t - t_{1,(i+1)\mathrm{mod}3}) * (t - t_{1,(i+2)\mathrm{mod}3})}{(t_{1,i} - t_{1,(i+1)\mathrm{mod}3}) * (t_{1,i} - t_{1,(i+2)\mathrm{mod}3})}$$

$$\int_{t_{1,0}}^{t_{1,2}} (I_{load})dt = (p_1(t_{1,2})W_{c1}(t_{1,2}) - p_1(t_{1,0})W_{c1}(t_{1,0})) + \left[p_2(t)\partial W_{c1} / \partial t\right]_{t_{1,0}}^{t_{1,2}} \quad \text{(6)a,b,c,d}$$

$$f(t_{1,0},t_{1,2})_{eff} = \frac{err_c(t_{1,0},t_{1,2})}{(t_{1,2} - t_{1,0})}$$

VI.4 Results

Table VI.2. Error statistics compared to SPICE of delay and slew for proposed and traditional techniques for various benchmark circuits.

Circuit	Weibull slew error μ+σ (%)	Thevenin slew error μ+σ (%)
C3540	4.6	7.7
C499	3.2	7.6
C2670	3.2	7.5
C1908	5.1	6.1
C880	2.3	5.7
Circuit	Weibull, delay error μ+σ (%)	Thevenin, delay error μ+σ (%)
C3540	3.2	7.2
C499	3.9	3.6
C2670	5	7
C1908	2.3	7.5
C880	3.5	7.4

We performed simulations on benchmark circuits synthesized in an industrial 90nm technology. The results of the Weibull-based analysis were compared with numerical

integration results based on the current source model. For comparison of accuracy with a model comparable in time efficiency, we use a Thevenin model. This converges in less than 4 iterations for most cases [50]. This model is at the heart of most of the timing analysis tools. Note that our experiment sought to compare performance of the two methods in moderate to high resistive shielding conditions, since these represent the most difficult cases traditionally. Hence, the benchmark circuits were synthesized targeting such load conditions so that the various approaches can be evaluated in a stringent environment.

Table 2 shows a comparison of the two methods for several ISCAS85 benchmark circuits [38]. As a result of the improvements shown, the error at the $\mu+\sigma$ percentile (68th percentile for normally distributed errors) is reduced by 20-150% in slew. For computing 50% delay the new approach provides up to 220% smaller error at the $\mu+\sigma$ percentile. Fig 4 and 5 show data for large ISCAS85 benchmark circuits. Fig 5 visually depicts how errors in slew rate estimation are reduced with this approach compared to a Thevenin-based flow. Fig. 4(a) shows slew rate error of our approach. Figs 4(b), (c) show the delay performance for our approach. Thevenin-based models are criticized for being unphysical in mapping any complex load to a single Ceff. This is precisely the factor that leads to larger errors in slew rate for the Thevenin case here. We have observed that errors in 10-30% and 70-90% transition time improve substantially because of the underlying physical approach of current source models. This coupled with comparable efficiency is the advantage of the proposed approach. As noted before, convergence of the Newton Raphson system occurs in 3-4 iterations, which is similar to the Thevenin approach.

VI.5 Conclusions

We have investigated the importance of various modeling decisions on the accuracy and complexity of CSMs. In particular we find that a bicubic spline approach to fitting DC current source as a function of input and output voltages is accurate and lends itself to efficient manipulation in timing analysis. Furthermore, we show that the use of a 2-piece internal capacitance model provides good accuracy, while remaining tractable. We then propose a Weibull-based method to perform waveform analysis using the suggested CSM. This technique allows the higher accuracy capabilities of current source models to be leveraged in efficient static timing analysis tools. We show that errors in delay and slew across gates in various benchmark circuits are reduced substantially (by $\mu + \sigma$ error quantile) compared to traditional Thevenin-based approaches. In addition, the approach retains computational efficiency as the Newton-Raphson approach converges in 3-4 iterations, as is the case in Thevenin-based timing flows. Also, very importantly, the approach can be scaled to other parasitic models that have been proposed with a reasonable complexity.

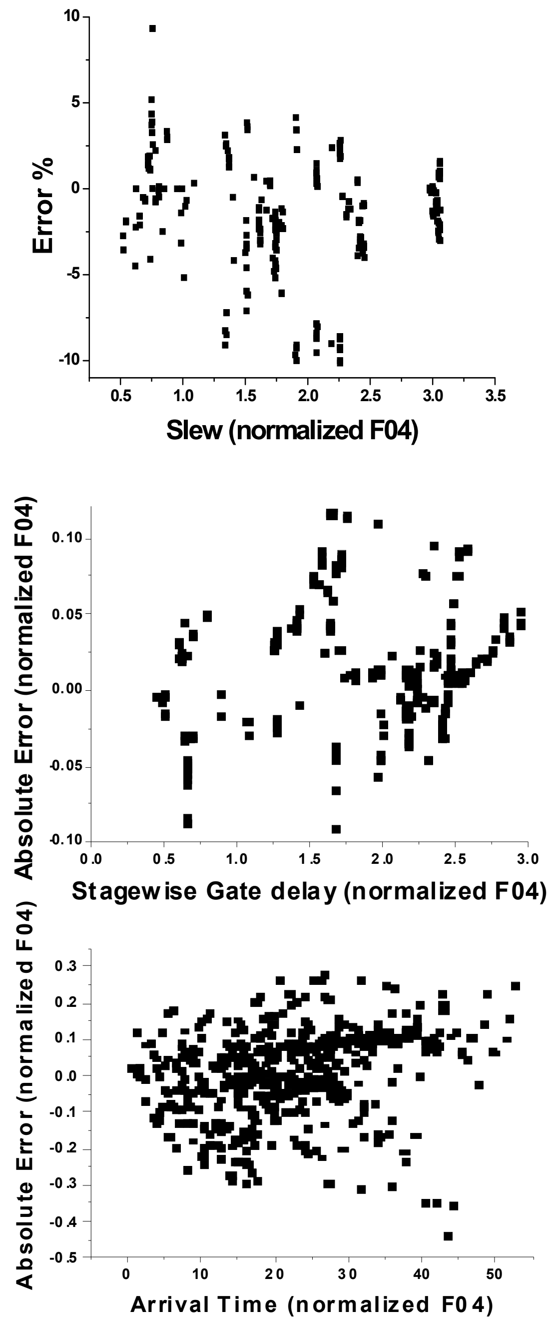


Figure VI.4. (a)% error in slew (b) Absolute error in gate delay (c) Absolute error in arrival time.

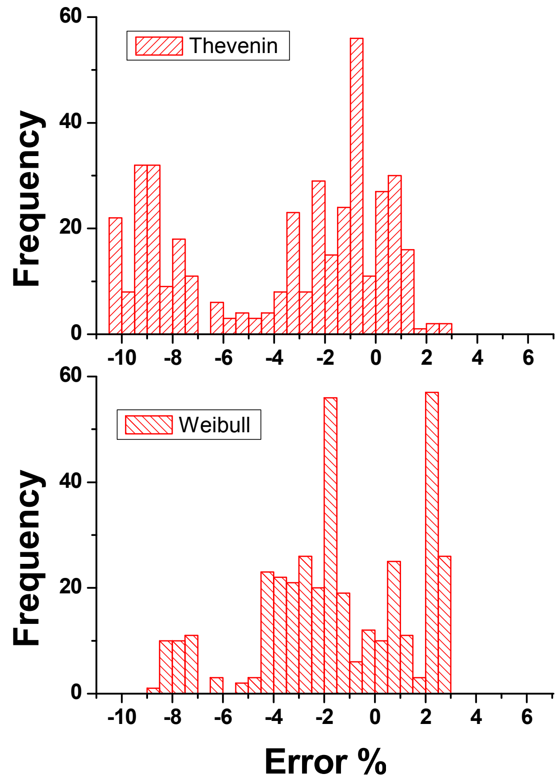


Figure VI.5. Error histograms for slew estimations in two large ISCAS85 circuits given a primary input excitation

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

In this chapter, we summarise our findings regarding the effectiveness of Monte Carlo based algorithms for performance analysis and optimization of digital circuits. We discuss the merits of intelligent sampling techniques for statistical timing analysis and the advantages of implementation on a massively parallel system such as a Graphics Processing Unit. These ideas are further extended to the case of statistical optimization. Using the proposed framework for comparison of statistical optimization techniques, inferences are drawn regarding the advantages of different algorithms in literature and the proposed SLOP algorithm. We conclude with suggestions for further work in the area to enable adoption of these methods in industrial Electronic Design Automation (EDA) tools.

VII.1 Smart Monte Carlo SSTA : SH-QMC

A Stratification + Hybrid Quasi Monte Carlo (SH-QMC) approach is introduced with the objective of improving the efficiency of MC based statistical static timing analysis. The proposed approach extracts timing criticality information of the circuit to intelligently select samples. On average 23.8X and up to 44X reduction is achieved in the number of samples required for timing estimation compared to a random sampling approach. When

implemented on a quad core processor, the approach is shown to be faster than traditional SSTA while achieving comparable accuracy. The scaling trends of SH-QMC with respect to circuit size are also favorable.

VII.2 Acceleration of SH-QMC on Graphics Processing Units (GPUs)

The proposed smart sampling based MC SSTA technique SH-QMC is implemented on a GPU system. It is shown that while approaches such as random sampling based MC SSTA can keep resources utilized on a GPU with a simple implementation in which samples are analyzed in parallel, smart sampling techniques lead to resource utilization with such an implementation owing to the reduced sample size. We propose a gate scheduling technique to expose additional parallelism in the application, while shared memory is utilized to reduce communication bandwidth bottlenecks associated with slow global memory. We have the impressive result that MC SSTA runtime on a multi-GPU is twice as fast as a single STA run on a CPU.

VII.3 Comparison of Statistical Design Optimization Techniques

A lower bound computation method to evaluate statistical design optimization techniques is proposed. Using this bound, we evaluated several current design optimization methods. We found that worst-corner based deterministic approaches result in a pessimistic design with an average area 20% greater than the theoretical lower bound motivating the need for statistically informed methods.

Among these, we compared a smart deterministic approach (Burns) and a robust statistical optimization technique (RGP), as well as a method proposed in this paper call SLOP that uses sample level view of the process variation space. Results show that all statistically aware technique have areas within 10% of the lower bound, on average. More statistically aware techniques (SLOP, RGP) do achieve lower areas; however the additional improvement is only 5.9% on average for RGP and are within 3.7% of the lower bound, with additional runtime cost of 41.5X compared to Burns. SLOP has higher area compared to RGP by 3.8% on average, however is faster than RGP by 5.6X. Overall, the lower bound shows that all statistical methods produce results that are provably close to the theoretical minimum and trade-off additional run time for approaching this minimum to within a couple percent.

VII.4 Smart Sampling based approach for full-chip leakage analysis

We addressed the problem of reducing the sample size for Monte Carlo based leakage analysis. A solution to estimate the total leakage distribution considering inter-die and intra-die components is proposed. It is demonstrated that a direct approach involving smart sampling leads to large sample size. Therefore, samples are reused using proper weights to obtain statistics with respect to different distribution functions. For the largest benchmark considered, the runtime is reduced from 23 hours for a random sampling based approach to 3 minutes for the proposed approach. Also, the library leakage characterization cost is shown to be scalable with respect to the number of process variation variables.

VII.5 Future Work

This work presents compelling arguments to support the adoption of Monte Carlo based algorithms for performance analysis and optimization of digital circuits. However, some challenges remain in their adoption into an industrial flow.

Critics of Monte Carlo based techniques often point to the lack of incremental capability in such techniques, for example, in the case of statistical timing analysis. Monte Carlo based techniques depend on samples generated in the process variation space to obtain statistics of the circuit delay distribution. If incremental changes are made to the design, the technique should ideally be capable of utilizing results from the previous analysis to speed up analysis of the modified design. This work discusses the incremental computation of a fixed percentile of the delay distribution after Engineering Change Order (ECO). However, if the designer performs a series of changes resulting in a significant change to the critical path of the design, full recomputation of samples is required.

To minimize the number of recomputations, one possible approach is to perform periodic checks on the design for changes to the ordering of principal components. A smart technique to perform this check can minimize the number of recomputations. Alternately, one could explore possibilities for incremental regeneration of samples with changes to the ordering of principal components. It may be noted that with the aid of massively parallel machines such as GPUs, the runtime of smart sampling based SSTA is significantly reduced as reported in this work. This reduces the need for incremental techniques for SSTA on such machines. However, incremental techniques could still provide significant value for general-purpose processor based design flows.

Another important requirement for adoption of smart sampling based SSTA in an industrial flow is to find an accurate lower bound on the number of samples required to achieve target accuracy in the performance metric. This is especially important in an optimization loop, where the critical graph changes continuously and could have a significant effect on the sample size required for target accuracy. One method for this computation would be to study the convergence of the circuit delay as more samples are analyzed, via a learning-based approach. An alternate possibility is to perform a theoretical analysis to bound the error for the smart Monte Carlo technique as a function of the number of samples, while considering circuit-specific information (such as slack distribution of top k critical gates, where k is a parameter).

Regarding statistical design optimization, it is noted in the work that smart deterministic approaches such as Burns, which use statistically generated guardbands, achieve results to within 10% of the computed lower bound for the cost objective. More statistically aware techniques, such as SLOP and RGP, provide moderate improvement in accuracy while trading off runtime. Whereas the straightforward parallelism available in SLOP enables speed up on a GPU, this is not true for RGP which does not have such natural parallelism. Conventionally, the primary focus of optimization research has been to improve the quality of optimization solutions especially using more and more statistical information, while computational efficiency is given secondary status at best. This new evidence suggests a paradigm shift for research in the area of statistical design optimization. It shows that the primary focus of research should now be to develop techniques to achieve higher computational efficiency through parallelism or other means,

while the solutions meet a specified quality criterion, say to within 5% of the lower bound metric on average for benchmark circuits considered.

BIBLIOGRAPHY

[1] G. Nanz and L. Camilletti, “Modeling of chemicalmechanical polishing: A review,” IEEE Trans. on Semiconductor Manuf., vol. 8, no. 4, 1995.

[2] C. Mack, “Understanding focus effects in submicrometer optical lithography: A review,” Optical Engineering, vol. 32, no. 10, 1993.

[3] L. Scheffer, “Physical cad changes to incorporate design for lithography and manufacturability,” Proc. ASP-DAC, 2004.

[4] S. Sapatnekar, *Timing*. Springer-Verlag New York, Inc., 2004.

[5] A. Chandrakasan, et al., Design of High-Performance Microprocessor Circuits, IEEE Press, 2000.

[6] C. Visweswariah, et al., "First-Order Incremental Block-Based Statistical Timing Analysis," *Proc. Design Automation Conference*, pp 331-336, 2004.

[7] H. Chang, and S.S.Sapatnekar, "Statistical Timing Analysis Considering Spatial Correlations using a Single Pert-Like Traversal," *Proc. International Conference on Computer-Aided Design*, pp. 621-625, 2003.

[8] K. Chopra, et al., "Parametric yield maximization using gate sizing based on efficient statistical power and delay gradient computation", *Proc. International Conference on Computer Aided Design*, pp 1023-1028, 2005.

[9] F. N. Najm,N. Menezes, "Statistical timing analysis based on a timing yield model," *Proc. Design Automation Conference*, pp. 460-465, 2004.

[10] Y. Zhan, A. Strojwas, X. Li, T. Pileggi, D. Newmark, and M. Sharma, “Correlation aware statistical timing analysis with non-gaussian delay distributions,” *Proc. Design Automation Conference*, 2005.

[11] V. Khandelwal and A. Srivastava, “A general framework for accurate statistical timing analysis considering correlations,” *Proc. Design Automation Conference*, 2005.

[12] J. Singh and S. Sapatnekar, “Statistical timing analysis with correlated non-gaussian parameters using independent component analysis,” *Proc. Design Automation Conference*, 2005.

[13] K. Heloue and F. Najm, “Statistical timing analysis with two-sided constraints,” *Proc. International Conference on Computer Aided Design,* 2005.

[14] A. Devgan and C. Kashyap, “Block-based static timing analysis with uncertainty,” *Proc. International Conference on Computer Aided Design,* 2003.

[15] V. Khandelwal, A. Davoodi, and A. Srivastava, “Efficient statistical timing analysis through error budgeting,” *Proc. International Conference on Computer Aided Design*, 2004.

[16] L. Scheffer, "The Count of Monte Carlo," *TAU*, 2004.

[17] J. Xiong, V. Zolotov, and L. He, “Robust extraction of spatial correlation,” *Proc. ISPD*, 2006.

[18] F. Liu, “How to construct spatial correlation models: A mathematical approach,” *Proc. TAU Int. Work. on Timing*, 2007.

[19] F. Liu, “A general framework for spatial correlation modeling in vlsi design,” International workshop on Timing issues in the specification and synthesis of digital systems, 2007.

[20] N. A. Cressie, *Statistics for Spatial Data*. Wiley Series in Probability and Statistics, 1993.

[21] B. Cline, K. Chopra, and D. Blaauw, “Analysis and modeling of cd variation for statistical static timing,”, *Proc. International Conference on Computer Aided Design*, 2006.

[22] S. Bhardwaj, S. Vrudhula, P. Ghanta, and Y. Cao, “Modeling of intra-die process variations for accurate analysis and optimization of nano-scale circuits,” *Proc. Design Automation Conference*, 2006.

[23] I. Jolliffe, *Principal Component Analysis*. Springer-Verlag New York, Inc., 2002.

[24] H. Chang, S.S.Sapatnekar, "Full-chip analysis of leakage power under process variations including spatial correlations", *Proc. Design Automation Conference,* pp. 523-528, 2005.

[25] X. Li, J.le, L.T.Pileggi, "Projection-based statistical analysis of full-chip leakage power with non-log-normal distributions", *Proc. Design Automation Conference,* pp. 103-108, 2006.

[26] K.R.Heloue, N.Azizi, F.N.Najm, "Modeling and estimation of full-chip leakage current considering within-die correlation", *Proc. Design Automation Conference,* pp. 93-98, 2007.

[27] T.Li, W.Zhang, Z.Yu, "Full-chip Leakage Analysis in Nano-scale Technologies: Mechanisms: Variation Sources, and Modeling", *Proc. Design Automation Conference,* pp. 594-599, 2008.

[28] R.Y. Rubinstein, Simulation and the Monte Carlo Method, John Wiley & Sons, Inc., 1981.

[29] V.Veetil, D.Sylvester, D.Blaauw, "Efficient Monte Carlo based Incremental Statistical Timing Analysis", *Proc. Design Automation Conference,* pp. 676-681, 2008.

[30] M. Keramat and R. Kielbasa, "Worst Case Efficiency of LHSMC Yield Estimator of Electrical Circuits." *Proc. ISCAS*, v. 3, pp. 1660 - 1663, 1997.

[31] R. Kanj, R. Joshi, and S. Nassif, "Mixture Importance Sampling and Its Application to the Analysis of SRAM Designs in the Presence of Rare Failure Events," *Proc. Design Automation Conference*, pp. 69-72, 2006.

[32] S. Tasiran and A. Demir, "Smart Monte Carlo for Yield Estimation," *TAU*, 2006.

[33] A. Singhee and R.A. Rutenbar, "From Finance to Flip Flops: A Study of Fast Quasi-Monte Carlo Methods from Computational Finance Applied to Statistical Circuit Analysis", *Proc. ISQED*, pp. 685-692, 2007.

[34] E. Hlawka,"Funktionen von beschrankter Variation in der Theorie der Gleichverteilung", *Ann. Mat. Pura Appl.*, 54, pp 325-333., 1961.

[35] I.M.Sobol, "The Distribution of Points in a Cube and the Approximate Evaluation of Integrals", *USSR Comp. Math and Math. Phys.,* 7(4), pp. 86-112, 1967.

[36] P. Bratley, B. Fox, "Algorithm 659: Implementing Sobol's Quasirandom Sequence Generator", *Trans. on Mathematical Software*, 14(1), pp 88-100, 1988.

[37] M. Stein, "Large Sample Properties of Simulations Using Latin Hypercube Sampling," *Technometrics*, 29,pp 143-151, 1987.

[38] F. Brglez and H. Fujiwara, "Neutral netlist of ten combinational benchmark circuits and a target translator in FORTRAN", Special session on ATPG and fault simulations, *Proc. ISCAS*, pp. 695-698,1985.

[39] F.N.Najm, "A Survey of Power Estimation Techniques in VLSI Circuits," *IEEE Transactions on Very Large Scale Integration(VLSI) Systems*, Vol. 2, pp. 446-455, 1994.

[40] A. Papoulis, Probability, Random Variables and Stochastic Processes, McGraw-Hill Inc., New York 1991.

[41] R.Rao, A.Devgan, D.Blaauw, D.Sylvester, "Parametric Yield Estimation Considering Leakage Variability," *Proc. Design Automation Conference,* pp. 442-447, 2004.

[42] J.F. Croix and D.F. Wong, "Blade and Razor: Cell and Interconnect Delay Analysis Using Current-Based Models", *Proc. Design Automation Conference*, pp. 386-389, 2003.

[43] I. Keller, K. Tseng and N. Verghese, "A robust cell-level crosstalk delay change analysis", *International Conference on Computer-Aided Design*, pp 147-154, 2004.

[44] B. Tutuianu , R. Baldick, and M.S. Johnstone, "Nonlinear Driver Models for Timing and Noise Analysis", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1510-1521, Nov. 2004.

[45] P.Li, E.Acar,"A Waveform Independent Gate Model for Accurate Timing Analysis", pp 363-365, ICCD 2005.

[46] C.S. Amin, C. Kashyap, N. Menezes, K. Killpack, and E. Chiprout, "A multi-port current source model for multiple-input switching effects in CMOS library cells", *Design Automation Conference*, pp. 247-252, 2006.

[47] H.Fatemi,S.Nazarian,M.Pedram, "Statistical logic cell delay analysis using a current-based model", *Design Automation Conference*, 2006.

[48] C.K. Tsai, M. Marek-Sadowska: "An Interconnect Insensitive Linear Time-Varying Driver Model for Static Timing Analysis," *International Symposium on Quality Electronics Design*, pp. 654-661, 2005

[49] C.S. Amin, F. Dartu, and Y.I. Ismail, "Weibull-based analytical waveform model," *International Conference on Computer-Aided Design*, pp. 161-168, 2003.

[50] F.Dartu, N.Menezes, L.T.Pileggi, "Performance Computation for Precharacterized CMOS Gates with RC Loads", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 544-553, May 1996.

[51] C.D.Boor, A Practical Guide to Splines, Springer, 2001.

[52] R. Gandikota, D. Blaauw, D. Sylvester, "Modeling crosstalk in statistical static timing analysis", Proc. Design Automation Conference, pp. 974-979, 2008.

[53] A.Singhee, S.Singhal and R.A.Rutenbar, "Practical, fast Monte Carlo statistical static timing analysis: why and how," Proc. International Conference on Computer-Aided Design, pp. 190-195, 2008.

[54] A.Singhee, S.Singhal and R.A.Rutenbar, "Exploiting Correlation Kernels for Efficient Handling of Intra-Die Spatial Correlation, with Application to Statistical Timing," Proc. Design, Automation and Test in Europe, pp. 856-861, 2008.

[55] V. Veetil, D. Sylvester and D.Blaauw, "Efficient Monte Carlo based Incremental Statistical Timing Analysis," Proc. Design Automation Conference, pp. 676-681, 2008.

[56] J.Jaffari, and M.Anis, "On efficient Monte Carlo-based statistical static timing analysis of digital circuits," Proc. International Conference on Computer-Aided Design, pp. 196-203, 2008.

[57] V.Veetil, D.Sylvester, D.Blaauw, S.Shah, and S.Rochel, "Efficient Smart Sampling-Based Full-Chip Leakage Analysis for Intra-Die Variation Considering State Dependence," Design Automation Conference, 2009.

[58] K. Gulati, S.P. Khatri, "Accelerating Statistical Timing Analysis Using Graphics Processing Units", Proc. ASP-DAC, pp. 260-265, 2009.

[59] J.D.Owens, et al., "GPU Computing" Proceedings of the IEEE, vol. 96(5), pp. 879-899, 2008.

[60] http://www.nvidia.com/object/cuda_home.html#

[61] J.Xiong et al, " Criticality computation in parameterized statistical timing", Proc. Design Automation Conference, pp. 63-68, 2006.

[62] http://www.nvidia.com/object/product_tesla_s1070_us.html

[63] C.Visweswariah, "Death, taxes and failing chips.", Design Automation Conference, 2003. pp. 343-347.

[64] J.Jaffari, M.Anis, "On efficient Monte Carlo-based statistical static timing analysis of digital circuits.", *International Conference on Computer Aided Design* 2008, pp. 196-203.

[65] Y.Abulatia, A.Kornfeld, "Estimation of FMAX and ISB in microprocessors.", *IEEE Transacations on VLSI Systems*, Oct 2005, pp. 1205-1209.

[66] A.Agarwal, K.Chopra, D.Blaauw, V.Zolotov , "Circuit optimization using statistical static timing analysis.", *Design Automation Conference* 2005, pp. 321-324.

[67] S.Borkar, T.Karnik, S.Narendar, V.De , "Parameter variations and impact on circuits and microarchitecture.", *Design Automation Conference* 2003, pp. 338-342.

[68] K.A.Bowman, S.G.Duvall, J.D.Meindl, "Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration.", IEEE J. Solid-State Circuits, Nov 2002, pp. 183-190.

[69] S.M.Burns, M.Ketkar, N.Menezes, K.A.Bowman, J.W.Tschanz, V.De , "Comparative Analysis of Conventional and Statistical Design Techniques.", *Design Automation Conference* , 2007. pp. 238-243.

[70] J.P.Fishburn, A.E.Dunlop , "TILOS: A posynomial programming approach to transistor sizing", *International Conference on Computer Aided Design*, 1985.

[71] M.Mani, A.Devgan, M.Orshansky, "An efficient algorithm for statistical minimization of total power under timing yield constraints.", *Design Automation Conference*, 2005. pp. 309-314.

[72] J.Singh, L.Zhi-Quan, S.S.Sapatnekar , "Robust Gate Sizing by Geometric Programming.", *Design Automation Conference*, 2005. pp. 315-320.

[73] A. Srivastava, D. Sylvester, and D. Blaauw, “Statistical optimization of leakage power considering process variations using dual-vth and sizing,” *Proc. Design Automation Conference*, 2004.

[74] A. Agarwal, K. Chopra, and D. Blaauw, “Statistical timing based optimization using gate sizing,” *Proc. DATE*, 2005.

[75] V. Sundararajan, S.Sapatnekar, K.K.Parhi , "Fast and Exact Transistor Sizing Based on Iterative Relaxation.", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2002, Vol. 21, No. 5.

[76] S.Boyd, S.J.Kim, L.Vandenberghe, A.Hassibi , "A tutorial on geometric programming.", Springer Netherlands, 2007, Vol. 8, No. 1.

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

[78] L.Pang, B.Nikolic, "Measurement and analysis of variability in 45nm strained-Si CMOS technology.", Custom Integrated Circuits Conference, 2008. pp. 129-132.

[79] S.Reda, S.R.Nassif, "Analyzing the impact of process variations on parametric measurements: Novel models and applications.", Design, Automation and Test in Europe, 2009. pp. 375-380.

[80] M.Kanno et. al, "Empirical Characteristics and Extraction of Overall Variation for 65-nm MOSFETs and Beyond", IEEE Symposium on VLSI Technology, 2007. pp. 88-89.

[81] S.H.Kulkarni, D.Sylvester, D.Blaauw, "A Statistical Framework for Post-Silicon Tuning through Body Bias Clustering", ICCAD 2006.

[82] J. Tschanz, J. Kao, S. Narendra, R. Nair, D. Antoniadis, A. Chandrakasan, and V. De, “Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage,” JSSC, vol. 37, November 2002.