UCLA
UCLA Electronic Theses and Dissertations

Title
Conventional and Machine Learning Assisted High Sigma Analysis

Permalink
https://escholarship.org/uc/item/67q1601f

Author
Wu, Wei

Publication Date
2016

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

University of California
Los Angeles

Conventional and Machine Learning Assisted High Sigma Analysis

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Electrical Engineering

by

Wei Wu

2016

© Copyright by
Wei Wu
2016

Abstract of the Dissertation

Conventional and Machine Learning Assisted High Sigma Analysis

by

Wei Wu

Doctor of Philosophy in Electrical Engineering

University of California, Los Angeles, 2016

Professor Lei He, Chair

Statistical circuit simulation exhibits increasing importance for circuit designs under process variations. In particular, high sigma analysis is needed to optimize highly-duplicated standard cells, where an extremely rare circuit failure event could lead to catastrophe of the entire chip. Conventional importance sampling (IS) approaches perform high sigma analysis efficiently at low dimensionality, but perform poorly either when there are a larger number of process variation variables, or when the failing samples are distributed in multiple regions.

In this dissertation, a series of high sigma analysis approaches have been proposed. First, a high dimensional importance sampling (HDIS) is presented to mitigate the dimensionality problem in traditional IS. A maximum entropy (MAXENT) based approaches is proposed to model the distribution of circuit performance under process variation. MAXENT models the distribution in overall, but does not specifically model the tail. To fix this issue, a piecewise distribution model (PDM) is proposed to consider the distribution as multiple segments and model each segment using MAXENT, hence improve high accuracy in the high sigma tail.

Moreover, two machine learning assisted approaches are proposed for high

sigma analysis. The rare-event microscope (REscope) trains classifier(s) to filter out the majority of the unlikely-to-fail samples and surgically look into those likely-to-fail ones, whose distribution is analytically modeled as a generalized pareto distribution to estimate failure probability. Finally, hyperspherical clustering and sampling (HSCS) algorithm is proposed to cluster failing samples and to perform importance sampling around those clusters to cover all failure regions. Experiment results demonstrate that the proposed approaches are 2-3 orders faster than Monte Carlo, and more accurate than both academia solutions such as IS, Markov Chain Monte Carlo, and industrial solutions such as mixture IS used by ProPlus Design Automation, Inc.

The dissertation of Wei Wu is approved.

Sudhakar Parmati

Miodrag Potkonjak

Lei He, Committee Chair

University of California, Los Angeles

2016

Table of Contents

1 Introduction **1**

1.1 Background 1

1.2 Motivation of Stochastic Circuit Analysis 3

1.3 Organization of the Dissertation 5

2 Stochastic Behavior Modeling and Analysis **8**

2.1 Overview of Stochastic Behavior Modeling 8

2.1.1 General Stochastic Modeling Method using SPICE Simulation 8

2.1.2 Preliminary knowledge 9

2.2 Existing Approach: Point Estimation Method 10

2.2.1 Moment Matching via Padé Approximation 11

2.2.2 Numerical Stability Issue and Scaling 12

2.3 Maximum Entropy 13

2.3.1 Finding a Distribution through Maximizing Entropy . . . 13

2.3.2 Algorithm Stability 15

2.3.3 Experiment Results 16

2.4 Conclusion 24

3 High Dimensional Importance Sampling for Fast Yield Analysis 25

3.1 Introduction of High Sigma Yield Analysis 25

3.2 Preliminary Knowledge of Importance Sampling 29

3.2.1 Formulation of Probability Estimation 29

3.2.2 Importance Sampling (IS) 30

3.2.3 Failure Analysis of Importance Sampling 31
3.3 High-Dimensional Importance Sampling Algorithm 32
3.3.1 Algorithm Overview 32
3.3.2 Shift and Reshape Sampling Distribution 34
3.3.3 Conditional Probability Calculation 37
3.3.4 Boundedness Analysis 37
3.4 Experiment Results 39
3.4.1 SRAM Circuit and Variation Modeling 39
3.4.2 SRAM Cell with Reading Failure 40
3.4.3 Sense Amplifier for Target Gain 41
3.5 Conclusion 45

4 Piecewise Distribution Model 47
4.1 Motivation 47
4.2 Piecewise Distribution Model 49
4.2.1 Building the Segment1 Distribution 50
4.2.2 Selecting the Optimal Segment1 Distribution 51
4.2.3 Shifting Input Distributions and Building the Segment2 Distribution 53
4.2.4 Reweighing Segment2 via Conditional Probability 54
4.3 Experiment Results 56
4.3.1 Experiment Settings 56
4.3.2 Experiment on Mathematically Known Distribution 58
4.3.3 Experiment on Circuits 61
4.3.4 Speedup Comparison 64

4.4 Conclusion 65

5 Rare-event Microscope: Aiding Yield Analysis with Classification Algorithms 66

5.1 Multiple Failure Regions 66

5.1.1 Limitation of Existing Works 66

5.2 Preliminary Knowledge 68

5.2.1 Modeling Rare Events using GPD 68

5.3 Rare-event Microscope Algorithm 71

5.3.1 Algorithm overview 71

5.3.2 Presampling 72

5.3.3 Parameter pruning 72

5.3.4 Nonlinear SVM classifier 73

5.3.5 Fitting the tail distribution to GPD 74

5.4 Experiment Results 75

5.4.1 Charge pump circuit and experiment setting 75

5.4.2 Handling multiple separate failure regions 77

5.4.3 Parameter weighing and pruning 79

5.4.4 Accuracy and Efficiency 80

5.5 Conclusion 81

6 Hyperspherical Clustering and Sampling: Aiding Yield Analysis with Hyperspherical Clustering 83

6.1 Existing Works that Handle Multiple Failure Regions 84

6.2 Locating Min-Norm Point for Importance Sampling 84

6.3 Hyperspherical Clustering and Sampling 85
6.3.1 Algorithm Overview 85
6.3.2 Hyperspherical Clustering 86
6.3.3 Multiple Mean-Shift Importance Sampling 89
6.4 Experiment Results 92
6.4.1 Evaluation on Mathematically Known Distribution 92
6.4.2 Experiments on Charge Pump Circuit 95
6.5 Conclusion 101

7 Summary 103

A Network Compression for Deep Learning Inference on Mobile Platforms 106

A.1 Introduction 107
A.2 Network Compression 109
A.2.1 Knowledge Transfer 109
A.2.2 Network Parameter Reduction 112
A.2.3 Algorithm Flow 116
A.3 Experiments 117
A.3.1 MNIST 117
A.3.2 CIFAR-10 121
A.3.3 Discussions 126
A.4 Conclusion 127

B FPGA Based for Deep Convolutional Neural Network 128

B.1 Introduction 129

B.2 Basic CNN Architecture 131
B.2.1 Preliminary of Convolutional Neural Network 131
B.2.2 Network Architecture of a Real-Life CNN 132
B.3 Uniform Accelerator for Convolutional and Fully-Connected Layer 134
B.3.1 Reformulation of the Convolution Operation 134
B.3.2 Design of Convolution-Flattening and Inner-Product Accelerators 135
B.4 FPGA based CNN with Uniform Accelerator 136
B.4.1 Overall Architecture 136
B.4.2 Accelerator allocation 138
B.4.3 Automation and Scalability 139
B.5 Experiments 139
B.5.1 Experiment Setup 139
B.5.2 Experimental Results 140
B.6 Conclusion 141

References **143**

List of Figures

1.1 Process, supply voltage, and temperature (PVT) variations in circuit designs 1

1.2 Variability induced by insufficient lithography resolution 2

1.3 Simulation results of threshold voltage at different technology nodes 3

2.1 General Stochastic Modeling Method utilizing SPICE Simulation 8

2.2 Performance and PDF scaling for stability 13

2.3 Schematics of 6T SRAM Cell Circuit 18

2.4 Schematics of Delay Chain Circuit 19

2.5 PEM lack of stability on SRAM circuit (200 samples) 19

2.6 PEM stability on SRAM circuit (250 samples) 20

2.7 PEM lack of stability on SRAM circuit (300 samples) 20

3.1 Basic idea in proposed algorithm. (Noted that $\mathcal{T} = \{Y|Y \geq t\}$ contains $\mathcal{S} = \{Y|Y \geq t_c\}$). 26

3.2 The scale illustration of likelihood ratios in importance sampling. 31

3.3 Basic idea in proposed algorithm. (Noted that $\mathcal{T} = \{Y|Y \geq t\}$ contains $\mathcal{S} = \{Y|Y \geq t_c\}$). 32

3.4 Overall flow in proposed algorithm. (Noted that $\mathcal{T} = \{Y|Y \geq t\}$ contains $\mathcal{S} = \{Y|Y \geq t_c\}$). 33

3.5 The distance between centroid points of two subsets along each parameter axis. 36

3.6 Functional diagram of an SRAM circuit. 40

3.7 The schematic of a sense amplifier circuit. 42

3.8 Comparison between different methods in terms of the failure probability estimation and figure of merit 43

4.1 PDM contains 2 Phases: building the Segment1 distribution and selecting the optimal Segment1 distribution; shifting input parameters to build the Segment2 distribution, and estimating the final probability 49

4.2 Slope of Gaussian vs Non-Gaussian Distribution 51

4.3 Spearman's Correlation of Distributions with Different Moments . 52

4.4 Segment1 Comparison using Spearman's Correlation Results . . . 53

4.5 Shape Issue in Conditional Probability 56

4.6 LogNormal PDF 58

4.7 LogNormal Sigma Behavior 59

4.8 Clock Path PDF 61

4.9 Clock Path Sigma Behavior 61

4.10 Op. Amp PDF 62

4.11 Op Amp Sigma Behavior 63

5.1 Mean-shifting based methods on a yield analysis problem with two disconnected failure regions 67

5.2 Model the tail of lognormal using GPD 70

5.3 The REscrope framework consists of four components: presampling, parameter pruning, classification, tail distribution estimation. 71

5.4 A block diagram of PLL 75

5.5 Simplified schematic of the charge pump circuit 76

5.6 How multiple failure regions are handled in HDIS [WGC14], SB [SR08], and REscope 78

5.7 Weight of all 108 process variations in charge pump circuit 80

5.8 Modeling the tail of the mismatch current distribution 81

6.1 The HyperSpherical Clustering and Sampling (HSCS) algorithm consists of two phases: 1) hyperspherical clustering, 2) multiple mean-shift importance sampling. 86

6.2 2-dimensional sample space with two disjoint failure regions $\mathcal{S}_1$ and $\mathcal{S}_2$ 92

6.3 Spherical presampling to collect failed samples 93

6.4 Spherical k-means might converge to local optimal with "improper" initial centroids 94

6.5 Sample coverage of the modified mixture importance sampling . . 94

6.6 Multiple failure region coverage test MC, HDIS [WGC14], Spherical IS [DQS08], and HSCS 97

6.7 Clustering maximization objective while changing the targeted number of clusters 99

6.8 Number of actually clusters may be small than the targeted number of clusters 99

6.9 Convergence curve of Monte Carlo, HDIS, Spherical IS, and the proposed HSCS 101

6.10 Robustness test of HSCS with 10 replications 102

A.1 Transferring the knowledge from the original cumbersome teacher neural network to thinner, shallower student networks 110

A.2 A fully-connected layer followed by ReLU activation functions . . 113

A.3 Singular value decomposition (SVD) 114
A.4 Decompose the weight matrix into two smaller matrices 115
A.5 Transferring the knowledge from LeNet to a shallow neural network with only one fully-connected layer 118
A.6 Transferring the knowledge from deep convolutional neural network to a shallow neural network 123

B.1 Convolutional Layer 131
B.2 An illustration of the architecture of AlexNet [KSH12] 132
B.3 Convolution Flattening 134
B.4 Convolution-Flattening (CF) Accelerator 135
B.5 Inner-Product (IP) Accelerator 136
B.6 Overall architecture 137

List of Tables

2.1 Variation Sources Modeled in CMOS Transistors 18

2.2 Accuracy Comparison with 200 sample for SRAM and 400 samples for delay chain 22

2.3 Accuracy Comparison with 300 sample for SRAM and 600 samples for delay chain 23

2.4 Speedup Comparison of MAXENT, PEM, and MC 24

3.1 Process Parameters of MOSFETs. 39

3.2 Comparison for SRAM bit-cell analysis with 90% target accuracy and confidence level. 40

3.3 Comparison for sense amplifier analysis with 90% target accuracy and confidence level 44

4.1 Parameters of MOSFETs 57

4.2 Sigma Error for LogNormal 60

4.3 Sigma Error for Circuits 64

4.4 Speedup Comparison 65

5.1 Comparison of the accuracy and efficiency on charge pump circuit 80

6.1 Process variation parameters for each transistor 96

6.2 Accuracy and efficiency evalution on 70-dimensional charge pump circuit 98

A.1 Comparison between the teacher network (LeNet) and student networks in terms of accuracy, # of weight parameters, and computational cost 119

A.2 Compressed the weight matrix by decompose it to the multiplication of two smaller matrices 119

A.3 Knowledge transfer to compression by removing inactive nodes for CIFAR-10 dataset 124

A.4 Compressed the weight matrix by SVD decomposition for CIFAR-10 dataset 124

B.1 A detail breakdown of the network parameter and computational load in each layer of AlexNet [KSH12] 133

B.2 Resource consumption of convolution-flattening accelerators at each layer 138

B.3 Overall Resource Utilization 140

B.4 Performance Comparison 141

Acknowledgments

During the development of the dissertation, I received support and encouragement from a great number of individuals. I would like to take this opportunity to acknowledge them with my full gratitude.

I would like to express the deepest appreciation to my advisor, Professor Lei He, for his generous help over the years. His continuous guidance and encouragement foster an environment of academic excellence that helps me develop the problem solving skills, and eventually grow into an independent researcher. This dissertation should not have come to its completion without his tremendous support.

I would like to thank my doctoral committee members, Professor Kung Yao, Professor Miodrag Potkonjak, and Professor Sudhakar Parmati for their suggestions to improve this work. Their time and effort are greatly appreciated.

My sincere thanks to the collaborators Dr. Fang Gong and Mr. Rahul Krishnan for your help on developing some of the algorithms and performing the experiments in this dissertation. I also want to express my thanks to Prof. Xin Li from CMU for sharing their source code for cross comparison.

I am also very grateful to my managers and mentors during the internships, including Mr. Pranav Mistry and Dr. Jeff Pierce from Samsung Research America, Dr. Shijie Zhang from Google, Dr. Jinjun Xiong and Dr. Vladimir Zolotov from IBM T.J. Watson Research Center. Thank you for your mentorship and your valuable industry insights.

I treasure the opportunity that I have worked with a group of wonderful and brilliant people at the UCLA Design Automation Lab. In particular, I thank Dr. Roy Lee, Dr. Wenyao Xu, Mr. Juexiao Su, Mr. Min Gao, Mr. Tianheng Tu, Mr. Webber Lee, Mr. Feng Shi, Mr. Zhuo Jia, Mr. Xiao Shi, Mr. Yingjun Wu, Mr. Tianyu Li, Ms. Monica Wang, Ms. Yunxuan Yu, and Mr. Junyi Xie for being a round with helpful hands.

Last, and most of all, I am grateful to my family. I thank my wife Shujuan Wang in particular. This dissertation becomes so humble compared to her continuous support and love.

Vita

2007	B.S. (Electrical Engineering), Beihang University
2010	M.S. (Electrical Engineering), Beihang University
2010	Research Assistant, Tsinghua University
2011	Research Assistant, Nanyang Technological Unviersity
2012–2016	Ph.D. Graduate Student Researcher/Teaching Assistant, Electrical Engineering Department, UCLA
2014	Research Intern, Samsung Research America
2015	Software Engineering Intern, Google Inc.
2015	Research Intern, IBM T.J. Watson Research Center

Publications

R. Krishnan, **W. Wu**, S. Bodapati, and L. He, "Accurate Multi-segment Density Estimation Through Moment Matching", *TCAD (minor revision).*

W. Wu, YL. Chen, Y. Ma, CN. Liu, JY. Jou, S. Pamarti, and L. He, "Wave Digital Filter based Analog Circuit Emulation on FPGA", *ISCAS, 2016.*

W. Wu, S. Bodapati, and L. He, "Hyperspherical Clustering and Sampling for

Rare Event Analysis with Multiple Failure Region Coverage", *ISPD, 2016*.

YL. Chen, **W. Wu**, CN. Liu, and L. He, "Incremental Latin Hypercube Sampling for Lifetime Stochastic Behavioral Modeling of Analog Circuits", *ASPDAC, 2015*.

W. Wu, X. Li, L. He, and D. Zhang, "Accelerating the Iterative Linear Solver for Reservoir Simulation on Multicore Architectures", *ICPADS, 2014*.

W. Wu, W. Xu, R. Krishnan, YL. Chen, and L. He, "REscope: High-dimensional Statistical Circuit Simulation towards Full Failure Region Coverage", *DAC, 2014*.

W. Wu, F. Gong, G. Chen, and L. He, "A Fast and Provably Bounded Failure Analysis of Memory Circuits in High Dimensions", *ASPDAC, 2014*.

R. Krishnan, **W. Wu**, F. Gong, and L. He, "Stochastic Behavior Modeling of Analog/Mixed-Signal Circuits by Maximizing Entropy", *ISQED, 2013*.

W. Wu, F. Gong, R. Krishnan, H. Yu, and L. He, "Exploiting Parallelism by Data Dependency Elimination: A Case Study of Circuit Simulation Algorithms", *Design & Test, 2013*.

C. Zhang, **W. Wu**, H. Huang, and H. Yu, "Fair Energy Resource Allocation by Minority Game Algorithm for Smart Buildings", *DATE, 2012*.

X. Chen, **W. Wu**, H. Yu, Y. Wang, and H. Yang, "An EScheduler-based Data Dependence Analysis and Task Scheduling for Parallel Circuit Simulation", *TCAS-II, 2011*.

CHAPTER 1

Introduction

1.1 Background

The integrated circuit (IC) technology has been scaled from the micrometer to nanometer level in the past decade. The lead foundries TSMC and Global-Foundries have announced their state-of-art CMOS process technology featuring FinFET transistors with 16nm and 14nm, respectively [Cou13]. Indeed, industry favors the newer and smaller technology because they are usually associated with higher integration density, lower power, and smaller gate capacitor (shorter delay) and etc [KAC12]. It has become, however, extremely challenging to guarantee high-precision and high-reliability in modern IC manufacturing due to inevitable uncertainties and variations.

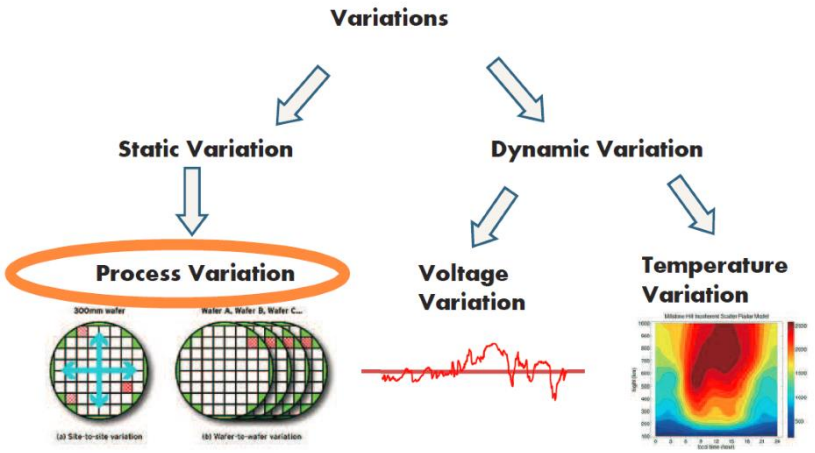


Figure 1.1: Process, supply voltage, and temperature (PVT) variations in circuit designs

Uncertainty in Manufacturing process, supply voltage, and temperature (PVT) variations are three major types of variations in circuit design, as illustrated in Figure 1.1. While the voltage and temperature variations are "dynamic" as they depend on operating environment, the process variations are "static" once the circuit is fabricated. The "static" process variations have been identified as the leading source introducing unavoidable uncertainties in circuit behavior and induce significant yield loss [EBS97, CCS04, BDM02].

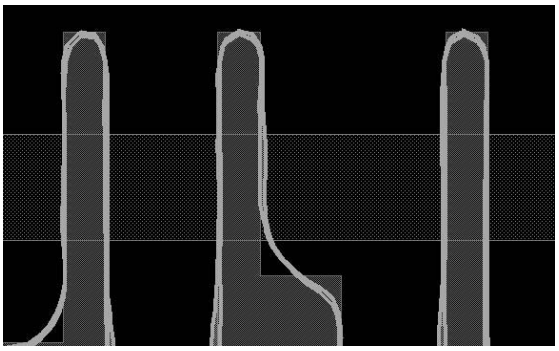
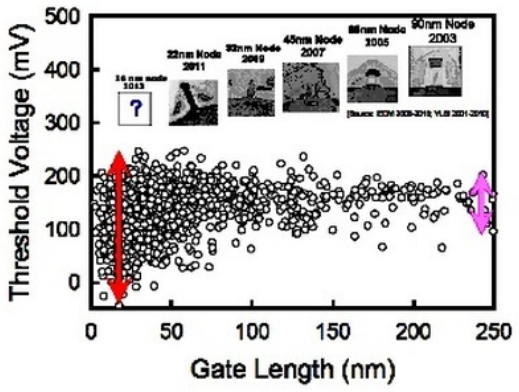


Figure 1.2: Variability induced by insufficient lithography resolution

Process variations are induced by the uncertainties during chemical mechanical polishing (CMP), etching, lithography and other manufacturing process. In particular, the resolution of the lithography was found to be at least 5-10nm, due to the limitation of secondary electron generation in electron-beam-irradiated solids [LYL09]. It is, obviously, not fine enough to fabricate the latest FinFET devices, whose channel length is only 10 to 16nm or even less [Cou13]. The gap between the design and manufacturing leads to the difference between a ideal design and actual fabricated circuit, which are illustrated in Figure 1.2 [CDH06]. It is difficult to guarantee the correctness of their physical design parameters, such as channel width, length and oxide thickness. The variations of these physical parameters could be translated into the threshold voltage, which induces large amount of variations in circuit behavior, such as leakage power, timing delay, output swing, etc.



Courtesy of Prof. Yiming Li, National Chiao Tung University, Taiwan

Figure 1.3: Simulation results of threshold voltage at different technology nodes

In Figure, 1.3, the simulation results of threshold voltage at different technology nodes under is plotted while considers the process variation. It is obvious that process variation continues increasing when the transistor size scales down. The variations on all the transistors in a circuit add up and impact the circuit performance, or even lead to significant yield loss.

1.2 Motivation of Stochastic Circuit Analysis

To compensate for the effects of process variations during the design phase, stochastic analysis tools are urgently sought to accurately characterize the random process variations and efficiently predict their effects on circuit behavior.

First, it is important to understand the stochastic behavior of circuit performance during the pre-silicon phase, rather than after the fabrication. A typical example is the flexible electronics, such as rollable displays, disposable RFID, etc, which suffer from the aging effects [HHC11]. To ensure the quality and durableness, circuit designers need to evaluate the stochastic behavior of circuit performance at each time step, and to find the optimal circuit parameters during the design time. To reliably estimate the probabilistic circuit performance, Monte

Carlo (MC) method [JL89] is the "gold standard", but at the cost of simulation runtime. It needs to run computational expensive transistor-level SPICE simulation on a huge number of samples so as to describe the probabilistic distribution (e.g., PDF and CDF) of circuit performance.

Several other statistical methodologies [Nas01, VWG06] are proposed to estimate the probabilistic performance. However, they either assume the linearity of the circuit, or assume that the unknown performance follows Gaussian distribution, which are not valid in practice.

Recently, asymptotic probability extraction (APEX) [LLG04] and Point Estimation based Method (PEM) [GYH11] take advantage of padé approximation by matching the probabilistic moments to time moments and calculate the distribution as the impulse response corresponding to the time moments [PR90]. However, both of these two approaches becomes numerically instable when high order moments is used [FF95]. A fast yet stable algorithm to analyze the stochastic circuit behavior is highly desired.

Second, standard circuit cell, which is duplicated for millions of times, and for critical circuits, such as phase locked loop (PLL), which stabilize clock for the entire chip, a extremely small failure probability may cause the catastrophe of the entire chip. In particular, SRAM cell design tends to adopt the most advanced process technology to achieve the minimum-sized cells and thus becomes more vulnerable to process variations. The failure probability of a SRAM cell should be kept extremely small (e.g., 10^{-4}-10^{-7}), making the failure event become a "rare event" [AN06]. For such a small failure probability, MC method becomes infeasible in practice. Fast statistical approaches for rare event modeling are proposed in the passed decade, which can be categorized in the following groups: 1) importance sampling based approaches [KJN06, DQS08, QTD10, KHT10, WGC14] 2) classification based approaches [SR09, SR08]. The former constructs a new "proposed" sampling distribution under which a "rare event" becomes "less rare"

so that more failures can be easily captured, while the latter uses a classifier to block the majority of the samples and only simulates the samples that are likely to fail.

Most of the existing approaches can be successfully applied to low-dimensional problems with small number of variables but, in general, perform poorly in high dimension. Moreover, none of these approaches considers the possibility that the fail samples fall in multiple failure regions. Therefore, an approach that could accurately analyze the rare failure probability with high-dimensional process variations and multiple separate failure regions is needed.

1.3 Organization of the Dissertation

The research presented in this dissertation mainly focuses on process variation modeling and analysis using numerical and statistical techniques, which studies two important issues mentioned in Chapter 1.2: stochastic behavioral modeling and analysis, rare failure probability analysis in high dimension, and realistic scenario where circuits failure are caused by disjoint parameter clusters.

The remaining parts of this dissertation are organized as follows:

- **Chapter 2: Stochastic Behavioral Modeling and Analysis** [KWG13]
  We first review the point estimation method (PEM) [GYH11] and discuss its limitation. In addition, a more stable maximum entropy based approach (MaxEnt) [KWG13] is proposed to accurately model the circuit performance distribution.

- **Chapter 3: High Dimensional Importance Sampling for Fast Yield Analysis** [WGC14]
  An improved high dimensional importance sampling (HDIS) [WGC14] is proposed in this chapter. Unlike conventional importance sample, which

might get unbounded estimation, the HDIS estimates the actual failure probability (yield rate) with a proven bounded result.

- **Chapter 4: Piecewise Distribution Model** [KWB16]

  Piecewise Distribution Model (PDM) tackles the yield analysis from a different avenue, it is based on the performance distribution modeling approach proposed in Chapter 2. Instead of modeling the performance distribution in one piece, it considers the distribution as multiple segments, and approximates each segment using MaxEnt. Hence the PDM can capture more details on the distribution tail, which represents the rare failure events.

- **Chapter 5: Rare-event Microscope: Aiding Yield Analysis with Classification Algorithms** [WXK14]

- **Chapter 6: Hyperspherical Clustering and Sampling: Aiding Yield Analysis with Hyperspherical Clustering** [WBH16]

  Chapter 5 and Chapter 6 propose two approaches that perform yield analysis with the aid of machine learning techniques. Moreover, these approaches consider the condition that failure samples might be distributed in multiple failure regions. In particular, Rare-event Microscope (REscope) presented in Chapter 5 use a nonlinear classifier to separate unlikely-to-fail samples and likely-to-fail ones. Experiment results indicate that the nonlinear classifier is able to figure out the boundary between unlikely-to-fail samples and multiple regions of likely-to-fail samples. While REscope relies on the nonlinear classifier, Hyperspherical Clustering and Sampling (HSCS) approach explicitly uses spherical clustering algorithm to locate the clusters of failure samples and considers those clusters as failure regions. Moreover, it also modifies mixture importance sampling to apply the mean-shift technique to multiple failure regions.

- **Chapter 7: Summary**

A summary of the highlight contributions is presented in the end of this dissertation.

- **Appendix**

  I also append two chapters about my recent work related to deep learning to this dissertation as an extension to my machine learning related research. The work in Appendix A discusses the approaches that compress a well-trained, but cumbersome network to a smaller one without sacrificing accuracy. The other work in Appendix B takes one step further, and proposes an FPGA based deep convolutional neural network, which is much faster than the conventional CPU based implementation, and it much more power efficient.

CHAPTER 2

Stochastic Behavior Modeling and Analysis

2.1 Overview of Stochastic Behavior Modeling

2.1.1 General Stochastic Modeling Method using SPICE Simulation

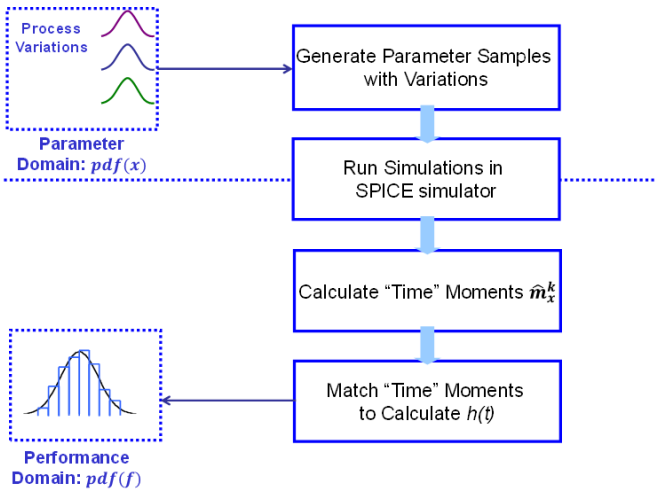


Figure 2.1: General Stochastic Modeling Method utilizing SPICE Simulation

The stochastic behavior modeling is desired because the Monte Carlo simulation is usually too time consuming to yield an accurate distribution of the circuit performance. The flow of a moment matching based stochastic behavior modeling is illustrated in Figure 2.1. First, in the parameter domain, it takes in the process variation parameters (such as channel length, channel width, oxide thickness), which are modeled as random variables. In the performance domain, the distribution of circuit performance is estimated as the output of the system. Clearly, the

transistor level SPICE simulation is the link/mapping between these two domains. The motivation of this work is to explore an efficient approach so as to accurately estimate the performance distribution with as few number of samples as possible, which means less simulation cost.

A well-formulated solution of this problem is moment matching. Several moment matching based approaches have been proposed in the past decade, such as APEX [LLG04], PEM [GYH11], and the most recent MaxEnt [KWG13].

2.1.2 Preliminary knowledge

In this section, we briefly introduce two different types of moments defined in the statistics community [CB01] and signal processing field [OWH96] as the preliminary knowledge for the discussions on moment matching based approaches.

2.1.2.1 Probabilistic Moments

In statistics, the k-th moment of variable x is defined as:

$$m_x^k = E(x^k) = \int_{-\infty}^{+\infty} x^k \cdot pdf(x)dx \tag{2.1}$$

where $E(\cdot)$ is the expectation operator and $pdf(x)$ is the probability density function (PDF) of variable x. In particular, the first four probabilistic moments are the mean, variance, skewness and kurtosis, respectively.

2.1.2.2 Time Moments

The "time moment" has been introduced in the signal processing field [OWH96] for a long time and has been successfully applied to circuit analysis in the past a few years [PR90, Elm48, VS83]. In fact, the time moment is the coefficient of a Taylor expansion of the homogeneous response in the Laplace domain and the

k-th time moment can be expressed as:

$$m_t^k = \frac{(-1)^k}{k!} \int_{-\infty}^{+\infty} t^k \cdot h(t)dt \tag{2.2}$$

where t is the time and $h(t)$ is the impulse response of a linear time invariant (LTI) system H. Moreover, the time moment can be further expanded using the residues and poles of this LTI system as [OWH96, PR90]:

$$m_t^k = -\sum_{r=1}^{M} \frac{a_r}{b_r^{k+1}} \tag{2.3}$$

where a_r and b_r are residues and poles of this LTI system, respectively. As such, the transfer function $H(s)$ of this LTI system can be represented as an M-order rational function (pole-residue format) as:

$$H(s) = \sum_{r=1}^{M} \frac{a_r}{s - b_r} \tag{2.4}$$

In addition, its impulse response $h(t)$ in the time domain can also be expressed with a_r and b_r as:

$$h(t) = \begin{cases} \sum_{r=1}^{M} a_r e^{b_r \cdot t} & (t \geq 0) \\ 0 & (t < 0) \end{cases} \tag{2.5}$$

The time moments have found extensive applications in circuit analysis in the past few years. For example, the work in [AOC99] makes use of the first two time moments to estimate the delay of simple RC networks; AWE [PR90] needs higher order of time moments to approximate the probabilistic distribution of circuit performance.

2.2 Existing Approach: Point Estimation Method

The point estimation method calculate the moment by matching the probabilistic moment to time Moments, and use the padé approximation to calculate the parameters in equation (2.5) to form closed-from probability density function (PDF).

2.2.1 Moment Matching via Padé Approximation

An interesting observation can be found by comparing the probabilistic moments in (2.1) with the time moments in (2.2): m_x^k is different from m_t^k due to a scaling factor $(-1)^k/k!$.

More importantly, if we treat the variable x in (2.1) as the time t in (2.2), the impulse response $h(t)$ can closely approximate $pdf(x)$. As such, we can multiply the probabilistic moments with a scaling factor:

$$\hat{m}_x^k = \frac{(-1)^k}{k!} \cdot m_x^k = \frac{(-1)^k}{k!} \cdot \int_{-\infty}^{+\infty} x^k \cdot pdf(x)dx \tag{2.6}$$

With this slight modification, we are able to extract the impulse response $h(t)$ of a LTI system with time moments $\hat{m}_x^k$ to approximate the $pdf(x)$ by the moment matching method.

With the first $2 * M$ time moments $\hat{m}_x^k$ in (2.6), the residues a_r and poles b_r can be formed by expanding all time moments $\hat{m}_x^k$ with equation (2.3) as:

$$\begin{cases} a_1 + a_2 + \cdots a_M = -\hat{m}_x^{-1} \\ \frac{a_1}{b_1} + \frac{a_2}{b_2} + \cdots \frac{a_M}{b_M} = -\hat{m}_x^0 \\ \frac{a_1}{b_1^2} + \frac{a_2}{b_2^2} + \cdots \frac{a_M}{b_M^2} = -\hat{m}_x^1 \\ \vdots \\ \frac{a_1}{b_1^{2M-1}} + \frac{a_2}{b_2^{2M-1}} + \cdots \frac{a_M}{b_M^{2M-1}} = -\hat{m}_x^{2M-2} \end{cases} \tag{2.7}$$

where the poles, b_i $(i = 1, 2, ..., M)$, and residues, a_i $(i = 1, 2, ..., M)$, are the $2 * M$ unknowns in above nonlinear system.

This nonlinear system can be solved as a linear system, which have been thoroughly discussed in [PR90]. Here, we briefly review the analytic solution of (2.7).

First, all the poles b_i $(i = 1, 2, ..., M)$ can be solved as the eigenvalues of the matrix

$$M = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -c_0 & -c_1 & -c_2 & \cdots & -c_{M-1} \end{bmatrix} \tag{2.8}$$

where c_i $(i = 0, 1, ..., M-1)$ are solved from the linear equations as:

$$\begin{bmatrix} \hat{m}_x^{-1} & \hat{m}_x^{0} & \cdots & \hat{m}_x^{M-2} \\ \hat{m}_x^{0} & \hat{m}_x^{1} & \cdots & \hat{m}_x^{M-1} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{m}_x^{M-2} & \hat{m}_x^{M-1} & \cdots & \hat{m}_x^{2M-3} \end{bmatrix} \begin{bmatrix} -c_0 \\ -c_1 \\ \vdots \\ -c_{M-1} \end{bmatrix} = \begin{bmatrix} \hat{m}_x^{M-1} \\ \hat{m}_x^{M} \\ \vdots \\ \hat{m}_x^{2M-2} \end{bmatrix} \tag{2.9}$$

When the poles b_i are available, the residues a_i can be solved from (2.7) with simple arithmetic operations. Therefore, the impulse response $h(t)$ can be calculated, with the obtained a_i and b_i in equation (2.5), which can be used as an approximation of $pdf(x)$, which is:

$$pdf(x) = \begin{cases} \sum_{r=1}^{M} a_r e^{b_r \cdot t} & (x \geq 0) \\ 0 & (x < 0) \end{cases} \tag{2.10}$$

2.2.2 Numerical Stability Issue and Scaling

As discussed in Chapter 2.2.1, the parameter of the Once the moments are available, it is possible to extract the desired probabilistic distribution using the moment matching method as described in Section 2.2. Theoretically, more moments provide higher accuracy. However, as investigated in [FF95], the conventional moment matching method with high order moments suffers from severe numerical instability issue in solving the linear system (2.9). In particular, the moments typically increase or decrease exponentially towards infinity or zero, which results in significant ill-condition of the moment matrix in (2.9).

For example, the 0-th moment m_f^0 (i.e., the entry in the left-top corner of moment matrix in (2.9)) equals 1 while the 15-th moment m_f^{15} (i.e., the entry in the right-bottom corner of moment matrix in (2.9)) could be as small as 5.3e-147. As such, the moment matrix is severely ill-conditioned and the solution of the related linear system in (2.9) is unreliable and inaccurate.

To improve the stability, several approach has been proposed, such as scale

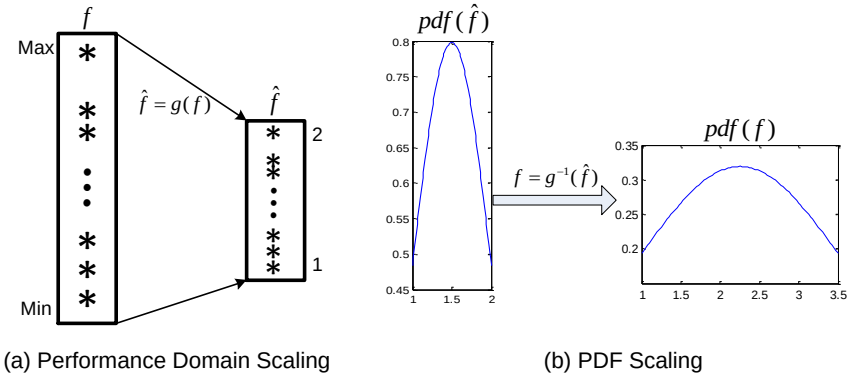


Figure 2.2: Performance and PDF scaling for stability

the performance value. In detail, a linear scaling in Figure 2.2(a) scales the performance f into a small interval $\hat{f} \in [1, 2]$ using a scaling function $\hat{f} = g(f)$. Afterwards, the high order moments can be calculated using $\hat{f}$ and the performance distribution (e.g., $pdf(\hat{f})$ and $cdf(\hat{f})$) can be recovered with the moment matching method as described in Chapter 2.2.1. In addition, the probabilistic distributions should be converted back into the original range of the performance f which is shown in Figure 2.2(b). This approach reduces the condition number. However, when the order of moments grows to 20, the conditional number will still be ˜10^{18} level [FF95], which indicate an unstable system. To solve this problem, an new algorithm without involving equation (2.9) in padé is desired.

2.3 Maximum Entropy

2.3.1 Finding a Distribution through Maximizing Entropy

Entropy is a measure of uncertainty. When choosing a distribution for a random variable (i.e. the circuit performance), one should choose a distribution that maximizes the entropy [Jay57]. By doing this, we can ensure that the distribution is uniquely determined to be maximally unbiased with regard to missing information, while still agreeing with what is known [Jay57].

$$W = \int -p(x) \log p(x) dx \tag{2.11}$$

Where W is the entropy and $p(x)$ is the distribution of random variable x. Maximizing this entropy is subject to moment constraints. In this approach, we consider the probabilistic moments with moment order k

$$\int x^i p(x) dx = \mu_i, \qquad i = 0, 1, ..., k. \tag{2.12}$$

The analytical solution to this convex optimization problem involves the use of Lagrangian Multipliers as stated in [KLI93] and takes the form

$$p(x) = \exp\left(-\sum_{i=0}^{k} \lambda_i x^i \right) \tag{2.13}$$

However, the solution to the above problem does not exist for values of $k >= 2$ [Wu03]. To remedy this issue, [GJM96] suggests transforming this constrained problem into an unconstrained problem by utilizing duality. Using this method results in the dual objective function below

$$\Gamma = \ln Z + \sum_{i=1}^{k} \lambda_i \mu_i \tag{2.14}$$

$$Z = \exp(\lambda_0) = \int \exp\left(-\sum_{i=1}^{k} \lambda_i \mu_i \right) dx \tag{2.15}$$

This problem can now be solved for any value of k. Most MAXENT approaches solve this problem using an iterative method such as Newton's method [Wu03, CHZ10]. Here, Newton's method is used to solve for the Lagrangian multipliers $\lambda = [\lambda_1, \lambda_2, ..., \lambda_k]'$ for moments $i, j = 1, 2, ..., k$ at iteration m

$$\lambda_{(m)} = \lambda_{(m)} - H^{-1} \frac{\delta \Gamma}{\delta \lambda} \tag{2.16}$$

Where the gradient (2.17) and Hessian (2.18) are defined as

$$\frac{\delta\Gamma}{\delta\mu_i} = \mu_i - \frac{\int x^i \exp\left(-\sum_{i=1}^{k} \lambda_i\mu_i\right) dx}{\int \exp\left(-\sum_{i=1}^{k} \lambda_i\mu_i\right) dx} = \mu_i - \mu_i(\lambda) \tag{2.17}$$

$$H_{ij} = \frac{\delta^2\Gamma}{\delta\mu_i\delta\mu_j} = \mu_{i+j}(\lambda) - \mu_i(\lambda)\mu_j(\lambda) \tag{2.18}$$

$$\mu_{i+j}(\lambda) = \frac{\int x^{i+j} \exp\left(-\sum_{i=1}^{k} \lambda_i\mu_i\right) dx}{\int \exp\left(-\sum_{i=1}^{k} \lambda_i\mu_i\right) dx} \tag{2.19}$$

Since the Hessian is positive definite, there exists a unique solution to the above problem [MP84]. Moreover, [MP84] also states that for a non-negative distribution $P(x)$ integrable in [0,1] with moments $\mu_0, \mu_1, ..., \mu_k$, then if $P_N(x)$ is the MAXENT density, we have the following result:

$$\lim_{N\to\infty} \int_0^1 F(x)P_N(x)dx = \int_0^1 F(x)P(x)dx \tag{2.20}$$

This is known as the Maximum Entropy Principle (MEP) [MP84]. As [CHZ10] explains, MEP indicates that the MAXENT density can be used to approximate the distribution arbitrarily well if the sample size is large enough to allow calculation of enough moments.

2.3.2 Algorithm Stability

To show that the distribution in (2.13) is stable for all cases we are concerned with, it is sufficient to show that it is non-negative and absolutely continuous in the interval [0,1] of the random variable x. Although our random variable x may fall outside of this range, it is easy to normalize it such that it falls in this interval. Moreover, we are only concerned with the interval [0,1], and not any subintervals

inside of it our outside of it. Showing that (2.13) is non-negative and absolutely continuous is rigorously explained in [MP84] and an overview of the fundamentals is as follows.

The dual problem shown in (2.14) is everywhere convex and has an absolute minimum [MP84]. First, the function is everywhere convex because the Hessian, the second derivative of (2.14), is positive definite. Second, the dual problem has an absolute minimum if the moments in (2.12) are monotonically increasing which holds true in the case of probabilistic moments. The proofs for both of these conditions can be found in [MP84].

It suffices to say that according to [MP84], if the dual problem in (2.14) is everywhere convex and has an absolute minimum, then the distribution that minimizes it, in this case (2.13) is non-negative and absolutely continuous in the interval. Since the distribution $p(x)$ is non-negative and absolutely continuous, it will not have a negative probability and it will not vanish over the interval. Therefore, $p(x)$ can be considered stable.

2.3.3 Experiment Results

We have implemented the proposed algorithm on MATLAB. The use two circuits to evaluate the performance of MAXENT, a 6-T SRAM bit-cell with 54 variables and a delay chain with 108 variables. HSPICE is used to evaluate these 2 circuits. PEM [GYH11] and MC [JL89] were also implemented using the code obtained from original authors.

2.3.3.1 Experimental Setup

First, here is a brief review of the algorithms evaluated in the experiments:

- **MC (Monte Carlo) [JL89]**: Use quasi-random sampling to gather a huge data set of "MC samples". Calculate the performance distribution from

these MC samples. Use a Figure of Merit to decide when we have enough MC samples for our ground truth. Figure of Merit: If the standard deviation of error between distribution n and distribution $n-1$ is *less* than 0.01, we determine the n^{th} distribution to be the ground truth.

- **PEM (Point Estimation Method) [GYH11]**: Use quasi-random sampling to gather a small data set in order to calculate the probabilistic moments of the random variables. Convert these probabilistic moments to time moments of the corresponding LTI system. Use AWE to perform moment-matching in order to calculate the performance distribution.

- **MAXENT (Maximum Entropy)**: Use quasi-random sampling to gather a small data set in order to calculate the probabilistic moments of the random variables. Use the Maximum Entropy [Jay57] formulation to perform moment-matching in order to calculate the performance distribution.

Process variations and circuits: The statistical data for the process variations are shown in Table 2.1. There are a total of 9 process variations in each transistor, meaning that there are 54 variables for the 6 transistor SRAM circuit, and 108 variables for the 12 transistor delay chain. Similar to other methods [GYH11, LLG04], we model the process variations as Gaussian distributions with various mean and sigma values as listed in Table 2.1.

6T SRAM bit-cell: Figure 2.3 depicts the 6T SRAM bit cell circuit overview. The reading operation of this cell is viewed as the circuit performance. The reading operation of the cell is determined by the voltage ΔV between BL and $\overline{BL}$. If this voltage is large enough to be sensed, it is deemed to be a successful read. The discharge behavior at $\overline{BL}$ plays a crucial role in the value of ΔV. Due to process variations in all transistors, the discharge behavior of $\overline{BL}$ may not be predicted and therefore the voltage ΔV may not be large enough.

Table 2.1: Variation Sources Modeled in CMOS Transistors

Variable Name	σ/μ	unit
Flat-band Voltage (V_{fb})	0.1	V
Gate Oxide Thickness (t_{ox})	0.05	m
Mobility (μ_0)	0.1	m^2/Vs
Doping concentration at depletion (N_{dep})	0.1	cm^{-3}
Channel-length offset (ΔL)	0.05	m
Channel-width offset (ΔW)	0.05	m
Source/drain sheet resistance (R_{sh})	0.1	Ohm/mm^2
Source-gate overlap unit capacitance (C_{gso})	0.1	F/m
Drain-gate overlap unit capacitance (C_{gdo})	0.1	F/m

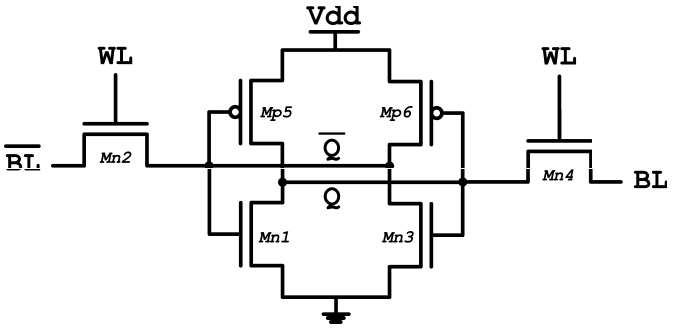


Figure 2.3: Schematics of 6T SRAM Cell Circuit

Delay Chain: Figure 2.4 depicts the delay chain circuit overview. The delay of this inverter chain is viewed as the circuit performance.

2.3.3.2 Stability

Figure 2.5 shows the performance distributions generated by MAXENT, PEM, and MC for the first 16 moments and first 18 moments using the 6T SRAM cell circuit using 200 samples. As we can see, MAXENT is stable under both conditions. The curves representing MAXENT for the first 16 moments and first 18 moments show very good overlap with the ground truth (MC) distribution. On the other hand, only the PEM curve corresponding to 16 moments is stable and overlaps with the ground truth distribution. The PEM curve corresponding to 18 moments is unstable, with a value of 0 through most of the distribution

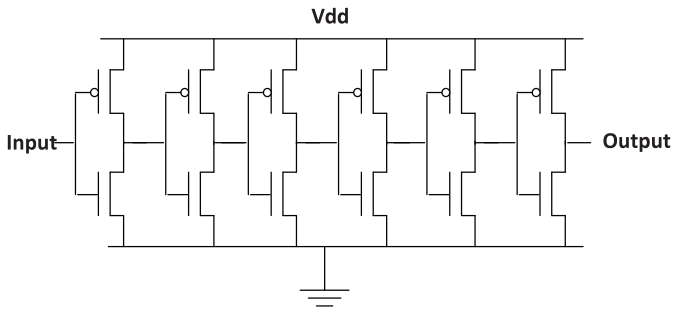


Figure 2.4: Schematics of Delay Chain Circuit

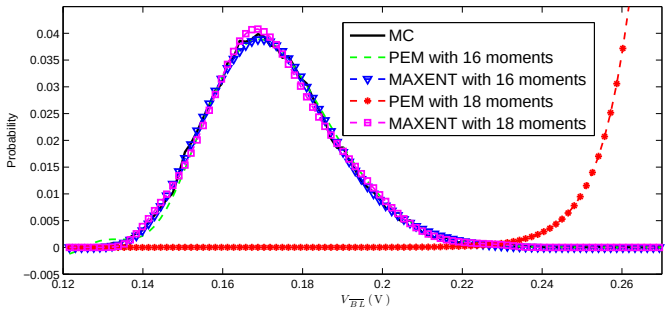


Figure 2.5: PEM lack of stability on SRAM circuit (200 samples)

until it blows up to infinity. The only value that changed between these curves are the order of moments that were used. The sample number, circuit topology, process variations, and all other inputs were held constant. These results imply that PEM is very sensitive to the moments that are used.

Figure 2.6 shows the performance distributions generated by MAXENT, PEM, and MC for the first 16 moments and first 18 moments using the 6T SRAM circuit using 250 samples. As we can see, MAXENT is stable under both 16 moments and 18 moments and overlap well with the ground truth distribution. Moreover, we see that PEM is now stable under both 16 moments and 18 moments and also overlap well with the ground truth distribution. previously, PEM was unstable for the SRAM circuit using 200 samples and 18 moments, whereas now it is stable for the SRAM circuit using 250 samples and 18 moments.

Figure 2.7 shows the performance distributions generated by MAXENT, PEM,

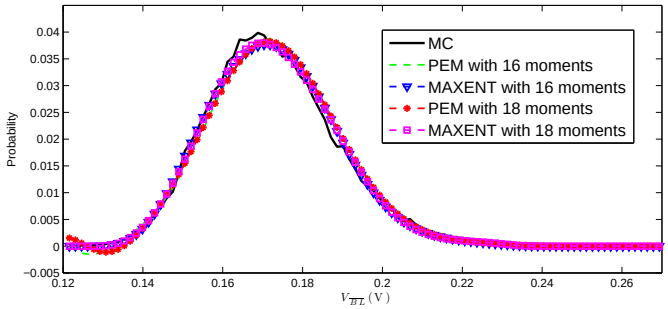


Figure 2.6: PEM stability on SRAM circuit (250 samples)

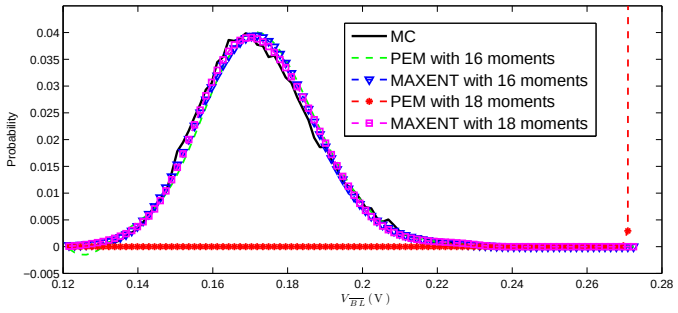


Figure 2.7: PEM lack of stability on SRAM circuit (300 samples)

and MC for the first 16 moments and first 18 moments using the 6T SRAM circuit using 300 samples. In this case, we have returned to the instability of PEM. We see that MAXENT is still stable as always, but PEM is now unstable with 18 moments.

The above results are due to two reasons: the inaccuracies of the Padé approximation and the generation of moments from samples. PEM uses the Padé approximation to approximate the performance distribution as a transfer function of an LTI system. In short, this transfer function is a ratio of polynomial functions as shown in (2.5). The poles of this transfer function are estimated using the eigenvalues of a system matrix that is solved by AWE. When solving for the eigenvalues, AWE leverages the Padé approximation. However, the Padé approximation does not give all of the true eigenvalues of the system. Instead, it will

generate poles (eigenvalues) that correspond to the dominant poles of the original system, and a few poles that do not correspond to the poles in the original system but account for the effects of the remaining poles [FF95]. Consequently, the Padé approximation may generate some positive eigenvalues. In the experiments above, the Padé approximation always generated 1 positive pole for the unstable cases, and 0 positive poles for the stable cases. Since the eigenvalues correspond to the poles of the transfer function, they will take the form of a sum of weighted exponential functions. It is clear that since the Padé approximation may generate positive poles, they will correspond to unbounded exponential terms that continue to grow and lead to instability.

The number of samples also plays a key role in the stability of PEM. Since PEM uses AWE and the Padé approximation to solve the set of nonlinear functions in (2.9), changing the values on the RHS of (2.9) will change the values of b_r which are the poles of the transfer function. Using 200 samples will generate a set of moments M_1 while using 250 samples will generate a set of moments M_2. These moments will have completely different values and will lead to a new set of solutions to (2.9). This new set of solutions (the poles of the transfer function) may be stable or unstable.

The key drawback of PEM is that it is unpredictable. The experimental results reinforce the idea that the instabilities in PEM are unpredictable and can occur with any number of samples that we use depending on the calculation performed by the Pade approximation. On the other hand, MAXENT is predictable. MAXENT does not use an LTI system model and does not use the Pade approximation, so it will not be subjected to this type of instability. More specifically, as was mentioned in the previous section, the distribution generated by MAXENT will always be non-negative and will always be absolutely continuous on the interval [0,1]. Clearly, the MAXENT distribution is more robust than the PEM distribution.

2.3.3.3 Accuracy

We also evaluated the accuracy of the MAXENT algorithm compared to PEM. Throughout our experiments, MAXENT consistently offers lower error relative to the ground truth than PEM does for any order of moments. We determine the error using the following equation:

$$error = \int (f_1(x) - f_2(x))\, dx \tag{2.21}$$

where $f_1(x)$ is our distribution from MAXENT or PEM and $f_2(x)$ is the ground truth distribution from MC. Table 2.2 displays the relative error for both MAXENT and PEM in the SRAM and Inverter Chain circuits. We note that moment orders 2 and 4 were excluded due to the high error in both algorithms.

Table 2.2: Accuracy Comparison with 200 sample for SRAM and 400 samples for delay chain

Circuit	# Samples	Moment Order	PEM Error(%)	MAXENT Error(%)
SRAM	200	6	46.349	11.85
		8	30.656	3.988
		10	15.577	3.281
		12	9.4457	3.394
		14	6.6038	3.181
		18	198.97	5.470
Inv. Chain	400	6	54.761	10.173
		8	38.021	7.830
		10	22.382	7.907
		12	15.052	7.679
		14	13.631	7.482
		18	199.62	7.383

Table 2.3 displays the relative error for both MAXENT and PEM in the SRAM and Inverter Chain circuits. The most noticeable trend is that MAXENT offers a lower relative error than PEM across all orders of moments. Both MAXENT and PEM utilized the same sampled values, and thus used the same moment values.

The only difference is the way they performed their moment matching. In fact, we can see that MAXENT seems to perform its moment matching very effectively and efficiently. MAXENT seems to converge to a steady-state value of error by the time it hits 8 moments, whereas PEM continues to decrease in error (still always having a higher value than MAXENT) until it becomes unstable. Moreover, we see in Table 2.3 that changing the number of samples does not affect the result of MAXENT having a smaller relative error. In fact, in one case PEM fails altogether and cannot create a distribution.

Table 2.3: Accuracy Comparison with 300 sample for SRAM and 600 samples for delay chain

Circuit	# Samples	Moment Order	PEM Error(%)	MAXENT Error(%)
SRAM	300	6	46.117	11.043
		8	30.251	5.331
		10	15.097	6.046
		12	11.341	5.818
		14	10.74	6.516
		18	200	6.222
Inv. Chain	600	6	55.037	9.653
		8	38.728	4.295
		10	22.46	4.306
		12	13.859	4.801
		14	10.695	5.846
		18	NaN	5.315

2.3.3.4 Speedup

To observe the efficiency of MAXENT, we compare the speedup with respect to Monte Carlo while regarding the loss of accuracy. Table 2.4 shows the speedup in comparison to Monte Carlo with the corresponding loss in accuracy. As explained above, we use a Figure of Merit to decide when we have enough MC samples for our ground truth. Figure of Merit: If the standard deviation of error between distribution n and distribution $n-1$ is *less* than 0.01, we determine the n^{th}

distribution to be the ground truth. This corresponded to 39000 samples with a FoM of 0.0094817 for the 6T-SRAM circuit, and 54000 samples with a FoM of 0.0094902 for the delay chain. For the 6T-SRAM circuit, we have a speedup of 195x while still maintaining a very small error of about 3%. For the delay chain, we have a speedup of 135x while still maintaining an error of about 7.3%.

Table 2.4: Speedup Comparison of MAXENT, PEM, and MC

Circuit	Method	Samples	Speedup	Error %
SRAM	Monte Carlo	(39 x 10^3	1x	0%
	MAXENT	200	195x	3.09%
Delay Chain	Monte Carlo	(54 x 10^3	1x	0%
	MAXENT	400	135	7.28%

2.4 Conclusion

In this Chapter, we discussed the approaches for stochastic behavior modeling, the PEM and MAXENT algorithm.

The PEM algorithm take advantage of Padé approximation to estimate the PDF of the circuit performance as the impulse result of a LTI system. However, due to the instability of the moment matrix involved in Padé approximation, it suffers from the numerical stability problem.

MAXENT approximates the exact behavioral distribution with a product of exponential distributions and finds the closest approximation by choosing the Lagrange Multipliers for these exponential distributions. To do so, the Shannons information entropy between them has been maximized so as to reduce the distance between these two distributions. With the extensive experiments, the proposed approach has shown significant improvement in stability and accuracy (up to 35% lower error) when compared to AWE based method [JL89] and up to 195x speedup when compared with MC method [GYH11].

CHAPTER 3

High Dimensional Importance Sampling for Fast Yield Analysis

3.1 Introduction of High Sigma Yield Analysis

Behavior model presented in Section 2 is sufficient to describe the overall behavior of a circuit with process variations. However, for critical circuits, such as PLL, which stabilizes clock for the entire chip, and standard circuit cells, which are duplicated for millions of times, an extremely rare failure event (10^{-4}-10^{-7}) could be transferred to large failure probability of the entire system, leading to catastrophe of the chip design.

The analysis of such small failure is also known as high sigma analysis, because sigma is the conventional notation of probability of Gaussian distribution. For single-tailed case, $n\sigma$ probability is calculated as

$$\text{Prob}_{n\sigma} = 1 - \text{CDF}_{(0,1)}(n) \tag{3.1}$$

where $\text{CDF}_{(0,1)}()$ is cumulative distribution function (CDF) of Gaussian distribution with zero mean and unit standard deviation.

As illustrated in Figure 3.1 (a), 2σ probability is about 97.7% for single-tailed case, corresponding to 2.23% in the tail. In typical high sigma analysis for memory circuit, the failure probability can scale up to 4-6 sigma, corresponding to failure probability from 3.17e-5 to 9.87e-10, which are indeed rare failure events.

Traditional circuit simulation performs worse case analysis (WCA) to deter-

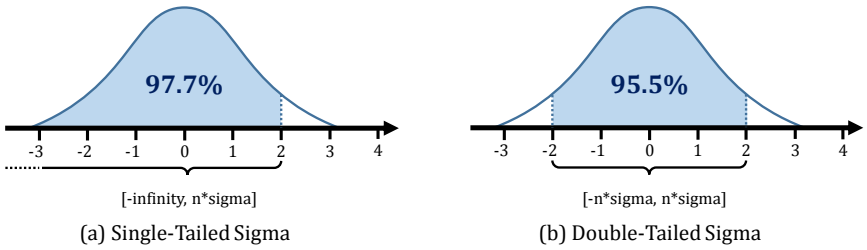


Figure 3.1: Basic idea in proposed algorithm. (Noted that $\mathcal{T} = \{Y|Y \geq t\}$ contains $\mathcal{S} = \{Y|Y \geq t_c\}$).

mine a safety margin during the design. However, only checking a few corners is insufficient to analyze extreme small failure probability [SR09]. Modern statistical circuit simulation approaches consider process variations and statistically simulate the circuit to estimate the probability that a circuit does not meet the performance metric. Among those approaches, Monte Carlo (MC) analysis remains the gold standard [JL89]. It repeatedly draws samples and evaluates circuit performance via transistor-level simulation. Even though the circuit simulation has been considerably accelerated [CWW11, WSC11, WGK13], it is, however, extremely time-consuming because millions of samples need to be simulated to capture one single failure when the failure is a rare event.

Instead of sampling randomly with standard MC, more efficient approaches only sample the statistically likely-to-fail case [LLG07, GYH11, KWG13, SR08, KJN06, DQS08, QTD10, KHT10, SR08, SR09, SWC08, DL11]. The aforementioned approaches can be categorized in the following groups:

1. **Moment Matching** [LLG07, GYH11, KWG13]: The approaches in this category only evaluate a small number of samples with SPICE simulation, and approximate the PDF of the circuit performance to an analytical expression by means of moment matching. However, some approaches in this category are known as numerically instable because the moment matrix solved during the moment matching process is usually ill-conditioned

[FF95, CN94]. Moreover, existing moment matching based approaches only match the overall shape of the PDF without surgically look into its tail, which contains information particularly for rare events. Therefore, these algorithms are usually applied to stochastic behavior modeling rather than rare event (high sigma) analysis.

2. **Importance Sampling**: To specifically look into the samples that cause a rare event, importance sampling based approaches [KJN06, DQS08, QTD10, KHT10] had been developed to construct a new "proposed" sampling distribution under which a "rare event" becomes "less rare" so that more failures can be easily captured. The critical issue is how to build an optimal proposed sampling distribution. Previous work investigated different approaches. For example, [KJN06] mixes a uniform distribution, the original sampling distribution and a "shifted" distribution centering around the failure region. The approaches in [DQS08, QTD10] shift the sampling distribution towards the point of failure region with a minimum L_2-norm. The work in [KHT10] uses "particle filtering" to tilt more samples towards the failure region. These importance sampling based methods are plagued by the curse of high dimensionality [AB03, BB05, RG09]. In general, they can only be used in low-dimensional problems (e.g., those with a scope of 6-12 variables) but become very untrustworthy for high-dimensional problems.

3. **Classification**: the approach in statistical blockcade (SB) [SR08, SR09] makes use of a classifier to block those Monte Carlo samples that are unlikely to cause failures and simulates the remaining samples. However, the SB approach uses the linear support vector machine (SVM), which can be easily fooled by high dimensionality [WGC14]. An improvement of SB [WXK14] along with more details about classification based approaches will be discussed in detail in the Chapter 5.

Clearly, most of the existing approaches can be successfully applied to low-dimensional problems with small number of variables but, in general, perform poorly in high dimension.

Among others, the Scaled Sigma Sampling (SSS) [SLL13] and subset simulation (SUS) [SL14] approach the rare failure probability via different avenues. SSS draws samples by scaling up the standard deviation (sigma) of the original distribution, while using the same mean. Failure probabilities are calculated at different scaling factors to extrapolate the failure probability under the original distribution, i.e. scaling factor equal to 1 [SLL13]. However, SSS is susceptible to accuracy loss due to the extrapolation requirement. Alternatively, SUS approaches the rare failure probability as the production of several, large conditional probabilities estimated in multiple phases [SL14]. Samples in each phase are generated with the aid of the Markov Chain Monte Carlo (MCMC) method.

In this Chapter, we propose a novel statistical algorithm to efficiently estimate the failure probability of memory circuits in high dimension, where tens or hundreds of random variables are present. In details, the proposed methodology first constructs a new subset of the sampling space that dominates the failure region for memory circuits and can be efficiently estimated with a few samples. Then, the failure probability of memory circuits can be evaluated by the product rule of conditional probability within this sampling subset space. More importantly, the estimation from the proposed method is proved to be always bounded in high dimensions. Experiments on a 54-dimensional SRAM cell circuit show that the proposed approach achieves 1150X speedup over Monte Carlo without compromising any accuracy. It is also 204X faster than the classification based method (e.g., Statistical Blockade [SR09]) by and 5X faster than existing importance sampling method (e.g., Spherical Sampling [DQS08, QTD10]). On another 117-dimension circuit, the classification based method fails to improve the performance by blocking "unlikely to fail" samples, and Spherical Sampling [DQS08, QTD10] method

completely fails to provide reasonable accuracy. Contrastingly, the proposed approach yields accurate result with 364X speedup over Monte Carlo.

3.2 Preliminary Knowledge of Importance Sampling

3.2.1 Formulation of Probability Estimation

Let $f(X)$ be a probability density function (PDF) for a random variable X (e.g., any process or electronic variable parameters) which is the input of a measurement process as shown in (3.2); the output Y is an observation (e.g., voltage, amplitude, period, etc.) with input X:

$$\underbrace{X}_{variable} \Rightarrow \boxed{Measurement,\ SPICE,\ etc.} \Rightarrow \underbrace{Y}_{observation} \tag{3.2}$$

Usually, it is of great interest to estimate the probability of Y from a small subset $\mathcal{S}$ of the entire sampling space. For example, a small subset is the "failure region" for SRAM design and includes all failed samples where performance constraints cannot be satisfied. Therefore, the probability $p(Y \in \mathcal{S})$ can be estimated as:

$$p(Y \in \mathcal{S}) = \int I(X) \cdot f(X) dX. \tag{3.3}$$

$$I(X) = \begin{cases} 0 & if\ Y \notin \mathcal{S} \\ 1 & if\ Y \in \mathcal{S} \end{cases}$$

where Y is the observation/performance with the input variable X and the indicator function $I(\cdot)$ identifies whether $Y \in \mathcal{S}$ or not. Note that the integral in equation (3.4) is intractable because the analytical formula of $I(X)$ is unavailable. Therefore, sampling based method must be used. For example, the MC method enumerates as many samples of X as possible (e.g., $x_1, \cdots, x_n$) according to $f(X)$ and evaluates their indicator function values to estimate $p(Y \in \mathcal{S})$ as:

$$\tilde{p}(Y \in \mathcal{S}) = \frac{1}{n}\sum_{i=1}^{n} I(x_i) \xrightarrow{\substack{a.s. \\ n\to+\infty}} p(Y \in \mathcal{S}). \tag{3.4}$$

Here $\tilde{p}(X \in \mathcal{S})$ is an unbiased estimate from sampling method and can be very close to $p(X \in \mathcal{S})$ with a large number of samples.

3.2.2 Importance Sampling (IS)

When $Y \in \mathcal{S}$ is a *rare event*, the MC method becomes extremely inefficient because most $I(x_i)$ are zeros. Millions or billions of samples of X are needed to capture only one failed sample from the failure region $\mathcal{S}$.

To deal with this issue, the *importance sampling* (IS) has been introduced to sample from a "proposed" sampling distribution $g(X)$ that tilts towards $\mathcal{S}$ where a rare-event becomes more likely to happen:

$$\begin{aligned} p_{IS}(Y \in \mathcal{S}) &= \int I(X) \cdot \frac{f(X)}{g(X)} \cdot g(X) dX \\ &= \int I(X) \cdot w(X) \cdot g(X) dX. \end{aligned} \tag{3.5}$$

Here, $w(X)$ is the "likelihood ratio" or the weight for each sample of X. $w(X)$ compensates for the discrepancy between $f(X)$ and $g(X)$ and unbiases the probability estimation under $g(X)$. Sampling based methods can be used to evaluate above integral as:

$$\tilde{p}_{IS}(Y \in \mathcal{S}) = \frac{1}{n}\sum_{j=1}^{n} w(\tilde{x}_j) \cdot I(\tilde{x}_j) \xrightarrow{\substack{a.s. \\ n\to+\infty}} p(Y \in \mathcal{S}). \tag{3.6}$$

$\tilde{x}_j$ $(j = 1, \cdots, n)$ follows the "proposed" sampling distribution $g(X)$ rather than the original distribution $f(X)$, because more *rare event* samples in the subset $\mathcal{S}$ can be easily chosen under the distribution $g(X)$.

Theoretically, $\tilde{p}_{IS}(Y \in \mathcal{S})$ is consistent with $p(Y \in \mathcal{S})$ in (3.4) if $supp(g(X)) \supset supp(I(X) \cdot f(X))$, where $supp(\cdot)$ denotes the support of a probabilistic distribution.

3.2.3 Failure Analysis of Importance Sampling

While importance sampling is, in principle, mathematically correct, the *degeneration* or *collapse* of the likelihood ratios leads to the failure of importance sampling in high dimensions as discussed in [BB05, RG09].

Let us consider a classical case, as shown in Figure 3.2, where $f(X)$ is the "original" sampling distribution and $g(X)$ is the "proposed" sampling distribution. The small circles with the same size within $g(X)$ are samples drawn from $g(X)$. In the bottom of Figure 3.2, a few circles with different sizes represent the illustrative scales of the likelihood ratios corresponding to the samples on top of them. Clearly, if $g(X)$ has thinner tails than $f(X)$, the likelihood ratios $w(X) = f(X)/g(X)$ approach infinity in the tails of $g(X)$. Hence, the likelihood ratios vary dramatically and have extremely large variance that leads to unstable probability estimate.

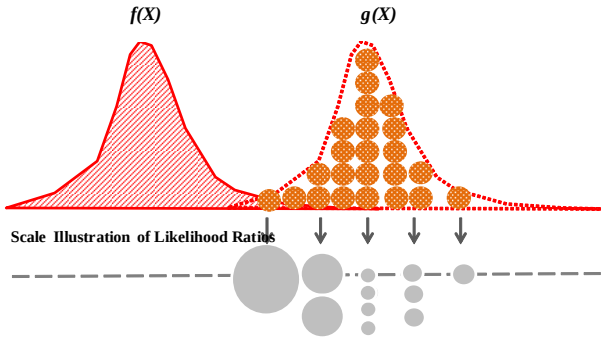


Figure 3.2: The scale illustration of likelihood ratios in importance sampling.

Moreover, the reason for the collapse of likelihood ratio can be explained from another perspective: when importance sampling shifts $g(X)$ towards the rare-event region that is typically in the tails of $f(X)$, $f(X)$ and $g(X)$ become mutually singular and have "disjoint" support [BB05]. Therefore, IS fails to retain its accuracy.

This collapse issue of likelihood ratios becomes much worse in high dimensions

because $w(X)$ is a product of probabilities for multiple parameters and consequently approaches infinity more quickly.

3.3 High-Dimensional Importance Sampling Algorithm

3.3.1 Algorithm Overview

We consider a small subset $\mathcal{S}$ as the failure region in SRAM design under the given performance constraint (e.g., the performance of SRAM circuit Y should be greater than certain performance threshold t_c). Hence the subset $\mathcal{S} = \{Y|Y \geq t_c\}$ in Figure 3.3 contains all failed samples that are "rare events".

The basic idea of the proposed algorithm is to construct a new subset $\mathcal{T}$ with a new threshold t (e.g., t = 0.99-quantile point). This new subset $\mathcal{T} = \{Y|Y \geq t\}$ includes "non-rare" events and dominates the "rare event" subset $\mathcal{S}$ (e.g., $supp(\mathcal{T}) \supset supp(\mathcal{S})$).

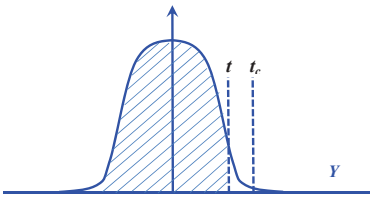


Figure 3.3: Basic idea in proposed algorithm. (Noted that $\mathcal{T} = \{Y|Y \geq t\}$ contains $\mathcal{S} = \{Y|Y \geq t_c\}$).

In this way, the failure probability of SRAM design can be estimated by a product rule from the probability theory [PP01]:

$$P(Y \geq t_c) = \quad P(Y \geq t) \cdot P(Y \geq t_c|Y \geq t). \tag{3.7}$$

The proposed algorithm has two stages and can be illustrated with Figure 3.4:

1) Initial Sampling with MC: This step aims to evaluate the probability $P(Y \in$

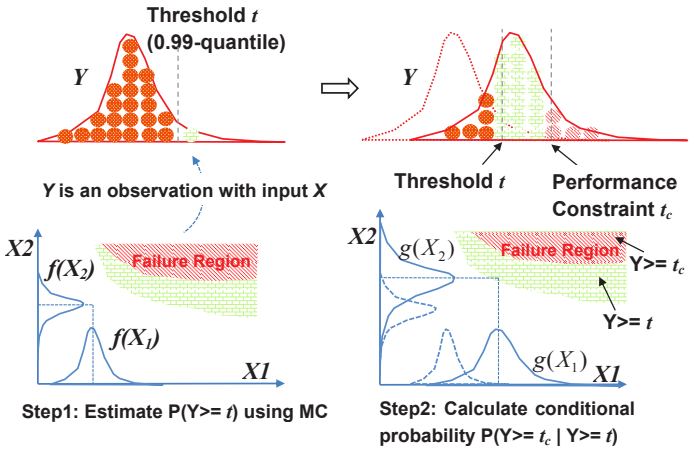


Figure 3.4: Overall flow in proposed algorithm. (Noted that $\mathcal{T} = \{Y|Y \geq t\}$ contains $\mathcal{S} = \{Y|Y \geq t_c\}$).

$\mathcal{T}) = P(Y \geq t)$ where t is the threshold, such as $t = 0.99$-quantile point shown in the left of Figure 3.4. Since the samples in $\mathcal{T}$ are "non-rare" events, this evaluation needs only a few samples using standard MC method.

2) Conditional Probability Estimation: The most challenging task is to efficiently evaluate the conditional probability $P(Y \geq t_c|Y \geq t)$ where sampling method must be used. To expedite the convergence rate of estimation, a "proposed" sampling distribution $g(X)$ that is close to the failure region shall be constructed by *shifting* and *reshaping* the "original" sampling distribution (shown in the right of Figure 3.4). More details will be discussed in the following section.

The overall algorithm flow is described in Algorithm(1). There are several issues that need to be resolved: 1) It is, at the moment, unclear how to *shift* and *reshape* the original sampling distribution $f(X)$ in order to build $g(X)$; 2) With the proposed sampling distribution $g(X)$, how to calculate the conditional probability; 3) It is of great interest to study whether the estimations of proposed algorithm is always bounded or not.

The following sections discuss how we solve these issues.

Algorithm 1 Overall Algorithm

Input: random variables X with sampling distributions $f(X)$ and performance constraints $Y \geq t_c$.

Output: the estimation of failure probability $p_{IS}(Y \geq t_c)$.

1: **/* 1: Initial Sampling with MC */**
2: Use few MC samples to find the threshold value t of performance (e.g., t = 0.99-quantile point).
3: Run standard Monte Carlo method to calculate $P_{MC}(Y \geq t)$ with certain accuracy level.
4: **/* 2: Conditional Probability Calculation */**
5: Shift the original sampling distribution $f(X)$ towards the failure region.
6: Reshape the shifted $f(X)$ by changing its standard deviation to construct $g(X)$.
7: Generate samples from $g(X)$ and evaluate conditional probability $P(Y \geq t_c | Y \geq t)$.
8: **/* 3: Failure Probability Estimation */**
9: Solve for the failure probability $p_{IS}(Y \geq t)$ as

$$P_{IS}(Y \geq t_c) = P_{MC}(Y \geq t) \cdot P(Y \geq t_c | Y \geq t).$$

3.3.2 Shift and Reshape Sampling Distribution

Mean-Shift Vector Selection Mean-shift is a typical way to move the sampling distribution towards the failure region where the failed samples are most likely to happen in previous works [KJN06, DQS08, QTD10, KHT10, GBD12]. The key is to find the mean-shift vector for the original sampling distributions $f(X)$.

To this end, we propose to shift $f(X)$ towards a "non-rare" subset $\mathcal{T} = \{Y | Y \geq t\}$, because our target is to evaluate the conditional probability $P(Y \geq t_c | Y \geq t)$ around the subset $\mathcal{T}$. More importantly, as $\mathcal{T}$ is usually not far away from the mean of $f(X)$, the shifted distribution shares almost the *same* support with $f(X)$ so as to avoid the "disjoint support" issue.

In addition, we adopt the insights from [GBD12] to find a close-to-optimal mean-shift vector in this work. Let us consider a 1-D problem as an example. The algorithm in [GBD12] starts with an initial parameterized distribution $\hat{f}(X, \hat{\mu})$ and tries to update the mean value iteratively to achieve a close-to-optimal sampling distribution $f^*(X, \mu^*)$ by an analytic formula:

$$\mu^* = \frac{\sum_{i=1}^{N} I(x_i) \cdot w(x_i) \cdot x_i}{\sum_{i=1}^{N} I(x_i) \cdot w(x_i)}. \tag{3.8}$$

Here x_i $(i = 1, \cdots, N)$ are samples drawn from $\hat{f}(X, \hat{\mu})$ and $w(x_i)$ are their likelihood ratios as $w(x_i) = f(x_i)/\hat{f}(x_i, \hat{\mu})$.

Intuitively, the updated mean value μ^* can be viewed as the coordinates of the *centroid point* in the failure region where the failed samples are most likely to happen. This interesting finding becomes more obvious if $\hat{f}(X, \hat{\mu})$ equals $f(X)$ and all likelihood ratios take on value 1. Hence, μ^* is:

$$\mu^* = \frac{\sum_{i=1}^{N} I(x_i) \cdot x_i}{\sum_{i=1}^{N} I(x_i)}. \tag{3.9}$$

Therefore, our mean-shift method tries to shift the sampling distribution towards the "centroid point" of the subset $\mathcal{T} = \{Y|Y \geq t\}$, which can be evaluated with available MC samples from the first step in Algorithm (1) and requires no extra sampling/simulation cost.

Standard Deviation Selection// Next, it is desired to *reshape* the shifted sampling distribution around the centroid of subset $\mathcal{T}$. In particular, the standard deviation for the proposed sampling distribution $g(X)$ must be properly chosen to reach the failure region $\mathcal{S} = \{Y|Y \geq t_c\}$, because the shifted and reshaped sampling distribution should *dominate* or completely cover the "rare-event" region $\mathcal{S}$.

As an illustration, let us consider a 2-D problem in Figure 3.5. The problem now becomes how to choose the standard deviation of the proposed sampling

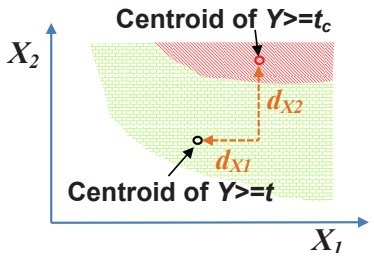


Figure 3.5: The distance between centroid points of two subsets along each parameter axis.

distribution $g(X)$ to obtain the samples in the "rare-event" region $\mathcal{S} = \{Y|Y \geq t_c\}$.

The proposed algorithm first approximates the centroid point of $\mathcal{S} = \{Y|Y \geq t_c\}$ using uniformly-distributed samples and then calculates the distance between these two centroid points along each parameter axis (e.g., d_{X_1} and d_{X_2} shown in Figure 3.5). Then, we choose $max(d_{X_i}, \sigma_{(0,X_i)})$ as the standard deviation of $g(X_i)$ for the variable X_i, where $\sigma_{(0,X_i)}$ is the original standard deviation of $f(X_i)$. This choice can be intuitively explained as follows:

- $d_{X_i} > \sigma_{(0,X_i)}$**:** the failure region $\mathcal{S}$ is very far away from the subset $\mathcal{T}$, therefore, the larger value d_{X_i} is used to extend the range of $g(X_i)$ and obtain the rare-event samples in the failure region. In the meantime, $g(X_i)$ has almost the same supports with $f(X_i)$ because its mean position locates at the centroid point of $\mathcal{T}$ and is not far away from $f(X_i)$.

- $d_{X_i} < \sigma_{(0,X_i)}$**:** Suppose the smaller one, d_{X_i}, is chosen as the standard deviation of $g(X_i)$, the proposed sampling distribution $g(X)$ will have much smaller sampling space, thereby, making it fail to keep the same supports with $f(X_i)$ and suffer from "disjoint supports" issue. The proposed algorithm chooses $\sigma_{(0,X_i)}$ as the standard deviation of $g(X_i)$ in this case.

3.3.3 Conditional Probability Calculation

With the proposed sampling distribution $g(X)$, it is desired to efficiently estimate the conditional probability in Algorithm(1). We can start with the product rule in the probability theory [PP01]:

$$P(Y \geq t_c | Y \geq t) = \frac{P(Y \geqslant t_c, Y \geqslant t)}{P(Y \geqslant t)}. \tag{3.10}$$

In addition, when samples x_i $(i = 1, \cdots, N)$ are generated from $g(X)$, both $P(Y \geqslant t_c)$ and $P(Y \geqslant t)$ can be estimated mathematically with the indictor function and likelihood ratios. Thus, the equation (3.10) becomes:

$$\begin{aligned} P_{MIS}(Y \geq t_c | Y \geq t) &= \frac{P(Y \geqslant t_c)}{P(Y \geqslant t)} \\ &= \frac{\frac{1}{N}\sum_{i=1}^{N} w(x_i) \cdot I_{\{Y \geqslant t_c\}}(x_i)}{\frac{1}{N}\sum_{i=1}^{N} w(x_i) \cdot I_{\{Y \geqslant t\}}(x_i)}. \end{aligned} \tag{3.11}$$

where $I_{\{Y \geqslant t_c\}}(\cdot)$ and $I_{\{Y \geqslant t\}}(\cdot)$ are indicator functions for subsets $Y \geq t_c$ and $Y \geq t$, respectively. $w(x_i)$ are likelihood ratios for these samples. In this way, the conditional probability can be efficiently evaluated under proposed sampling distribution $g(X)$.

3.3.4 Boundedness Analysis

Importance Sampling

Let us first investigate the existing importance sampling and assume samples x_j $(j = 1, \cdots, M)$ are generated from the proposed sampling distribution $g(X)$.

We find the upper bound of probability estimate from the conventional importance sampling according to Boole's inequality (also known as the union bound from probability theory [PP01]) as:

$$
\begin{aligned}
P(Y \geqslant t_c) &= P_f(\sum_{j=1}^{M} I_{\{Y \geqslant t_c\}}(x_j)) \leqslant \sum_{j=1}^{M} P_f(x_j) \cdot I_{\{Y \geqslant t_c\}}(x_j) \\
&= \sum_{j=1}^{M} w(x_j) \cdot I_{\{Y \geqslant t_c\}}(x_j). \qquad (3.12)
\end{aligned}
$$

In 3.12 P_f stands for the probability estimation under sampling distribution $f(X)$. As discussed in [BB05, RG09], the likelihood ratios $w(x_j)$ can vary dramatically in high dimension and be any random quantities. Therefore, the union bound of the estimation $P(Y \geqslant t_c)$ in (3.12) approaches infinity and importance sampling becomes unreliable and untrustworthy.

Proposed Algorithm

The proposed algorithm constructs a subset $\mathcal{T} = \{Y|Y \geq t\}$ that *dominates* the failure region $\mathcal{S} = \{Y|Y \geq t_c\}$ (i.e., $\mathcal{T} \supset \mathcal{S}$). Therefore, the upper bound of conditional probability can be derived as:

$$
\begin{aligned}
P(Y \geqslant t_c|Y \geqslant t) &= \frac{P(Y \geqslant t_c)}{P(Y \geqslant t)} \\
&= \frac{\sum_{j=1}^{N} w(x_j) \cdot I_{\{Y \geqslant t_c\}}(x_j)}{\sum_{j=1}^{N} w(x_j) \cdot I_{\{Y \geqslant t\}}(x_j)} \leqslant 1. \qquad (3.13)
\end{aligned}
$$

Note that no matter how likelihood ratios $w(x_j)$ vary, the same likelihood ratios for samples in the failure region $\mathcal{S} = \{Y|Y \geq t_c\}$ would appear in both numerator and denominator in (3.13) if and only if the calculations of both $P(Y \geqslant t_c)$ and $P(Y \geqslant t)$ utilize the *same* set of samples x_j $(j = 1, \cdots, M)$ drawn from $g(X)$. Clearly, the conditional probability estimation of proposed algorithm is always bounded by the upper bound 1. Thereby, the propose algorithm can reliably provide bounded estimation results.

Table 3.1: Process Parameters of MOSFETs.

Variable Name	σ/μ	unit
Flat-band Voltage (V_{fb})	0.1	V
Gate Oxide Thickness (t_{ox})	0.05	m
Mobility (μ_0)	0.1	m^2/Vs
Doping concentration at depletion (N_{dep})	0.1	cm^{-3}
Channel-length offset (ΔL)	0.05	m
Channel-width offset (ΔW)	0.05	m
Source/drain sheet resistance (R_{sh})	0.1	Ohm/mm^2
Source-gate overlap unit capacitance (C_{gso})	0.1	F/m
Drain-gate overlap unit capacitance (C_{gdo})	0.1	F/m

3.4 Experiment Results

We investigate its performance of the proposed algorithm for failure analysis of memory circuits (e.g., SRAM bit-cell and sense amplifier) in this section. All experiments are performed using MATLAB and Hspice with BSIM4 transistor model. The proposed algorithm is named as HDIS (high-dimensional importance sampling) in this section. In addition, Monte Carlo (MC), statistical blockade (SB)[SR09], and spherical sampling (SS) [DQS08, QTD10] have been implemented for comparison purpose.

3.4.1 SRAM Circuit and Variation Modeling

A functional diagram of SRAM circuit with one bit-cell column is shown in Figure 3.6, which consists of a decoder, bit-cells, a sense amplifier and a delay chain [PS08]. During the reading operation: the bit-cells store the data in forms of '0' or '1'; the decoder generates an address of a specific bit-cell and releases a read enable signal. Therefore, the chosen bit-cell starts to discharge the bit-lines (i.e., the lines that connect to all bit-cells) to produce a voltage difference between two bit-lines. the sense amplifier reads out the stored data by capturing and magnifying the voltage difference on bit-lines.

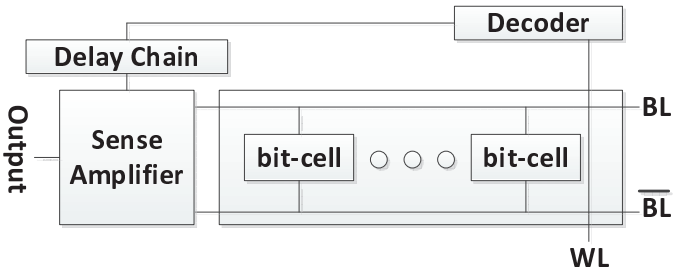


Figure 3.6: Functional diagram of an SRAM circuit.

Table 3.2: Comparison for SRAM bit-cell analysis with 90% target accuracy and confidence level.

	Monte Carlo (MC)	Spherical Sampling (SS)[QTD10]	Statistical Blockade (SB)[SR09]	Proposed method (HDIS)
failure probability	2.413E-05 (0%)	2.8415E-05 (+17.7%)	2.7248e-05 (+12.9%)	2.4949E-05 (+3.39%)
#sim. runs	4.6e+6 (1150X)	2e+4 (5X)	8.16e+5 (204X)	4e+3 (1X)

The process variations are introduced into each transistor of SRAM circuit, which are same to those 9 process parameters used in Table (2.1).

3.4.2 SRAM Cell with Reading Failure

Here, the same 6-T SRAM cell circuit is used as what we discussed in Chapter 2.3.3.1. We also evaluate the reading error with the same scenario. We perform different methods (MC, SS[QTD10], SB[SR09], proposed) on this SRAM bit-cell example to predict the reading failure probability under process variations and the comparison results are shown in Table 3.2.

3.4.2.1 Accuracy Comparison

At a first glance, we would be very surprised to find that SS[QTD10] method based on conventional importance sampling framework can provide accurate failure rate predictions in this 54-dim problem!

However, this comparison cannot allow us to reach that conclusion, because

this SRAM bit-cell example is a "pseudo" high-dimensional problem for two-fold reasons: (1) during the reading operation, not all transistors are active. In fact, both $Mp5$ and $Mn3$ are shut off, therefore, the process variations on these two transistors have no effect on discharge behavior of bit-lines at all; (2) without loss of generality, assuming $\bar{BL}$ = '0' and BL='1', the discharge current flows from $\bar{BL}$ to the ground through $Mn2$ and $Mn1$ so that to pull down the voltage of $\bar{BL}$. As such, the process variations in $Mn2$ and $Mn1$ have more significant effects on the discharge behavior of bit-lines and can potentially mask the variation effects in $Mp6$ and $Mn4$. In this way, there are only 18 "effective" variable parameters, which suggests that this example is a problem with modest dimension.

When compared with MC results, the proposed method provides the most accurate failure probability estimation with only 3.39% relative error, while the estimations from SS[QTD10] and SB[SR09] have more than 10% relative error.

3.4.2.2 Efficiency Comparison

From Table 3.2 we also compare the efficiency of these methods: MC is very time-consuming and requires nearly 4.6 millions transistor-level SPICE simulations; SB[SR09] can provide 6X complexity reduction by screening out and simulating those "most-likely-to-fail"samples; SS[QTD10] method is made more efficient (230X speedup over MC) by better choosing failed samples using importance sampling algorithm; the proposed algorithm achieves the best convergence rate (1150X faster than MC) by efficiently spreading more samples into the failure region using a sampling distribution with a large-standard-deviation in high dimensions.

3.4.3 Sense Amplifier for Target Gain

Next, we consider a sense amplifier example which includes 13 transistors as shown in Figure 3.7.

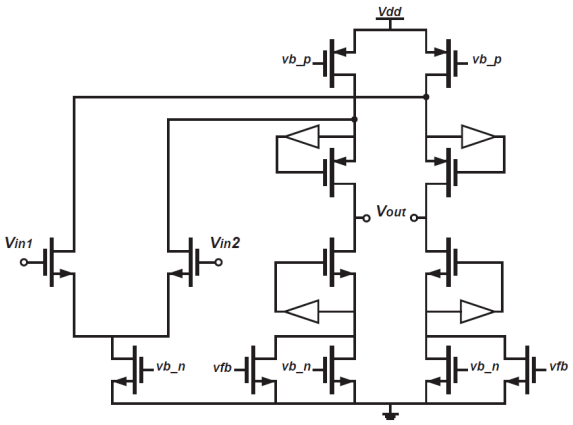


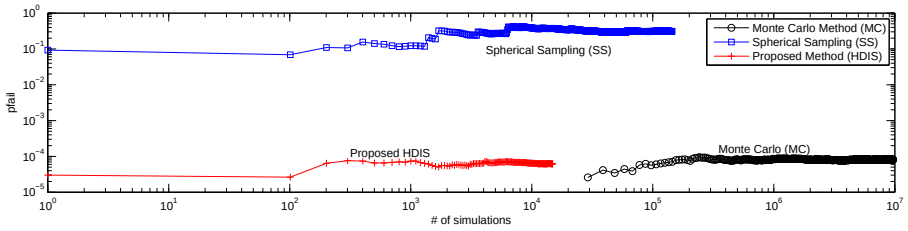
Figure 3.7: The schematic of a sense amplifier circuit.

In a SRAM circuit, the sense amplifier is designed to magnify the voltage difference between $\bar{BL}$ and BL. If the gain is too small, the output of this amplifier might be too weak to be read by the decoder circuit. Therefore, a reading failure happens. With the variation modeling summarized in Table 3.1, the sense amplifier example has 117 random variables in total. More importantly, all of these variable parameters are "effective" because the transistors are active and process variations on each transistor can significantly change the gain, which is a truly high-dimensional problem.

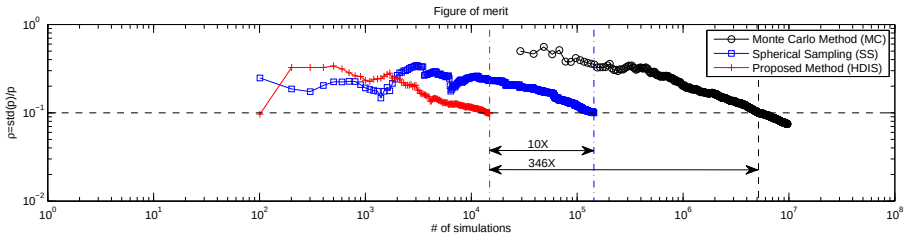
3.4.3.1 Accuracy Comparison

To validate the accuracy of the proposed algorithm, we apply different methods (MC, SS[QTD10], SB[SR09] and proposed) on this 117-dim problem to predict the timing failure probability. Here, MC serves as the "gold standard". SB[SR09], is not included in the further comparison, because the classifier used in SB[SR09] fail to block any Monte Carlo sample. Therefore, Considering the complexity of running the classifier, the SB [SR09] involves even higher computation complexity than MC method.

The evolution of the probability estimation in different methods are plotted in Figure 3.8(a). Several observations can be made:



(a) failure probability



(b) figure of merit

Figure 3.8: Comparison between different methods in terms of the failure probability estimation and figure of merit

First, this figure shows the failure of conventional importance sampling (i.e., SS[QTD10]). In fact, due to the degeneration or collapse of likelihood ratios, SS[DQS08, QTD10] method converges to a random quantity which is obviously wrong and far away from the MC result. Moreover, SS[QTD10] does not have a mechanism for improving accuracy even though more samples are added.

The proposed method builds an effective proposed sampling distribution to choose more failed samples easily and its estimation is theoretically bounded due to the proposed evaluation of conditional probability. Therefore, the proposed algorithm can reliably estimate the failure probability that matches with MC results.

3.4.3.2 Efficiency Comparison

Even though the Figure 3.8(a) provides a rough comparison of efficiency, the detailed comparison can be shown in Figure 3.8(b), where different methods try

Table 3.3: Comparison for sense amplifier analysis with 90% target accuracy and confidence level

Target Failure Probability		Monte Carlo (MC)	Spherical Sampling (SS) [QTD10]	Proposed Method (HDIS)
8e-3	prob:(failure)	8.136e-4	0.2603	7.861e-3 (3.4%)
	#sim. runs	4.800e+4 (24X)	16000 (8X)	2000
8e-4	prob:(failure)	8.044e-4	0.2541	8.787e-4 (9.2%)
	#sim. runs	4.750e+5 (36X)	8.330e4 (6.4X)	1.300e4
8e-5	prob:(failure)	8.089e-5	0.3103	8.186e-5 (1.2%)
	#sim. runs	5.156e+6 (346X)	1.430e+5 (10X)	1.500e+4

to achieve the "comparable" accuracy. Note that circuit simulation is the most time-consuming part and the runtime cost of the remaining computation becomes negligible. As such, the required number of circuit simulations for the same accuracy and confidence level serves as a measurement of the efficiency.

First, the Figure-Of-Merit (FOM) is used to quantify the accuracy of probability estimation as [DQS08, QTD10]:

$$\rho = \frac{\sqrt{\sigma^2_{p(fail)}}}{p(fail)}. \tag{3.14}$$

where $p(fail)$ is the failure probability and $\sigma_{p(fail)}$ is the standard deviation of $p(fail)$. In fact, the FOM can be viewed as a *relative error* so that lower FOM means higher accuracy of probability estimation.

We compare the evolutions of FOM for different methods in Figure 3.8(b) and draw a dash line to indicate the 90% accuracy with 90% confidence ($\rho = 0.1$). And we can have following observations:

First, SS[QTD10] has reached $\rho = 0.1$ but its estimation is completely wrong. Clearly, it cannot detect the failure at all. The same observation is applied to other existing importance sampling methods due to the boundedness analysis in Section 3.3.1.

Second, The proposed algorithm can provide the accurate estimation of failure probability with only a few thousands samples, which dramatically relieves the requirements of computing and storage efforts. As shown in this figure, the proposed method can achieve $708X$ speedup over Monte Carlo and be $17X$ faster than statistical blockade method [SR09].

3.4.3.3 Comparison for Different Failure Probabilities

We study various methods on the sense amplifier example with three different failure probabilities summarized in Table 3.3. It is obvious that SS[QTD10] method fails to achieve any reasonable accuracy in all these cases. This demonstrates the failure of conventional importance sampling method. On the contrary, the estimates from the proposed method match the MC result.

In addition, the table reveals that the proposed method provides the fastest convergence speed in all these cases and, more importantly, offers substantial complexity reduction as the failure probability becomes smaller. This property makes our proposed algorithm suitable for industrial problems where exist "rare events" with extremely small probability.

3.5 Conclusion

The HDIS [WGC14] is designed to handle higher dimensional problem by introducing a a different way to reweight the "important" samples, but it still cannot handle the problem with multiple failure regions. It has been successfully applied to the failure probability prediction of memory circuits (e.g., SRAM bit-cell, sense amplifier) and demonstrates significant complexity reduction without compromising the accuracy. Experiments on a 54-dimensional SRAM cell circuit show that the proposed approach achieves 1150X speedup over Monte Carlo without compromising any accuracy. It is also 204X faster than the classification based

method (e.g., Statistical Blockade [SR09, SR08]) by and 5X faster than existing importance sampling method (e.g., Spherical Sampling [DQS08, QTD10]). On another 117-dimension circuit, the statistical blockade [SR08] fails to improve the performance by blocking "unlikely to fail" samples, and Spherical Sampling [DQS08, QTD10] method completely fails to provide reasonable accuracy. Contrastingly, the proposed approach yields accurate result with 364X speedup over Monte Carlo.

CHAPTER 4

Piecewise Distribution Model

4.1 Motivation

As industry moves towards more energy efficient chips, minimizing power consumption becomes increasingly important. In such designs, low supply voltages (VDD) are often used to reduce power. However, while VDD is explicitly reduced the overdrive voltage ($V_{gs} - V_{th}$) is implicitly reduced [KAH12]. In the presence of V_{th} variations from the manufacturing process, transistors may enter the sub-threshold operation region causing a strongly non-linear circuit behavior. This non-linear behavior translates to circuit behavior distributions becoming strongly non-Gaussian (see Figure 4.8). Consequently, when modeling this behavior for yield analysis, it is necessary to consider the inherent non-linearity that arises due to the aforementioned reasons.

The moment matching based performance models, e.g. MAXENT [KWG13], do capture the overall shape of the PDF, but they are often inaccurate in the tail region where rare events are modeled. This limitation is because MAXENT uses only one set of moments that are accurate in the low sigma region but inaccurate in the tail. Obtaining moments that are accurate in the tail of the distribution (also known as the high sigma region) requires both a large number of samples to obtain accurate moments and knowledge of which exact moments reflect behavior in the tail of the distribution, which is often unknown [Dur10]. Consequently, the distribution that MAXENT uses is formulated on a global optimization framework

that attempts to minimize overall error, making it difficult to capture the high sigma behavior in non-Gaussian distributions.

To address both the issue of high-dimensionality *and* non-Gaussian distributions while maintaining high accuracy and efficiency, we propose a piecewise distribution model (PDM) that uses moment matching via maximum entropy to build multiple separate, region-based distributions of circuit behavior [KWB16]. Without loss of generality, we consider a distribution as two segments in the rest of this paper. The first distribution, Segment1, matches moments that are accurate only in the body/bulk of the distribution. The second distribution, Segment2, matches moments that are accurate only in the high sigma/tail region of the distribution and models the tail of circuit behavior. Both distributions are constructed using the maximum entropy moment matching technique but differ by using two *different sets* of moments. The moments in Segment1 are obtained by using circuit behavior sample moments calculated directly from the original input (process variation) distributions. The moments in Segment2 are obtained using sample moments calculated from input distributions that are *shifted* towards regions that are more likely to fail.

The optimal Segment1 distribution is selected using Spearman's rank correlation coefficient to analyze the monotonic behavior of the CDF. The Segment2 distribution is assumed to be an exponential distribution. Because this distribution is constructed from shifted moments, its probability must be re-weighed and is done so using conditional probability and a scaling factor that corrects for continuity between the Segment2 distribution and the true model of the tail distribution.

PDM has a constant complexity in terms of input dimensions as it works solely in the output (circuit behavior) domain. Experiments on both a mathematically known distribution and circuits demonstrate the method is accurate up to 4.8 sigma for non-Gaussian distributions with more than 2 orders of speedup relative

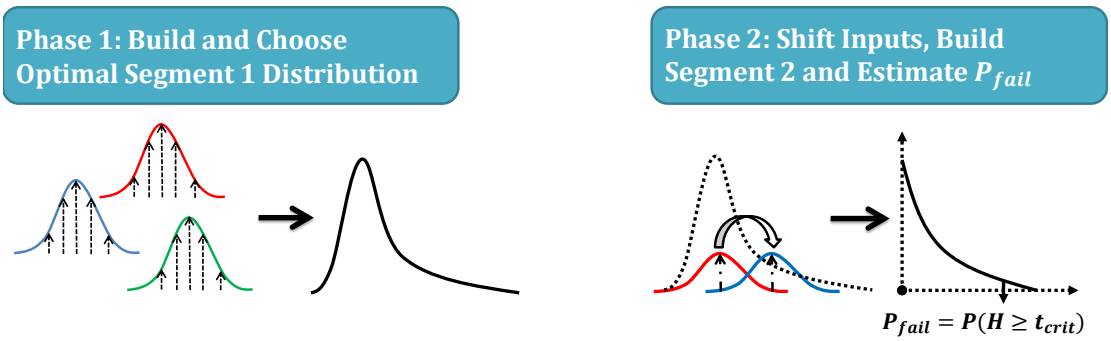


Figure 4.1: PDM contains 2 Phases: building the Segment1 distribution and selecting the optimal Segment1 distribution; shifting input parameters to build the Segment2 distribution, and estimating the final probability

to Monte Carlo, which is typically sufficient for analog circuits that are reused, such as differential amplifiers, bias circuit, or even PLLs, level shifters, etc.

More details about the proposed Piecewise Distribution Model (PDM) are elaborated in Section 4.2.

4.2 Piecewise Distribution Model

As presented in Chapter 2, we observe that MAXENT is a robust method for statistical circuit performance modeling. It guarantees stability for monotonic moments and offers high accuracy compared to other statistical modeling algorithms [LLG07, GYH11]. However, we note that MAXENT is a global moment matching approach which offers high accuracy in the bulk of the distribution, but is unlikely to capture the accuracy in the tail (high sigma) region of the distribution. To this end, MAXENT is an insufficient approach when modeling the high sigma behavior of circuit performance distributions.

In the following discussion, we propose piecewise distribution model (PDM) to accurately and effectively model the high sigma portion of non-linear distributions from circuits in high dimensionality. The motivation behind PDM is to accurately

model the tail distribution of circuit behavior by using region specific moments. In general, moment matching techniques such as [LLG07, GYH11, KWG13] use moments that may accurately reflect the bulk or body of the distribution. However, these global approximation methods use general probabilistic moments which give very little information about the high sigma areas and thus fail to accurately model the tail distribution. To this end, PDM utilizes moment matching to approximate the high sigma distribution by using *region specific moments* which capture highly accurate information in regions of interest. In general, an *arbitrary* number of segments can be used to model the overall distribution. Without losing generality, we break the total distribution into two segments - the first distribution (Segment1) matches the low sigma region and is accurate in the body (typically $\leq 4\sigma$) while the second distribution (Segment2) matches the high sigma region and is accurate in the tail (typically $\geq 4\sigma$). The flow of the method is shown in Figure 4.1 while details are given below.

4.2.1 Building the Segment1 Distribution

To build the Segment1 distribution, we first draw samples $q_i; i = \{1, ..., N_1\}$ from input parameter distributions $f(x_j); j = \{1, ..., p\}$ where p is the number of variables. Next, we simulate these samples using a circuit simulator to obtain circuit behavior outputs $y_i; i = \{1, ..., N_1\}$. Finally, sample probabilistic moments μ_k are calculated and matched using MAXENT as outlined in [KWG13, MP84]. Depending on the number of moments that are matched, we will obtain different Segment1 distributions. However, the exact number of moments to be matched is unknown because we do not know which set of moments map to different areas of the distribution [Dur10]. Consequently, we sweep across a range of values $k = 5, 7, 9, ..., K$ to build multiple Segment1 distributions and select a single, "optimal" Segment1 distribution as explained below.

4.2.2 Selecting the Optimal Segment1 Distribution

One of the key characteristics of non-Gaussian distributions is that the gradient of their CDFs are monotonically increasing, i.e. the change in circuit behavior for a fixed change in probability continuously increases as the sigma value increases. Here, the sigma value is simply the standard Z-score of a Standard Normal distribution, $P(Z \geq \sigma)$. On the other hand, the gradient is constant for a Gaussian distribution. This is illustrated in Figure 4.2 which shows the gradient of the CDF for a LogNormal (non-Gaussian) distribution vs a Gaussian distribution. Here, although the LogNormal distribution is a mathematical distribution, we label the $y - axis$ of the figure as *Circuit Behavior* to emphasize that this type of circuit behavior is of interest. Consequently, we select the optimal Segment1 distribution by choosing the one with a monotonically increasing gradient.

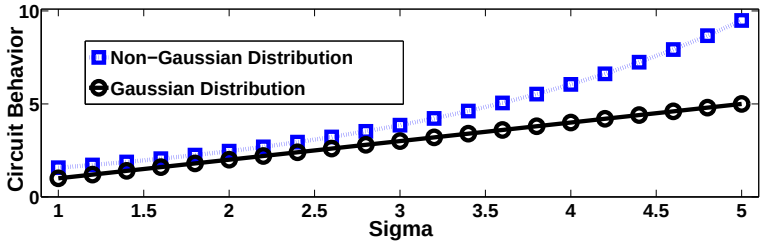


Figure 4.2: Slope of Gaussian vs Non-Gaussian Distribution

In order to gauge the monotonicity of the gradient, we turn to Spearman's rank correlation coefficient [Ken48]. Unlike the conventional Pearon correlation coefficient, which directly measures the correlation between two sets of variables, we utilize Spearman's rank correlation coefficient because it measures the *monotonic* relationship between two sets of variables. Specifically, the correlation coefficient ρ is a measure of how well a set of data can be described using a monotonic function. A coefficient of +1 indicates strong correlation to a monotonically increasing function while a coefficient of -1 indicates strong correlation to a monotonically decreasing function. To this end, we measure the gradient of the CDF for various

body distributions and compare the data set to a monotonically increasing set using Spearman's Coefficient ρ and select the distribution with the largest, positive coefficient. Figure 4.3 compares various Segment1 distributions, each built with a different number of moments, that are used for approximating a non-Gaussian distribution. We see that the coefficient for 5 of 6 distributions indicates that the gradient data set is monotonically decreasing or uncorrelated. However, there is a single distribution using 14 moments with a coefficient of $\rho = 0.98$, indicating it is a monotonically increasing set and should be used as the optimal Segment 1 distribution. In general, the optimal Segment 1 distribution may not have 14 moments.

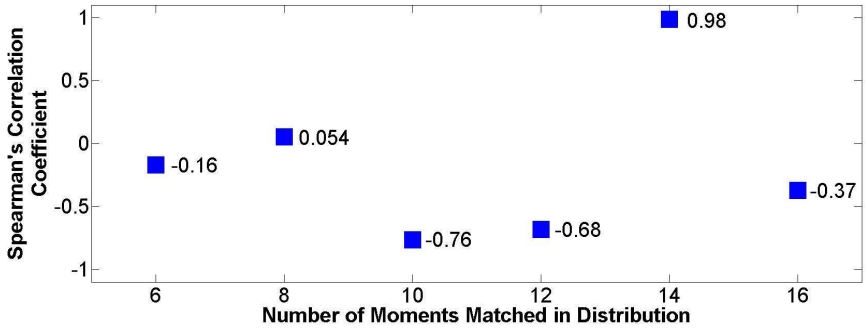


Figure 4.3: Spearman's Correlation of Distributions with Different Moments

To confirm that this is the optimal choice of the above example, we compare the estimated data from the selected Segment1 distribution (strong Spearman's correlation), one non-selected distribution (poor Spearman's correlation), and the ground truth values as shown in Figure 4.4. We see that the selected distribution matches very well with the ground truth because both distributions are non-Gaussian and exhibit monotonically increasing gradients. On the other hand, the distribution with poor correlation is very inaccurate. We utilize this combination of gradient and Spearman's correlation to select the optimal Segment1 distribution used in PDM.

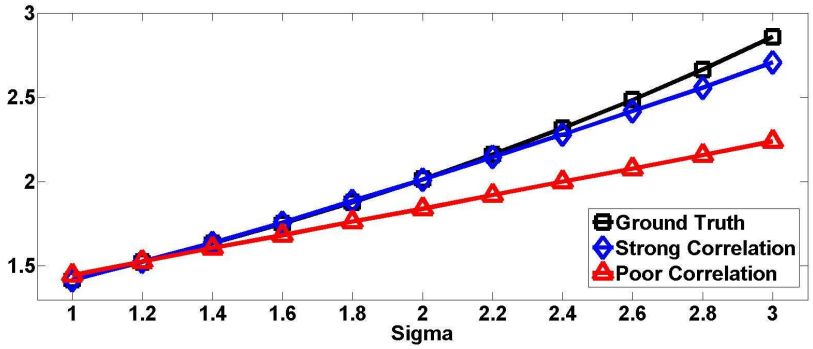


Figure 4.4: Segment1 Comparison using Spearman's Correlation Results

4.2.3 Shifting Input Distributions and Building the Segment2 Distribution

The motivation behind shifting the input distributions is to draw more samples that yield an output in the tail of the original circuit behavior distribution. By generating more samples in this region, we can generate region specific moments that are highly accurate in the tail. To obtain moments ν_l that are specific to the tail of the distribution, we must shift the mean of the input parameter distributions from m to $\hat{m}$ for each input parameter individually. To shift the mean, we first find the largest circuit behavior y_{max} from the set y_i used when building the Segment1 distribution. The value of y_{max} is directly impacted by the sampling algorithm and number of samples in N_1. Finding the optimal y_{max} is out of the scope of this paper, and the y_{max} used in the proposed application attempts a shift towards the general vicinity of parameter samples that produce tail-like circuit behaviors. Each circuit behavior y_i has a corresponding set of input samples q_j for each input parameter $j = 1, ..., p$. The largest circuit behavior y_{max} will have a sample value q_j^* for each input parameter $j = 1, ..., p$. To obtain the shifted distributions, we simply shift the mean m_j of parameter j to the sample q_j^*.

Once the input parameters are shifted, an additional N_2 samples $\hat{q}_i; i = 1, ..., N_2$ are drawn and simulated yielding an output $\hat{y}_i; i = 1, ..., N_2$. To ensure that the

moments ν_l are comprised of information *only* in the tail distribution, we must first screen the simulated data $\hat{y}_i$ such that only samples that lay in the tail are used. To do this, we simply pick a circuit behavior t^* that separates the Segment1 distribution and the next distribution, in this case Segment2. The value of t^* is obtained by selecting a sigma point s in the Segment1 distribution and extracting the corresponding circuit behavior. Typically, s is chosen to be a sigma value between 3 and 4 as this is where the long, flat region of the tail begins as shown in Figure 4.2. Next, the circuit behavior values are screened to obtain $w_k = \hat{y}_i \geq t^*; k = 1, ..., N_3$ where N_3 is the number of points beyond t^*. Because the output was screened, we ensure that the moments ν_l shall only be reflective of the tail distribution's domain and not be polluted by information outside of it.

Finally, to build the Segment2 distribution, we calculate $l = 4$ moments using $\mu_i = \int x^i p(x)dx$ and match them using maximum entropy as in [KWG13, Wu03, CHZ10, MP84]. The motivation behind using only 4 moments is that this forces the maximum entropy method to yield an exponential distribution as shown in [Con13]. The exponential distribution is a good approximation of the tail as it is monotonically decreasing and can easily be obtained using the maximum entropy method.

4.2.4 Reweighing Segment2 via Conditional Probability

Once the Segment2 distribution is obtained, the probability for a specified circuit behavior t_{spec} can be obtained; however, it will be inherently biased because the input parameters were shifted to draw more important samples. To resolve this issue, we use conditional probability to "re-weigh" probabilities as follows

$$P(H \geq t_{spec}) = P(H \geq t_{spec} | B \geq t^*) * P(B \geq t^*) \tag{4.1}$$

Where H is the random variable associated with the Segment2 distribution, B is the random variable associated with the Segment1 distribution, t_{spec} is the circuit

behavior whose probability is of interest, and t^* is the circuit behavior for sigma point s. The conditional probability relationship in (4.1) works well when the two distributions H and B are identical, i.e. if we are calculating conditional probability under one distribution, or if they share the same mean. However, this equation does not hold true in the proposed algorithm. This is demonstrated by rearranging (4.1) as shown in (4.2).

$$P(B \geq t^*) = \frac{P(H \geq t_{spec})}{P(H \geq t_{spec} | B \geq t^*)} \tag{4.2}$$

For a new point $t^{'}_{spec}$, the relationship is

$$P(H \geq t^{'}_{spec}) = P(H \geq t^{'}_{spec} | B \geq t^*) P(B \geq t^*) \tag{4.3}$$

$$P(B \geq t) = \frac{P(H \geq t^{'}_{spec})}{P(H \geq t^{'}_{spec} | B \geq t^*)} \tag{4.4}$$

Rearranging (4.2) and (4.4) and equating the common term yields

$$P(B \geq t) = \frac{P(H \geq t_{spec})}{P(H \geq t_{spec} | B \geq t^*)} = \frac{P(H \geq t^{'}_{spec})}{P(H \geq t^{'}_{spec} | B \geq t^*)} \tag{4.5}$$

Clearly this relationship holds perfectly when the distributions from the numerator and denominator (joint and conditional, respectively) are identical as in importance sampling algorithms such as [WGC14]. However, because PDM performs the re-weighing process in the output domain, the modeled tail and the true distribution may be shaped extremely differently. In other words, because the B and H distributions are necessarily two different random variables, the relationship in (4.1) must be modified to account for the shape mismatch that inherently arises due to the unknown shape of the distributions. Consequently, we propose a dynamic scaling technique that additionally reweighs the probability under the Segment2 distribution by a scaling factor β. The scaling factor acts as a heuristic correction factor that is calculated based on the indicator function of the subset w_k of the entire circuit behavior space, and the total number of outputs N_3 as shown in (4.7). Each approximation of different t_{spec} values has a different value

of beta due to different values of the indicator function (4.6).

$$I(w_k) = \begin{cases} 0 \text{ if } w_k < t_{spec} \\ 1 \text{ if } w_k \geq t_{spec} \end{cases} \tag{4.6}$$

$$\beta = \sum_{k=1}^{N_3} \frac{I(w_k)}{N_3} \tag{4.7}$$

Using this scaling factor yields the final probability of a specified circuit behavior t_{spec} as (4.8)

$$P(H \geq t_{spec}) = P(H \geq t_{spec} | B \geq t^*) * P(B \geq t^*) * \beta \tag{4.8}$$

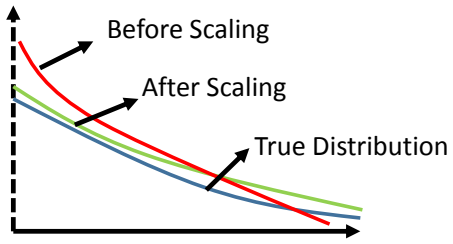


Figure 4.5: Shape Issue in Conditional Probability

Figure 4.5 shows an example of the difference in shape between the true tail distribution, the unscaled Segment2 distribution and the scaled Segment2 distribution. Additionally, we note that both Segment1 and Segment2 distributions are guaranteed to be stable, i.e. they will have a non-negative probability and therefore the CDF is guaranteed to be monotonic. This naturally arises because both distributions are calculated using the maximum entropy method and all moments in both segments are monotonically increasing.

4.3 Experiment Results

4.3.1 Experiment Settings

We implemented PDM in MATLAB using simulation outputs from HSPICE. PDM is compared with Monte Carlo, moment matching algorithm MAXENT [KWG13],

High Dimensional Importance Sampling (HDIS) [WGC14], and subset simulation (SUS) [SL14] to demonstrate that it offers significant speedup while maintaining higher accuracy than other methodologies that are targeted towards modeling the high sigma behavior of circuits.

The algorithm was tested against the mathematically known LogNormal distribution, along with the high sigma delay of a six stage clock path circuit and gain of an Operational Amplifier. The results show the estimated sigma for multiple t_{spec} values and are compared to Monte Carlo as ground truth. The Monte Carlo results were generated with roughly $8E6$ samples for the Time Critical Path and $2.5E6$ samples for the Operational Amplifier. Additionally, we compare the results to the MAXENT algorithm to show the improvements using a piecewise distribution model rather than a global approach. We also compare the results to HDIS to show that the re-weighing portion of PDM is accurate and robust for high dimensional circuits because it is independent of dimensionality. The independence is due to the re-weighing process occurring in the output domain where there is only a single variable. The source code of SUS is also obtained from its original authors for cross evaluation. Table 4.1 gives an overview of the variables used in each circuit.

Table 4.1: Parameters of MOSFETs

Variable Name	Time Critical Path	OpAmp
Flat-band Voltage	†	
Threshold Voltage		†
Gate Oxide Thickness	†	†
Mobility	†	†
Doping concentration at depletion	†	
Channel-length offset	†	†
Channel-width offset	†	
Source/drain sheet resistance	†	†
Source-gate overlap unit capacitance	†	†
Drain-gate overlap unit capacitance	†	†

The time critical path circuit has six stages and nine process parameters per

transistor for a total of 54 variables, while the circuit behavior of interest is the delay from input to output. Figure 3.7 displays a schematic of the two-stage differential cascode operational amplifier, and is the same circuit as in [KWG13]. The circuit has a total of thirteen transistors and four gain boosting amplifiers. In total, only ten transistors are considered to be independently varied. However, transistors in the gain boosting amplifiers are also varied, though due to the mirrored properties of the circuit they are varied simultaneously and are counted as one variation. As such, although each transistor has seven process parameters resulting in a total of 70 variables, the true number of variables is much higher. In the proposed algorithm, the circuit behavior of interest is the gain $\frac{V_{out1}}{V_{in1}}$.

4.3.2 Experiment on Mathematically Known Distribution

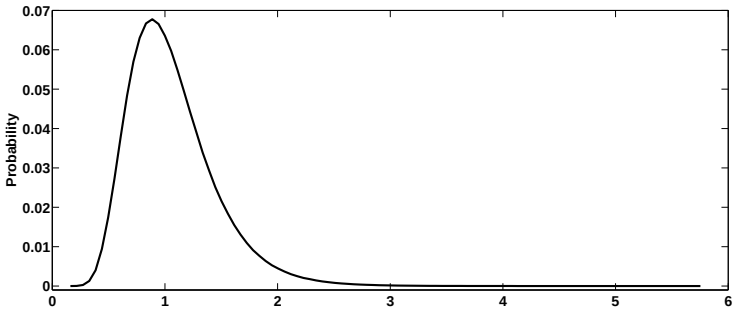


Figure 4.6: LogNormal PDF

To illustrate the capability of modeling strongly non-Gaussian distributions, we use PDM to model a LogNormal distribution. The LogNormal distribution with mean and sigma parameters $\mu = 0$, $\sigma = 0.35$ was selected because of its strongly non-Gaussian behavior. A plot of the PDF of this distribution is presented in Figure 4.6. The distribution appears to be Gaussian for a small portion due to the bell shaped curve, but it has a very long tail, giving it the non-Gaussian properties that are of interest.

Figure 4.7 shows the high sigma modeling results for Monte Carlo, PDM,

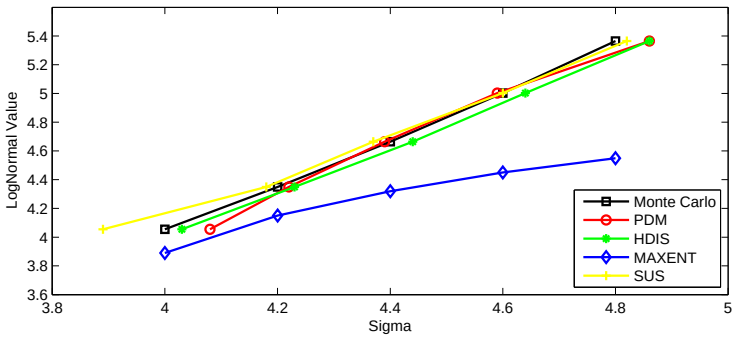


Figure 4.7: LogNormal Sigma Behavior

HDIS, MAXENT, and SUS at multiple t_{spec} points. The figure is the CDF zoomed into the tail area with the x-axis as sigma and y-axis as the value of the random variable, precisely circuit behavior. Here sigma is used to represent probability, i.e. $4\sigma \approx 0.000064$ in the tail. The motivation for this type of plot is to best represent the non-linear behavior of a non-Gaussian PDF. Additionally, it shows only the high sigma behavior rather than the overall distribution because that is the motivation and focus behind this algorithm.

While the number of samples required for SUS ranges from 5800 to 7400 in the experiment setup, HDIS, MAXENT and PDM each used a total of 4000 samples, with PDM using 3000 samples to calculate the Segment1 distribution and 1000 samples to calculate the Segment2 distribution. In this case, the point s that separates Segment1 and Segment2 is selected to be the 4 sigma point, i.e. whatever circuit behavior that corresponds to a tail probability of $6.4E-5$ in the Segment1 distribution. By introducing the Segment2 distribution at the point s, PDM is able to avoid any errors that MAXENT suffers from, allowing PDM to match almost identically with the Monte Carlo results up to 4.8 sigma. By utilizing region specific moments and doing a piecewise approximation of the distribution, PDM keeps consistently small errors. On the other hand, the MAXENT algorithm begins to lose accuracy and fails to capture the tail of the distribution because it only uses one distribution to model the overall behavior.

Furthermore, we see that HDIS and SUS has accuracy comparable to both PDM and Monte Carlo. At the 4 sigma point, we see that HDIS is slightly more accurate. However, between 4 and 4.8 sigma we see that PDM is more accurate, with both algorithms exhibiting the same, good accuracy at the 4.8 sigma point. These results intuitively make sense as the LogNormal distribution has only 1 variable so HDIS does not suffer from the curse of dimensionality. Moreover, because it has only 1 variable, it is able to find a good shift. Similarly, PDM is able to maintain very high accuracy because it matches region specific moments. Table 4.2 shows the error in estimated sigma for PDM. The error is between -0.25% and 2% all the way to the 4.8 sigma point.

We also note that MAXENT and PDM do *not* assume the distributions to be matched are Gaussian distributions because they do not match only 3 moments. [Con13] outlines that the maximum entropy moment matching method can be forced to assume a Gaussian distribution if we match exactly 3 moments. However, because we sweep through a wide range of moments for both MAXENT and PDM, we, in general, will never pick a Gaussian distribution because it does not agree with the gradient criteria selected by Spearman's correlation coefficient. Consequently, the high error that MAXENT suffers from is due to its limitation of using one set of moments, not from any assumptions about its model.

Table 4.2: Sigma Error for LogNormal

True Sigma	Estimated Sigma	% Error
4.0	4.0786	1.9650%
4.2	4.2224	0.5333%
4.4	4.3886	-0.2591%
4.6	4.5888	-0.2435%
4.8	4.8569	1.1854%

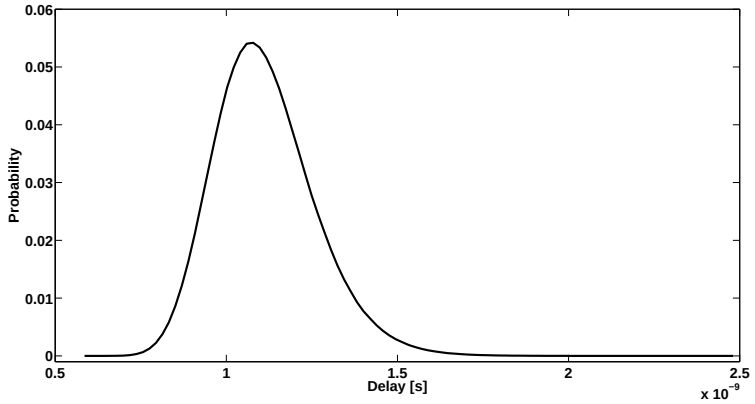


Figure 4.8: Clock Path PDF

4.3.3 Experiment on Circuits

The Monte Carlo distribution of the time critical path circuit delay is presented in Figure 4.8. Because the circuit operates at a very low VDD level, it behaves in a slightly non-linear way. The distribution, while not as long tailed as the LogNormal, has a more elongated tail than a Gaussian distribution.

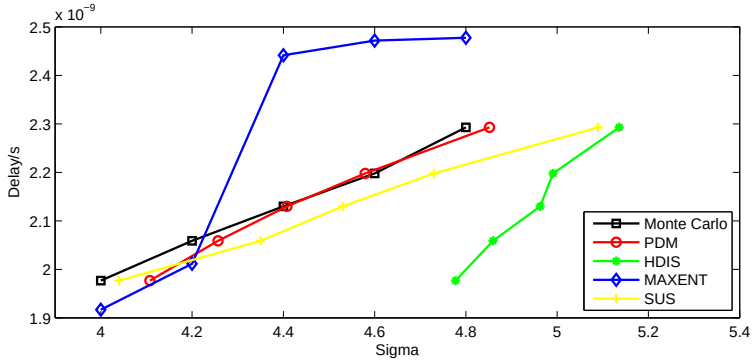


Figure 4.9: Clock Path Sigma Behavior

SUS uses between 5803 and 9010 samples for different sigma points. With a slightly larger number of samples, the probability estimated by SUS is accurate in overalthe bulk of the tail, but is optimistic with respect to Monte Carlo and PDM. At the same thresholds, the sigma calculated by SUS is between 0.04 and 0.2 larger than the MC results.

Furthermore, we see that the results from HDIS are completely inaccurate compared to both Monte Carlo and PDM. HDIS is unable to come anywhere near the proper sigma value for any of the points that it estimates. This is likely inaccurate from a combination of high dimensionality and an inaccurate shift in the mean and sigma of the new sampling distribution that causes the re-weighing process to again become inaccurate. Simply put, if the shifting method is inaccurate the results from HDIS will be inaccurate. If a larger number of samples is used, then the shift and corresponding samples drawn from the new distribution will be more accurate; however, due to the run time prohibitive nature of high dimensional circuits, it is imperative to minimize the number of samples. On the other hand, the shifting method in PDM is more robust because the re-weighing process is performed in the output domain and is performed using conditional probability rather than as a ratio of two distributions. Table 4.3 shows the error in sigma between PDM and the ground truth from Monte Carlo. We see a worst case error of 2.7% at 4 sigma but significantly less errors at higher sigma values.

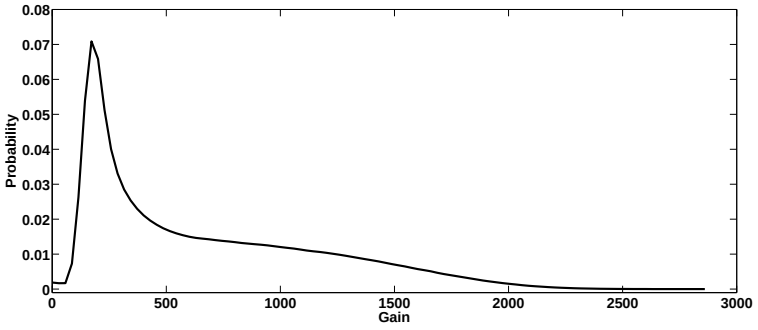


Figure 4.10: Op. Amp PDF

The Monte Carlo distribution of the Operational Amplifier circuit gain is shown in Figure 4.10. The distribution is heavily skewed and has a very sharp peak near the beginning and proceeds to drop very quickly, However, it also has a slightly flatter portion that eventually decreases to a long, flat region of the tail. It clearly has a long tail and behaves in a strongly non-Gaussian way.

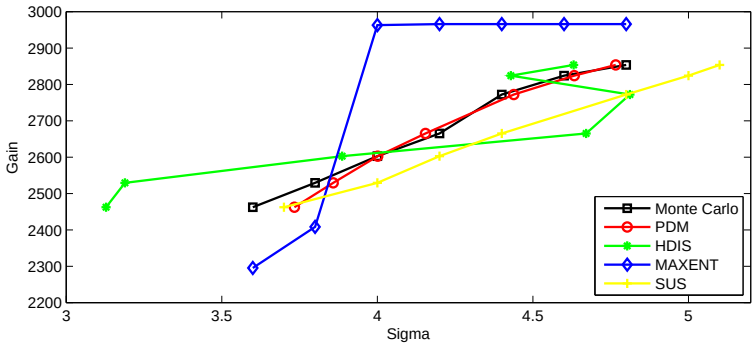


Figure 4.11: Op Amp Sigma Behavior

Figure 4.11 shows the high sigma modeling results for Monte Carlo, MAXENT, and PDM at multiple t_{spec} points. The figure shows only the high sigma behavior rather than the overall distribution because that is the motivation and focus behind this algorithm. Both MAXENT and PDM used a total of 3000 samples, with PDM using 2000 samples to calculate the Segment1 distribution and 1000 samples to calculate the Segment2 distribution. In the case of the OpAmp, the point s was determined to be the 3.6 sigma point rather than the 4 sigma point as in the previous cases due to the extremely long-tailed nature of the distribution. Before the point s, it's clear that PDM has a larger error (roughly 5%) than in previous cases. However, when we introduce the Segment2 distribution, PDM is able to immediately recover and match the 3.8 sigma point closely and continues to match larger sigma points and the overall shape of the Monte Carlo curve very well. By introducing this second "piece" to model the distribution, we are able to get a significant increase in accuracy. On the other hand, the MAXENT method has a large error, blows up and returns noise values because it is unable to capture the tail of the distribution as it does not use moments that are specific to that region. We again note that MAXENT does not assume the distribution is a Gaussian model because it matches more than 3 moments. Hence, its error is due to limitations of using one set of moments to model the total distribution.

The SUS algorithm used between 5004 and 8216 samples at different sigma

points. The sigma estimated by SUS is pessimistic with respect to Monte Carlo and tends to slightly overestimate the true sigma value, with small pessimism at lower sigmas (3.7 vs 3.6) and higher pessimism at larger sigmas (5.1 vs 4.8). However, like PDM, SUS is able to capture the overall trend and shape of the Monte Carlo results.

Table 4.3: Sigma Error for Circuits

Time Critical Path			Op Amp		
True Sigma	Estimated Sigma	% Error	True Sigma	Estimated Sigma	% Error
4.0	4.1077	2.693%	4.0	4.0015	0.0375%
4.2	4.2571	1.360%	4.2	4.1547	-1.0786%
4.4	4.4080	0.182%	4.4	4.4386	0.8773%
4.6	4.5793	-0.450%	4.6	4.6329	0.7152%
4.8	4.8517	1.077%	4.8	4.7662	-0.7042%

Moreover, we observe that the results from HDIS are inaccurate and at one point has a huge jump in its results and is simply noisy throughout. Although the Operational Amplifier circuit is not as high dimensional as the Clock Path, HDIS is still unable to properly model the high sigma region. Again, the inaccuracy is most likely from an inaccurate shift in the mean and sigma of the new sampling distribution that causes the re-weighing process to again become inaccurate. Table 4.3 shows the error in estimated sigma between PDM and the ground truth from Monte Carlo. We see very accurate results with a worst case error of about -1% at 4.2 sigma.

4.3.4 Speedup Comparison

To analyze the efficiency of the proposed method, we compare the number of samples required by PDM to the number of samples used for Monte Carlo. Since the LogNormal distribution is a mathematically known circuit and requires no Monte Carlo simulations, we exclude that speedup comparison. In the clock path circuit, PDM requires a total of 4000 samples - 3000 samples for the body distribution

and 1000 for the hybrid distribution. In the Operational Amplifier, PDM requires a total of 3000 samples - 2000 samples for the Segment1 distribution and 1000 for the Segment2 distribution. Table 4.4 compares the Monte Carlo and PDM runtime requirements and the speedup for all circuit examples. We note that the speedup of the algorithm compared to Monte Carlo will vary based on the number of samples that are used; however, it is clear that PDM offers a significant speedup at very little loss in accuracy.

Table 4.4: Speedup Comparison

Circuit	Monte Carlo Runtime	PDM Runtime	Speedup
Clock Path	8,000,000	4000	$2000x$
Op. Amp.	2,500,000	3000	$833x$

4.4 Conclusion

In this Section, we proposed PDM - a piecewise distribution model that performs region based moment matching to extract the PDF of circuit performance. PDM is able to model the high sigma regions of the circuit performance PDF. In particular, we introduced a second distribution based on a set of moments that are accurate in the tail of the PDF leads to significantly improved accuracy over the basic MAXENT [KWG13] with little error compared to Monte Carlo.

CHAPTER 5

Rare-event Microscope: Aiding Yield Analysis with Classification Algorithms

5.1 Multiple Failure Regions

5.1.1 Limitation of Existing Works

The aforementioned HDIS (in Chapter 3) and PDM 4 tackle the high sigma yield analysis problem via different way. While HDIS finds an "important" region and shifts the sample mean to capture more samples in that region and to calculate the reweighed failure probability, the PDM breaks the performance distribution into multiple segments and calculates failure probability via conditional probabilities. To collect enough sample for the next segment, PDM also shifts the sample mean to cover the "important" region.

Note that one common steps involved in HDIS and PDM is shifting the sample mean to collect more important samples. The same mean-shifting technique has also been applied in several other approaches, including mixture importance sampling [KJN06], spherical sampling [DQS08, QTD10], while the sample mean of the "important" region is located in different ways.

The mean-shifting based approaches are able to produce accurate estimation when failed samples are located in a single connected region. However, they do not consider the condition that the failure samples fall in multiple failure regions. As illustrated in Figure 5.1, mean-shifting based approaches fail to cover

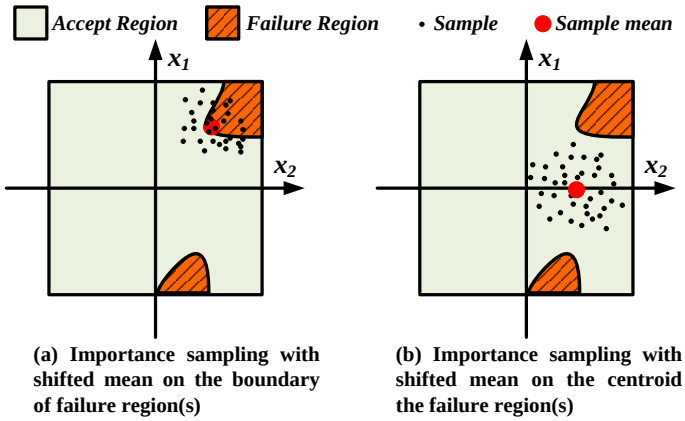


Figure 5.1: Mean-shifting based methods on a yield analysis problem with two disconnected failure regions

the failed samples distributed in two disconnected failure regions. For example, spherical sampling methods [DQS08, QTD10] search the fail sample with minimal Euclidean distance to the nominal value. Hence it locates the sample mean on the boundary of the failure region on the top right corner, while totally ignore the failed samples in the other failure region. As illustrated in Figure 5.1(b), other importance sampling approaches take the centroid of the failed samples in the presampling phase, leading to a shifted sample mean even not within any failure region.

Beside the mean-shifting based approaches, the basic classification based approach, statistical blockade (SB) [SR08], utilizes a classifier to block samples that are unlikely to fail, leaving only likely-to-fail samples to simulate. More recently, recursive SB [SWC08] and REscope [WXK14] are proposed to tackle problem with multiple failure regions. However, recursive SB assumes that each failure region is associate with different label, which does not hold for several circuits [MAL14, ML14].

In this Chapter, we proposed a new classification based approach, rare event microscope (REscope). REscope zooms into the failure regions and models the

circuit performance distribution of likely-to-fail samples into a generalized pareto distribution (GPD), which is known as a good model of the tail of the PDF [SR08, HW87]. It prunes the less useful process variation parameters in a high-dimensional problem by considering the contribution of each parameter to the performance metrics. Furthermore, we applied a nonlinear SVM classifier which is capable of identifying multiple disjoint failure regions. On a 108-dimension charge pump circuit in the phase lock loop (PLL) design, the proposed method outperforms the importance sampling approach and is 389x faster than the Monte Carlo approach. Moreover, it estimates the failure rate accurately, while importance sampling totally fails because the failure regions are not correctly captured.

The highlights of the proposed REscope are summarized as follows:

- Fix the high dimension problem by performing feature ranking according to their contribution to the circuit performance, and only select the important features.
- Enable the classifier to handle multiple regions efficiently by using a nonlinear mapping function as the SVM kernel. Therefore the boundary between pass region and failure region do not have to be a linear hyperplane as in SB [SR09, SR08].
- More robust way to matching the tail to a GPD distribution. (Use probability-weighted moment matching results as initial value and iteratively refine the result via maximum likelihood optimization)

5.2 Preliminary Knowledge

5.2.1 Modeling Rare Events using GPD

In statistical circuit simulation, it is called circuit failure event when a circuit performance metric does not meet the requirement. Mathematically, given a cir-

cuit with several process variations $S = \{X_1, X_2, ..., X_N\}$, the statistical circuit simulation analyze the failure probability, i.e., a performance metric Y exceeds certain failure threshold y_f. The failure probability P_f can be represented as

$$P_f = P(Y > y_f) = 1 - F(y_f) \tag{5.1}$$

where $F(y)$ is the cumulative distribution function (CDF) of performance metric Y.

A typical way to efficiently statistically model Y is simulate a small size of Monte Carlo samples and apply moment matching to fit the simulation result into certain analytical form $F_{mm}(y)$ [LLG07, GYH11, KWG13]. These approaches may correctly capture the body shape of the distribution, it is, however, difficult to exactly fit the tail. The failure probability estimated by moment matching, $1 - F_{mm}(yf)$, could be very inaccurate. Hence, we need to particularly model the tail of the distribution.

To simplify the discussion, let's assume that the performance metric$Y$ belongs to a "lognormal" distribution, which is usually used to model circuit performance, i.e., memory read/write time. The PDF of a lognormal distribution is defined as

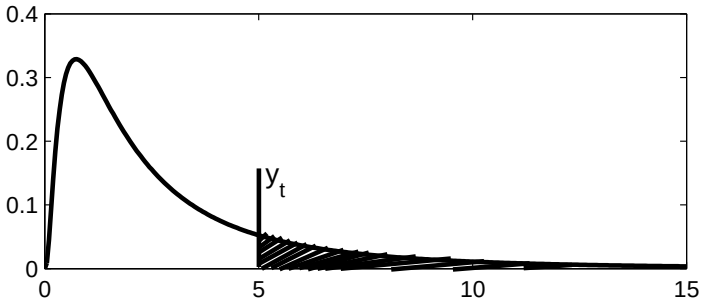
$$f_{\mu,\sigma}(y) = \frac{1}{y\sigma\sqrt{2\pi}} \exp(-\frac{(\ln y - \mu)^2}{2\sigma^2}) \tag{5.2}$$

where μ and σ are the mean and standard deviation, respectively. A lognormal distribution with $\mu = \ln 2$ and $\sigma = 1$ is presented in Figure 5.2(a). Suppose y_t is a threshold that separates a tail from the body of the PDF function $f(y)$, the conditional CDF on the tail can be expressed as

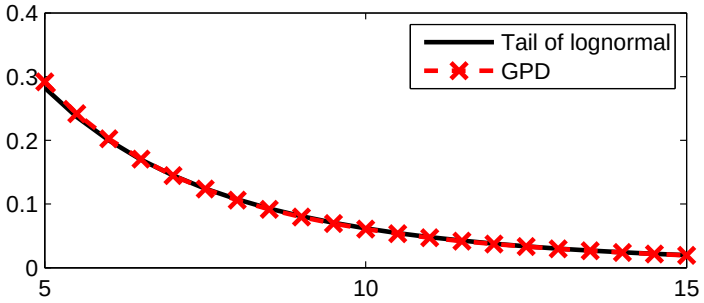
$$F_t(y) = P(Y > y | Y > y_t) = \frac{F(y) - F(y_t)}{1 - F(y_t)} \tag{5.3}$$

If we know the $F(y_t)$, the failure probability of the given threshold y_f can be calculated as:

$$P_f = (1 - F(y_t))(1 - F_t(y_f)) \tag{5.4}$$



(a) Lognormal distribution



(b) Conditional PDF of the Lognormal tail

Figure 5.2: Model the tail of lognormal using GPD

Fortunately, $F(y_t)$ can be accurately estimated by a few thousand samples because the event of $Y > y_t$ is not that rare. Therefore, the remaining problem is to correctly model the conditional CDF $F_t(y)$.

For several decades, the generalized pareto distribution (GPD) has been known as a good model for the distribution of the exceedence to a certain threshold in another distribution, i.e., the tail of $F(y)$ [HW87]. The CDF function of the GPD is defined as

$$F_{(\xi,\mu,\sigma)}(y) = \begin{cases} 1 - (1 - \frac{\xi(y-\mu)}{\sigma})^{\frac{1}{\xi}} & for\, \xi \neq 0 \\ 1 - \exp(-\frac{(y-\mu)}{\sigma}) & for\, \xi = 0 \end{cases} \tag{5.5}$$

where ξ is the shape parameter, σ is the scale parameter, and μ is the starting point of the tail, which is y_t in this example. In particular, the tail of the a lognormal random variable Y, can be accurately modeled by a GPD distribution with $\xi = 0.27$ and $\sigma = 3.5$, which is shown in Figure 5.2(b).

Given that the GPD can be used to model rare event, the remaining problems turn out to be 1) how to effectively draw samples in the tail to model the GPD under high dimension, 2) how to deal with the the problem of multiple failure regions, and 3) how to accurate fit the tail distribution into GPD distribution.

5.3 Rare-event Microscope Algorithm

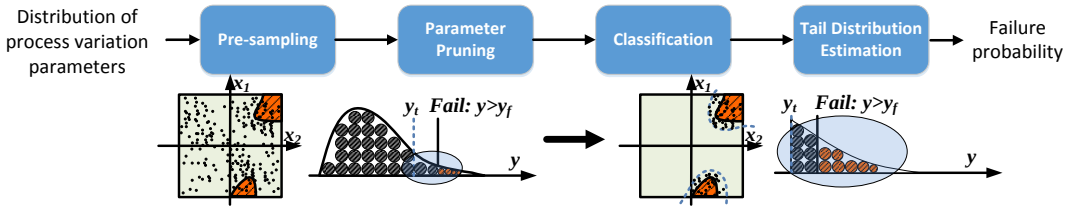


Figure 5.3: The REscrope framework consists of four components: presampling, parameter pruning, classification, tail distribution estimation.

5.3.1 Algorithm overview

In this section, we propose, REscope, to identify multiple separate failure regions in the high dimensional circuit simulation. It falls in the category of classification based methods. The REscope framework consists of four components, (1) presampling, (2) parameter pruning, (3)classification, and (4) tail distribution estimation, as shown in Figure 5.3. REscope takes in the distribution of the process variation parameters, $S = \{X_1, X_2, ..., X_N\}$, of a test circuit, and outputs the estimated failure probability of a given requirement on performance metric, i.e. $P_f = P(Y > y_f)$, where y_f is the threshold that determines whether to accept or fail this circuit. In the remaining part of this section, we will elaborate the design of each component in detail.

5.3.2 Presampling

The purpose of the presampling is to approximately sketch the circuit behavior. Without loss of generality, we use M (typically a few thousand) Monte Carlo samples, $\mathcal{S} = \{s_1, s_2, ..., s_M\}$, subject to the original distribution of S. Next, transistor level SPICE simulation is performed to evaluate the performance metric Y of the test circuit using these samples $\mathcal{S}$. A relaxed threshold y_t is chosen to determine the tail from the main PDF, and probability that a sample falls in the tail $F(y_t) = P(Y > y_t)$ is calculated.

5.3.3 Parameter pruning

With the design complexity up-scaling and advanced process technology, there are a sea of parameters in the circuit simulation. Parameter pruning, which is a way to map the high-dimensional circuit description to a low-dimension space, can effectively improve the accuracy and efficiency of circuit simulation and analysis. Existing approach, such as principle component analysis, reduces the dimension by examining the correlation among input parameters, and project them to a smaller, orthogonal base. It cannot help if each dimension of the process variation parameter, X_i and X_j, are mutually independent.

We leverage the ReliefF algorithm [KSR97] to prune parameters in REscope. More specifically, each parameter are analyzed in terms of how sensitive it is to cause a circuit failure. The sensitivity is quantified as a weight parameter. In particular, for a data set $\mathcal{S} = \{s_1, s_2, ..., s_M\}$ with M samples, where each sample $s_i = \{x_1, x_2, ..., x_N\}$ consists of N variation parameters. It starts with a N-long weight vector ,W, of zeros, and iteratively updates W. At each iteration, it take a random sample s_i, and find the closest samples (in terms of Euclidean distance) in two decision regions respectively. The closest sample in the same region is called "near-hit", and the other one is called "near-miss". The weight vector is then

updated as

$$W_i = W_i + (x_i - nearMiss_i)^2 - (x_i - nearHit_i)^2 \tag{5.6}$$

The weight vector makes sense because it increases if a feature differs from the nearby sample in the different region more than the sample in the same region, and decreases in the reverse case. It only requires a linear time in the number of given features and training instances, and is noise-tolerant and robust to feature interactions.

Different from the general sensitively analysis that only looks at the overall sensitively of the performance metric to a parameter, the ReliefF specifically looks at the sensitivity around the decision boundary of circuit pass and failure, which yields more important information than general sensitivity analysis.

5.3.4 Nonlinear SVM classifier

In the third step, a nonlinear classifier is adopted to identify whether a sample s_i falls in the failure region or accept region. Therefore, we can skip the unlikely-to-fail samples and focus on the samples in the tail.

As mentioned in Chapter 5.1, most of mean-shifting based methods assumes only one failure regions in the sample space. For instance, [DQS08] draws samples around the boundary of the failure region. While others, sush as the HDIS [WGC14], performs the importance sampling by shift the sample mean to the centroid of the failure regions. The "importance samples" may easily cover all failure samples if there is only one failure region. In reality, there might be multiple separate failure regions, the centroid of all fail samples might fall in somewhere outside the real failure regions, as shown in Figure 5.1.

A previous work based on the classification considering the existence of multiple failure regions [SWC08]. In [SWC08], the authors assume that the samples in different failure regions yield different type of failures. Therefore, they applied

the linear classifier to identify different failure types in a binary decision fashion. However, the assumption in [SWC08] loses the generality because it limits that the samples in different failure regions always yield different type of failures.

In REscope, we also consider a classification method to co-recognize the multiple failure regions. Different from [SWC08], our method is not constrained to one failure type in one region, and is also applicable to the case with various failure types. Considering the intrinsic non-linearity of circuit behavior, we employ a non-linear classifier to tackle the multiple-region multiple-type failure sample classification challenges. More specifically, we use a Gaussian radial basis function kernel (RBF) base support vector machine (SVM) to train and classify samples. The reason to choose RBF kernel rather than linear or other polynomial kernel is that in high-dimensional circuits, the decision boundary between good and failure samples is usually non-linear. RBF with radial arc boundary is more capable to adapt and discover the decision boundary.

5.3.5 Fitting the tail distribution to GPD

By performing classification, we can efficiently collect the likely-to-fail samples. Assuming $Y_p = \{y_{p1}, y_{p2}, ..., y_{pn}\}$ are the simulation outputs form the samples in the previous step that satisfy $y_{pi} > y_t$, step4 models the distribution of Y_p into a GPD. As given in 5.5, there are only three parameters, ξ, μ, and σ, to determine the CDF of GPD. While in this example, the parameter μ is known as the start point of tail, y_t. There are three approaches to approximate ξ and σ in the CDF, moment matching [HW87], probability-weighted moment (PWM) matching [HWW85], and maximum likelihood estimation (MLE) [Hos85].

The moment matching and PWM matching only use the first two order of moments to estimate these two parameters, which may lead to a mismatch in high order statistics. On the other hand, the MLE iteratively approaches the $\hat{\xi}$

and $\hat{\sigma}$ using Newton's method towards a maximum log likelihood function [Hos85]:

$$\log L(Y_p; \xi, \sigma) = -n \log(\sigma) - (1 - \xi) \sum_{i=1}^{n} z_i \tag{5.7}$$

where $z_i = -\xi^{-1} \log(1 - \xi y_{pi}/\sigma)$.

The drawback of MLE is that it may take a lot a iterations before the results finally converged to $\hat{\xi}$ and $\hat{\sigma}$.

In REscope, we use PWM matching results, ξ_0 and σ_0, as the initial solution of the Newton's method. Next, MLE is applied to iteratively approach the $\hat{\xi}$ and $\hat{\sigma}$ which maximize the log likelihood function [Hos85]. The number of iteration is reduced due the configuration of the initial value.

5.4 Experiment Results

5.4.1 Charge pump circuit and experiment setting

The performance of the REscope is evaluated using a charge pump (CP) circuit, which is a critical sub-circuit of the the phase-locked loop (PLL). The block diagram of a PLL is presented in Figure 5.4.

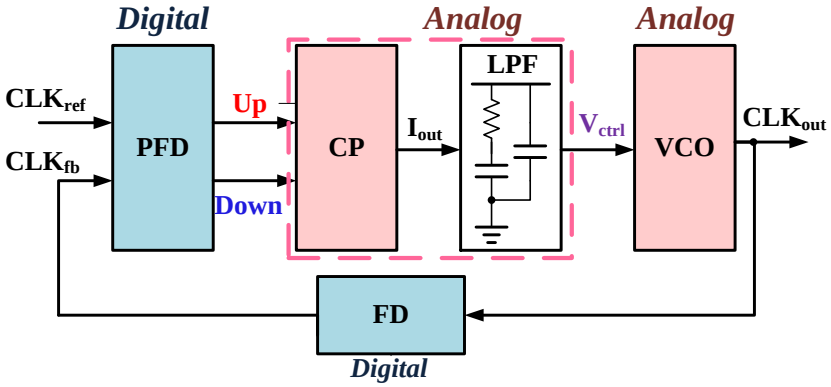


Figure 5.4: A block diagram of PLL

In this PLL, a phase frequency detector (PFD) is used to detect the phase difference between reference clocks (CLK_{ref}) and feedback clocks (CLK_{fb}). The

CP takes in the PFD output (up/down) and generate a charge or discharge current to the loop filter (LPF), which will affect the node voltage (V_{ctrl}) on LPF. This V_{ctrl} is controls the frequency of the output clock (CLK_{out}) through a voltage controlled oscillator (VCO). The output of the VCO is sent back to the PFD through a frequency divider (FD), which divide the CLK_{out} into a lower frequency CLK_{fb}, to complete the feedback loop.

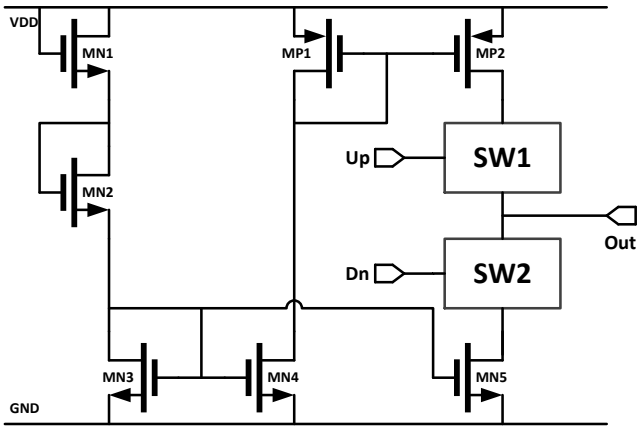


Figure 5.5: Simplified schematic of the charge pump circuit

As a sub-circuit of the PLL, CP adjusts the frequency of the output clock signal, CLK_{out}, via a charge/discharge capacitance and VCO. A simplified schematic of the charge pump consisting of two switched current sources is presented in Figure 5.5. Ideally, MN3, MN4, and MN5 on the bottom of Figure 5.5 are designed with the same dimension. The drain current flowing through these three NMOS transistors should be identical because they are imposed the same gate voltage. The same current also flows through MP1 since it shares the same branch with MN4. Similarly, on the top of Figure 5.5, two PMOS transistors form another current mirror, so that the current can be copied from MP1 to MP2. In this scenario, the charge current flowing through MP2 should be identical to the discharge current through MN5 when both switches are turned on, leading to zero net current.

In reality, it is, however, difficult to guarantee those transistors exactly the

same dimension because of the process variation effects during chip fabrication. Mismatches on these transistors, especially on MP2 or MN5, could result in a nonzero net current at the output node. It could cause large fluctuation at the control voltage, also known as "jitter", which severely affect the PLL system stability. In the following experiments, we consider a failure if there is a big enough mismatch between the charge and discharge current, mathematically

$$\max(\frac{I_{Charge}}{I_{Discharge}}, \frac{I_{Charge}}{I_{Discharge}}) > \gamma \tag{5.8}$$

where γ is a threshold of this performance matric.

The CP circuit is designed using TSMC 45nm technology and simulated with HSPICE with BSIM4 transistor model. In each transistor, we consider 4 parameters, channel-length offset (δL), channel-wight offset (δW), gate oxide thickness (t_{ox}), and flat-band voltage (V_{fb}), as the source of process variation as suggested by the foundry.

REscope are used to evaluate the mismatch current of the CP circuit in Figue 5.5. In addition, Monte Carlo (MC) method, statistical blockade (SB) [SR09] has been implemented, and the the source code of HDIS [WGC14] from its authors for accuracy and efficiency comparison. To evaluate the efficieny by counting the total number of simulations that are required to yield a stable (or confident) failure rate. In REscope, we generate a large number of MC samples and filter them by the classifier to make sure we can get enough samples on the tail. In our implementation, REscope stops when there are 1000 samples fall on the tail. The MC converges when the relative standard deviation of the failure probability, $\sigma_r = \frac{std(p_f)}{p_f}$, is smaller than 0.1.

5.4.2 Handling multiple separate failure regions

The CP is a typical circuit with multiple failure regions. To illustrate the capability of REscope on handling multiple failure regions, we use a simplified process

variation model, which only consider the threshold voltage (V_{th}) of MP2 and MN5 in Figure 5.5 as the source of process variations. When the V_{th} of MN5 is lower than the nominal value and V_{th} of MP2 is higher than the nominal, there will be a mismatch as $I_{Discharge}$ can be larger than I_{Charge}, and vice verse.

In this experiment, the threshold γ is configured to ensure a 5% failure rate. Under these configurations, the failure regions can be clearly visualized on a 2-D space, as shown in Figure 5.6(a).

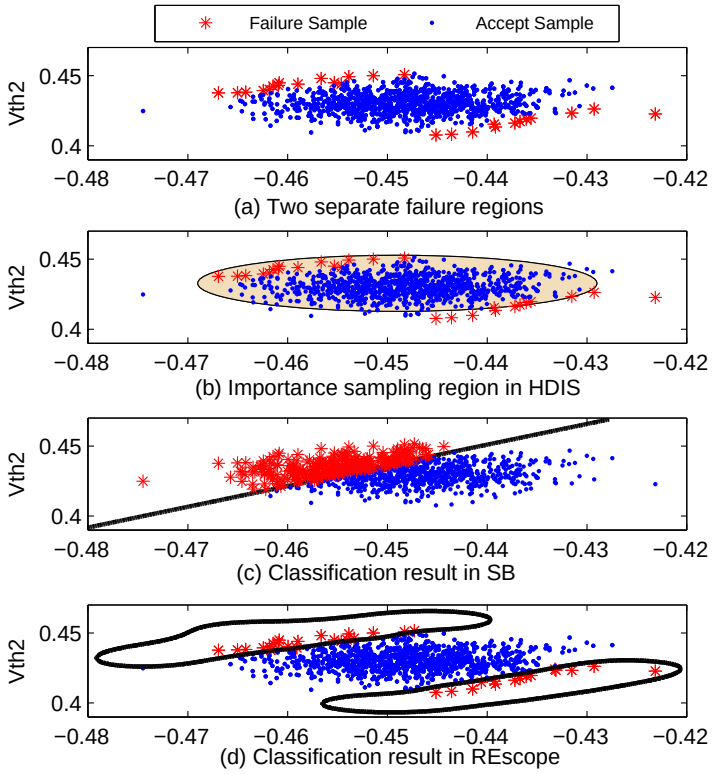


Figure 5.6: How multiple failure regions are handled in HDIS [WGC14], SB [SR08], and REscope

The importance sampling region of HDIS, along with the classification results of SB and REscope are illustrated in Figure 5.6(b), (c), and (d), respectively.

It is easy to notice that the HDIS failed to effectively capture the "importance" samples, because it attempts to draw "important" samples around the centroid of

the failure region. When there are 2 failure regions as illustrated in Figure 5.6(a), however, the centroid falls almost in the middle of a success region, which need a very large sample space to cover sufficient "important" samples.

The SB adopted a linear classifier, which is essentially find a linear hyperplane separating the successful region and fail region. However, in this example, it is impossible to separate all failure samples from the successful ones using just a linear hyperplane. In Figure 5.6(c), SB draws a boundary in the successful region, which only covers the failure region on the top-left corner and misclassifies the true samples on the bottom-right corner of the sample space. In the meantime, it also introduce a lot of over-classifications on the top-left sample space.

Taking advantage of the nonlinear classifier, REscope is able to classify all real fail samples in the sample space, which is illustrated in Figure 5.6(d).

5.4.3 Parameter weighing and pruning

In the following discussion, we model the δL, δW, t_{ox}, and V_{fb} in all 27 transistors of the charge pump circuit as process variation source, and evaluate the current mismatch. On this 108-dim problem, the REscope is compared with MC since HDIS does not succeed in capturing the failure region, nor does SB correctly classify the fail samples.

ReliefF is performance to to reduce the dimension before constructing the classifier. The weights of the ranked process variables are illustrated in Figure 5.7. It is easy to notice that the maximal weights can be more than 10x greater than the minimum. Using a higher number of input variables may fool the classifier.

In practice, we normalize the weights and setup a threshold to prune the parameters with smaller weights than the threshold. In this example, we kept the first 27 parameters and used them to build the classifier.

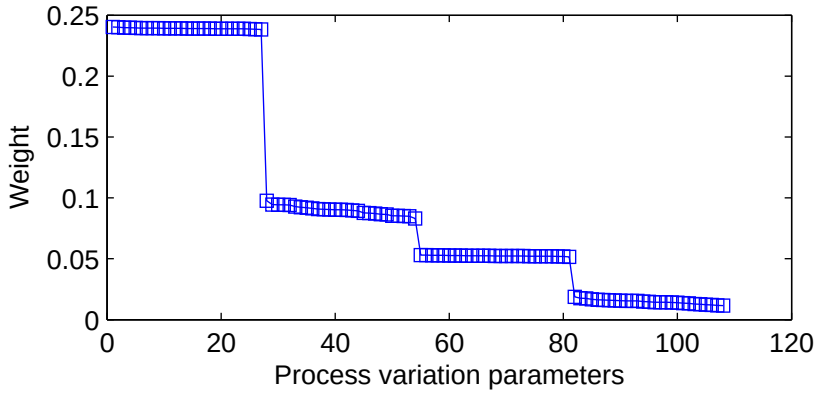


Figure 5.7: Weight of all 108 process variations in charge pump circuit

Table 5.1: Comparison of the accuracy and efficiency on charge pump circuit

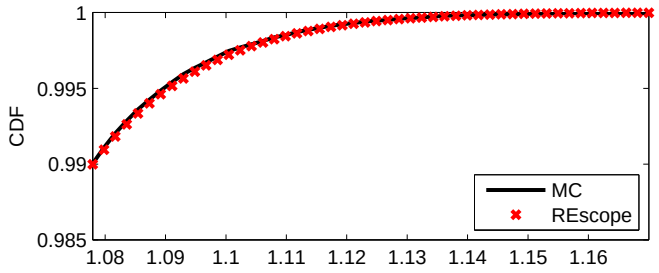
	Monte Carlo (MC)	Importance sampling (HDIS)[WGC14]	Proposed approach (REscope)
failure probability	2.279e-5 (0%)	1.136e-3	2.256e-5 (+1.05%)
#sim. runs	1.4e+6 (389x)	2e+4 (5.6x)	3.6e+3 (1x)

5.4.4 Accuracy and Efficiency

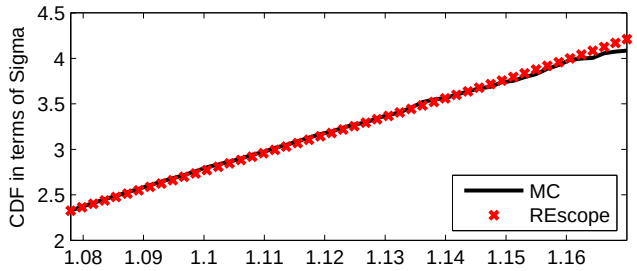
On this 108-dim problem, the REscope is compared with MC and HDIS on efficiency and accuracy, while the results are listed in Table 5.1. SB is excluded from the comparison the linear classifier generated by SB accepts all the MC samples, which makes no difference between a MC simulation. Since the HDIS didn't find the shift the mean at a desired place, it outputs a nearly random failure probability with 20 thousand simulations. The REscope accurately calculates the failure probability as 2.256e-5, with only 1.05% relative error compared with MC.

On the efficiency side, the MC needs 1.4 million to reach a confident estimation of the failure probability 2.279e-5, which is around 4.07 sigma. Beyond, 4.07 sigma, the MC result may be unreliable. On the other hand, the REscope only run 2000 samples to construct the nonlinear classifier. Next, 100,000 MC samples are generated for evaluation, but only 1621 are actually simulated, including 630 over-classified samples to avoid misclassification. Therefore, the REscope achieves

389x speedup compared with MC almost without sacrificing the accuracy.



(a) CDF tail in linear scale



(b) CDF tail in log scale

Figure 5.8: Modeling the tail of the mismatch current distribution

To examine the approximation accuracy scales when the failure probability becomes rarer, we plot the tail of mismatch current CDF function estimated by REscope in Figure 5.8(a), which perfectly fit to the MC result. In Figure 5.8(b), the fitting results are illustrated more clearly after represent the CDF in terms of sigma. The REscope estimate the probability of rare event accurately for upto 4.2 simga, which is about 1.22e-5 in terms of probability. Beyond 4.2 sigma, the accuracy of Monte Carlo cannot be guarantee as only 1.4 million MC samples are available so far.

5.5 Conclusion

In this chapter, REscope is proposed for statistical circuit simulation with rare failure event. Given a circuit with a large number of process variation parameters,

REscope first leverages the ReliefF algorithm to evaluate each parameter, and prune those that have little contribution to the circuit failure. Furthermore, we applied a nonlinear classifier which is capable of identifying disjoint multiple failure regions. Because of the classification, the computation complexity is reduced by only simulating samples that are classified as likely-to-fail samples. When sufficient samples are simulated, the simulation results are approximated to a GPD, which is usually used model the rare event.

On a 108-dimension charge pump circuit, the proposed method outperforms the importance sampling approach and is more than 2 orders faster than the Monte Carlo approach. Moreover, it estimates the failure rate accurately, while importance sampling totally fails because the failure regions are not correctly captured.

CHAPTER 6

Hyperspherical Clustering and Sampling: Aiding Yield Analysis with Hyperspherical Clustering

REscope presented in Chapter 5 solves the multiple failure region problem. Its performance is, however, heavily rely on the classifier, which works as a block box and out of user's control. In this chapter, a hyperspherical clustering and sampling approach [WBH16], HSCS in short, is proposed to effectively handle the challenges of both multiple failure regions and high dimensionality. As the first step, HSCS identifies multiple failure regions by grouping the failure samples into multiple clusters. Instead of clustering in a high dimensional open space, we sample spherically and develop a weighted spherical k-means algorithm to identify clusters only on a set of hyperspheres. Searching for min-norm points in these clusters is much easier than conventional spherical IS. Next, a modified mixture importance sampling shifts the sample mean to the min-norm points of multiple clusters so as to cover multiple failure regions.

HSCS is evaluated and compared with MC and other IS based implementations in terms of accuracy, efficiency, and robustness. On a small 2-dimensional problem with mathematically known distribution, HSCS yields very accurate results compared with mathematically calculated groundtruth. On a 70-dimensional charge pump circuit, HSCS is about 3 orders faster than MC and provides the same level of accuracy, while other IS based approaches either fail to converge or converge to wrong results. Furthermore, on both examples, HSCS demonstrates excellent robustness by generating consistent results in multiple replications.

6.1 Existing Works that Handle Multiple Failure Regions

The REscope presented in Chapter 5 is an improvement over statistical blockade (SB) [SR09] and recursive SB [SR08]. Using a nonlinear classifier with parameter pruning techniques and better tail modeling algorithm, the REscope can identify multiple failure regions at high dimensionality. However, on the other hand, REscope [WXK14] relies on support vector machine (SVM) with radial basis function (RBF) kernel to identify failure regions, but SVM works as a black box model and is out of user's control. Excessively training the SVM to identify multiple regions could easily lead to overfit.

Among others, [DL11, SL14] uses a set of sample "chains" to explore the failure region with the aid of the Markov Chain Monte Carlo (MCMC) method. However, it is difficult to cover the entire failure region with several chains of MCMC samples, particularly when tens or hundreds random variables are considered. Multi-cone approach [KJL12] deterministically breaks the original sample space into multiple non-overlapping cones, and sums up the analytically calculated failure probability in each cone. It does consider multiple failure regions, but the number of cones grows exponentially to the dimensionality, limiting it only effective for low dimensional problems.

Those drawbacks motivate the HSCS, which explicitly handles multiple failure regions.

6.2 Locating Min-Norm Point for Importance Sampling

As presented in Chapter 3.2, the essence of importance sampling is to find a proposed distribution $g(X)$ that tile towards $\mathcal{S}$ where a rare-event becomes less rare to happen. As samples are generated according to $g(X)$ rather than the original distribution$f(X)$, we need to use a weight parameter $w(X)$ to compensate

the discrepancy between $f(X)$ and $g(X)$ and unbiases the probability estimation under $g(X)$, where $w(X) = f(X)/g(X)$.

It is obvious that the samples closer to the nominal value are more desirable [DQS08] because they are associated with greater probability $f(X)$ and likelihood ratio $w(X)$, hence have more significant impact on the estimated failure probability $\tilde{P}_{IS}$. In practice, most of the existing approaches shift the sample mean to the point that is closest to the origin on the accept/fail boundary, which is also known as the minimum-norm (min-norm) point [DQS08]. However, the mean-shift IS implementations suffer from the following two drawbacks:

First, they search the min-norm points by constructing an accept/fail boundary in the open space, which may take prohibitively long runtime, especially at high dimensionality.

Moreover, while existing approaches [KJN06, DQS08, KHT10, WGC14] shift the sample mean to a more important point, they totally neglect that failed samples might be distributed in multiple disjoint regions. As illustrated in Figure 5.1, one shifted distribution might be insufficient to cover all the failures, hence leading to a biased estimation of $\tilde{P}_{IS}(Y \in \mathcal{S})$ in (3.6).

To improve the mean-shift IS, the remaining challenges turn out to be 1) identifying failure regions in high dimensional sample space, 2) effectively sampling to cover multiple failure regions.

6.3 Hyperspherical Clustering and Sampling

6.3.1 Algorithm Overview

In this section, we present the proposed hyperspherical clustering and sampling approach (HSCS). It consists of two major phases, (1) hyperspherical clustering, (2) importance sampling around multiple min-norm points, as illustrated in Figure

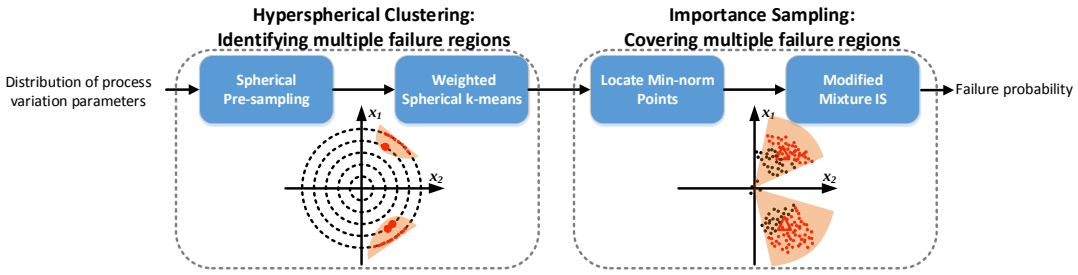


Figure 6.1: The HyperSpherical Clustering and Sampling (HSCS) algorithm consists of two phases: 1) hyperspherical clustering, 2) multiple mean-shift importance sampling.

6.1. HSCS takes in the process variation parameters, and outputs the estimated failure probability $\tilde{P}_{IS}$ based on given requirements on performance metric Y.

To accurately estimate $\tilde{P}_{IS}$, we attempt to cover more samples that are closer to the nominal value. During the clustering phase, a weighted clustering algorithm is designed to bias the cluster centers towards those samples. In the second phase, sample means are shifted to the min-norm points of multiple clusters for two purposes: 1) capture more samples with greater weights, 2) cover the failed samples in multiple failure regions.

In the remaining part of this section, we will elaborate each phase of the algorithm.

6.3.2 Hyperspherical Clustering

The hyperspherical clustering phase includes a spherical presampling step and a weighted hyperspherical k-means step to cluster the failed samples. Algorithms in this phase are targeted to find the direction of failure regions, so that statistical approaches can be applied afterwards to estimate the failure probability with a better failure region coverage.

6.3.2.1 Spherical Presampling

In order to identify multiple failure regions, it is intuitive to collect a number of likely-to-fail samples (typically samples in the quantile of the performance distribution), and to cluster them into several aggregations according to their locations in the sample space. However, clustering samples that are randomly generated in high dimensional open space is challenging. Even in the same cluster, samples may still be far apart from each other. In this scenario, a cluster centroid does not necessarily mean more failed samples, leading to meaningless clusters.

Alternatively, we restrict the samples to a few hyperspherical surfaces by sampling spherically. In this scenario, clustering algorithms can be performed on a more restricted area rather than the high dimensional open space.

As illustrated in the left part of Figure 6.1, samples are randomly generated on hyperspheres with gradually increasing radius to capture samples in the quantile. During the implementation, we generate 1000 samples on each hypersphere surface and stop expanding the hypersphere until 5% or more samples on the current hypersphere surface fall in the 1% quantile.

6.3.2.2 Weighted Hyperspherical K-means

Conventional clustering algorithms (e.g. k-means) group samples to optimal clusters by minimizing the sum of Euclidean distance [HW79] between samples and their corresponding cluster centers, as defined in (6.1).

$$EuclideanDistance(X^{(1)}, X^{(2)}) = \left\| X^{(1)} - X^{(2)} \right\| \tag{6.1}$$

$$CosineDistance(X^{(1)}, X^{(2)}) = 1 - \frac{{X^{(1)}}^T X^{(2)}}{\|X^{(1)}\| \, \|X^{(2)}\|} \tag{6.2}$$

As we generate samples on hyperspheres, Euclidean distance makes less sense because the distance between samples and the origin is the same. It is more desirable to cluster samples based on the directions those samples pointing to

rather than Euclidean distance. Therefore we use cosine distance, defined in (6.2), as the distance metric, leading to a hyperspherical version of k-means algorithm.

Furthermore, a naive hyperspherical k-means algorithm only makes use of the samples on the outermost hypersphere, without incorporating the failed samples captured on the inner hyperspherical surfaces, which are usually associated with greater likelihood ratio according to (3.5), i.e. higher importance. To take full advantage of all the failed samples, we propose a weighted hyperspherical k-means algorithm. Each failed sample is normalized to unit length and associated with a weight calculated based on its probability density. With a targeted number of clusters k, the proposed algorithm returns the cluster assignment for each input failed sample.

As the first step of Algorithm 2, a set of initial cluster centroids are randomly generated. Next the algorithm iteratively updates the cluster label assigned for all samples, cleans up empty cluster, and recalculates the centroids, until the label assignment remains unchanged after one iteration. During the cluster assignment step, the algorithm checks the cosine distance between a sample and all cluster centroids. The cluster j that maximizes ${X^{(i)}}^T \mu^{(j)}$, which is equivalent to minimizing the cosine distance, will be selected. Moreover, we assign samples different weights in the centroid update process in step 5, therefore the centroids are biased to samples with higher importance.

One caveat is that k-means searches for the cluster assignment $\mathcal{Y}$ in a greedy fashion, resulting in convergence to the local optimal instead of guaranteeing global optimum. The proposed weighted hyperspherical k-means is not an exception. In practice, we start from multiple set of randomly initialized cluster centroids $\mathcal{U}$, and choose the one leading to minimal sum of cosine distance as the solution. Hence, the final solution could be more prone to take the global optimum.

Also, the number of clusters, k, is unknown before the clustering. In practice,

Algorithm 2 Weighted Spherical K-Means Algorithm

Input: A set of M failed samples: $\mathcal{X} = \{X^{(1)}, X^{(2)}, ..., X^{(M)}\}$

Sample weights: $w^{(1)}, w^{(2)}, ..., w^{(M)}$

Number of initial clusters: k

Output: Cluster label for samples: $\mathcal{Y} = \{y^{(1)}, y^{(2)}, ..., y^{(M)}\}$

Updated number of clusters: k

1: Randomly initialize the unit length cluster centroids $\mathcal{U} = \{\mu^{(1)}, \mu^{(2)}, ..., \mu^{(k)}\}$;

2: **repeat**

3: **Cluster Assignment** (update $\mathcal{Y}$):

For each sample $X^{(i)}$, set $y^{(i)} = \underset{j}{\text{argmax}}\ {X^{(i)}}^T \mu^{(j)}$;

4: **Remove Empty Clusters** (update k)

Remove $\mathcal{X}_j$ if $\mathcal{X}_j = \{X^{(i)} | y^{(i)} = j\} = \emptyset$;

Update number of cluster k;

5: **Weighted Centroid Update** (update $\mathcal{U}$):

For cluster k, let $\mathcal{X}_j = \{X^{(i)} | y^{(i)} = j\}$, update centroid as $\mu^{(j)} = \sum_{X^{(i)} \in \mathcal{X}_j} w^{(i)} X^{(i)}$;

$\mu^{(j)} = \mu^{(j)} / \left\| \mu^{(j)} \right\|$;

6: **until** ¡$\mathcal{Y}$ remains unchanged¿

7: Return $\mathcal{Y}$ and k;

we try a number of different k and choose the one with a trade off between the model complexity and goodness of fit. In the machine learning community, k is empirically chosen to be $\sqrt{M}$ [MKB79], where M is the total number of samples to be clustered. More discussion on choosing k is included in the experiment section with concrete example.

6.3.3 Multiple Mean-Shift Importance Sampling

The previous phase generates normalized cluster centers, i.e. the direction of failure regions. In this phase, we locate the min-norm points of multiple failure

regions and apply a modified mixture importance sampling (MixIS) approach to sample in all the failure regions and to estimate the overall failure probability.

6.3.3.1 Locating the Min-norm Points using Bisection

To locate the min-norm points more accurately and efficiently, we only search towards the direction of the clusters given that they have been identified.

Mathematically, all the samples in the same cluster can be covered by a cone defined in (6.3).

$$\mathcal{C} = \{X | CosiceDistance(X, \mu) \leq d_{max}\} \tag{6.3}$$

As illustrated in the right part of Figure 5.3, the opening angle of cone $\mathcal{C}$ is constrained by d_{max}, the largest cosine distance between failed samples in this cluster and the cluster centroid μ.

Algorithm 3 Locate min-norm points for each cluster with bisection

Input: Minimal radius of existing failure samples, R

Output: Radius of min-norm point: R_{min}

1: $R_{max} = R$;
2: $R_{min} = 0$;
3: **repeat**
4: $R = (R_{max} + R_{min})/2$;
5: simulate a small set of samples at Radius = R in current cluster;
6: **if** any failed sample captured **then**
7: $R_{max} = R$;
8: **else**
9: $R_{min} = R$;
10: **end if**
11: **until** $R_{max} - R_{min} < R_{threshold}$
12: Return R;

Next, we apply bisection to search the minimal radius that leads to a failure

in each cone, as presented in Algorithm 3. Starting with a lower bound of 0, and upper bound at the minimal radius of the existing failure samples, the algorithm bisects the radius and only simulates a small number of samples at this radius. It will reduce the upper bound to search the lower half region if any failure is captured during the simulation, otherwise, it will go to the upper half.

After locating the minimal radius R_i of a cone, the min-norm point of the corresponding cluster is calculated as $Cm_i = \mu_i * R_i$, where μ_i is the normalized cluster center that indicates the direction of this cluster.

6.3.3.2 Modified Mixture Importance Sampling

Next, we modifie the MixIS and shift the sample mean to all these min-norm points found in the previous step. The proposed distribution $g(x)$ is defined as

$$g(X) = \alpha f(X) + (1-\alpha)\sum_{i=1}^{k} \beta_i f(X - Cm_i) \tag{6.4}$$

where

$$\beta_i = \frac{\sum_{X^{(i)} \in \mathcal{X}_k} w^{(i)}}{\sum_{\forall X} w^{(i)}} \tag{6.5}$$

is the weight for each failure region (cluster), which is calculated based on the sum of sample weights in the cluster.

Note that we also keep a small ratio (α) of $f(x)$ in the proposed distribution $g(x)$, so that IS likelihood ratio

$$\frac{f(X)}{g(X)} = \frac{f(X)}{\alpha f(X) + (1-\alpha)\sum_{i=1}^{k} \beta_i f(X - Cm_i)} < \frac{1}{\alpha} \tag{6.6}$$

is bounded by $1/\alpha$. It prevents the likelihood ratio from going to infinity at certain X, and preserves the numerical stability of the modified MixIS.

6.4 Experiment Results

The proposed HSCS is first evaluated using a mathematically known 2-dimensional normal distribution with 2 disjoint failure regions. Next, we verify HSCS using a more realistic high-dimensional charge pump circuit, which is known to have multiple failure regions.

6.4.1 Evaluation on Mathematically Known Distribution

On a sample space with 2-dimensional normal distribution, two disjoint failure regions, $\mathcal{S}_1$ and $\mathcal{S}_2$, are defined as follows:

- $\mathcal{S}_1 = \{X | \, \|X\| > 3.8 \text{ and } \phi(X) \in [\frac{2}{3}\pi, \frac{3}{4}\pi]\}$
- $\mathcal{S}_2 = \{X | \, \|X\| > 3.9 \text{ and } \phi(X) \in [\frac{4}{3}\pi, \frac{3}{2}\pi]\}$

where $\|X\|$ is the 2-norm of the sample, i.e. the Euclidean distance between the sample and the origin, and $\phi(X)$ is the phase of the 2-D sample. These two failure regions are illustrated in Figure 6.2[1].

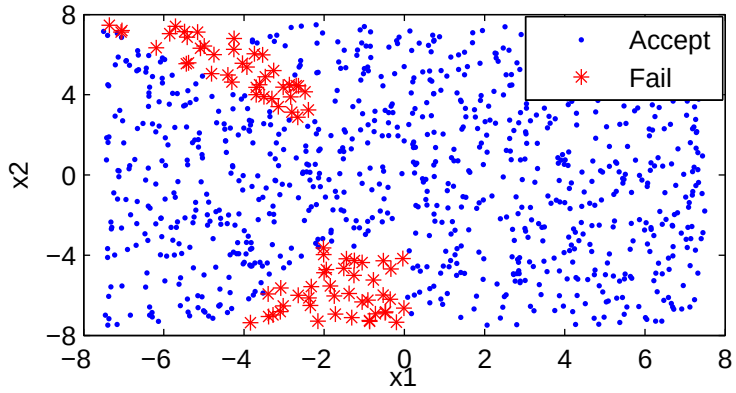


Figure 6.2: 2-dimensional sample space with two disjoint failure regions $\mathcal{S}_1$ and $\mathcal{S}_2$

[1]Figure 4 is plotted using uniformly distributed samples for better illustration.

Since the PDF and the failure regions are mathematically known, the failure probability can be calculated by integrating PDF function in (6.7),

$$P_F = \int_{X \in \{\mathcal{S}_1, \mathcal{S}_2\}} f(X)dX \approx 7.199e-5 \tag{6.7}$$

leading a failure probability of 7.199e-5, which is close to 4 sigma.

As the first step, HSCS gradually increases the radius of the sphere to search for failed samples and stops expanding until enough failed samples are collected. As illustrated in Figure 6.3, the presampling step converges at 4-sigma sphere in this particular example.

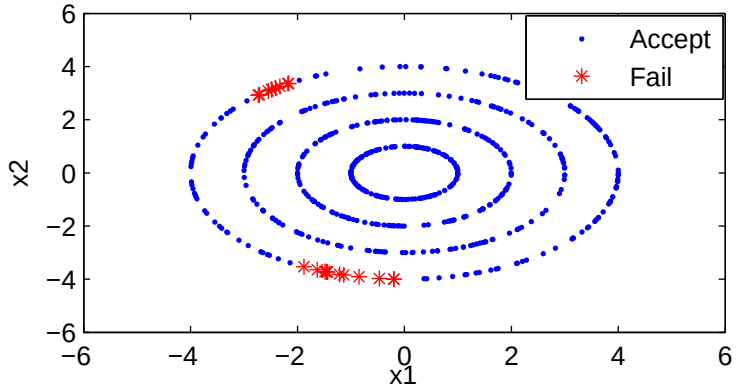


Figure 6.3: Spherical presampling to collect failed samples

Obviously, those failed samples are aggregated in two separate regions in Figure 6.4(a). The weighted hyperspherical k-means updates the cluster assignments in a greedy fashion by always assigning a sample to its closest centroid. Hence, if the initial centroids are improperly selected, it is possible that the iterative cluster assignments end up with assigning all samples in one cluster as illustrated in Figure 6.4(b), while leaving the other cluster empty (the empty cluster is removed in step 3 of Algorithm 2).

This problem has been well addressed in the machine learning community by randomly creating multiple set of initial centroids and applying the same cluster

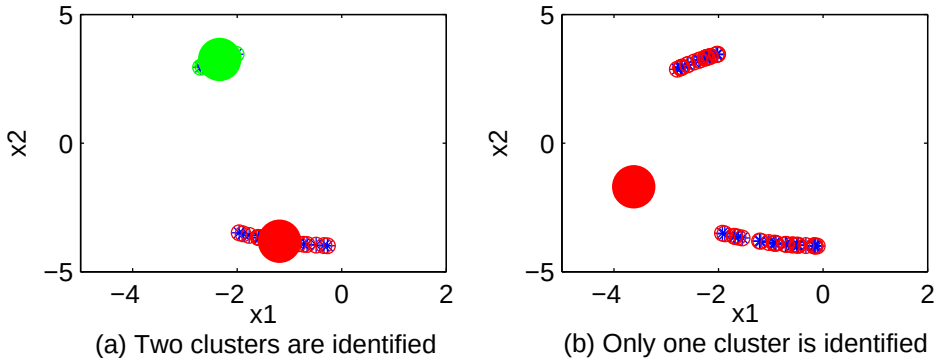


Figure 6.4: Spherical k-means might converge to local optimal with "improper" initial centroids

algorithm to all these set of samples. Only the cluster assignment with best optimization target, i.e. the smallest sum of cosine distance, will be chosen.

Next, bisection is applied to locate the min-norm points. In this example, we generate 20 samples only at each Radius. In each cluster, the algorithm ends up with 5 iterations and converges to radius at 3.9375 and 3.8125, which are very close to the groundtruth.

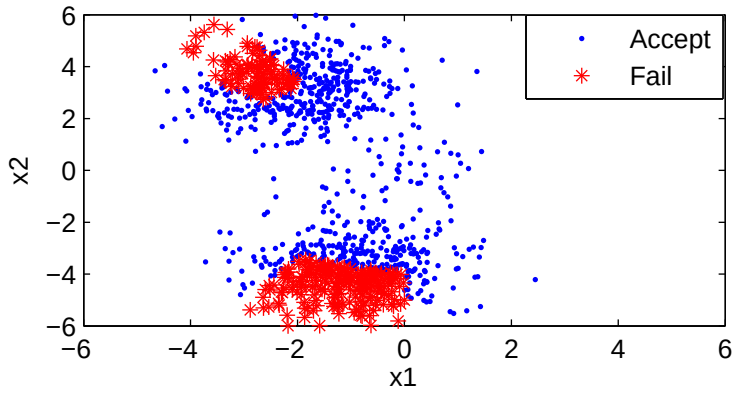


Figure 6.5: Sample coverage of the modified mixture importance sampling

The modified mixture importance sampling shifts the sample means to the min-norm points of both failure regions. Samples drew by importance sampling in Figure 6.5 indicate that both failure regions are accurately and fully covered.

The failure rate is estimated at 7.109e-5 using HSCS, which is quite close to the mathematically calculated ground truth, 7,199e-5. As a stochastic algorithm, we also run the HSCS with 100 replications to verify the stability. The estimated failure probability ranges from 5.54e-5 to 9.05e-5, with an average of 7.21e-5.

6.4.2 Experiments on Charge Pump Circuit

6.4.2.1 Charge Pump Circuit and Experiment Setting

The same charge pump circuit used in Chapter 5 is redesigned using using PTM 22nm high performance technology model [SYC12] and simulated in HSPICE. The CP circuit is a typical circuit known to have multiple failure regions [MAL14, ML14, WXK14]. We analyze this circuit with two different process variation setups.

- In the first setup, we map the variations to threshold voltage (V_{th}), and only model the V_{th} of MP2 and MN5 as variation source. Hence, the failure regions can be visualized in a 2-dimensional space.
- A more comprehensive model with 10 parameters, as listed in Table 6.1, are considered as variation source for each of those 7 transistors in Figure 5.5. Variations in those two digital switches are not accounted. In the second setup, there are a total of 70 variation parameters in the circuit, which is a relatively high dimensional problem.

In addition to the HSCS, Monte Carlo (MC) is included as the gold reference of the experiment. We also implement the high-dimensional importance sampling (HDIS) [WGC14] and spherical importance sampling (SpIS) [DQS08] for accuracy and efficiency comparison. The HDIS and Spherical IS are two typical mean shifting approaches that shift the sample mean to the centroid and min-norm point of the failure region respectively.

Table 6.1: Process variation parameters for each transistor

Variable Name	σ/μ	Unit
Flat-band Voltage (V_{fb})	0.05	V
Threshold Voltage (V_{th0})	0.05	V
Gate Oxide Thickness (t_{ox})	0.03	m
Mobility (μ_0)	0.05	m^2/Vs
Doping concentration at depletion (N_{dep})	0.05	cm^{-3}
Channel-length offset (ΔL)	0.03	m
Channel-width offset (ΔW)	0.03	m
Source/drain sheet resistance (R_{sh})	0.05	Ohm/mm^2
Source-gate overlap unit capacitance (C_{gso})	0.05	F/m
Drain-gate overlap unit capacitance (C_{gdo})	0.05	F/m

The efficiency is evaluated by counting the total number of simulations required to yield a stable failure rate. All the aforementioned approaches converge at the same criterion, i.e. the relative standard deviation of the estimated failure probability,

$$\sigma_r = \frac{std(p_f)}{p_f}, \tag{6.8}$$

gets smaller than 0.1.

6.4.2.2 2-D Setup with Visualized Failure Regions

In this setup, instead of investigating very rare failure event, we configure the threshold γ to target a 5% failure probability. Under this configuration, two failure regions can be easily visualized when we plot the accepted MC samples against failed ones in 2-dimensional sample space, as shown in Figure 6.6(a).

With only 1000 samples, the coverage of HDIS, Spherical IS, and the proposed HSCS are illustrated in Figure 6.6(b), (c), and (d), respectively. Sample means of these 3 importance sampling approaches are marked as upward-pointing triangulars in the Figures.

It is easy to notice that HDIS fails to shift the sample mean to any of the

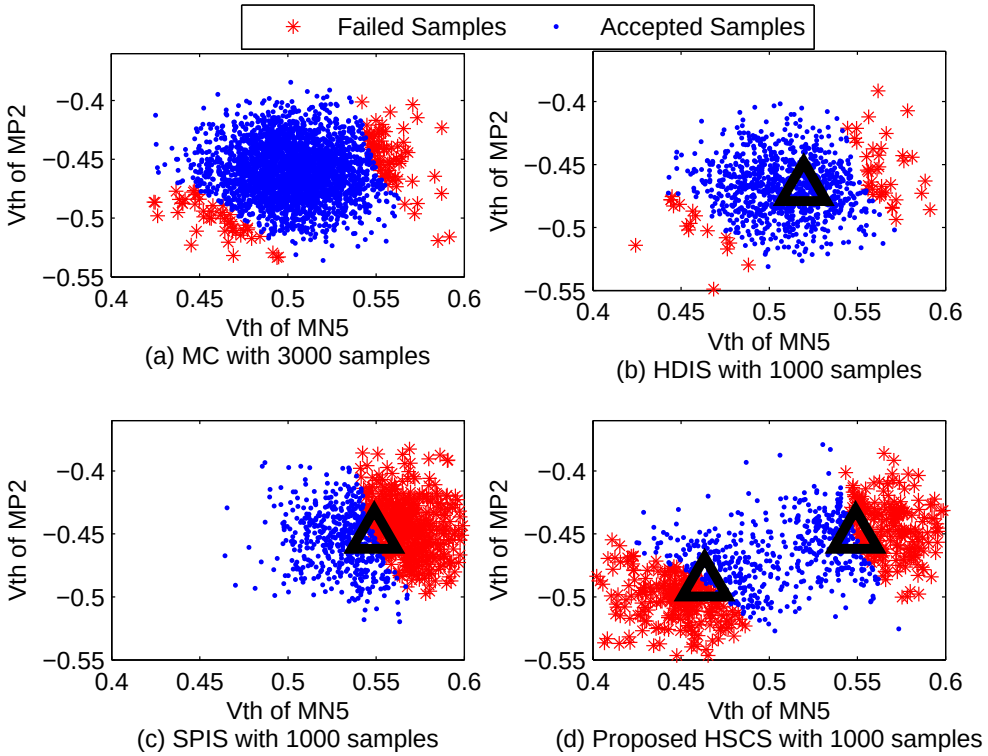


Figure 6.6: Multiple failure region coverage test MC, HDIS [WGC14], Spherical IS [DQS08], and HSCS

failure regions. As illustrated in 6.6(b), it attempts to draw samples around the centroid of the failed samples. The centroid of those failed samples, however, falls almost close to the origin, which is obviously not in the failure region, leading to a poor coverage on those truly "important" samples.

Spherical IS shifts the sample mean to the existing sample with minimal norm. It correctly locate the min-norm point, as shown in Figure 6.6(c), but Spherical IS only samples one failure region while leaving the other one totally untouched.

The samples drew by the proposed HSCS are plotted in Figure 6.6(d). Samples generated during presampling and min-norm points searching are not included in this Figure. While the majority of samples are centered at the min-norm points of those two failure regions, HSCS still preserves a few samples around the origin

Table 6.2: Accuracy and efficiency evalution on 70-dimensional charge pump circuit

	Monte Carlo	HDIS [WGC14]	SpIS [DQS08]	Proposed HSCS with 10 replications
failure probability	**4.904e-5**	**3.9e-3**	**8.788e-7**	3.89e-5 $\sim$ 5.88e-5 (mean **4.82e-5**)
Total #sim. runs	**1.584e7**	**3.8e4**	**>7.4e5**	4.6e3 $\sim$ 5.5e4 (mean **2.3e4**)
#sim. for presampling	-	1.1e4	4e3	4.2e3
#sim. for IS		3.8e4	>7e5	410 $\sim$ 5.1e4 (mean 1.9e4)

to keep a small ratio of the original distribution according to equation (6.4) and avoids numerical instability in likelihood ratio calculation.

6.4.2.3 Hyperspherical Clustering with 70 Process Variation Parameters

In the following discussion, we model 10 process variation parameters on 7 transistors shown in Figure 5.5 in the CP circuit, leading to a 70-dimension problem. Transistors in two digital switches are not considered.

To collect enough samples for clustering, we generate 1000 samples at each hypersphere surface and gradually increase its radius and search for the samples on the 1% quantile. Until 6 sigma hypersphere, a total of M = 144 samples are collected, including 41 failed samples captured on 5 sigma hypersphere, and 103 failed samples on 6 sigma hypersphere. The weighted spherical k-means algorithm is applied on these 144 samples to group them into clusters. Note that the actual number of clusters (k_{actual}) generated by the algorithm could be small than k_{target}, as some clusters may become empty during the cluster assignment and are removed.

To determine the optimal number of clusters, we start with different k_{target} and evaluate the value of the maximization objective (also referred as profit) under those k_{target}. As shown in Figure 6.7, there is a big jump when k_{target} increases from 1 to 2. Afterwards, the slope becomes gentler and almost flat when k_{target}

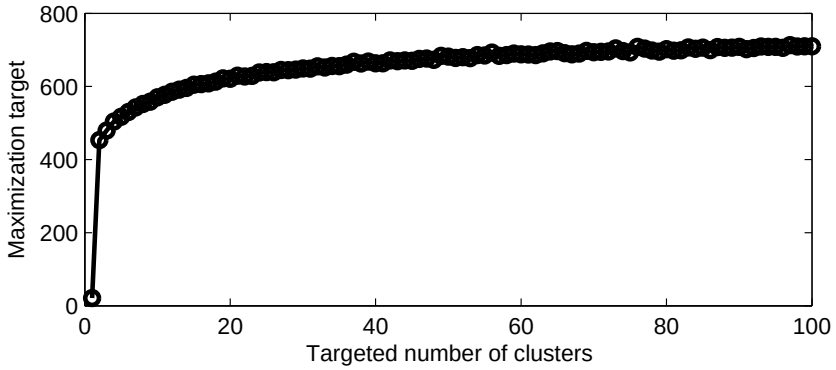


Figure 6.7: Clustering maximization objective while changing the targeted number of clusters

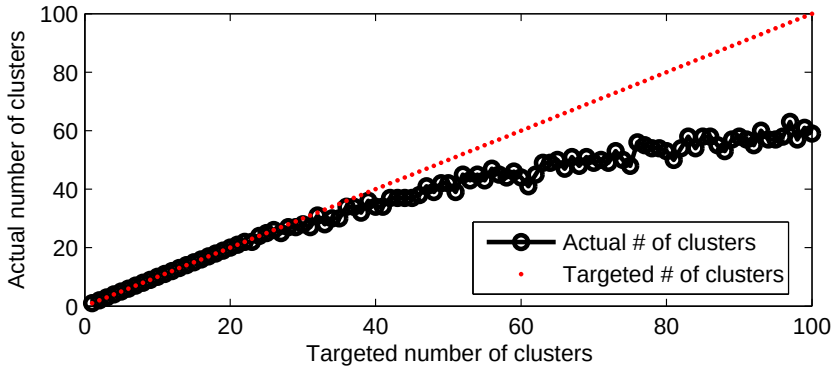


Figure 6.8: Number of actually clusters may be small than the targeted number of clusters

reaches 30.

A lot of information can be interpret from this Figure. First, the big jump indicates that the failed samples are located in two major clusters. When we use two centroids instead of one, the samples becomes much closer to the centroids, leading to a remarkable increase in the profit. Of course, these two big clusters can be further decomposed into smaller ones, but the profit generated by increasing k_{target} is smaller. When k_{target} is beyond 30, we do not benefit from increasing the cluster numbers.

The number of actually generated clusters k_{actual} is plotted against k_{target} in

Figure 6.8, which helping us understand Figure 6.7 better. When k_{target} is small, the algorithm generates whatever number of clusters we ask for. Therefore, k_{actual} is overlapped with k_{target}. However, excessively increasing k_{target} results in a lot of redundant clusters, which are not assigned any samples and removed from the targeted clusters. These redundant clusters account for the gap between k_{actual} and k_{target}. In this particular problem, any k_{target} between 2 and 30 could be reasonable. As expected, the empirical guess, $k = \sqrt{M} = 12$ falls in this range.

6.4.2.4 Accuracy, Efficiency, and Robustness

The HSCS is also compared with MC, HDIS [WGC14], and SpIS [DQS08] in terms of both efficiency and accuracy. Their convergence curves are plotted in Figure 6.9[2], including one figure for the estimated failure probability (P_{fail}) and the other one for deviation of the estimation.

To generate the groundtruth, MC takes nearly 16 million simulations to get confident estimation of P_{fail} at 4.904e-5. The HDIS converges with only 4.9e4 samples (11k samples for pre-sampling and 38k for IS), but unfortunately, to a wrong estimation as shown in Figure 6.9(a). The Spherical IS is terminated since it does not show any sign of convergence after 7.4e5 samples being simulated. The poor performance of HDIS and SpIS is not a surprise because they fail to draw samples to comprehensively cover the failure regions, hence leading to fluctuant or event deviated estimations. More quantitative results of these approaches are presented in Table 6.2. Contrasting to HDIS and SpIS, the proposed HSCS achieves very promising estimation about 2.3e4 samples. In short, it estimates P_{fail} at MC accuracy with $\sim$3 order speedup.

To ensure that HSCS can consistently generate accurate estimation, we executed the same program with 10 replications and presented their convergence

[2]Note that the convergence curves of HDIS, SpIS, and HSCS start from different points because they need different # of samples in the presampling step.

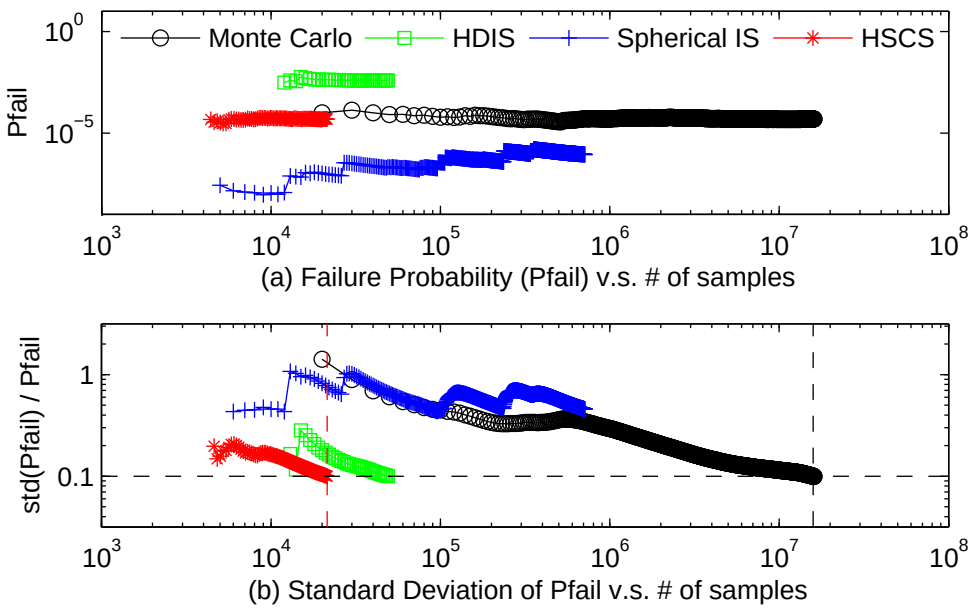


Figure 6.9: Convergence curve of Monte Carlo, HDIS, Spherical IS, and the proposed HSCS

curves in Figure 6.10. We notice that the failure probabilities estimated by these replications converge to the ground truth, the dashline in Figure 6.10(a). As detailed in Table 6.2, the estimated failure probability ranges from 3.89e-5 to 5.88e-8, with an average of 4.82e-5. This is very close to the MC result. Also, it only takes an average of 2.3e4 samples to converge the simulation, which is about 3 orders faster than MC.

6.5 Conclusion

In this chapter, HSCS is presented to tackle the challenging statistical circuit simulation problems with multiple failure regions and high dimensionality, which are the shortcomings of the existing importance sampling and classification based approaches. HSCS first applies spherical presampling and clustering to identify multiple failure regions. Next, it locates the min-norm points of each failure

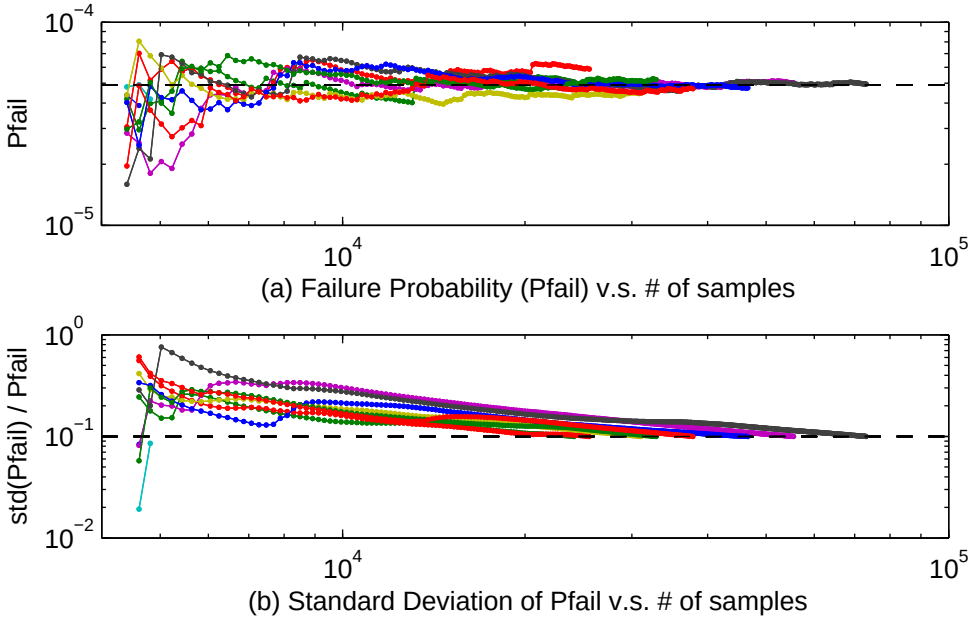


Figure 6.10: Robustness test of HSCS with 10 replications

region and leverage a modified MixIS that shifts the sample mean to those min-norm points. Therefore, the importance samples cover multiple failure regions. In the experiments on a 70-dimensional charge pump circuit, HSCS achieves $\sim$3 orders speedup over MC providing the same level of accuracy, while other IS based approaches either fail to converge or converge to wrong results. Furthermore, HSCS demonstrates excellent robustness by generating consistent results in multiple replications.

CHAPTER 7

Summary

As the process technology scales down to nanometer, analog circuits are more prone to process, voltage and temperature (PVT) variations. Stochastic circuit analysis simulates circuits while considering PVT variations. It helps circuit designers to shift the post-silicon verification to pre-silicon phase debug, which is more cost friendly and also significantly shortens the time to market. This dissertation presented the following several pieces of research related to stochastic circuit analysis.

The dissertation starts from understanding the performance of a circuit under PVT variation. We develop a maximum entropy (MaxEnt) algorithm that models the distribution of circuit performances in a very efficient way. MaxEnt builds up the circuit performance distribution by matching the probabilistic moments of only a small number of samples, and achieves 10^2 to 10^3 times speedup compared with the Monte-Carlo (MC) simulation and better accuracy than the existing moment matching approaches.

Take one step further, if the circuit under analysis uses a large number of duplicated cells or is critical to the entire system, a rare failure event needs to be considered. The rare event modeling (or yield analysis) is extremely challenging when the number of variation sources is large, i.e. the dimension is high. Several algorithms are proposed to tackle the high dimensional yield analysis problem.

First, a high-dimensional importance sampling (HDIS) approach is developed to estimate the probability of rare failure events. Compared with the conventional

important sampling (IS) approaches, the failure probability estimated by HDIS is provably bounded but the estimation from conventional IS methods are not necessarily bounded.

Second, we proposes a piecewise distribution model (PDM) that tackle this problem from a different angle. Based on MaxEnt, PDM models performance distribution as multiple segments so that it can also capture enough details on the tail of the distribution, which contains the information we need for yield analysis.

Both HDIS and PDM apply collect "important" samples via mean-shift and assume failure samples are distributed in one single failure region, which is not true in real circuits. We address this issue by two machine learning aided approaches, rare-event microscope (REscope) and Hyperspherical Clustering and Sampling (HSCS).

REscope uses a nonlinear classifier to identify multiple failure regions. To make sure the classifier works properly, we adopted the RELIEF-F algorithm to rank process variation variables as of their "importance" with respect to the performance metric under study, and to separate the failure samples from the accepted samples only according to those important variables. The REscope exhibits good accuracy on a charge pump circuit while other non-Monte-Carlo approaches fail to make the correct estimation.

REscope achieves good performance on both accuracy and efficiency, but it relies on a nonlinear classifier, which works as a black box model and is out of user's control. Excessively training the SVM to identify multiple regions could easily lead to overfit. To overcome this issue, the HSCS identifies multiple failure regions explicitly through a reweighted spherical k-means algorithm, which clusters failed samples on a set of hyperspheres, rather than the high dimensional open space. Next, a modified mixture importance sampling is designed to draw samples at those clusters to achieve multiple failure region coverage. The pro-

posed HSCS is evaluated using both mathematical and circuit-based examples. It achieves about 3-order speedup over Monte Carlo with the same level of accuracy, while other importance sampling based approaches either fail to converge or converge to wrong results. Furthermore, HSCS demonstrates excellent robustness by generating consistent results in multiple replications.

The algorithms included in this dissertation present comprehensive solutions for yield analysis, and tackle the real-life challenge such as high dimensionality and multiple failure regions. Those algorithms can be integrated in existing memory/standard cell design tools, and exhibit promising potentials in the deisgn considering pervasive PVT variations.

APPENDIX A

Network Compression for Deep Learning Inference on Mobile Platforms

The Appendix (This chapter and the next one) presents two approaches that help deploy the deep neural network on resource constraint mobile and embedded platforms, by compressing the network size and by developing specific hardware to accelerate the inference, respectively.

In the past decade, deep learning has attracted increasing interests and favored to a variety of applications due to its improved accuracy and flexibility. Deep learning, however, requires high computational cost not only in training, but also in inference because of the complex and deep pipelines. This makes its deployment to resource constraint mobile platforms a challenge. In this Appendix, we propose two network compression techniques to reduce the deep learning model complexity without losing much accuracy so that it can be easily deployed to those resource constrained environment. First, we transfer the knowledge from a well-trained complex model to a thinner and shallower model. Second, the network is further compressed by removing the inactive neurons and the redundancy in the weight parameters. On MNIST and CIFAR-10 datasets, we demonstrate 30x reduction in computational cost with negligible accuracy loss.

A.1 Introduction

Deep neural networks have set up new performance standard in a variety of applications, including visual object recognition [CMS12, KH09, SLJ15], speech recognition [MDH12], artificial intelligence games [CS14], etc. In the meantime, those areas are undergoing a transition from using small, low-quality, compressed data set to large, high resolution, high fidelity ones. To keep pace with the ever increasing data set, the deep neural networks scale up (with respect to the number of model parameters) to represent more features hidden between the data. Such large deep neural networks require high computational cost not only during the training phase, but also in the inference phase. For instance, GoogleNet [SLJ15] need about 2 GFLOPs to infer one single 227x227 image, while a high definition image might need to be inferred multiple times at different position and scale of the image. Industry's answer to such networks is computing clusters consisting of a vast amount of CPUs and high-performance GPUs [CWV14].

It can also be observed that machine learning applications are migrating towards mobile devices and embedded platforms. As examples, smart phones are increasingly operated via speech recognition, autonomous driving vehicles perform visual object recognition and react in real-time [GLU12]. In contrast to the industrial-size cluster, mobile and embedded devices are design with limited computational capacity and memory size to feature long battery life. While it is possible to train deep and sophisticated neural networks off-line on industrial-size clusters, it is difficult to deploy them on mobile device for inference [LG15]. For certain applications, acquired data can be transmitted back to the cloud and inferred by powerful clusters. However, there are also a variety of application scenarios, where internet connection is unavailable, inconvenient (the online transmission cause unpredictable delay), or insecure to transmit the data to the cloud. Under such circumstance, we need to perform inference in realtime on

resource-constraint mobile devices.

Model compression techniques are targeted to solve this problem. [BC14] demonstrates that deep neural networks can be compressed into "shallow" single-layer networks by training the shallow networks on the logistic output of the deep network using a large amount of data [BCN06]. This idea is proven and enhanced recently in [HVD15, RBK14] by adjusting the temperature parameter of the softmax layer during the training process.

Among others, [CBD14, GAG15] train neural networks with reduced bit precision, hence optimize memory usage and open the potential of design very efficient hardware dedicated for deep learning. HashedNets [CWT15] group network connections via a hash function and reduce the model size by applying the same weight value for all connections grouped in the same hash bucket. All these existing works suggest that there are plenty of redundancy within network weight parameters.

In this chapter, we present two approaches that compress the network without sacrificing much accuracy. Starting from the original sophisticated, but cumbersome, neural network (teacher network), we first use a knowledge transfer approach to convert the teacher network to a relatively shallower and thinner student network. While training the student network, we mixed the original labeled data with a large number of unlabeled ones to ensure more generality.

Without a deep pipeline of sub-sampling layers, the shallower student network has a large number of neurons on the fully-connected layer, resulting very large weight matrices at the very end of the network. To tackle this problem, we proposed two approaches to compress them. First, we analyze the activity of each neuron on that layer, and remove the less active ones. Second, we approximate the original weight matrix by an approximated one, which can be decomposed to two smaller matrices by applying singular value decomposition (SVD) and remove the dimensions with smaller singular values. Surprisingly, removing the redundancy

from neurons and weight matrices almost do not sacrifice any accuracy, but get even better generality. The compressed network still maintains very high accuracy.

We implement the proposed approaches based on Caffe [JSD14] platform and evaluate the proposed knowledge transfer and network compression approach on MNIST [LBB98a] and CIFAR-10 [KH09] datasets. On MNIST, we reduce the total computational cost to 4.4% with almost no loss in accuracy. On CIFAR-10, the knowledge transfer step introduces about 6% accuracy loss with 4x reduction (reduce to 25%) in computational cost. The following network compression step introduce another 4.2x reduction with only 0.05% loss in accuracy, which is almost negligible.

The remaining part of this chapter is organized as follows. We present the detail of knowledge transfer and network parameter reduction approaches in Section A.2. Experiment results is presented in Section A.3. And the chapter is concluded in Section A.4.

A.2 Network Compression

A.2.1 Knowledge Transfer

We use knowledge transfer as the first step of our network compression approach. It transfer the knowledge from an accurate, but cumbersome, neural network (teacher network) to a thinner and shallower student network, as illustrated in Figure A.1. The essential of knowledge transfer is that instead of directly training the student network from label, it trains the student network using the soft targets (e.g. logits, class probabilities) generated by the accurate teacher network. In this case, the student network is trained to minimize the "difference" compared with the teacher network, rather than directly minimize classification error.

During the knowledge transfer we pay close attention to the following two

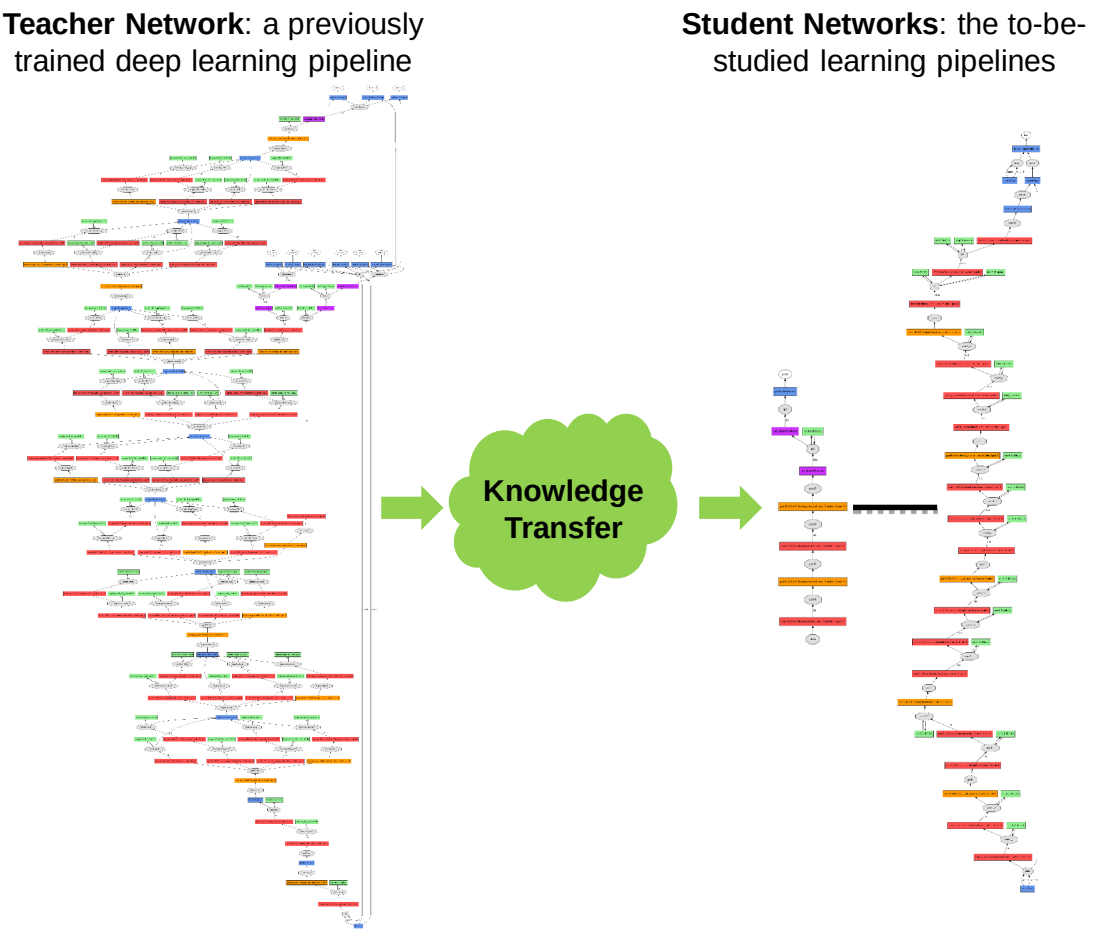


Figure A.1: Transferring the knowledge from the original cumbersome teacher neural network to thinner, shallower student networks

aspects. First, we care about the soft targets that are used to train the student network. Second, we also discussed about how to organize the training data to achieve a better student network.

A.2.1.1 Learning Target in Knowledge Transfer

Neural networks typically process the input data through a pipeline of layers and end up with a vector of logits, z_i. These logits can be converted to the class

probability p_i through a "softmax" layer as follows:

$$p_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \tag{A.1}$$

where temperature parameter T is set to 1 in most cases.

Instead of direct training the student network by minimizing the classification error, the knowledge transfer approach in [BC14] minimizes the euclidean distance between the logits of the original network and the student network. This approach is generalized to minimizing the cross entropy between the class probabilities of the original and the student network [HVD15], while minimizing the Euclidean distance between logits can be considered as a special case of minimizing the cross entropy between class probabilities. By setting the temperature at a high value, the gradient of cross-entropy between class probabilities approximates to the gradient of Euclidean distance between logits, which matches every logits of the student network to the original network. On the other hand, at low temperature, the knowledge transfer pays less attention to matching logits that are much smaller than the average. This could be helpful because logits are almost completely unconstrained by the cost function used for training the original network so they could be very noisy.

A.2.1.2 Organizing Training Data for Knowledge Transfer

In addition to the learning target, we also pay close attention to the data that are used to train the student network. As mentioned in [BCN06] and [BC14], using a large amount of unlabeled data (artificial data or authentic data without label) can be helpful for knowledge transfer because more data bring in better generality, and avoid over-fitting.

However, we discover that inserting original labeled data in between the unlabeled data could improve the test accuracy of the student network. That is because the teacher network is trained to minimize the classification error using

the original training data. Thus, the soft targets are closer to the ideal conditioning, i.e. the true class has higher logit and probability, while the false classes are associated with smaller logits and class probabilities. Compared with those unlabeled data, which have not gone through the training process, the class probabilities are softer, i.e. is less similar to the ideal binary class probability. While the softer learning target is better for generality, the sharper learning target in the original training data could enforce more similarity between the class probability of student network and ideal class probability.

In practice, we acquire a large amount of unlabeled data, and insert the original training data in the unlabeled data set. With a hybrid data set, the student network is alternatively trained by original training data and unlabeled data. And a higher test accuracy can be achieved.

A.2.2 Network Parameter Reduction

The student network trained from the original teacher network usually features a shallower pipeline of sub-sampling layers, i.e. convolutional or pooling layer with stride larger than 1. Without enough subsampling, there are usually a large number of neurons on the fully-connected layers, leading to a huge matrices at the very end of the network.

The network parameter reduction techniques are presented to reduce the number of parameters and computational cost in those fully-connected layers. We perform network parameter reduction from two aspects, by removing the inactive nodes on those layers and compressing the weight matrices respectively.

A.2.2.1 Remove Inactive Neurons

A typical fully-connected layer performs the following two steps as illustrated in Figure A.2. It first project the input neurons to a vector of output through

linear projection, which is calculated through matrix vector production (MVP). Second, each value of in the vector is passed to an activation function to model non-linearity. Typical activation functions include rectified linear unit (ReLU) [NH10], sigmoid or tanh [LBO12]. Take ReLU in equation (A.2) as example:

$$f(x) = \max(x, 0) \tag{A.2}$$

It activates a neuron only if its value is larger than 0. For a trained deep neural network, a neuron on a fully connected layer can be activated at certain input and inactive at others.

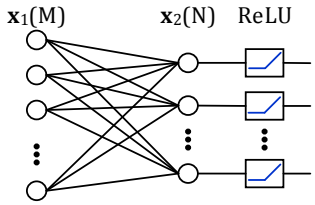


Figure A.2: A fully-connected layer followed by ReLU activation functions

In practice, we found that some neurons have very small activation rate, or even never activate for all the training data. Removing a neuron that never activates has no impact to the neural network output. Moreover, it reduces the noise while classifying unknown data, because the meaningless neurons might be activated by unknown data and introduce noise to the network.

In our implementation, we reduces neurons with small activation rate until it has notable impact on training error. Take the fulling connected layer in Figure A.2 for example, by reducing the number of neurons on $\mathbf{x}_2$ from N to N_R, forming $\mathbf{x}_{2R}$, it compresses the number of parameters in the weight matrix from MN to MN_R, and reduce the computation cost at the current layer from $2MN$to $2MN_R$.

A.2.2.2 Compress Weight Matrices

Beside checking the activation rate of a neuron on the feature map vector $\mathbf{x}_i$, we also study the parameter value in the weight matrix $\mathbf{W}$, and we found that a large number of parameters are zero or has a very small value. In other words, the weight matrix $\mathbf{W}$ is a relatively sparse matrix. Of course we can compute the MVP between feature map vector and weight matrix as sparse MVP, theoretically it does reduce the total number of floating point operations (FLOPS). However, it is typically slower because of the irregular data accessing pattern in matrix operation.

In this work, instead of calculating the MVP based on its sparse pattern, we approximate the original weight matrix $\mathbf{W}$ by $\tilde{\mathbf{W}}$ which has a lower rank and the MVP better $\mathbf{x}_i$ and $\tilde{\mathbf{W}}$ can be calculated in an easier way.

We start from taking a singular value decomposition (SVD) over the original weight matrix $\mathbf{W}$:

$$\mathbf{W} = \mathbf{U}\mathbf{S}\mathbf{V}^T \tag{A.3}$$

where $\mathbf{S}$ is a diagonal matrix consists of singular values, each column of $\mathbf{U}$ is an eigen vector of $\mathbf{W}\mathbf{W}^T$, and each row of $\mathbf{V}$ is an eigen vector of $\mathbf{W}^T\mathbf{W}$, as illustrated in Figure A.3.

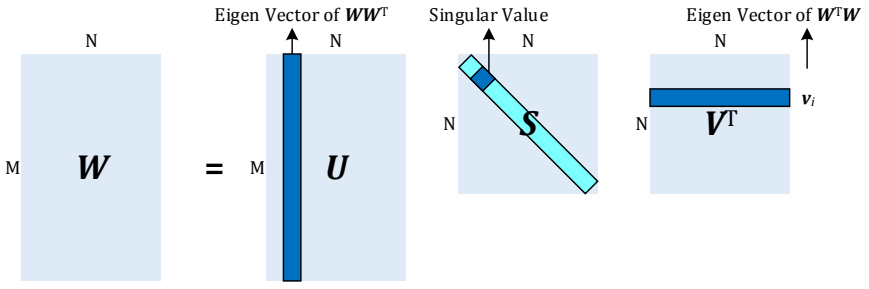


Figure A.3: Singular value decomposition (SVD)

To compress the matrix based on the SVD results, we only keep those N_R large singular values in matrix $\mathbf{S}$, only keep those eigen vectors in $\mathbf{U}$ and $\mathbf{V}$ respectively.

Therefore matrix $\mathbf{U}$ and $\mathbf{S}$ is reduced to $\mathbf{U}_R$ and $\mathbf{S}_R$ as follows:

$$\mathbf{U}_R = \mathbf{U}(:, 1 : N_R) \tag{A.4}$$

$$\mathbf{V}_R = \mathbf{V}(:, 1 : N_R) \tag{A.5}$$

$$\mathbf{S}_R = \mathbf{S}(1 : N_R, 1 : N_R) \tag{A.6}$$

After absorb $\mathbf{S}_R$ into $\mathbf{V}_R^T$ as

$$\mathbf{V'}_R^T = \mathbf{S}_R \mathbf{V}_R^T \tag{A.7}$$

The original weight matrix $\mathbf{W}$ can be approximated by $\tilde{\mathbf{W}}$, which is calculated as

$$\mathbf{W} \approx \tilde{\mathbf{W}} = \mathbf{U}_R \mathbf{V'}_R^T \tag{A.8}$$

where $\mathbf{U}_R$ is an M-by-N_R matrix and $\mathbf{V'}_R^T$ is an N_R-by-N matrix, as illustrated in Figure A.4.

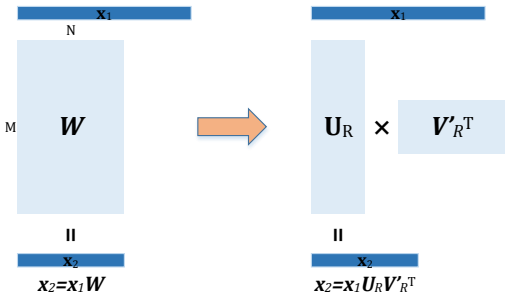


Figure A.4: Decompose the weight matrix into two smaller matrices

Another interpretation of (A.8) is to insert a linear layer before the nonlinear fully-connected layer (inner-product+ReLU). The linear layer projects the original feature map $\mathbf{x}_i$ to a smaller $\mathbf{x}'_i$ using matrix $\mathbf{S}_R$:

$$\mathbf{x}'_i = \mathbf{x}_i \mathbf{U}_R \tag{A.9}$$

The second matrix $\mathbf{V}_R^T$ and the ReLU work as the nonlinear layer afterwards. Even though the linear layer can be absorbed to the nonlinear layer by multiplying $\mathbf{U}_R$

and $\mathbf{V'}_R^T$, calculating them separately actually results in smaller computational cost and less weight parameters.

After decomposing $\mathbf{W}$, the number of weight parameter has been updated from NM to $N_R(M+N)$, and the number of computational operation is updated from $2NM$ to $2N_R(M+N)$. When $N_R \ll \min(M, N)$, both computational cost and parameter size are reduced.

A.2.3 Algorithm Flow

After the network is compressed by those aforementioned approaches, we finally go through a refinement process to improve the accuracy of the student network. Here refinement is defined as using the compressed network parameters as initial values, and training the compressed network using very small learning rate.

As the compressed network is reduced based on a trained network, the parameters should be somewhere close to the optimal. Fine tuning the parameter using a small learning rate could lead the network to optimal parameters.

Given the discussion in this section, the proposed algorithm can be summarized as the following steps:

1. Transfer the knowledge from a pre-trained network to a shallower and thinner network

2. Apply network compression methods to further reduce network parameters and computational cost

3. Refine the compressed network to improve accuracy

A.3 Experiments

In this section, we evaluate the network compression approaches on two visual object recognition datasets, MNIST [LBB98a] and CIFAR-10 [KH09]. The knowledge transfer and network parameter reduction approaches are implemented based on Caffe [JSD14] with a MATLAB interface.

A.3.1 MNIST

For MNIST dataset, we used all the 60,000 examples in the training set for training and 10,000 test examples for testing.

We use the well-known LeNet [LBB98b] as teacher network. LeNet has two convolutional with 20 and 50 kernels respectively. Each convolution layer is followed by a subsampling (pooling) layer. After convolution and subsampling, the feature map is processed by a fully-connected inner-product layer to 500 neurons, and then reduced to 10 logits (Conv20+Conv50+FC500+FC10). With a total of 431,080 weight parameters, it takes about 4.6 MFLOPS to infer one 28x28 gray-scale image. The test accuracy 0f LeNet is 99.10% over those 10000 test samples included in MNIST dateset.

A.3.1.1 Knowledge Transfer

To compress the network, we first apply knowledge transfer to train student networks, which are shallower and associated with less computational cost.

Experiment results on a shallow network with one fully-connected layer are illustrated in Figure A.5. First, the student network converges faster and ends up with a high test accuracy at 98.81% if we apply knowledge transfer. Contrastingly, directly training the same student network only achieves 98.0% test accuracy. Second, we notice that slightly high accuracy can be achieved by removing the

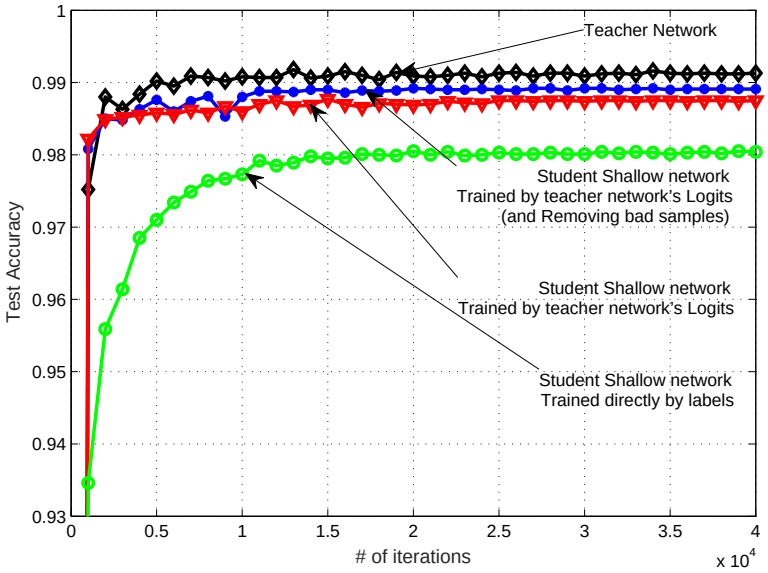


Figure A.5: Transferring the knowledge from LeNet to a shallow neural network with only one fully-connected layer

training data that the teacher network mis-classifies. That is reasonable because those mis-classified examples might confuse the student network. A teacher using only positive training examples trains a better student networks.

We also compare the accuracy of applying knowledge transfer to different student networks. Here we use shallow networks with only 1 hidden layer (fully-connected), but different number of neurons as student networks. Comparisons between student networks directly trained from labels and those trained from the soft targets (logits of teacher network) are presented in Table A.1.

We can notice that, for the same student network, higher accuracy can be achieved if it is trained towards the soft targets generated by teacher network, rather than directly trained by the labels. Even for the student network with only one hidden layer of 200 neurons, 98.67% accuracy is achieved by training via knowledge transfer. That is very close to the teacher network.

In terms of the network size and computational cost, the student network

Table A.1: Comparison between the teacher network (LeNet) and student networks in terms of accuracy, # of weight parameters, and computational cost

Network Parameters		Computational cost		# of weight params		Test Accuracy				Activation
		kFLOPS	%	x10^3	%	By label	By soft target	Compression	Refine	rate[2]
Teacher Lenet		4,631.7	100.00%	431.1	100.00%	99.10%				-%
Student Network[1]	FC500	795.5	17.18%	397.5	92.21%	98.01%	**98.81%**			0.00%
	FC400	636.4	13.74%	318.0	73.77%	97.92%	**98.86%**	98.81%	98.76%	0.01%
	FC300	477.3	10.31%	238.5	55.33%	98.01%	98.69%	98.30%	**98.76%**	11.34%
	FC200	318.2	6.87%	159.0	36.89%	98.01%	**98.67%**	90.86%	**98.64%**	19.46%
	FC150	238.7	5.15%	119.3	27.67%	97.85%	98.48%	79.17%	**98.49%**	22.76%
	FC100	159.1	3.44%	79.5	18.44%	97.82%	98.27%	67.35%	**98.37%**	27.96%
	FC80	127.3	2.75%	63.6	14.76%	97.54%	97.89%	61.15%	**98.24%**	30.56%
	FC50	79.6	1.72%	39.8	9.22%	**97.62%**	97.23%	49.02%	97.43%	35.68%

[1]: Student networks (FCxxx) are shallow networks with only one hidden-fully-connected layer, where "xxx" is the number of neurons on the fully-connected layer.

[2]: For all neurons on the fully-connected layer, only keep those whose activation frequency is higher than the number in this column.

Table A.2: Compressed the weight matrix by decompose it to the multiplication of two smaller matrices

Network Parameters		Computational cost		# of weight params		Test Accuracy			
		kFLOPS	%	x10^3	%	By label	By soft target	Compression	Refine
Teacher Lenet		4,631.7	100.00%	431.1	100.00%	99.10%			
Student FC500		795.5	17.18%	397.5	92.21%	98.01%	98.81%	-%	-%
Student Network[1]	L400_FC500	1,039.5	22.44%	519.5	120.51%	98.03%	98.80%	**98.81%**	**98.81%**
	L300_FC500	782.5	16.90%	391.0	90.70%	97.92%	98.78%	98.80%	**98.83%**
	L250_FC500	654.0	14.12%	326.8	75.80%	97.98%	98.79%	98.80%	**98.81%**
	L200_FC500	525.5	11.35%	262.5	60.90%	97.97%	98.81%	98.77%	**98.82%**
	L150_FC500	397.0	8.57%	198.3	45.99%	97.86%	98.79%	98.78%	**98.83%**
	L100_FC500	268.5	5.80%	134.0	31.09%	97.89%	98.80%	98.80%	**98.86%**
	L80_FC500	217.1	4.69%	108.3	25.13%	97.88%	98.79%	98.79%	**98.88%**
	L50_FC500	140.0	3.02%	69.8	16.18%	97.90%	98.72%	98.59%	**98.83%**

[1]: Student networks (Lxxx-FC500) consist of two linear fully-connected layer, and a ReLU layer after the second fully-connected layer. "xxx" stands for the number of neurons on the linear layer.

"FC200" reduces the computational cost to only 6.87% and # of weight parameters to only 36.89% while still providing an acceptable accuracy at 98.49%.

A.3.1.2 Network Compression by Removing Inactive Neurons

Instead of knowledge transfer, we can also get small shallow networks by compressing the FC500 network, which has a relatively larger fully-connected layer. We approach the network compression by two methods mentioned in Section A.2.2.

To quantitatively analyze how often a neuron is activated on the hidden layer of FC500 network, we include the activation rate at the last column of Table A.1. The activation rate here is defined as how often a neuron is activated given all 60,000 training samples. And the proposed network reduction approach only keeps the neurons whose activation rate is higher than the value in the table. For instance, FC400 keeps the neurons with activation rate higher than 0.01%. In other words, it removes all neurons that activate at less than 0.01% of inputs.

An interesting observation on this column is that there are a large portion of neurons that are very inactive. For instance, those 100 neurons in the compressed FC400 network are almost always inactive, which explains that by the accuracy almost remains the sames after removing those 100 neurons. In fact, after looking at the data, we found that 71 neurons on the fully connected layer of FC500 are always inactive (activation rate = 0). In other words, removing those neurons has no impact on the test accuracy, moreover, it improves the generality of the neural network. If we go ahead and remove the inactive nodes aggressively, the accuracy starts to drop.

Another work we did is to refine the compressed network by setting the compressed weight parameters as initial state and training the network with small learning rate. The test accuracy after refine step is also included in Table A.1. For each network, we compare the test accuracy achieved by different training approaches on the same network and highlight the one with highest accuracy. We can notice that refining the compressed network can achieve the highest accuracy at most time.

A.3.1.3 Weight Matrix Compression

Analysis on the activation rate indicates that there are a large portion of redundancy residing in the network.

Another ways to removing the redundancy is through the SVD decomposition. Instead of reducing the weight matrix to a smaller one, the SVD converts the original matrix to the multiplication between two smaller matrices. The results of weight matrix compression via SVD are presented in Table A.2.

We can observe that when a linear layer with 400 nodes is added before the FC500 layer, the number of weight parameters actually increases. And it starts to decrease when we continue reduce the number of nodes on the linear layer. Even when we keep only 50 nodes on the linear layer, it still maintains a very high accuracy at 98.59%. With this accuracy, it has compress the original LeNet to only 3.02% of its original computational cost, and 16.18% of its original number of parameters.

We also compare the accuracy of training the network directly verse compression and a refining step. As presented in Table A.2, the entire column of the refine option are highlighted, indicating that it always leads to the highest accuracy by compressing the weight matrix and followed by refining the compressed network with small learning rate.

On the MNIST dataset, the combination of knowledge transfer, network compression, refinement can 33.2x reduction (reduce to 3.02%) on computation cost almost without loss in accuracy.

A.3.2 CIFAR-10

We also test the same approach on a relatively larger data set, the CIFAR-10, with 50,000 training images, and 10,000 test images. Each image is 32x32 with 3 color channels (RGB). Besides the original CIFAR-10 dataset, we also use a large

number of unlabeled data from the Tiny Images Dataset [TFF08] to enhance the knowledge transfer. In particular, we uses the first 1.5 million unlabeled examples from the Tiny Images Dataset.

We build a network similar to Network In Network (NIN) [LCY13] as the teacher network. The teacher network has 3 convolutional layers with 128 kernels at each layer, followed by a dropout layer and a fully-connected layer reducing the feature map to 10 logits. The teacher network is trained to reach 84.97% test accuracy.

A.3.2.1 Knowledge Transfer

The samples from the Tiny Images Dataset is similar to CIFAR-10, but it is still not following the exactly same style. The student network can be biased if the training data set is dominated by the unlabeled data from the Tiny Images Dataset. In our implementation, we reorganize the training data by inserting original training data in between each bunch of unlabeled data from the Tiny Images Dataset. In this case, after the student network is trained by unlabeled data for a while to improve the generality, it will be retrained by the original data to enforce the student network to learn the knowledge from CIFAR-10 dataset. In short, the training data structure ensures good accuracy on CIFAR-10 without losing generality.

The comparisons between the same student network trained directly using labels and using the probability generated by teacher networks are illustrated in Figure A.6.

Unlike MNIST, it is difficult to train a decent shallow network for CIFAR-10 directly using the labels. [BC14, DB13, ERF13] add a base convolution and pooling layer to study different deep architectures. Especially in [BC14], it use a convolutional and a pooling layer followed by fully-connected layer with a large

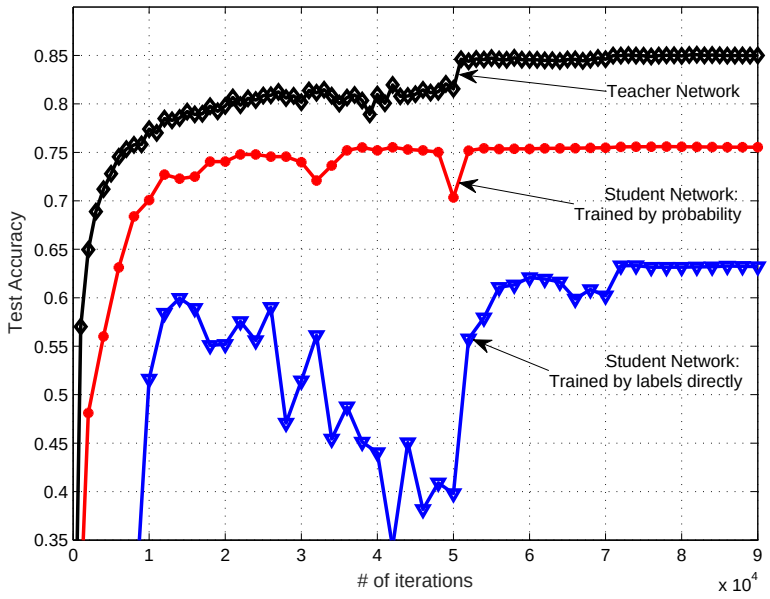


Figure A.6: Transferring the knowledge from deep convolutional neural network to a shallow neural network

number of non-linear units (ReLU layer), so as to benefit from the feature extracted from convolution, but also keeping the network simple and shallow. In our implementation, we follow the idea in [BC14] and build shallow networks with one convolutional and pooling layers as student network. Particularly, we configure 32 kernels in the convolutional layer, e.g. student network Conv32_FC4000 in Figure A.6 has a convolutional and pooling layer in the front, followed by a fully-connected layers with 4000 nodes.

As shown in Figure A.6, We can only reach about 63.76% accuracy by training Conv32_FC4000 using the original label. However, if we train the same student using soft targets (class probabilities), the accuracy can be improved to 78.54%.

A quantitative analysis with more student networks at different sizes is presented in Table A.3. Similar to MNIST results, knowledge transfer (column "by soft target" in the table) results in decent accuracy on the student networks. One thing to notice is that the student network are somehow "larger" than the teacher

Table A.3: Knowledge transfer to compression by removing inactive nodes for CIFAR-10 dataset

Network Parameters		Computational cost		# of weight params		Test Accuracy				Activation
		MFLOPS	%	x10^6	%	By label	By soft target	Compression	Refine	rate[2]
Teacher Network		282.44	100.00%	0.85	100.00%	84.97%				-%
Student Network[1]	Conv32_FC4000	70.66	25.02%	32.81	3862.00%	63.76%	78.54%			0.00%
	Conv32_FC3600	64.10	22.69%	29.53	3475.83%	62.38%	78.05%	78.50%	**79.11%**	14.39%
	Conv32_FC3200	57.54	20.37%	26.25	3089.66%	59.40%	78.27%	78.32%	**78.87%**	21.51%
	Conv32_FC2800	50.97	18.05%	22.97	2703.49%	63.38%	78.44%	77.62%	**79.02%**	25.91%
	Conv32_FC2400	44.41	15.72%	19.69	2317.32%	62.90%	78.62%	75.29%	**78.95%**	29.16%
	Conv32_FC2000	37.85	13.40%	16.41	1931.15%	67.28%	78.47%	72.00%	**78.79%**	31.88%
	Conv32_FC1600	31.28	11.08%	13.13	1544.97%	61.88%	78.21%	67.29%	**78.48%**	34.39%
	Conv32_FC1200	24.72	8.75%	9.85	1158.80%	54.89%	78.09%	60.29%	**78.38%**	37.16%
	Conv32_FC800	18.16	6.43%	6.56	772.63%	63.44%	**78.02%**	50.50%	77.72%	40.21%

[1]: Student networks (Conv32_FCxxx) denotes shallow networks with one convolutional layer (32 kernels) followed by one fully-connected layer, where "xxx" is the number of neurons on the fully-connected layer.

[2]: For all neurons on the fully-connected layer, only keep those whose activation frequency is higher than the number in this column.

Table A.4: Compressed the weight matrix by SVD decomposition for CIFAR-10 dataset

Network Parameters		Computational cost		# of weight params		Test Accuracy		
		MFLOPS	%	x10^6	%	By soft target	Compression	Refine
Teacher Network		282.44	100.00%	0.85	100.00%	84.97%		
Student Conv32_FC4000		70.66	25.02%	32.81	3862.00%	78.54%	-%	-%
Student Network[1]	Conv32_L3000_FC4000	78.28	27.72%	36.63	4310.53%	77.74%	78.56%	**78.98%**
	Conv32_L2000_FC4000	53.89	19.08%	24.43	2875.51%	77.81%	78.54%	**78.84%**
	Conv32_L1000_FC4000	29.50	10.45%	12.24	1440.49%	77.27%	78.60%	**78.92%**
	Conv32_L800_FC4000	24.63	8.72%	9.80	1153.48%	77.03%	78.52%	**79.03%**
	Conv32_L400_FC4000	14.87	5.27%	4.92	579.47%	76.23%	78.42%	**79.05%**
	Conv32_L200_FC4000	10.00	3.54%	2.49	292.47%	75.56%	78.31%	**78.82%**
	Conv32_L150_FC4000	8.78	3.11%	1.88	220.72%	73.89%	78.13%	**78.78%**
	Conv32_L100_FC4000	7.56	2.68%	1.27	148.97%	74.08%	77.39%	**78.55%**
	Conv32_L80_FC4000	7.07	2.50%	1.02	120.27%	73.45%	74.11%	**78.06%**

[1]: Student networks (Conv32_Lxxxx_FC4000) denotes a network with 3 layers, a convolutiona layer with 32 kernels, a linear layer with xxxx nodes and a fully-connected nonlinear layer with 4000 nodes. (pooling, normalization layers is not considered)

network in terms of the number of weight parameters, but if the computational cost has been reduced considerably, e.g. inferring the Conv32_FC1600 takes only

11.08% of the computational cost compared with the original teacher network.

A.3.2.2 Network Compression

The results of compressing the network by removing inactive nodes and decompose the weight matrix are presented in Table A.3 and Table A.4 respectively.

For each student network, the training approach resulting the highest accuracy is highlighted in these tables. We can notice that refining the network after network compression leads to the highest accuracy for almost all the student networks.

Another thing is that the compressed network are actually larger than the original teacher network in terms of the parameter size. The parameters in the student network Conv32_FC4000 is actually 38.6x larger than the convolutional teacher network. That is because the convolutional neural network features parameter sharing and local connectivity. And the pooling layer sub-samples the feature map, and further reduces the data size. Those characteristics reduce the number of parameters in the convolutional neural network. Instead, the student network puts a fully connected layer right after only one convolutional layer, which has a relatively large feature map, resulting a large weight matrix represent the full connection between the feature map and the neurons on the hidden layer.

Even though the weight parameter size is not reduced, the computational cost is compressed drastically. For instance, through weight matrix decomposition, we can reduce the network to Conv32_L80_FC4000, which has only 2.5% computational cost compared with the original network, but still maintains 78.06% accuracy.

A.3.3 Discussions

A.3.3.1 Why Refinement Helps?

In the experiments, we notice that the most accuracy student networks are almost always generated by refining the compressed network model. Especially for Table A.3 and Table A.4, the refined networks dominate the student networks in terms of accuracy.

It is not surprising that the refined student network can be more accurate than the same compressed network without refinement. As we removed the redundancy in by reducing the network parameters, the compressed parameters in the resulting network could be close to the optimal, but are not necessary at the optimal. A refinement with a very small learning rate gradually tiles the weight parameters towards the optimal. It is important to use a small learning rate for refinement, otherwise, the network parameters may jump around the optimal but can hardly approach it.

It is also reasonable that refinement can be more accurate compared with training the student network directly, no matter using the original label or the soft targets of the teacher network. Directly training the network is an optimization problem starting from randomly initialized parameters. However, refinement is an optimization process starting from a good initial guess that is close to the optimal, (even without refinement, the compressed networks already have a high accuracy). Of course, the refinement can generate a better accuracy than directly training.

A.3.3.2 Which Compression Approach is More Effective?

It is also interesting to study which compression approach is more effective, removing inactive nodes, or compressing the weight matrix.

By comparing the experiment data in Table A.3 and Table A.4, we can conclude that compressing the weight matrix via SVD is better than removing the inactive nodes. For instance, at the same accuracy level of 78.5%, Table A.3 only compress the network to Conv32_FC1600, which still takes 11.08% of computational cost and 15.45x weight parameters of the original teacher network. Contrastingly, the student network Conv32_L100_FC4000 in Table A.4 has only 2.68% FLOPS and 1.49x weight parameters compared with the teacher network, but also achieves 78.55% accuracy.

Similar empirical conclusion can be made on MNIST by comparing the results in Table A.1 and Table A.2.

A.4 Conclusion

Deep neural network achieves very good accuracy at the cost of high computational complexity and larger parameter size. However, it is difficult to deploy an accurate but cumbersome network model on resource-constrained mobile devices. Such dilemma motivates network compression.

In this Chapter, we present a flow of multiple approaches that compress the deep neural networks in terms of both computational cost and parameter size. We first transfer the knowledge from a well-trained complex model to a thinner and shallower model. Second, the network is further compressed by removing the inactive neurons and the redundancy in the weight parameters. Third, the compressed network is then refined to achieve an accuracy that is closed to the original teacher network. Experiments on MNIST and CIFAR-10 datasets demonstrate more than 30x reduction in computational cost with negligible accuracy loss.

APPENDIX B

FPGA Based for Deep Convolutional Neural Network

In the past decade, deep convolutional neural networks (CNN) have set new performance standards in several high-impact applications. On the other hand, deep CNNs gain breakthrough in accuracy at the price of high computational cost, which motivates FPGA based acceleration. However, most existing FPGA accelerated CNNs use external DRAM heavily due to the limited on-chip memory size, which suppresses performance. Moreover, accelerators are usually designed exclusively for certain network layers, preventing the computing resource from being shared between layers. In this chapter, we reformulate the convolution computation by flattening it to large-scale matrix multiplication between feature maps and convolution kernels, which can be computed as inner product (IP). With this formulation, the accelerators cross all layers can be unified to enhance resource sharing. The proposed design benefits from the uniform accelerators in two folds. First, it maximizes utilization of computing resources because accelerators are fully used at every layer. Second, the proposed design process one layer at a time, requiring only data in current layer. Thus, it minimizes the access to off-chip DRAM by buffering the required data on-chip. As a result, the performance of the proposed design only depends on the computational resources and is almost not constrained by DRAM bandwidth. Experimental results indicate that the proposed design not only outperforms existing designs in terms of performance (119.32 GFLOPS), it also achieves very high performance density (66.29

MFLOPS/DSP Slice).

B.1 Introduction

Deep convolutional neural networks (CNNs) have exhibited promising accuracy in a variety of applications, such as visual object recognition [SLJ15], speech recognition [MDH12], etc. However, it gains breakthrough in accuracy at the price of high computational cost, which motivates accelerations by computing clusters [DCM12], GPUs [CWV14], and FPGAs [ORK15].

It has also been observed that applications of deep CNN have shifted towards mobile and embedded devices. As examples, smart phones are increasingly operated by speech recognition [Sch10], autonomous driving cars perform visual object recognition in real-time [GLU12]. In contrast to GPUs or CPU clusters, such devices are designed for lower power and long battery life. Under these scenarios, FPGA accelerated CNNs stand out due to the low power consumption, high flexibility and computational capability of FPGA.

The key to a high performance FPGA design is to make full use of on-chip resources, including both computing resources, i.e. DSP slides, and enormous on-chip memory bandwidth. Several FPGA based deep CNNs are proposed in the past few years [FPH09, CMB10, CSJ10, PSM13, ZLS15]. Among these designs, [FPH09] uses FPGA as a vectorial arithmetic unit, and implements CNN mainly on a 32bit soft processor for flexibility. Works in [ORK15, CMB10, CSJ10] design specific accelerator for each layer, while accelerators cannot be shared between layers[1].

Using specific accelerators for each layer actually creates a dilemma between maximizing computational capability and memory bandwidth. To keep all the

[1]Work in [ZLS15] designs uniform accelerator for convolutional layers, but the accelerator cannot be used for the fully-connected layer.

accelerators busy (optimize computational capability), accelerators have to be designed as a pipeline, which requires intermediate data for each accelerator. Due to the data size, they are typically stored in off-chip memory, whose bandwidth will become the bottleneck when multiple accelerators access them simultaneously.

This dilemma motivates the proposed FPGA architecture which speeds up CNN using uniform accelerators. Benefiting from uniform accelerators, we can process one layer at a time, but using all the computing resource. Moreover, the data required for one layer can be easily buffered in on-chip memory with massive bandwidth, which optimizes the memory access.

Naturally, convolutions with different kernel size or stride cannot be implemented by the same accelerator. To tackle this problem, we reformulate the convolution to two separate steps, 1) convolution flattening (CF) to flatten the feature map and kernel from high-dimensional matrices to big 2-D matrices, 2) inner product (IP) between each rows and columns of the flattened feature map and kernel matrices. In our design, we use separate CF accelerators at each layer as they only involve light-weight memory manipulation, while uniform IP accelerators are shared cross all layers, even including the fully connected (FC) layers (matrix-vector production operations). Experimental results indicate that the proposed design achieves 119.32 GFLOPS, which is 1.84x over the state-of-the-art work. Meanwhile, our implementation achieves a high performance density of 66.29 MFLOPS/DSP.

Moreover, our design flow is semi-automated. The CF and IP accelerators with given parameters can be automatically synthesized, which make it easier to adopt the proposed architecture to new CNNs and new FPGA devices.

The remaining part of this chapter is organized as follows: Section II introduces the background of CNN. Section III presents the reformulation of convolution and the design of uniform accelerators. Section IV describes the proposed architecture and implementation details. Section V presents the experimental results and

comparison with related works. Section VI concludes this chapter.

B.2 Basic CNN Architecture

B.2.1 Preliminary of Convolutional Neural Network

The area of Artificial Neural Networks (ANN) was originally inspired by the goal of modeling biological neural systems and has achieved good results in classification [RDS15]. Regular ANN has an input vector and several hidden layers, where each layer has a set of neurons fully connected to the neurons in previous layer, resulting a large parameters matrix.

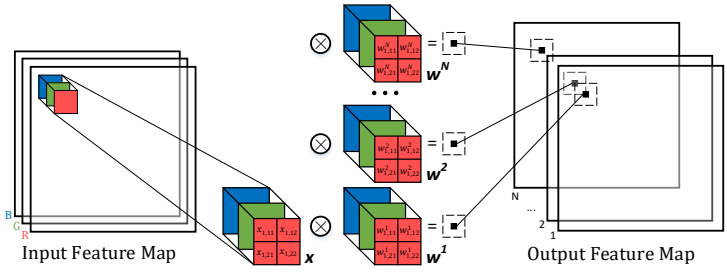


Figure B.1: Convolutional Layer

Convolutional neural network (CNN) is proposed to reduce the number of parameters in regular ANN by featuring local connectivity and parameter sharing. As illustrated in Fig. B.1, first, each neuron at convolutional layer is only connected to a local region of the input instead of the entire input feature map. Second, all neurons in a single depth slice are using the same weight vector. Those two features are reasonable for high dimensional data, such as images, where one pixel is only relevant to the pixels nearby.

B.2.2 Network Architecture of a Real-Life CNN

A typical CNN mainly consist of 3 types of layers: convolutional layer, pooling layer and fully-connected (FC) layer. The architecture of a real-life CNN, AlexNet [KSH12], is illustrated in Fig. B.2. It is a winning top-5 classifier in ILSVRC 2012 [RDS15], which categorizes the objects from high-resolution images into 1000 different classes.

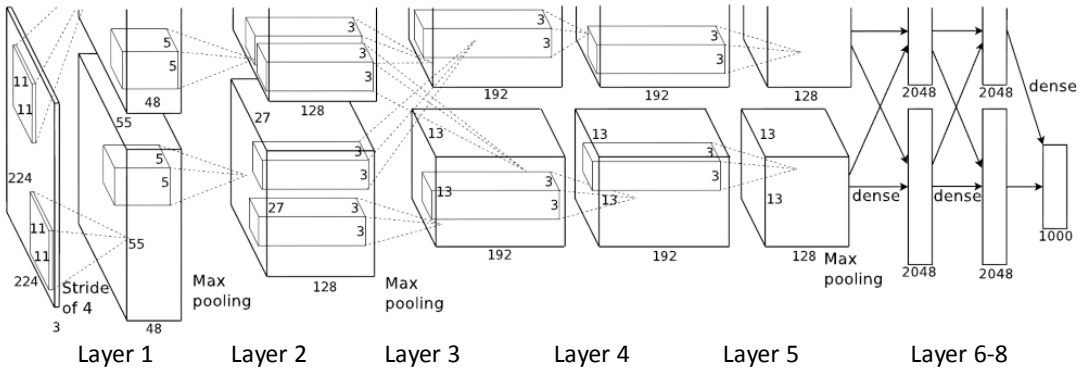


Figure B.2: An illustration of the architecture of AlexNet [KSH12]

As shown in Fig. B.2, AlexNet consists of 5 convolutional layers and 3 FC layers. Other layers, such as rectifier and normalization layers, are not included. Using 227x227x3 image as input, the first convolutional layer produces 55x55x96 output feature map, which is then compressed to 27x27x96 by a max pooling layer. After going through 5 convolutional layers and 3 pooling layers, the feature map at pool5 is reduced to 6x6x256. This feature map is then processed by 3 FC layers to generate an 1x1x1000 output, which can be used to infer the object class in the original image.

A detailed breakdown of the CNN parameters at each layer is presented in Table B.1. It is obvious that convolutional layers and fully-connected layers account for the majority (99%) of computational load.

Using specific accelerators for each layer actually creates a dilemma between

Table B.1: A detail breakdown of the network parameter and computational load in each layer of AlexNet [KSH12]

Layer Name		Ker. size / Stride	Output Size	F_I	F_O	# of P. (x10^6)	Mega FLOPS	Percentage
L1	conv1	11x11/4	55x55x 96	3	96	0.03	210.83	14.54%
	pool1	3x3/2	27x27x 96	96	96		0.63	0.04%
L2	conv2	5x5/1	27x27x256	48	256	0.31	447.90	30.89%
	pool2	3x3/2	13x13x256	256	256		0.39	0.03%
L3	conv3	3x3/1	13x13x384	256	384	0.88	299.04	20.62%
L4	conv4	3x3/1	13x13x384	192	384	0.66	224.28	15.47%
L5	conv5	3x3/1	13x13x256	192	256	0.44	149.52	10.31%
	pool5	3x3/2	6x 6x256	256	256		0.08	0.01%
L6	fc6		1x1x4096			37.75	75.50	5.21%
L7	fc7		1x1x4096			16.78	33.55	2.31%
L8	fc8		1x1x1000			4.10	8.19	0.57%
Total						60.95	1449.92	100.00%

F_I/F_O: Number of input/output feature maps

# of P.: Number of parameters

maximizing computational capability and memory bandwidth, which motivates the design of uniform accelerators.

B.3 Uniform Accelerator for Convolutional and Fully-Connected Layer

Naturally, convolutions with different kernel sizes or strides cannot be accelerated by the same accelerators. In this section, we reformulate the convolution operation and design uniform accelerators to speed up convolution at different layers.

B.3.1 Reformulation of the Convolution Operation

As shown in Fig. B.3, the convolution between a local region of feature map (x) and a kernel (w^i) is essentially inner-product (IP) when they are flattened to vectors xf and wf^i.

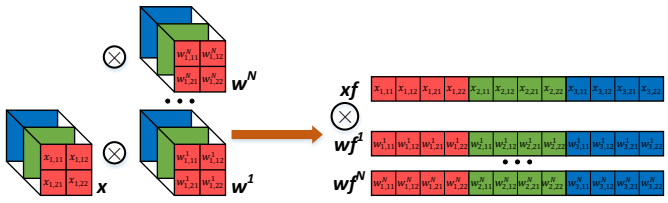


Figure B.3: Convolution Flattening

Note that wf^i can be pre-calculated since the kernels remain unchanged for different inputs, but xf needs to be generated in real-time when input changes. It is acceptable to flatten feature maps at different layer by different accelerators because it only involves memory manipulations, which can be handled in FPGA at low cost.

On the other hand, the acceleration of IP is very expensive as it involves all the floating-point operations. At different layers, the lengths of flattened vectors xf and wf^i are typically different. However, the flattened vectors xf and wf^i

are typically very long, e.g. for layer 2, the length is 2400 (5x5x96). Given this characteristic, we can break the long vectors into shorter uniform segments, and design uniform accelerators to calculate inner product for each segment.

B.3.2 Design of Convolution-Flattening and Inner-Product Accelerators

According to the reformulation of convolution operation, each convolutional layer has its own light-weight CF accelerators, while all layers share the same uniform IP accelerators, which take care of all the floating-point operations.

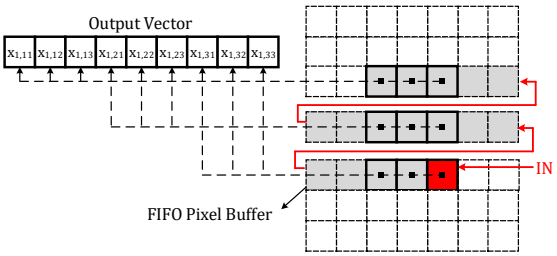


Figure B.4: Convolution-Flattening (CF) Accelerator

B.3.2.1 Convolution-Flattening Accelerator

Even though the CF operation only accounts for memory manipulation, it is non-trivial to design a CF accelerator. The key of designing an efficient CF accelerator is to reuse the data when the convolutional filter (kernel) slides on the feature map.

As shown in Fig. B.4, the CF accelerator is designed mainly using a set of shift registers. At each clock cycle, the shift register takes in 1 pixel in the 7x7 feature map, and output a 1x9 vector on the left hand side (flattened from the data in the 3x3 window). To generate a vector that is long enough for the inner product, a convolutional layer may consist of multiple CF accelerators.

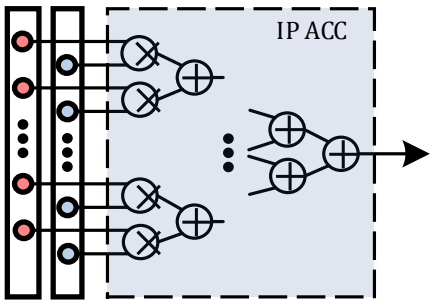


Figure B.5: Inner-Product (IP) Accelerator

B.3.2.2 Inner-Product Accelerator

The design of IP accelerator is relatively straightforward. As shown in Fig. B.5, it uses a set of multipliers to calculate the production between each element of two input vectors. The multiplier outputs are summed up to a single result using an adder tree. This IP accelerator is fully streamable, which can process two input vectors at each clock cycle.

The key of achieving a high computational capability is to keep the computing resource (DSPs) always busy. In this design, all convolutional layers shares the same IP accelerators. At each layer, the CF accelerators continuously stream data to the IP accelerator, resulting a high computational capability.

Moreover, we developed tools to automatically generate the CF and IP accelerators with given parameters such as kernel size, stride, IP vector length. Such tools facilitate the adoption of this architecture to other CNNs or FPGA devices.

B.4 FPGA based CNN with Uniform Accelerator

B.4.1 Overall Architecture

The overall architecture of the proposed FPGA based CNN is illustrated in Fig. B.6. Clearly, the architecture can be partitioned into 3 parts mainly consists of

logic, DSPs and on-chip memory blocks, respectively.

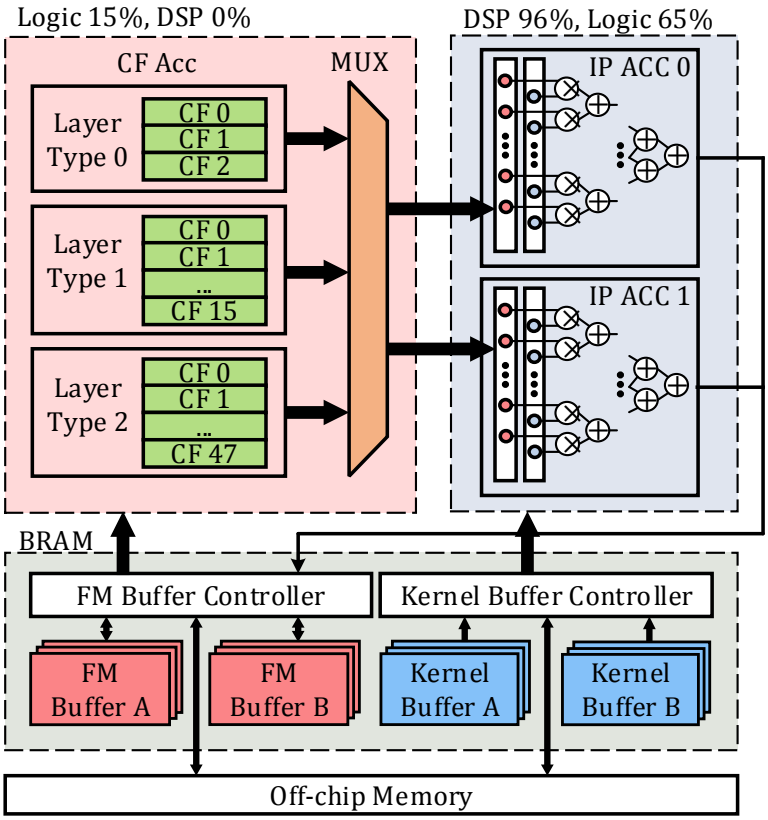


Figure B.6: Overall architecture

In this architecture, we use two page-flip buffers to store the feature map. While one FM buffer stores the input of the CF accelerators, the other one accepts the output from IP accelerators. Those buffers are implemented as distributed BRAMs on FPGA, allowing multiple 32-bit numbers to be loaded in one cycle. Moreover, once the original image is loaded from off-chip DRAM, the feature map data always stay in the on-chip memory, which saves off-chip memory bandwidth drastically.

We also use two buffers to load kernel data from off-chip memory. While using the kernel data in one buffer, the controller loads the kernel data of the next layer to the other buffer. We can achieve very high off-chip memory bandwidth because only blockwise reading operation is involved for kernel loading.

Table B.2: Resource consumption of convolution-flattening accelerators at each layer

Convolution Flatten Accelerator					Layer Latency			
Size	LUT	FF	N	Len	#	F_I	F_O	latency
11x11	5.63%	2.75%	3	363	1	3	96	145,307
5x5	0.034%	0.044%	16	400	2	96	256	559,979
3x3	0.016%	0.017%	48	432	3	256	384	194,795
					4	192	384	129,899
					5	192	256	86,635

N: Number of CF accelerator instances

Len: Length of the output vector at this layer

The logic part consists of CF accelerators that flattens the input feature maps. Note that we use only three types of CF accelerators since layer 3-5 actually share the same type of CF accelerators. Those CF accelerators are packed into layer blocks, which shares the same IP accelerators on the right hand side.

This design minimizes the off-chip memory access by keeping feature map data always on chip, and only load kernel data. It also shares uniform IP accelerators across layers, which keeps the DSP always busy and maximizes the performance.

B.4.2 Accelerator allocation

For a given FPGA device, it is straightforward to determine the number and size of the IP accelerators based on available DSPs. The CF accelerator consumes very little resource, thus allocation of accelerators is actually a matter of optimizing the usage of IP accelerator, rather than minimizing the resource consumption of CF accelerators.

As a concrete example, the allocation of CF accelerators for each layer in AlexNet is presented in table B.2. In this particular example, the kernel vector at the first layer is at most 363 (11x11x3), thus using any IP accelerator longer than

363 is a waste of resource. In this example, we instantiate two IP accelerators, while each of them calculates the IP between two 432x1 vectors. The IP size is chosen as 432 to best fit the kernel size at Layer 3-5, i.e. using 48 CF accelerators, 432 (3x3x48) 32-bit number can be generated and streamed to each of the IP accelerator at every clock cycle.

B.4.3 Automation and Scalability

The proposed architecture involves a semi-automation flow of synthesizing an FPGA design once CNN parameters are given. By far, we have developed the tools that automatically generate the CF and IP accelerators, and pack the CF accelerators to a layer. The ongoing work is to synthesize the controllers which interface between the layers, memory and IP blocks.

It is also convenient to applied the proposed architecture to larger network, or implemented on more advanced FPGA. Since we only store the kernel of the current and the next layer on chip, larger CNN can be easily fit in this architecture, let alone one important trend of FPGA is 3D stacking large amount of memory to FPGA logic [LM13]. When more advanced FPGAs are available, we can just use more CF accelerators and larger IP accelerators, which are automatically generated by tools.

B.5 Experiments

B.5.1 Experiment Setup

The proposed design is synthesized and implemented with Vivado (v2015.4) while targeting to a Xilinx Ultrascale XCVU190 FPGA. The kernel data are extracted from Caffe [JSD14] and stored in off-chip DRAM. We also use Caffe to validate intermediate results and final results calculated from FPGA.

B.5.2 Experimental Results

B.5.2.1 Resource Utilization

The proposed design is implemented with timing constraint configured at 100MHz. The resource utilization after implementation is presented in Table B.3. DSP resources are very heavily used, i.e. 96.33% of the DSP resource are used in this design.

Table B.3: Overall Resource Utilization

	CLB LUTs	CLB FFs	BRAMs	DSPs
used	839576	1093926	3003	1734
available	1074240	2148480	3780	1800
Percentage	78.16%	50.92%	79.44%	96.33%

B.5.2.2 Performance

We also calculated the throughput of the design. At the current stage, only the layers before pool5 in Table B.1 are implemented, which accounts for a total of 1.333 GFLOP.

The proposed design takes a total of 1,116,896 clock cycles to finish one round of process. Therefore the throughput can be calculated as

$$\text{Throughput} = \frac{F_{clk}}{N_{cycle}} \times N_{FLOP} = 119.32 \text{ GFLOPS}, \tag{B.1}$$

which is substantially higher than the state-of-the-art [CSJ10, ZLS15] as presented in Table B.4.

B.5.2.3 Performance Density

An design can achieve different performances given different FPGA devices. To account in the impact of devices, we evaluate both performance and performance

density, i.e. the performance at unit computing resource (FLOPS/DSP).

Given sufficient logic resources, our design can make efficient use of DSPs by enlarging the inner product accelerator. Thus, throughput is mainly bounded by available DSPs. Comparison between our design and various existing designs of FPGA based CNN accelerator is shown in Table B.4. High level synthesis is used in [ZLS15] to unroll loops and results in 22.01 MFLOPS/DSP density. Our inner product accelerators make use of more than 96% of available DSPs and all dedicated DSPs are kept busy running in 99% of clock cycles when processing images. This high utilization of DSPs significantly improves performance density to 66.29 MFLOPS/DSP.

Table B.4: Performance Comparison

	ISCA2010[CSJ10]	FPGA2015[ZLS15]	Our Design
Precision	48bit fixed	32bit float	32bit float
FPGA	Virtex5 SX240T	Virtex7 VX485T	Virtex Ultrascale XCVU190
Throughput (GOPS)	16	61.62	119.32
Available DSPs	1056	2800	1800
Utilized DSPs	N/A	2240	1734
Density1 (MOPS/DSP)	15.15	22.01	66.29

B.6 Conclusion

CNN achieves very good accuracy at the cost of high computational complexity, which motivates FPGA based acceleration. This chapter proposed an FPGA based CNN using uniform Inner-Product (IP) accelerators shared across multiple convolutional layers. The uniform IP accelerator creates two benefits. First, the throughput is maximized because the uniform IP accelerators (DSPs) can be fully utilized while processing each layer. Second, it minimizes the off-chip memory access. Instead of accessing the kernel data from all the layers, we only need to

process one layer at a time to make use of all the DSP. This make us easier to buffer the data on chip since we only use the data from only layer. We also developed a semi-automation flow that enable automatic synthesis of the accelerators and layer blocks. Experimental results shown that the proposed design achieves 119.32 GFLOPS throughput which is 1.94x over the state-of-the-art implementation [ZLS15]. Meanwhile, our implementation achieves a high performance density of 66.29 MFLOPS/DSP.

References

[AB03] S.K. Au and J.L. Beck. "Important sampling in high dimensions." *Structural Safety*, **25**(2):139 – 163, 2003.

[AN06] Kanak Agarwal and Sani Nassif. "Statistical analysis of SRAM cell stability." In *Proceedings of the 43rd annual Design Automation Conference*, pp. 57–62. ACM, 2006.

[AOC99] E. Acar, A. Odabasioglu, M. Celik, and L.T. Pileggi. "S2P: a stable 2-pole RC delay and coupling noise metric [IC interconnects]." In *Ninth Great Lakes Symposium on VLSI*, pp. 60 –63, mar 1999.

[BB05] T. Bengtsson B. Li and P. Bickel. "Curse-of-dimensionality revisited: Collapse of importance sampling in very high-dimensional systems." *Technical Report No.696,Department of Statistics, UC-Berkeley*, 2005.

[BC14] Jimmy Ba and Rich Caruana. "Do Deep Nets Really Need to be Deep?" In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 2654–2662, 2014.

[BCN06] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. "Model compression." In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pp. 535–541, 2006.

[BDM02] K.A. Bowman, S.G. Duvall, and J.D. Meindl. "Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration." *Solid-State Circuits, IEEE Journal of*, **37**(2):183–190, Feb 2002.

[CB01] G. Casella and R.L. Berger. *Statistical inference*. Duxbury Press, 2001.

[CBD14] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. "Low precision arithmetic for deep learning." *CoRR*, **abs/1412.7024**, 2014.

[CCS04] R. Chang, Yu Cao, and Costas J. Spanos. "Modeling the electrical effects of metal dishing due to CMP for on-chip interconnect optimization." *Electron Devices, IEEE Transactions on*, **51**(10):1577–1583, Oct 2004.

[CDH06] Ke Cao, Sorin Dobre, and Jiang Hu. "Standard cell characterization considering lithography induced variations." In *Proceedings of the 43rd annual Design Automation Conference*, pp. 801–804. ACM, 2006.

[CHZ10] B. Chen, J. Hu, and Y. Zhu. "Computing Maximum Entropy Densities: A Hybrid Approach." *Signal Processing: An International Journal (SPIJ)*, **4**(2):114, 2010.

[CMB10] Srihari Cadambi, Abhinandan Majumdar, Michela Becchi, Srimat Chakradhar, and Hans Peter Graf. "A programmable parallel accelerator for learning and classification." In *PACT*, pp. 273–284. ACM, 2010.

[CMS12] Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. "Multi-column deep neural networks for image classification." In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3642–3649. IEEE, 2012.

[CN94] E. Chiprout and M.S. Nakhla. *Asymptotic waveform evaluation and moment matching for interconnect analysis*. Kluwer Academic Publishers, 1994.

[Con13] Keith Conrad. "Probability distributions and maximum entropy." *retrieved November*, **14**:2013, 2013.

[Cou13] Rachel Courtland. "3-D transistors for all." *Spectrum, IEEE*, **50**(1):11–12, 2013.

[CS14] Christopher Clark and Amos Storkey. "Teaching deep convolutional neural networks to play go." *arXiv preprint arXiv:1412.3409*, 2014.

[CSJ10] Srimat Chakradhar, Murugan Sankaradas, Venkata Jakkula, and Srihari Cadambi. "A dynamically configurable coprocessor for convolutional neural networks." In *ACM SIGARCH Computer Arch. News*, volume 38, pp. 247–257, 2010.

[CWT15] Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. "Compressing Neural Networks with the Hashing Trick." *CoRR*, **abs/1504.04788**, 2015.

[CWV14] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. "cuDNN: Efficient Primitives for Deep Learning." *CoRR*, **abs/1410.0759**, 2014.

[CWW11] Xiaoming Chen, Wei Wu, Yu Wang, Hao Yu, and Huazhong Yang. "An EScheduler-Based Data Dependence Analysis and Task Scheduling for Parallel Circuit Simulation." *IEEE Trans. Circuits Syst. II, Exp. Briefs*, **58**(10):702 –706, oct. 2011.

[DB13] Yann N Dauphin and Yoshua Bengio. "Big neural networks waste capacity." *arXiv preprint arXiv:1301.3583*, 2013.

[DCM12] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. "Large scale distributed deep networks." In *Advance in Neural Info. Processing Systems*, pp. 1223–1231, 2012.

[DL11] Changdao Dong and Xin Li. "Efficient SRAM failure rate prediction via Gibbs sampling." In *Proceedings of the 48th DAC*, 2011.

[DQS08] Lara Dolecek, Masood Qazi, Devavrat Shah, and Anantha Chandrakasan. "Breaking the simulation barrier: SRAM evaluation through norm minimization." In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '08, pp. 322–329, 2008.

[Dur10] Rick Durrett. *Probability: theory and examples*, volume 3. Cambridge university press, 2010.

[EBS97] M. Eisele, J. Berthold, D. Schmitt-Landsiedel, and R. Mahnkopf. "The impact of intra-die device parameter variations on path delays and on the design for yield of low voltage digital circuits." *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, **5**(4):360–368, Dec 1997.

[Elm48] WC Elmore. "The transient response of damped linear networks with particular regard to wideband amplifiers." *Journal of applied physics*, **19**(1):55–63, 1948.

[ERF13] David Eigen, Jason Rolfe, Rob Fergus, and Yann LeCun. "Understanding deep architectures using a recursive convolutional network." *arXiv preprint arXiv:1312.1847*, 2013.

[FF95] P. Feldmann and R.W. Freund. "Efficient linear circuit analysis by Padé approximation via the Lanczos process." *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, **14**(5):639–649, 1995.

[FPH09] Clément Farabet, Cyril Poulet, Jefferson Y Han, and Yann LeCun. "Cnp: An fpga-based processor for convolutional networks." In *FPL*, pp. 32–37. IEEE, 2009.

[GAG15] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. "Deep Learning with Limited Numerical Precision." *arXiv preprint arXiv:1502.02551*, 2015.

[GBD12] Fang Gong, Sina Basir-Kazeruni, Lara Dolecek, and Lei He. "A fast estimation of SRAM failure rate using probability collectives." In *Proceedings of the 2012 ACM international symposium on International Symposium on Physical Design*. ACM, 2012.

[GJM96] A. Golan, G.G. Judge, and D. Miller. *Maximum entropy econometrics: robust estimation with limited data*. Series in financial economics and quantitative analysis. Wiley, 1996.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? the kitti vision benchmark suite." In *Computer Vision and Pattern Recognition (CVPR)*, pp. 3354–3361. IEEE, 2012.

[GYH11] F. Gong, H. Yu, and L. He. "Stochastic Analog Circuit Behavior Modeling by Point Estimation Method." In *Proceedings of the 2011 international symposium on Physical design*, pp. 175–182. ACM, 2011.

[HHC11] Tsung-Ching Huang, Jiun-Lang Huang, and Kwang-Ting Cheng. "Robust Circuit Design for Flexible Electronics." *IEEE Design and Test of Computers*, **28**(6):8–15, 2011.

[Hos85] JRM Hosking. "Algorithm AS 215: Maximum-likelihood estimation of the parameters of the generalized extreme-value distribution." *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, **34**(3):301–310, 1985.

[HVD15] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. "Distilling the Knowledge in a Neural Network." *CoRR*, **abs/1503.02531**, 2015.

[HW79] John A Hartigan and Manchek A Wong. "Algorithm AS 136: A k-means clustering algorithm." *Applied statistics*, pp. 100–108, 1979.

[HW87] Jonathan RM Hosking and James R Wallis. "Parameter and quantile estimation for the generalized Pareto distribution." *Technometrics*, **29**(3):339–349, 1987.

[HWW85] JRM Hosking, James R Wallis, and Eric F Wood. "Estimation of the generalized extreme-value distribution by the method of probability-weighted moments." *Technometrics*, **27**(3):251–261, 1985.

[Jay57] E.T. Jaynes. "Information theory and statistical mechanics." *Physical review*, **106**(4):620, 1957.

[JL89] C. Jacoboni and P. Lugli. *The Monte Carlo method for semiconductor device simulation*. Springer Verlag, 1989.

[JSD14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. "Caffe: Convolutional Architecture for Fast Feature Embedding." *arXiv preprint arXiv:1408.5093*, 2014.

[KAC12] Kelin J Kuhn, Uygar Avci, Annalisa Cappellani, Martin D Giles, Michael Haverty, Seiyon Kim, Roza Kotlyar, Sasikanth Manipatruni, Dmitri Nikonov, Chytra Pawashe, et al. "The ultimate CMOS device and beyond." In *Electron Devices Meeting (IEDM), 2012 IEEE International*, pp. 8–1. IEEE, 2012.

[KAH12] Himanshu Kaul, Mark Anders, Steven Hsu, Amit Agarwal, Ram Krishnamurthy, and Shekhar Borkar. "Near-threshold voltage (NTV) design: opportunities and challenges." In *Proceedings of the 49th Annual Design Automation Conference*, pp. 1153–1158. ACM, 2012.

[Ken48] Maurice George Kendall. "Rank correlation methods." 1948.

[KH09] Alex Krizhevsky and Geoffrey Hinton. "Learning multiple layers of features from tiny images.", 2009.

[KHT10] Kentaro Katayama, Shiho Hagiwara, Hiroshi Tsutsui, Hiroyuki Ochi, and Takashi Sato. "Sequential Importance Sampling for Low-Probability and High-Dimensional SRAM Yield Analysis." In *IEEE/ACM International Conference on Computer-Aided Design*, 2010.

[KJL12] Rouwaida Kanj, Rajiv Joshi, Zhuo Li, Jerry Hayes, and Sani Nassif. "Yield estimation via multi-cones." In *Proceedings of the 49th DAC*, 2012.

[KJN06] Rouwaida Kanj, Rajiv Joshi, and Sani Nassif. "Mixture importance sampling and its application to the analysis of SRAM designs in the presence of rare failure events." In *in Proceedings of the 43rd annual Design Automation Conference*, pp. 69–72, 2006.

[KLI93] GEORGE J. KLIR. *A review of: ?NTROPY OPTIMIZATION PRINCIPLES WITH APPLICATIONS? by J. N. Ka-pur and H. K. Kesavan. Academic Press, San Diego, 1992. xix + 408 pages.*, volume 21. 1993.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[KSR97] Igor Kononenko, Edvard Šimec, and Marko Robnik-Šikonja. "Overcoming the myopia of inductive learning algorithms with RELIEFF." *Applied Intelligence*, **7**(1):39–55, 1997.

[KWB16] Rahul Krishnan, Wei Wu, Srinivas Bodapati, and Lei He. "Accurate Multi-segment Probability Density Estimation Through Moment Matching." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (Minor Revision)*, 2016.

[KWG13] Rahul Krishnan, Wei Wu, Fang Gong, and Lei He. "Stochastic behavioral modeling of analog/mixed-signal circuits by maximizing entropy." In *ISQED*, pp. 572–579, 2013.

[LBB98a] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, **86**(11):2278–2324, 1998.

[LBB98b] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, **86**(11):2278–2324, 1998.

[LBO12] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. "Efficient backprop." In *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

[LCY13] Min Lin, Qiang Chen, and Shuicheng Yan. "Network in network." *arXiv preprint arXiv:1312.4400*, 2013.

[LG15] Nicholas D. Lane and Petko Georgiev. "Can Deep Learning Revolutionize Mobile Sensing?" In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, HotMobile '15, pp. 117–122, New York, NY, USA, 2015. ACM.

[LLG04] Xin Li, Jiayong Le, P. Gopalakrishnan, and L. T. Pileggi. "Asymptotic probability extraction for non-normal distributions of circuit performance." pp. 2–9, 2004.

[LLG07] Xin Li, Jiayong Le, Padmini Gopalakrishnan, and Lawrence T Pileggi. "Asymptotic probability extraction for nonnormal performance distributions." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **26**(1):16–37, 2007.

[LM13] Steve Leibson and Nick Mehta. "Xilinx UltraScale: The Next-Generation Architecture for Your Next-Generation Architecture." *Xilinx White Paper WP435*, 2013.

[LYL09] Kyu Won Lee, SM Yoon, SC Lee, W Lee, IM Kim, Cheol Eui Lee, and DH Kim. "Secondary electron generation in electron-beam-irradiated solids: Resolution limits to nanolithography." *J Korean Phys Soc*, **55**:1720–1723, 2009.

[MAL14] Parijat Mukherjee, Chirayu S Amin, and Peng Li. "Approximate property checking of mixed-signal circuits." In *Proceedings of the 51st DAC*, 2014.

[MDH12] Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton. "Acoustic modeling using deep belief networks." *Audio, Speech, and Language Processing, IEEE Transactions on*, **20**(1):14–22, 2012.

[MKB79] Kantilal Varichand Mardia, John T Kent, and John M Bibby. *Multivariate analysis*. Academic press, 1979.

[ML14] Parijat Mukherjee and Peng Li. "Leveraging pre-silicon data to diagnose out-of-specification failures in mixed-signal circuits." In *Proceedings of the 51st DAC*, 2014.

[MP84] L.R. Mead and N. Papanicolaou. "Maximum entropy in the problem of moments." *Journal of Mathematical Physics*, **25**:2404, 1984.

[Nas01] S. Nassif. "Modeling and analysis of manufacturing variations." pp. 223–228, 2001.

[NH10] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines." In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, 2010.

[ORK15] Kalin Ovtcharov, Olatunji Ruwase, Joo-Young Kim, Jeremy Fowers, Karin Strauss, and Eric S Chung. "Accelerating deep convolutional neural networks using specialized hardware." *Microsoft Research Whitepaper*, **2**, 2015.

[OWH96] Alan V. Oppenheim, Alan S. Willsky, and S. Hamid. "Signals and Systems." *Prentice Hall*, 1996.

[PP01] A. Papoulis and S. Pillai. "Probability, Random Variables and Stochastic Processes." *McGraw-Hill*, 2001.

[PR90] L.T. Pillage and R.A. Rohrer. "Asymptotic waveform evaluation for timing analysis." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **9**(4):352 –366, apr 1990.

[PS08] Andrei Pavlov and Manoj Sachdev. "CMOS SRAM Circuit Design and Parametric Test in Nano-Scaled Technologies: Process-Aware SRAM Design and Test." *Springer Publisher*, 2008.

[PSM13] Maurice Peemen, Arnaud Setio, Bart Mesman, Henk Corporaal, et al. "Memory-centric accelerator design for convolutional neural networks." In *ICCD*, pp. 13–19. IEEE, 2013.

[QTD10] M. Qazi, M. Tikekar, L. Dolecek, D. Shah, and A. Chandrakasan. "Loop flattening and spherical sampling: Highly efficient model reduction techniques for SRAM yield analysis." In *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, pp. 801 –806, 2010.

[RBK14] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. "FitNets: Hints for Thin Deep Nets." *CoRR*, **abs/1412.6550**, 2014.

[RDS15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge." *IJCV*, **115**(3):211–252, 2015.

[RG09] R. Y. Rubinstein and P. W. Glynn. "How to deal with the curse of dimensionality of likelihood ratios in monte carlo simulation." *Stochastic Models*, **25**:547 – 568, 2009.

[Sch10] Mike Schuster. "Speech recognition for mobile devices at Google." In *PRICAI 2010: Trends in Artificial Intelligence*, pp. 8–10. Springer, 2010.

[SL14] Shupeng Sun and Xin Li. "Fast statistical analysis of rare circuit failure events via subset simulation in high-dimensional variation space." In *Computer-Aided Design (ICCAD), 2014 IEEE/ACM International Conference on*, pp. 324–331. IEEE, 2014.

[SLJ15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions." In *CVPR 2015*, 2015.

[SLL13] Shupeng Sun, Xin Li, Hongzhou Liu, Kangsheng Luo, and Ben Gu. "Fast statistical analysis of rare circuit failure events via scaled-sigma sampling for high-dimensional variation space." In *Proceedings of the International Conference on Computer-Aided Design*, pp. 478–485. IEEE Press, 2013.

[SR08] Amith Singhee and Rob A Rutenbar. "Statistical blockade: a novel method for very fast Monte Carlo simulation of rare circuit events, and its application." In *Design, Automation, and Test in Europe*, pp. 235–251, 2008.

[SR09] Amith Singhee and Rob A Rutenbar. "Statistical blockade: very fast statistical simulation and modeling of rare circuit events and its application to memory design." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **28**(8):1176–1189, 2009.

[SWC08] Amith Singhee, Jiajing Wang, Benton H Calhoun, and Rob A Rutenbar. "Recursive Statistical Blockade: an enhanced technique for rare

event simulation with application to SRAM circuit design." In *21st International Conference on VLSI Design*, pp. 131–136. IEEE, 2008.

[SYC12] Saurabh Sinha, Greg Yeric, Vikas Chandra, Brian Cline, and Yu Cao. "Exploring sub-20nm FinFET design with predictive technology models." In *Proceedings of the 49th DAC*, 2012.

[TFF08] Antonio Torralba, Rob Fergus, and William T Freeman. "80 million tiny images: A large data set for nonparametric object and scene recognition." *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **30**(11):1958–1970, 2008.

[VS83] J. Vlach and K. Singhal. *Computer methods for circuit analysis and design*. Springer, 1983.

[VWG06] S. Vrudhula, J. M. Wang, and P. Ghanta. "Hermite polynomial based interconnect analysis in the presence of process variations." pp. 2001–2011, 2006.

[WBH16] Wei Wu, Srinivas Bodapati, and Lei He. "Hyperspherical Clustering and Sampling for Rare Event Analysis with Multiple Failure Region Coverage." In *Proceedings of the 2016 Symposium on International Symposium on Physical Design (to appear)*. ACM, 2016.

[WGC14] Wei Wu, Fang Gong, Gengsheng Chen, and Lei He. "A Fast and Provably Bounded Failure Analysis of Memory Circuits in High Dimensions." In *19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2014.

[WGK13] Wei Wu, Fang Gong, R. Krishnan, Lei He, and Hao Yu. "Exploiting Parallelism by Data Dependency Elimination: A Case Study of Circuit Simulation Algorithms." *Design Test, IEEE*, **30**(1):26–35, Feb 2013.

[WSC11] Wei Wu, Yi Shan, Xiaoming Chen, Yu Wang, and Huazhong Yang. "FPGA Accelerated Parallel Sparse Matrix Factorization for Circuit Simulations." In Andreas Koch, Ram Krishnamurthy, John McAllister, Roger Woods, and Tarek El-Ghazawi, editors, *Reconfigurable Computing: Architectures, Tools and Applications*, volume 6578 of *Lecture Notes in Computer Science*, pp. 302–315. Springer Berlin / Heidelberg, 2011.

[Wu03] X. Wu. "Calculation of maximum entropy densities with application to income distribution." *Journal of Econometrics*, **115**(2):347–354, 2003.

[WXK14] Wei Wu, Wenyao Xu, Rahul Krishnan, Yen-Lung Chen, and Lei He. "REscope: High-dimensional statistical circuit simulation towards full failure region coverage." In *Proceedings of the 51st Annual Design Automation Conference*, pp. 1–6. ACM, 2014.

[ZLS15] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. "Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks." In *FPGA*, pp. 161–170. ACM, 2015.