ALGORITHMS FOR VERIFICATION OF ANALOG AND MIXED-SIGNAL INTEGRATED CIRCUITS

A Dissertation

by

HONGHUANG LIN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Peng Li
Committee Members,	Gwan Choi
	Samuel Palermo
	Duncan M. Walker
Head of Department,	Miroslav Begovic

August 2016

Major Subject: Computer Engineering

Copyright 2016 Honghuang Lin

ABSTRACT

Over the past few decades, the tremendous growth in the complexity of *analog and mixed-signal* (AMS) systems has posed great challenges to AMS verification, resulting in a rapidly growing verification gap. Existing formal methods provide appealing completeness and reliability, yet they suffer from their limited efficiency and scalability. Data oriented machine learning based methods offer efficient and scalable solutions but do not guarantee completeness or full coverage. Additionally, the trend towards shorter time to market for AMS chips urges the development of efficient verification algorithms to accelerate with the joint design and testing phases.

This dissertation envisions a hierarchical and hybrid AMS verification framework by consolidating assorted algorithms to embrace efficiency, scalability and completeness in a statistical sense. Leveraging diverse advantages from various verification techniques, this dissertation develops algorithms in different categories.

In the context of formal methods, this dissertation proposes a generic and comprehensive model abstraction paradigm to model AMS content with a unifying analog representation. Moreover, an algorithm is proposed to parallelize reachability analysis by decomposing AMS systems into subsystems with lower complexity, and dividing the circuit's reachable state space exploration, which is formulated as a satisfiability problem, into subproblems with a reduced number of constraints. The proposed modeling method and the hierarchical parallelization enhance the efficiency and scalability of reachability analysis for AMS verification.

On the subject of learning based method, the dissertation proposes to convert the verification problem into a binary classification problem solved using *support vector machine* (SVM) based learning algorithms. To reduce the need of simulations for

training sample collection, an active learning strategy based on probabilistic version space reduction is proposed to perform adaptive sampling. An expansion of the active learning strategy for the purpose of conservative prediction is leveraged to minimize the occurrence of false negatives.

Moreover, another learning based method is proposed to characterize AMS systems with a sparse Bayesian learning regression model. An implicit feature weighting mechanism based on the kernel method is embedded in the Bayesian learning model for concurrent quantification of influence of circuit parameters on the targeted specification, which can be efficiently solved in an iterative method similar to the *expectation maximization* (EM) algorithm. Besides, the achieved sparse parameter weighting offers favorable assistance to design analysis and test optimization.

To my parents.

ACKNOWLEDGEMENTS

This dissertation is based upon work supported by the National Science Foundation under Grant No. 1117660 and by the Semiconductor Research Corporation (SRC) through Texas Analog Center of Excellence at the University of Texas at Dallas (Task IDs: 1836.128 and 1836.058).

Over the past five years, my advisor, Professor Peng Li, has been continuously guiding and supporting me throughout my work towards the PhD degree and my life in the Texas A&M University. I would like to express my profound gratitude to him not only for his enlightening advice to my research, but also for his patient tutoring of being a researcher. His insightful perspective and comprehensive expertise are always beneficial to my research, and his everlasting enthusiasm, ambition and diligence constantly inspire me to keep pursuing higher goals.

Besides, I would like to thank Professor Duncan M. Walker, Professor Gwan Choi and Professor Samuel Palermo for serving on my PhD committee and offering valuable suggestions on my preliminary exam. I am also grateful to my mentors Asad Khan, Ravi Makam and Zhipeng Ye from the Texas Instruments Inc. for exposing me to the industrial practice of advanced AMS verification techniques.

In our research group, graduate students are moving forward at their own pace, yet the interaction and collaboration are persistently exciting and inspiring. Dr. Boyuan Yan and Dr. Leyi Yin gave me helpful advice and encouragement when I was a fresh PhD student. Besides, Dr. Leyi Yin, Dr. Suming Lai and Dr. Parijat Mukherjee provide me motivating discussions and technical support with their broad expertise, some of which initiate a strong momentum of my research and directly contribute to my work. I would also like to thank Dr. Yong Zhang, Dr. Tong

Xu, Qian Wang, Jimmy Jin and Jingwei Xu for informative discussions on a variety of research topics that broaden my academic horizons and enhance my professional skills.

Last but not the least, I would like to give my very special thanks to my parents for their eternal love, absolute support and deep-rooted faith in me. I am fearless to face any challenge because I believe in the staunch backing from my family. I would like to dedicate this dissertation to my parents, with my sincere wishes for their great health and happiness.

TABLE OF CONTENTS

Page

ABSTRACT ii

DEDICATION iv

ACKNOWLEDGEMENTS v

TABLE OF CONTENTS vii

LIST OF FIGURES x

LIST OF TABLES xv

1. INTRODUCTION 1

1.1 Analog and Mixed-Signal (AMS) Verification 1
1.2 Hierarchical and Hybrid AMS Verification Challenges 5
1.3 Dissertation Contributions 7
1.4 Dissertation Organization 10

2. MODELING OF AMS SYSTEMS AND VERIFICATION PROBLEMS . . 12

2.1 AMS Systems 12
2.2 Existing Formal AMS Verification Techniques 14
2.2.1 Proof Based Symbolic Methods 14
2.2.2 Equivalence Checking 15
2.2.3 Model Checking 16
2.3 Existing Simulation Based AMS Verification Techniques 18
2.3.1 Assertion Based Verification 19
2.3.2 Learning Based Verification 20

3. PARALLEL HIERARCHICAL REACHABILITY ANALYSIS 22

3.1 Introduction 23
3.2 Model Abstraction 25
3.2.1 DIA Verification Challenges 26
3.2.2 Digital Abstraction with Analog Variables 28
3.2.3 Error Interval Estimation via KRR 31

3.2.4 State Space Representation with Support Functions 33
3.3 Preliminary Approach Towards Efficient Reachability Analysis 34
3.3.1 Simulation-assisted SMT Method with Support Function Representations 35
3.3.2 System Partition 37
3.3.3 Analysis of the Linear Subsystem 39
3.3.4 Analysis of the Nonlinear Subsystem 42
3.3.5 Hybrid Reachability Analysis 43
3.4 Proposed System Decomposition 44
3.4.1 Overapproximation Error 44
3.4.2 Concepts of Extended State Spaces 46
3.4.3 Support Function Representations of Reachable Extended State Spaces and Their Manipulation 48
3.5 Proposed Parallel Algorithm 50
3.5.1 Subsystem-level Parallelism 51
3.5.2 SAT-level Parallelism 52
3.6 Experimental Results 53
3.6.1 PWM DC-DC Converter 53
3.6.2 PLL 56
3.7 Summary 62

4. ACTIVE LEARNING GUIDED SUPPORT VECTOR MACHINE BASED CIRCUIT VERIFICATION 63

4.1 Introduction 63
4.2 SVM 66
4.3 Active Learning Guided SVM 71
4.3.1 Version Space 71
4.3.2 Proposed Accelerated Active Learning with Probabilistic Version Space 76
4.3.3 Error Bound 80
4.4 Asymmetric Active Learning 82
4.4.1 Cost Asymmetric SVM 83
4.4.2 Safety Level Evaluation 84
4.4.3 Asymmetric Active Learning and Performance Optimization . 85
4.4.4 Dynamic Model Selection 86
4.5 Experimental Results 90
4.5.1 LC Based DC/DC Converter Design Classification 90
4.5.2 PLL Verification 94
4.5.3 Prediction of Peak Chip Temperature Using a Limited Number of On-chip Thermal Sensors 99
4.6 Summary 103

5. STATISTICAL AMS VERIFICATION AND PARAMETRIC ANALYSIS VIA BAYESIAN LEARNING . . . 104

5.1 Introduction . . . 105
5.1.1 Challenges from the Complex AMS Nature . . . 105
5.1.2 Challenges from the Feature Weighting Problem . . . 106
5.1.3 Overview . . . 108
5.2 Relevance Kernel Machines . . . 109
5.2.1 Kernel Methods . . . 109
5.2.2 Feature Kernel Weighting . . . 111
5.2.3 Relevance Kernel Machine . . . 116
5.3 Sparse Relevance Kernel Machine for AMS Characterization . . . 118
5.3.1 Relevance Vector Machine . . . 119
5.3.2 Sparse Relevance Kernel Machine . . . 122
5.3.3 Efficient Training Algorithm . . . 126
5.4 Experiments . . . 130
5.4.1 Variability Analysis of an LDO . . . 132
5.4.2 PLL BIST Scheme Optimization . . . 142
5.5 Summary . . . 148

6. CONCLUSION AND FUTURE WORK . . . 149

6.1 Conclusion . . . 149
6.2 Future Work . . . 151
6.2.1 Semi-Formal Verification with Statistical Models . . . 152
6.2.2 Road Map Towards Hierarchical and Hybrid AMS Verification 153

REFERENCES . . . 157

LIST OF FIGURES

FIGURE Page

1.1 AMS design to production flow. 2

1.2 Overview of the two categories of AMS verification techniques. 3

1.3 Hierarchical AMS verification. 5

1.4 Improvements achieved by the three proposed algorithms. 8

3.1 Block diagram for a DI-PLL. 26

3.2 Time-to-digital converter proposed in [115]. 27

3.3 Second order IIR filter. 28

3.4 Error interval prediction via KRR. 33

3.5 Support function representation. 35

3.6 System partition of DI-PLL. 38

3.7 Approximate support function value of arbitrary direction l_x. 41

3.8 Over-approximation of manifest system partition. 45

3.9 Reachability analysis on extended decomposed state spaces for a 2D system. 48

3.10 The proposed reachability analysis flow with two-level parallelism. . . 51

3.11 Parallel counterexample exploration. 53

3.12 Simplified model of a DC-DC converter. 54

3.13 Transient reachability analysis: (a) simulation trajectories and the bounds of v; (b) initial state space; (c) reachable space evolution in the first $1\mu s$; (d) reachable space evolution in the last $1\mu s$.[83] 55

3.14 The analog behavioral model of PLL. 56

3.15 Speedup in solving a single SAT problem: (a) system decomposition; (b) SAT parallelism. 58

3.16 Speedup of the entire flow for various numbers of threads [83]. 59

3.17 Convergence of the reachability analysis on $\Delta\phi(t)$ and the PLL lock-time verification. 60

3.18 Convergence of the reachability analysis on $V_1(t)$. 61

3.19 Convergence of the reachability analysis on $V_2(t)$. 61

4.1 Active-learning guided SVM. 65

4.2 (a) Hard margin SVM. Circles and squares are instances of the two classes. $f(x) = 0$ is the separating hyperplane; (b) Differences between soft margin and hard margin SVM. The dot line is the hyperplane of hard margin SVM with margin m'. The solid line is the hyperplane of soft margin SVM with larger margin m. The solid circle and square violate the margin m with slack variable ξ_a and ξ_b respectively. 68

4.3 Version Space in 2-dimensional W Space: (a) x_1,x_2 and x_3 are cutting hyperplane associated with training data. x_1 and x_2 are support vectors. the solid arc is the version space. (b) $\hat{w}^*$ and version space between x_1 and x_2 correspond to hard margin SVM. In soft margin SVM, hyperplane of x_2 is shift to x_2' with a vertical offset of $\xi_2/\|w\|$. the version space is enlarged by the arc between x_2 and x_3. $\hat{w}^{*\prime}$ corresponds to the optimal classifier of soft margin SVM. x_3 becomes the support vector as well. 74

4.4 A simplified model of an LC-based DC/DC converter. 91

4.5 The sampling procedure of active learning in the parameter space $L : [1nH, 1uH] \times C : [10nF, 10uF]$ (red star denotes positive instances, blue circle denotes negative instances, square denotes initial training samples): (a) 20 samples selected; (b) 50 samples selected; (c) 100 samples selected; (d) 200 samples selected; (e) 10,000 random instances 93

4.6 Comparison in the required number of simulations (AL short for Active Learning, RS short for Random Sampling) where the global accuracy is defined as the fraction of the correct predictions on 10K random test samples. (e.g. 0.1 means 10%) 94

4.7 Block diagram of PLL 95

4.8 Timing diagram of the phase detector and the resulting output of the charge pump and loop filter. 96

4.9 Results of PLL experiments (AL short for active learning, RS short for random sampling) where the global accuracy and the error on positive class are defined as the fraction of the correct predictions, with a test set whose size is 100K. (e.g. 0.1 means 10%): (a) number of simulations for certain accuracy; (b) number of simulations for safe prediction; (c) runtime to achieve certain accuracy; (d) runtime to achieve safe prediction 98

4.10 Samples selected by active learning: (a) the projection of the selected samples in the continuous 3-D $(V_1, V_2, \Delta\phi)$space, where V_1 and V_2 are the initial voltages on the two capacitors, and $\Delta\phi$ is the initial phase difference; (b) the projection of the selected samples in the (V_1, V_2) space. 99

4.11 Functional blocks placement, sensor positions and experiment flow . . 100

4.12 Results of chip thermal experiments (AL short for active learning, RS short for random sampling) where the global accuracy and the error on positive class are defined as the fractions of the correct predictions, using a test data set whose size is 100K. (e.g. 0.1 means 10%): (a) number of simulations for certain accuracy; (b) number of simulations for safe prediction; (c) runtime to achieve certain accuracy; (d) runtime to achieve safe prediction 102

5.1 A mapping example from a 2-D space to a 3-D space with its corresponding kernel function. Different colors denote different classes. . . 110

5.2 An example with an irrelevant feature, where blue circles and red stars denote two different classes respectively. It cannot be linearly solved by the kernel function corresponding to the mapping $\boldsymbol{x} = (x_1, x_2) \rightarrow \hat{\boldsymbol{x}} = (x_1, x_2, x_1^2 + x_2^2)$. The kernel function corresponds to the mapping $\boldsymbol{x} \rightarrow \tilde{\boldsymbol{x}}$ is feasible, with a linear solution denoted by the purple plane. Here $\tilde{x}_1 = x_2$, $\tilde{x}_2 = 0$, and $\tilde{x}_3 = x_2^2$. 112

5.3 An example with irregular feature relevancy, where blue circles and red stars denote two different classes respectively. It cannot be linearly solved by the kernel function corresponding to the mapping $\boldsymbol{x} = (x_1, x_2) \rightarrow \hat{\boldsymbol{x}} = (x_1, x_2, x_1^2 + x_2^2)$. The kernel function corresponds to the mapping $\boldsymbol{x} \rightarrow \tilde{\boldsymbol{x}}$ is feasible, with a linear solution denoted by the purple plane. Here $\tilde{x}_1 = x_1$, $\tilde{x}_2 = \upsilon x_2$, and $\tilde{x}_3 = x_1^2 + \upsilon x_2^2$, where υ is a small positive weight.. 113

5.4 The network model of the RVM where $K_{ij} = K(x_i, x_j)$. Circles denote random variables and squares denote deterministic model parameters. 119

5.5 The conceptual SRKM model. Small black dots denote training data, with each row representing a vector and each column representing a feature. Circles denote random variables and squares denote deterministic model parameters. 123

5.6 Development of SRKM Bayesian network where $K_{ijk} = K_k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Circles denote random variables and squares denote deterministic model parameters. 124

5.7 Efficient SRKM model with network reduction. Circles denote random variables and squares denote deterministic model parameters. 128

5.8 Performance comparison in feature selection: (a) detection rate; (b) false alarm rate. 132

5.9 Transistor level schematic of an LDO. 133

5.10 Regression performance comparison. 134

5.11 Weights of transistor's channel length variation in the model of: (a) quiescent current; (b) undershoot; (c) load regulation. Red lines represent the means of the probabilistic weights while the higher and lower bounds represent the 95% confidence bound estimated by SRKM. 135

5.12 Weights for process variations. 137

5.13 Weights for transistors' channel length variations with strong correlations and uniform sampling. 138

5.14 Weights for transistors' channel length variations with more realistic spatial correlations. 140

5.15 Weights for all the variations with correlations across different types of variations. 141

5.16 A PLL and three BIST schemes. 143

5.17 Classifier performance comparison. 144

5.18 Signature ranking for jitter prediction with Scheme 1. 145

5.19 Signature ranking for jitter prediction with all three schemes. 146

6.1 Roles of the proposed algorithms in the design flow. 150

6.2 Characterization model built by the statistical algorithm. 152

6.3 Road map towards hierarchical and hybrid AMS verification. 154

6.4 Envisioned top level full statistical verification developed from block level models. 155

LIST OF TABLES

TABLE Page

5.1 Test time optimization of Scheme 1 145

5.2 BIST scheme synthesis 147

1. INTRODUCTION

Over the past few decades, the semiconductor community has witnessed tremendous growth in the complexity of *very-large-scale integration* (VLSI) designs. As circuits and systems become increasing sophisticated, researchers and engineers are faced with challenges that continue to evolve at an unprecedented pace. Among those challenges, verification of *analog and mixed-signal* (AMS) circuits is one of those arduous tasks, whose difficulties come from not only continued technology scaling, more stringent specifications, and escalating design complexity, but also the need for addressing *process-voltage-temperature* (PVT) variations and device aging.

1.1 Analog and Mixed-Signal (AMS) Verification

AMS systems have been so pervasive in all sorts of modern products such as computers, automotive, portable and wearable devices, biomedical equipment, robots, and enormous systems like *Internet of things* (IoT) or *cyber physical systems* (CPS). The importance of AMS systems is rapidly increasing, which is due both to their indispensability as interfaces of electronic systems interacting with the real world, and to their irreplaceable functions complementing the digital content.

AMS verification is a methodology that checks the correctness of given AMS designs by investigating if the designs fulfill specified properties [54], which is usually considered as a pre-silicon process. As shown in Fig. 1.1, verification is an crucial stage in the flow from design to production. Since fabrication is usually expensive and there is a pressing need of shorter time-to-market, the role of the verification stage is becoming more and more important. The pre-silicon design-verfication iterations are expected to capture and fix errors as much as possible. Occasionally, in view of optimizing the whole flow, preliminary tasks of post-silicon testing and yield analysis

may be included in the verification stage as well.

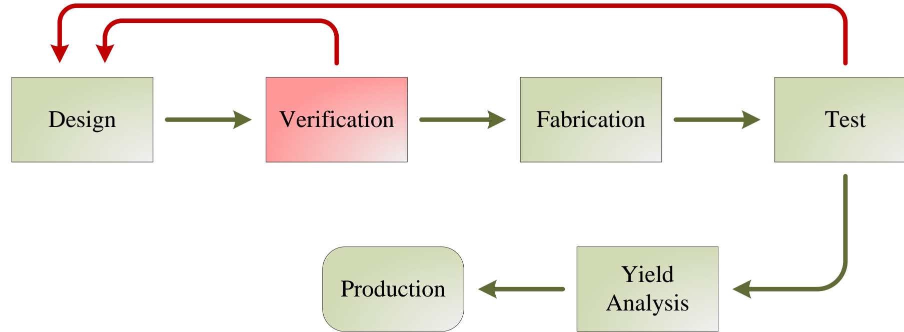


Figure 1.1: AMS design to production flow.

Verification of digital circuits has greatly advanced over the decades. Historically, it started with the development of transistor or gate level simulation techniques, then evolved to *register-transfer level* (RTL) that focuses on hardware description languages like Verilog [122], and later to automatic testbench generation with specific verification languages and libraries, where SystemVerilog [114] and *universal verification methodology* (UVM) become the mainstream in the industry [47] nowadays. Currently, digital verification is carried out at system levels involving multiple engines for various needs, such as simulations, emulations and *field-programmable gate array* (FPGA) prototypes.

Compared to digital verification, AMS verification is still immature due to the fact that analog signals are by nature more complicated than digital signals. The continuities in time, amplitude, frequency and phase of analog signals makes the parameter space or searching space of AMS verification infinitely large, which easily leads to the *curse of dimensionality*. Besides, unlike digital functionality that can be atomized into basic logical operations, analog functions and characteristics are de-

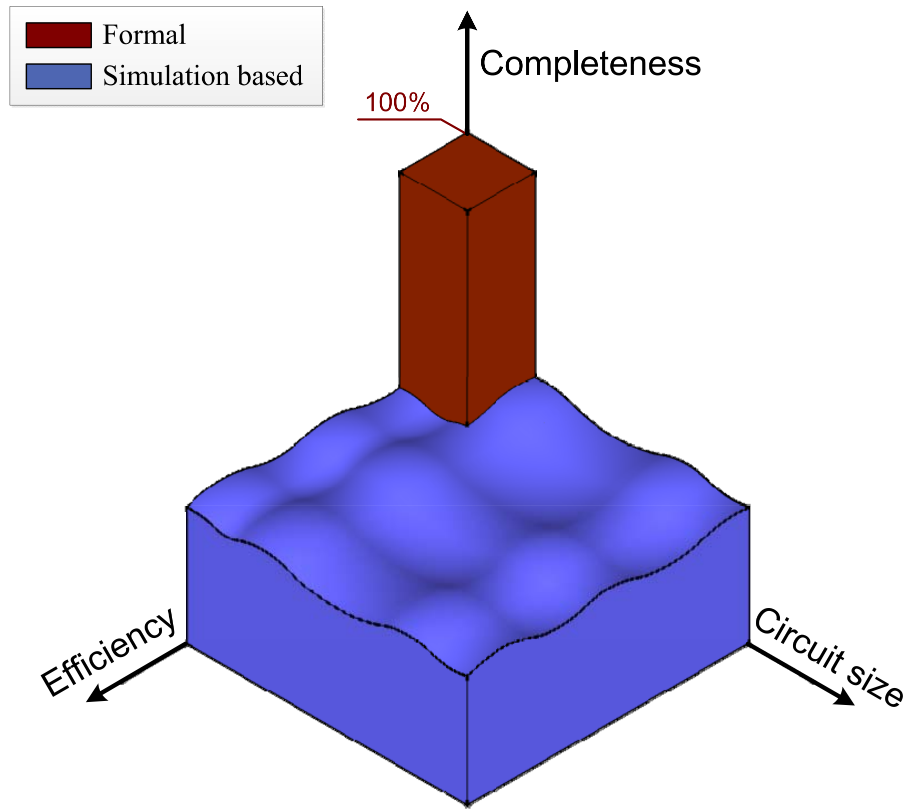


Figure 1.2: Overview of the two categories of AMS verification techniques.

scribed by continuous and usually nonlinear mathematical expressions. Furthermore, analog designs are still mostly custom designs relying on designers' innovations, due to the facts that standardizing analog designs or constraining them with a definite set of rules is impractical, hampering generic analog synthesis and fully-automated analog verification. Nonetheless, in addition to the constantly increasing complexity, there is a pressing need of shorter time-to-market of AMS ICs. Therefore, the demand of efficient algorithms for AMS verification has been greatly increased in recent years.

It is widely accepted that AMS verification techniques can be categorized into two groups [12]: simulation based verification and formal verification. Simulation

based methods verify circuits with a finite number of simulations by either embedding checkers in the process of simulations or measuring the performance from the output of the simulations. They are very efficient but cannot ensure coverage or completeness. On the other hand, formal methods verify circuits in a rigorous manner via approaches like mathematical proof. These methods are so expensive that large circuits are still beyond their capability, yet they are more reliable since the completeness is guaranteed. Fig. 1.2 demonstrates the differences between the two kinds of AMS verification techniques in terms of completeness, efficiency and the circuit size they are capable to handle.

Simulation based methods are more traditional but still prevailing over formal methods in the industry nowadays. Similar to digital verification based on testbenches, those methods verify AMS designs by running simulations across "corners" and then check if the outputs of the simulations hit or miss the given specifications. Due to the inherent continuity of analog signals and parameters, the state spaces or searching spaces of the AMS verification problems are infinitely large and hence the coverage provided by corner cases simulations is incomplete.

Recently, the investigation and development of formal methods in the AMS context has been attracting more and more interests from researchers. Formal methods are prominent in rigorous verification, and ensuring 100% coverage or completeness of the verification via mathematical proofs. Whereas the past decades have witnessed considerable advances in academic research of formal AMS verification, the efficiency of the formal methods is still a bottleneck and their applicability is limited to circuits of small sizes. Consequently, although noticeable growth of the adoption of formal methods has been observed in the industry of digital ICs [47], the industrial practice of such techniques in AMS applications is still minimal.

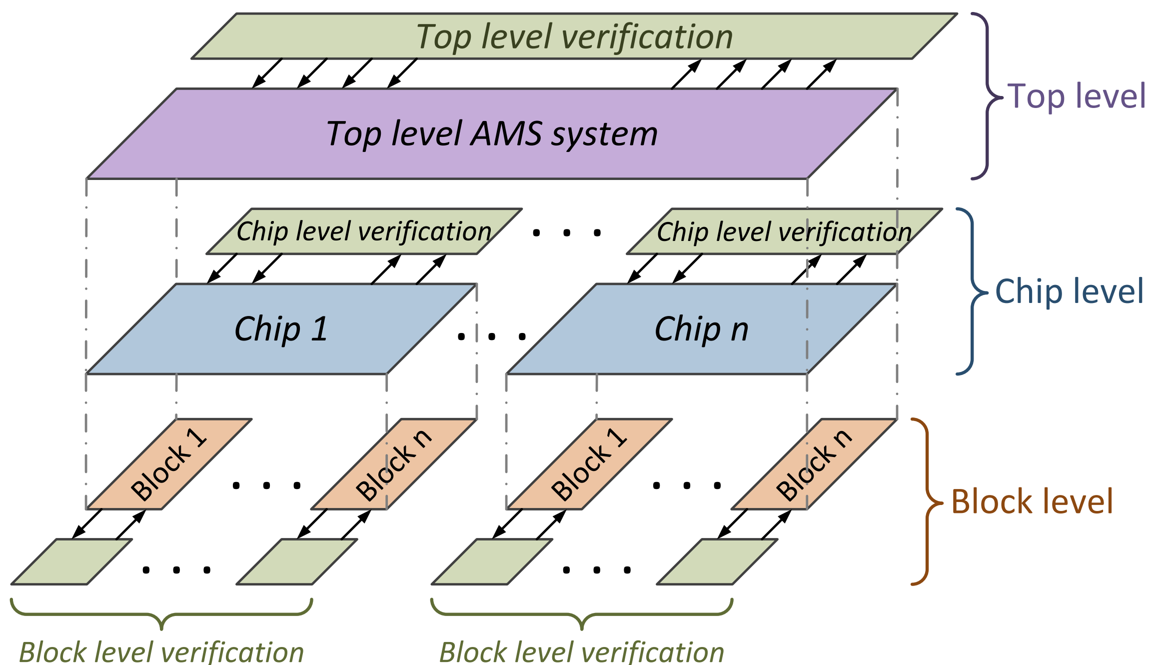


Figure 1.3: Hierarchical AMS verification.

1.2 Hierarchical and Hybrid AMS Verification Challenges

To deal with the explosively growing complexity in AMS designs that consist of an overwhelming amount of devices, designers are resorting to hierarchical methodologies that break a large design into smaller and sometimes reusable blocks. Since AMS verification also faces the same complexity challenges, typically a verification plan copes with a complex AMS system in its design hierarchy. For example, as illustrated in Fig. 1.3, a complete verification plan is conducted hierarchically by developing verification schemes for different levels including block level, chip level and the top level.

While complex AMS systems may involve multiple functional blocks with diverse functionality, it is impractical to develop one verification scheme for the entire top level verification. The hierarchical structure demonstrated in Fig. 1.3 is able to

segment the system level tasks into block level subroutines with lower complexity, and allows flexible adoption of various verification plans based on the characteristics or special needs of each block. For example, for critical blocks, formal methods are more favorable for their appealing reliability, while simulation-based methods can be used to handle non-critical but complex blocks in favor of better efficiency and scalability. This suggests a framework involving hybrid strategies for hierarchical AMS verification.

One of the challenges is the low efficiency of formal methods. Even after the complex system is segmented into blocks with much lower complexities, practically, the applicability of formal methods to the block level subsystems is still immensely restricted by the expensive cost of time and computing resources. Consequently, although it is widely recognized as an arduous challenge, it is remarkably favorable to enhance the efficiency and scalability of general purpose formal methods for AMS verification without any loss in the completeness or convergence.

For simulation-based methods, there usually is a lack of effective models that leverage simulations more than just as repeated checks and are able to provide useful feedback based on simulation results. Since each simulation can be treated as a specific format of circuitry data, machine learning models can be built to perform predictions for the purpose of verification based on the collected simulation data. One of the challenges of adopting learning based methods in the context of AMS verification is to reshape learning models to remedy the distinctive complication of AMS verification problems such as the requirement of conservativeness (e.g. minimizing acceptance error by aggressively rejecting uncertain circuit instances), the limited accessibility of data due to expensive simulations, and the enormous number of parameters under consideration. Besides, as illustrated in Fig. 1.1, there may be a considerable number of design-verification iterations in the flow from design to pro-

duction. In this light, the reusability of the verification model is of great significance in improving the efficiency of verification tasks after re-design.

Additionally, once an AMS design fails the verification process, providing hints from the verification perspective to the designers may help the process of diagnosis and re-design, and also improve the efficiency of design-verification iterations. Similar situations also exist between the testing and verification tasks since testing plans can usually be verified or optimized in the verification phase. Associating the design, the testing plans and the verification suggests unifying circuit analysis models, which can be achieved by learning based methods, and requires further development of learning models in the verification environment for parameter and specification analysis.

1.3 Dissertation Contributions

This dissertation focuses on developing generic AMS verification algorithms for better scalability and higher efficiency under the contexts of formal verification and learning based verification, which can be readily embedded into the modern hierarchical AMS verification framework described in the previous section. There are three series of new algorithms proposed in this dissertation and their improvements compared to legacy formal or simulation based methods is demonstrated in Fig. 1.4.

The first set of algorithms is developed under the reachability analysis framework [1, 149], which is a prominent formal verification framework, to enhance the efficiency and scalability of the reachability analysis by a new AMS model abstraction method, a system decomposition mechanism that partitions the reachability analysis problem into less complex subproblems, and hierarchical parallelization utilized among both the decomposed subsystems and the internal constraints of satisfiability models. In the hierarchical AMS verification framework, the combination of these algorithms serves as a formal checker that can be applied to formally verify critical blocks in the

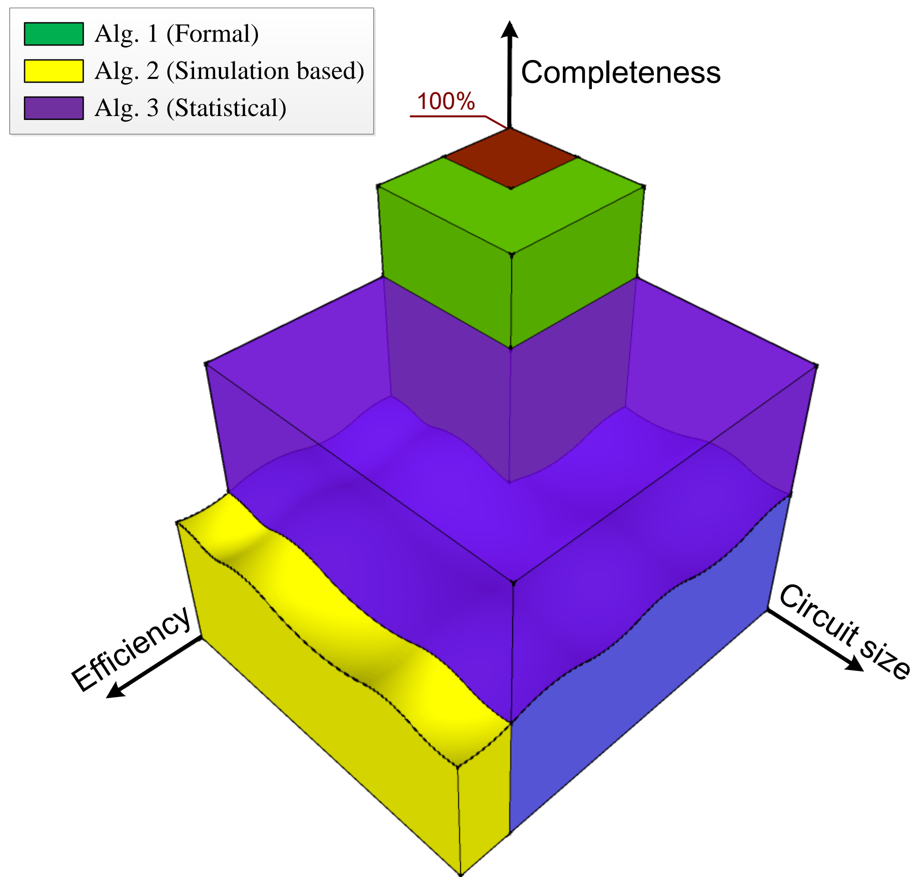


Figure 1.4: Improvements achieved by the three proposed algorithms.

AMS system.

The second set of algorithms demonstrate a novel active learning guided machine learning approach for learning based verification. One of the algorithms is employed under the context of support vector machines and is able to dramatically reduce the need of simulations and samples, leading to significant reduction of the overall training cost. It is further extended to an asymmetric algorithm to enable conservative prediction. Due to the superior efficiency of the algorithm and the reusability of the produced model, the algorithm is particularly suitable for preliminary verification of designs and any potential re-designs under the same specifications, or for quick validation of the feasibility of test plans.

The third algorithm is another learning based method that develops statistical regression model to accurately approximate the characteristics of AMS systems and reliably capture complex dependencies of circuit performances on essential circuit and device parameters. This sparse model produces accurate models learned from a moderate amount of simulations or measurement data, and provides probabilistical predictions which are beneficial to the conservativeness of verification tasks. Besides, it is able to compute a probabilistically inferred weighting factor for each parameter. Such factor quantifies the criticality of the parameter as part of the overall learning framework, hence offering a powerful enabler for variability modeling, failure diagnosis, and test development. In this sense, it is a valuable asset to the whole flow shown in Fig. 1.1. This statistical verification methodology can be leveraged in the hierarchical AMS verification framework to handle all the systems/subsystems that exceed the capability of formal checkers and provide helpful parametric analysis results to designers and testing engineers to aid the debug or re-design process and the optimization of test plans, respectively.

1.4 Dissertation Organization

In this dissertation, Section 2 discusses the existing AMS verification methods in the contexts of formal methods and simulation-based methods. First, a general definition of AMS systems is provided. Then, formal methods including proof based symbolic methods, equivalence checking and model checking are introduced and definitions are provided to illustrate problems addressed by different techniques. After that, a brief introduction to simulation based verification methods and the two popular routes including assertion based and learning based methods are provided.

In Section 3, the development of the first set of algorithms for formal AMS verification firstly proposes a model abstraction method that is specifically effective for *digitally-intensive analog* systems, which accounts for a substantial subset of AMS systems. Then, the concept and the potential improvement of system partitioning and parallelization is introduced and discussed, followed by a generic system decomposition method that is able to divide the original satisfiability problem into subproblems with lower complexity. The efficiency of the algorithm is demonstrated in reachability analysis of a simplified DC-DC converter and a digitally-intensive phase-locked loop.

Section 4 proposes the second set of algorithms by formulating the verification problems into binary classification problems which can be handled by support vector machines. An active learning scheme based on version space reduction is developed to guide the sampling process during the generation of training samples, resulting in simulation reduction in the verification tasks. In addition, an asymmetric extension of the active learning framework is proposed to achieve conservative prediction by adding safety level evaluation of the model into the active learning strategy. The advantages of the reduced number of the required simulations and the resulted training

process acceleration is manifested by verifying an LC based DC-DC converter and a charge pump phase-locked loop. The algorithm is also extended to validate a testing scheme of on-chip peak temperature monitoring.

In addition, a statistical regression method is proposed in Section 5 to simultaneously characterize AMS systems and capture the performance dependencies on various parameters. To achieve both objectives concurrently, it faces challenges from the complex AMS nature and the feature weighting problem, which are introduced firstly. Then, a novel feature kernel is defined to embed feature weighting mechanism into kernel machines. Under the Bayesian learning framework, a learning model called sparse relevance kernel machine is developed to produce sparse and accurate predicting models efficiently. The effectiveness of the algorithm is demonstrated in the statistical variability modeling of a low-dropout regulator. And the method is extended to the built-in self-test development and optimization of a charge-pump phase-locked loop.

The dissertation is concluded in Section 6, with discussions of potential future works towards a complete road map of hierarchical and hybrid AMS verification.

2. MODELING OF AMS SYSTEMS AND VERIFICATION PROBLEMS

Before diving into detailed AMS verification algorithms, this section provides definitions and brief descriptions of AMS systems and different verification problems for the purpose of illustrating different verification frameworks.

2.1 AMS Systems

Generally, a model of an AMS system consists of both analog and digital variables along with continuous and discrete transitions and state mapping functions. Based on the definition provided in [85], a new definition of AMS systems is provided as follows:

Definition 1 *An AMS system is a tuple* $H_{AMS} = (X_a, X_d, t, F_a, F_b)$*, where:*

- $X_a \subseteq \mathbb{R}^n$ *is a set of continuous variables representing analog content of the system;*
- $X_d \subseteq \{0,1\}^m$ *is a set of boolean variables representing digital content of the system;*
- $t \in \mathbb{R}$ *is a time variable;*
- $F_a \in \mathbb{R}^n \times \{0,1\}^m \times \mathbb{R}^2 \rightarrow \mathbb{R}^n$ *is the mapping function from the state* (X_a, X_d) *at time* t_1 *to the continuous portion* X_a *at time* t_2*;*
- $F_d \in \mathbb{R}^n \times \{0,1\}^m \times \mathbb{R}^2 \rightarrow \{0,1\}^m$ *is the mapping function from the state* (X_a, X_d) *at time* t_1 *to the discrete portion* X_d *at time* t_2*.*

To represent digital signals and arithmetic computations, X_d can be expressed in integers (i.e., $X_d \in \mathbb{Z}^m$). Alternatively, for behavioral description, the digital

content can also be defined by a nested tuple (X_d, R_{res}) where $X_d \subseteq \mathbb{R}^m$ is expressed similarly to X_a but comes with a resolution $R_{res} \in \mathbb{R}$. If $m = 0$, meaning $X_d = \emptyset$, there is no digital content and thus the system is pure analog. On the other hand, if $m \gg n$, meaning that digital content is the major portion, such systems are named as *digitally-intensive analog* (DIA) systems.

The two mapping functions F_a and F_b describe the transient behaviors of the AMS system. In Definition 1, F_a and F_b are generally defined to take both *time-variant* and *time-invariant* systems into account with the following forms:

$$X_a(t_2) = F_a(X_a(t_1), X_d(t_1), t_1, t_2),$$
$$X_d(t_2) = F_d(X_a(t_1), X_d(t_1), t_1, t_2).$$

Time variant models are usually for specific applications, such as time-varying capacitors integrated in *microelectromechanical systems* (MEMS) [101], photodiodes with temporal noise in image sensors, memristive cells for hardware implementation of neural networks [135], and so on, which only account for a small portion of AMS systems. Time-invariant components like filters, amplifiers, data converters, etc., are more widely used in AMS systems. For time-invariant systems, the state at t_2 may be determined by the state at t_1 and the time step $\Delta t = t_2 - t_1$. As a result, the two mapping functions can be simplified to the following forms:

$$X_a(t_2) = F_a(X_a(t_1), X_d(t_1), \Delta t) \in \mathbb{R}^n \times \{0,1\}^m \times \mathbb{R} \to \mathbb{R}^n,$$
$$X_d(t_2) = F_d(X_a(t_1), X_d(t_1), \Delta t) \in \mathbb{R}^n \times \{0,1\}^m \times \mathbb{R} \to \{0,1\}^m.$$

2.2 Existing Formal AMS Verification Techniques

As illustrated in the previous section, formal methods account for an important category of AMS verification techniques. The objective of this section is to provide definitions of different formal verification methods and a brief overview of some existing methods.

2.2.1 Proof Based Symbolic Methods

Symbolic analysis was the first step towards circuit design as well as formal verification by nature, and numerous proof based symbolic methods have been proposed in the AMS verification domain [149, 107]. Such methods verify design properties of AMS circuits with formal deduction based theorem proving. In the context of propositional logic [16], the AMS system H_{AMS} can be considered as a set of formulas and conditions, namely *axioms*, and the design properties P is a *theorem*. Given a set of *rules of inference* R_{infer}, this kind of verification methods can be consequently defined as:

Definition 2 *Proof based symbolic verification is a methodology finding a proof that* P *is deducible from* H_{AMS} *via* R_{infer}*, denoted by:*

$$H_{AMS} \vdash P.$$

And a proof is a tuple $H_{proof} = (A_1, A_2, ..., A_n, P)$ *with finite length, where* A_i *is either an axiom in* H_{AMS} *or an inference from its predecessors* $(A_1, A_2, ..., A_{i-1})$ *in* H_{proof} *via* R_{infer}*.*

The applicability of traditional symbolic methods is limited by the facts that they usually require substantial expertise and human innovation. Recent research has been focused on handling these problems with theorem provers. For example, a high

order logic proof checker PVS [106] is used in [50] to perform symbolic verification by formally proving that an approximated piecewise linear model of a synthesized netlist is functionally equivalent to the properties of a behavioral model extracted from VHDL-AMS specifications. Another symbolic verification method proposed in [4] uses Mathematica [139] to prove the correctness of the properties of AMS designs. With circuits and properties described by generalized recurrence equations, an iterative procedure is applied to keep generating counterexamples until the proof is achieved. In addition to the usage of theorem provers, some other symbolic methods focus on the development of data structures to efficiently manipulate large symbolic representations, most of which lie in the framework of *binary decision diagram* (BDD) [107] that eliminates data redundancy in the symbolic analysis, avoids exhaustive enumeration, and provides efficient numerical evaluations.

2.2.2 Equivalence Checking

AMS systems are usually modeled and compared on different levels of abstraction such as transistor level schematics, SPICE netlists, behavioral models [109], macromodels [82], etc. Equivalence checking is to prove or disprove the functional equivalence or similarity between two models of an AMS system. Since the functional equivalence is often verified with respect to the input-output behavior of the AMS systems, an abstraction of H_{AMS} can be expressed as a function $M(x;p)$ that takes inputs x and parameters p of the system and mimics the output of the system. Assuming the set of all possible inputs is Φ_I, and the set of all plausible parameters or configurations is Φ_P, equivalence checking can be defined as:

Definition 3 *Given two abstractions M_1 and M_2 of H_{AMS}, equivalence checking is to prove or disprove the following statement:*

For functional equivalence,

$$\forall x \in \Phi_I, \forall p \in \Phi_P, M_1(x;p) = M_2(x;p),$$

or for functional similarity,

$$\forall x \in \Phi_I, \forall p \in \Phi_P, M_1(x;p) \approx M_2(x;p).$$

Linear AMS systems like filters and amplifiers are usually modeled with transfer functions, and several works [11, 104] verify the transient response of linear analog circuits by performing equivalence checking between the transfer function representing the implementation and the transfer function representing the specification. For nonlinear AMS systems, *ordinary differential equations* (ODEs) or *differential algebraic equations* (DAEs) may be used to build the models [60, 32, 118] describing the behavior of the systems. To verify the functional similarity instead of equivalence, authors of [109] propose to formulate the equivalence checking problem as an optimization problem by minimizing the errors between the behavioral model and the implementation.

2.2.3 Model Checking

As a powerful tool for automatic verification, initially, model checking [10, 149] was developed for discrete systems with finite-state models and formal properties. Typically, the system under consideration is firstly modeled and the produced model is usually expected to function as a simulator of the system for quick assessment. Then, for AMS verification, a model checker is employed to determine if all the properties are satisfied by completely exploring the whole state space. Therefore, given properties or specifications P, model checking can be defined as:

Definition 4 *Model checking is to build a model M from H_{AMS} and then check if P holds in the model M, denoted by*

$$M \models P$$

Generally, P is formulated as a temporal logical formula, and model checking is usually conducted by analyzing the reachable state spaces of the extracted model, leading to a group of model checking methods called *reachability analysis* [1, 149]. This kind of methods is defined as follows:

Definition 5 *Given an initial state space Φ_0, reachability analysis is to compute a sequence of the resulting reachable state space $(\Phi_0, \Phi_1, ..., \Phi_n)$ of the extracted model M, until P is satisfied or violated in Φ_n, meaning the satisfiability of the following formula can be determined:*

$$(\Phi_0, \Phi_1, ..., \Phi_n) \models P$$

By fixing a time step and a state transition relationship, an AMS system H_{AMS} can be characterized by an initial state space Φ_0 and a transition function $Tr()$. Then, the reachable state space Φ_{i+1} from the current state space Φ_i can be defined as:

$$\Phi_{i+1} = \{Tr(x) | x \in \Phi_i\}$$

For nonlinear AMS systems, it is often arduous to compute the exact reachable state space. A more practical way is to overapproximate Φ_i with $\hat{\Phi}_i$ defined as:

$$\hat{\Phi}_{i+1} \supseteq \{Tr(x) | x \in \hat{\Phi}_i\}$$

and for conservativeness, P should be checked that it is not violated in any of those

approximations. One of the key challenges posed here is the accumulation of over-approximation which may collapse the convergence of the procedure of reachability analysis. As a result, numerous schemes have been developed to achieve tighter overapproximation of the state spaces, including hypercubes [57, 119, 146], convex polygons [86, 132], support function representations [78, 83], projectahedrons [53], zonotopes [7], and so on.

Reachability analysis for linear AMS systems is effective since the reachable state spaces can be efficiently computed [8, 52] and the error accumulation is well controlled. For nonlinear AMS systems, reachability analysis is much more complicated and requires massive efforts to achieve automatic and efficient computation of the reachable state spaces. For example, methods proposed in [7, 113] use a discrete-time linear model to approximate the nonlinear system and perform efficient reachability analysis via zonotope manipulation. Greatly benefited from the advances in boolean satisfiability problems (abbreviated as SAT) in the digital domain, *satisfiability modulo theories* (SMT) [14, 96] have been developed to decide the satisfiability of formulas with both integers and real numbers. By converting reachability analysis problems in the AMS verfication into SMT problems, methods in [132, 126, 145, 150] utilize a variety of SMT solvers (e.g. Yices [42], iSAT [48], Z3 [36], etc.) to explore the reachable spaces.

2.3 Existing Simulation Based AMS Verification Techniques

Despite the superior coverage or completeness provided by formal methods, regarding the efficiency and the scalability, simulation based methods are in a dominant position in the industry. Usually, they take the form of simulations at different levels (e.g. SPICE level or behavioral level) with configurable parameters varying in given ranges to reflect process variations, initial conditions and design parameters.

Industrial practice is to select the minimal, nominal and maximal values for each parameter and run simulations for all permutations of these so-called corners.

Designing testbenches in the simulation environment and evaluating performances of AMS systems based on simulation results is not yet fully automated. And running simulations exhaustively can be extremely expensive. Research on simulation based methods often aims at higher level automation, better versatility, optimization of the sampling process, and extraction of more useful information from simulation data. Simulation based methods are not well classified in the literature as formal methods since models and approaches utilized in those methods are less rigorous but more diverse. Nonetheless, some methods are similar in the techniques they utilized and the following subsections will briefly cover two groups of simulation based methods.

2.3.1 Assertion Based Verification

Assertion based verification methods focus on improving the simulation environment or creating new tools for the purpose of AMS verification. A common implementation of these methods is to embed AMS assertions or checkers in the simulation flow so as to automatically invoke property checking during every simulation process. For example, authors in [3] suggest extending the *property specification language* (PSL) [65] assertions from the digital domain to the AMS environment, and an implementation for SPICE level simulation has emerged [97]. Additionally, syntax and semantics introduced in [58] expand and generalize *SystemVerilog Assertion* (SVA) [114] regular expressions to continuous-time systems. Another AMS verification flow proposed in [144] implements embedded AMS checkers with Verilog-A [46] and VerilogAMS [75] to support top level functional coverage checking.

Overall, most of the assertion based methods focus on the AMS verification tools or implementations for the ease of use or rich functionality to capture more analog

behaviors. Those tools and implementations provide infrastructures to support and improve verification flows built upon them.

2.3.2 Learning Based Verification

Machine learning techniques have been evolving for decades and have been widely used in various domains like bioinformatics, medical imaging, information retrieval, and circuit applications, showing exceptional power in sophisticated systems for tasks such as classification, regression, clustering, decision making, etc. Lately, a machine learning algorithm even masters the ancient board game Go against human world champion [108], which is widely considered as another milestone of the progress of machine learning. Benefited from the advances of machine learning, emerging techniques leverage machine learning as a powerful toolbox to solve circuit problems including AMS verification.

For any circuit simulation, there are two types of outcomes: the circuit under consideration with a certain configuration complies with all the given specifications, namely passing, or at least one of the given specifications is violated, namely failure. Naturally, this can be formulated as a binary classification problem and authors in [39, 84] train classifiers to verify the feasibility and the performance of AMS circuits.

Another approach towards machine learning based verification is to characterize AMS circuits with regression models, reducing the burden of analyzing and extracting specific models for various circuits. For example, a modeling method proposed in [5] partitions large circuits into intermediate behavioral models and applies *support vector regression* (SVR) [129] to capture continuous values in the verification flow. Another method proposed in [35] extends *recursive vector fitting* (RVF) techniques [38] to model the time-domain response of nonlinear circuits and to automatically extract analytical behavioral models for the acceleration of simulation based flows

including verification.

Additionally, learning methods have been adopted in a variety of techniques related to verification tasks or may potentially be desirable for verification improvements. For instance, *finite state machine* (FSM) models are derived from transistor-level circuit descriptions by learning from input/output trajectories in [55]. And an algorithm called *fast function extraction* (FFX) [92] is proposed to build accurate variation-aware models by augmenting design models with nonlinear equations learned via pathwise regression method [49]. For post-silicon debug, an atomic model learning approach [37] based on supervised learning is proposed to diagnose design bugs by inserting modeling artifacts into different nodes and learning behaviors along signal flow paths.

Although machine learning is beneficial to the efficiency improvement of simulation based verification procedures, addressing AMS verification with machine learning techniques introduces new challenges including both the problems from the context of machine learning like over-fitting problem, and the problems from the AMS verification perspective, such as the limited availability of training data and safety or conservativeness consideration. Consequently, integrating machine learning into the AMS verification framework is a challenging task that requires innovation from both domains.

3. PARALLEL HIERARCHICAL REACHABILITY ANALYSIS *

Reachability analysis is an effective AMS formal verification technique [149] that automatically verifies dynamic properties, which can be formally expressed by temporal logic [28], of the AMS system. Similar to other formal methods, it also suffers from state space explosion for complex AMS circuits.

Avenues towards efficiency enhancement include lowering the complexity of the problem and increasing the computation power or resources. This work proposes a parallel hierarchical *satisfiability modulo theories* (SMT) based reachability analysis technique built on top of circuit decomposition. To achieve a less complex model, an abstraction method is proposed to model AMS systems with pure analog representations. Furthermore, the abstracted model is systematically decomposed into subsystems with less complex transient transitions, whose reachability analysis can be formulated as satisfiablity problems with fewer variables. The reachability analysis flow is then parallelized at the decomposed subsystem level and the satisfiability problem level. Overall, the framework lowers the complexity via model abstraction and system decomposition, and invokes more computing resources via parallelism, hence leads to appealing speedup compared to the traditional reachability analysis flow.

*Part of this section is reprinted with permission from: (1) "Verification of digitally-intensive analog circuits via kernel ridge regression and hybrid reachability analysis" by Honghuang Lin, Peng Li and Chris J. Myers, 2013. *Proceedings of the 50th Annual Design Automation Conference*, Page 1-6, ©2013 ACM. (2) "Parallel hierarchical reachability analysis for analog verification" by Honghuang Lin and Peng Li, 2014. *Proceedings of the 51st Annual Design Automation Conference*, Page 1-6, ©2014 ACM.

3.1 Introduction

Reachability analysis is an automatic formal framework in AMS circuit verification, yet it is often incapable of verifying complex circuits due to the exponentially increasing cost for high dimensional systems. Generally, the capability of formal methods is determined by the computational power and the complexity of the algorithm. For example, the efficiency of the nonlinear SMT based methods [126, 145, 85, 83] which convert the verification task into a *satisfiability* (SAT) problem heavily depends on the performance of the adopted SMT solver [48, 13]. Since there is no polynomial time solution for solving SAT problems, it leaves much opportunity for improvement through other avenues such as lowering the complexity of the modeled SAT problems and parallel computing.

Designing AMS circuits in highly scaled CMOS technologies is hampered by increasing *Process-Voltage-Temperature* (PVT) variations and worsening device characteristics. As a result, the so-called digitally-assisted or *digitally-intensive analog* (DIA) design methodology has emerged, which minimizes the pure analog content of the design while relying more on digital processing [93]. However, inclusion of increased digital content in such designs adds new complications to the existing challenges of AMS circuit design verification, which are resulted from effects such as nonlinear dynamical characteristics and complex interactions between digital and analog signals. Therefore, new model abstraction methods should be developed to remedy the complex "digital" effects.

Furthermore, parallelizing the reachability analysis flow requires careful consideration of possible additional errors, since breaking down the full system into subsystems may introduce overapproximations whose accumulation in the flow may hamper the convergence of the reachability analysis. Consequently, it is favorable to develop

a parallel algorithm with less complex subproblems for potential further efficiency improvement, but without introducing additional overapproximations to avoid error accumulation issues.

This work aims to develop a hierarchical parallel verification technique for AMS circuits under a formal reachability analysis framework [6, 145, 85, 83]. The proposed method decreases the complexity of the system transient behavior via a general system decomposition methodology and increases the computational power by leveraging parallel computing techniques to enhance the efficiency of reachability analysis.

The first main idea of this work is to "unify" the two types of signals by converting digital signals into approximate analog signals. More precisely, this work leverages machine learning to find the potential error of the conversion on a given digital block and bound its output using a continuous analog signal and a tight error interval. It turns out kernel based learning methods such as *support vector machines* (SVM) [128, 129] are useful tools for this purpose. In particular, this work adopts kernel ridge regression [103, 120] and the related confidence interval computing algorithm [34] to construct such "analog" models for the digital block and estimate the error interval, the latter of which ensures the conservativeness of this conversion as required by formal verification. The accuracy of this "digital-to-analog" modeling, e.g. the length of the error interval, is tightly controlled in the learning process.

Secondly, this work decomposes a nonlinear analog system into subsystems with lower dimensional transient behavior such that the transient verification of the circuit using the SMT-based reachability analysis can be drastically sped up. Partitioning an analog circuit into a set of smaller blocks (subsystems) makes intuitive sense as analog circuits tend to have a well-defined signal flow structure among different blocks. However, partitioning the circuit for reachability analysis gives rise to significant

complications to maintain key correlations among state variables. Furthermore, large or even unacceptable overapproximation may occur in the full system reconstruction from the lower dimensional reachable spaces of all the subsystems, which may lead to the deceleration of the convergency of the computed state trajectories and the failure in the reachability analysis altogether.

3.2 Model Abstraction

The interaction between digital (Boolean) and continuous-valued analog signals, which is intensified in DIA circuits, presents a key challenge in AMS verification. In addition, to fully capture "digital" effects such as finite resolution (inherent in any analog-to-digital conversion), round-off error and overflow (inherent in additions/multiplications due to finite word length effects), additional state variables need to be introduced, blowing up the dimensionality of the state space and slowing down the the reachability analysis. To address the problem, this work proposes to extract an abstract continuous model with a lower dimensionality for a given AMS or DIA design through the use of learning-based regression. The correctness of our approach is guaranteed by the use of error intervals that are part of the abstract model.

While the proposed abstraction method can be applied to generic AMS circuits and models to speedup transient reachability analysis, it turns out to be particularly appealing to DIA circuits. DIA circuits are constructed to have high digital content with minimum use of pure analog-based processing. Digital blocks such as filters are designed to be "linear" and implemented in robust digital logic. However, this linearity disappears in the presence of round-off errors due to finite word length effects. The use of the machine learned conservative analog models re-establishes this lost linearity and allows application of the proposed support function approach

to a more dominant portion of the design, which is modeled as the linear sub-system.

3.2.1 DIA Verification Challenges

Taking the *digitally-intensive phase-locked loop* (DI-PLL) shown in Fig. 3.1 as an example, the circuit consists only of digital functional blocks, yet involves an analog feedback path from the *digitally controlled oscillator* (DCO) to the *phase detector* (PD). The PD uses an accumulator and a *time-to-digital converter* (TDC) to detect the only analog variable, which is the phase difference $\Delta\phi$ between the reference clock REF and the output signal CKV, and then outputs the phase difference to the loop filter to control the DCO. Neglecting the digital effects and assuming all functional blocks have an ideal continuous characteristics, the verification of the system can be performed on a model involving a few state variables such as ones to capture the input/output/internal signals of the loop filter. However, to fully capture those digital effects, more state variables are needed and hence the model has a much higher complexity.

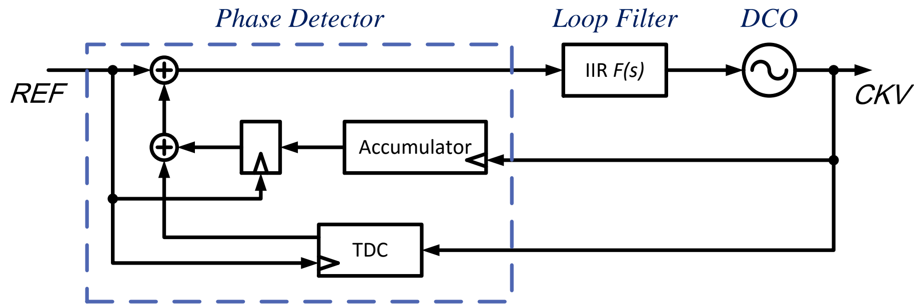


Figure 3.1: Block diagram for a DI-PLL.

For example, a TDC shown in Fig. 3.2 is used to measure the fractional phase difference between CKV and REF. Theoretically, the output of the TDC should be

proportional to the fractional phase difference, making the transition of the phase detector linear. However, the finite delay of the inverters limits the achievable resolution of the TDC. As a result, the phase detector with a finite TDC resolution has a staircase transition curve instead of an ideal linear one.

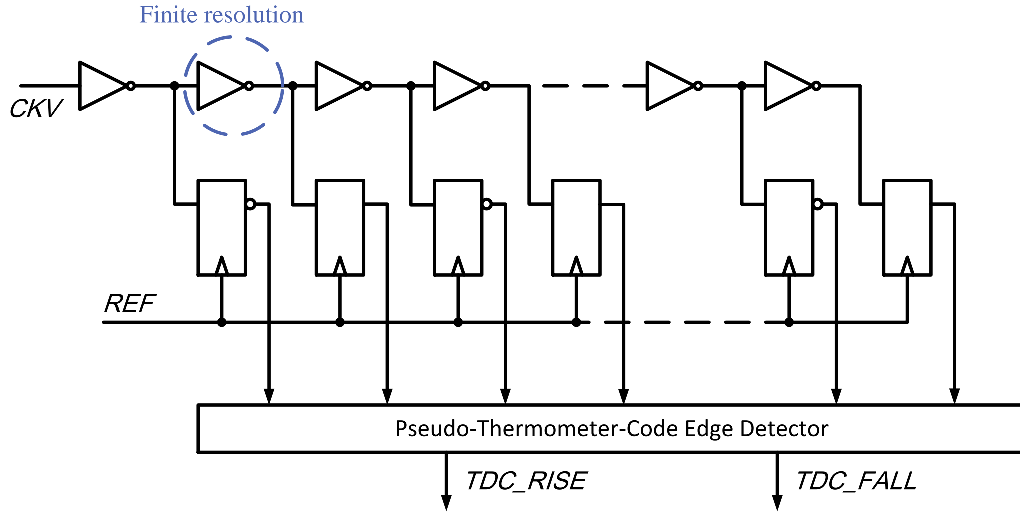


Figure 3.2: Time-to-digital converter proposed in [115].

Digital filters such as *finite impulse response* (FIR) or *infinite impulse response* (IIR) filters are often designed from a linear z-domain transfer function. To model the ideal transfer function of a second order IIR loop filter shown in Fig. 3.3, four state variables are sufficient. But to fully model the finite word length effects such as overflow and round-off error of the filter, 8 more variables should be assigned to the output nodes of all the internal adders and multipliers. The dimension of the system is thus greatly increased.

Inspired by the highly developed and advanced verification techniques in the pure digital domain, an intuitional approach towards AMS verification is to discretize the

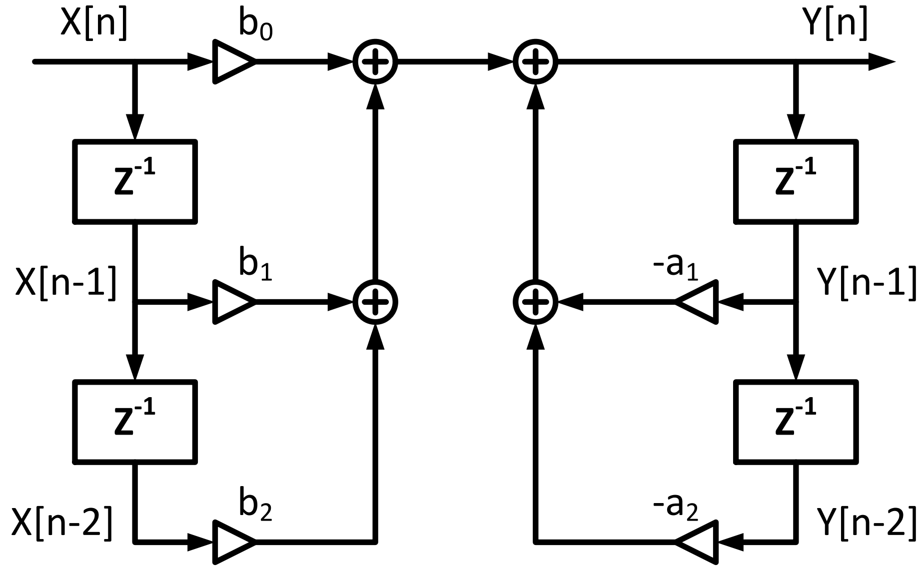


Figure 3.3: Second order IIR filter.

analog content [3] so as to address the problem with methods similar to digital techniques. However, discretization based method can only provide very limited coverage. Although it appears to bump heads with the DIA design trend that tends to replace analog functional blocks with digital ones, based on the DIA verification challenges posed above, modeling digital functional blocks with proper analog abstraction may potentially simplify the complexity of the formal verification task while preserving its completeness.

3.2.2 Digital Abstraction with Analog Variables

A general definition of an AMS system is provided in Definition 1 in the previous section as a tuple $H_{AMS} = (X_a, X_d, t, F_a, F_b)$. Assuming H_{AMS} is time-invariant and the time step for the transitional relationship is fixed, the time variable t can be discarded out of H_{AMS}. Furthermore, as suggested in Section 2.1, digital signals can be expressed as real value variables with pre-defined resolutions, resulting in a definition of a DIA system as follows:

Definition 6 *An DIA system is a tuple $H_{DIA} = (X_a, X_d, R_{res}, F_a, F_b)$, where:*

- *$X_a \subseteq \mathbb{R}^n$ is a set of continuous variables representing analog content of the system;*
- *$X_d \subseteq \mathbb{R}^m$ is a set of real-value variables representing digital content of the system, where $m \gg n$ is expected;*
- *$R_{res} \in \mathbb{R}^m$ is a set of resolutions corresponding to the digital variables;*
- *$F_a \in \mathbb{R}^{n+m} \to \mathbb{R}^n$ is the transition function from the full state (X_a, X_d) to the analog portion X_a in a time step;*
- *$F_d \in \mathbb{R}^{n+2m} \to \mathbb{R}^m$ is the transition function from the full state and the resolution (X_a, X_d, R_{res}) to the digital portion X_d in a time step.*

As discussed in Section 3.2.1, the size of X_d and F_d in a DIA system is usually large, which may likely blow up the dimension of the state space for verification. From the design perspective, to focus on the key properties of the digital functional blocks, it is common to approximate mapping functions in F_d with continuous mapping functions while independent error variables are added into X_a to cover the approximation error. Fox example, a TDC is routinely modeled to have a linear conversion characteristics and additive quantization errors, the latter of which is used to analyze the TDC induced noise [116].

Motivated by the above design analysis technique, our method converts key variables in X_d into analog variables and approximates F_d with continuous mappings to simplify the system. However, the goal of our method is to find a conservative approximation for the entire reachable state space rather than analyzing the "average" noise behavior of the circuit. This prompts us to find conservative error intervals, ideally tight, for our abstract models.

Let us define the abstracted model produced by our proposed method as:

Definition 7 *An abstraction of H_{DIA} is a tuple $H_C = (X, F, E_u, E_l)$ where:*

- *$X \subseteq \mathbb{R}^N$ consists of representations of analog content of the system X_a and continuous approximations of digital content of the system X_d;*
- *$F \in \mathbb{R}^N \to \mathbb{R}^N$ is the transition functions that consists of analog transition F_a and continuous approximations of digital transition F_d;*
- *$E_u \in \mathbb{R}^N \to \mathbb{R}^N$ is the upper bound of the error intervals of $F(X)$;*
- *$E_l \in \mathbb{R}^N \to \mathbb{R}^N$ is the lower bound of the error intervals of $F(X)$.*

To extract an abstract continuous model from the DI-PLL, all the digital variables in X_d such as the input/output of the phase detector and the loop filter are approximated by the corresponding continuous variables in X. Then, for all the digital blocks, the transition function will be approximated by the ideal characteristics. For example, the transition of the phase detector is modeled as a linear function that produces an output proportional to the detected phase difference. The transition of the filter is modeled by its ideal z-domain transfer function, which is also a linear mapping function that maps the input and internal storage of the filter to its output at the next sampling clock cycle.

Error accumulation is an inherent problem in reachability analysis. If the system abstraction comes with loose error intervals, i.e. large $E_u - E_l$, the reachability analysis may converge slower or, even worse, may no longer converge. If the error intervals are too narrow, i.e. small $E_u - E_l$, the abstraction may lose conservativeness and the result of the reachability analysis is not ensured to cover all the possible cases. So it is important and essential to find a tight interval that covers most of the errors

with small over approximation. The next subsection leverages *kernel ridge regression* (KRR) to compute such tight intervals.

3.2.3 Error Interval Estimation via KRR

KRR [103], a.k.a. *least squares support vector regression* (LS-SVR)[120] is a very effective statistical learning method and has been applied to error estimation. It is formulated as:

$$\min_{w,b,e} w^T w + \gamma \sum_{i=1}^{n} e_i^2 \tag{3.1}$$

subject to

$$Y_i = w^T \phi(X_i) + b + e_i, i = 1, ..., n. \tag{3.2}$$

where (X_i, Y_i) are training data and ϕ is the mapping function that maps X_i into a higher dimensional space. By using Lagrange multipliers, one can solve the problem by defining a kernel function $K(X_i, X_j) = \phi(X_i) \cdot \phi(X_j)$ instead of defining an exact ϕ. This model produces the following decision function:

$$\hat{m}(x) = \sum_{i=1}^{n} \alpha_i K(x, X_i) + b_i \tag{3.3}$$

Given enough training data and a suitable parameter γ, KRR can train an accurate model for error prediction. Considering the system abstraction for DI-PLL, errors may come from the continuation of digital variables and the mapping function approximation by an ideal continuous characteristic. For instance, let $x_i \in X$ denote the output of the phase detector in the abstract continuous model for the DI-PLL, our method assumes that the abstracted mapping function of the phase detector is $x_i = F_i(X)$ and its original mapping function is $x_i = F_{ad,i}(X)$. This method samples some random system states $X_1, X_2, ..., X_n$ and computes the error of the abstraction on x_i as $Y_j = F_i(X_j) - F_{ad,i}(X_j)$. Then, this method can get an error estimation of

x_i by applying KRR on the training data set(Y_j, X_j) such that $j = 1, 2, ..., n$. By repeating the sampling and training process, error prediction functions are produced by KRR for every state variable.

To find a tight interval of the prediction, an efficient algorithm for computing confidence intervals of KRR prediction has been given in [34]. The algorithm treats KRR as a linear smoother to derive the formulation of the bias and variance of the prediction. A linear smoother is defined as an estimator $\hat{m}$ of m in which there exists a smoother vector $L(x) = (l_1(x), ..., l_n(x))$ such that:

$$\hat{m}(x) = \sum_{i=1}^{n} l_i(x) Y_i, \forall x \in \mathbb{R}^d \tag{3.4}$$

KRR prediction is a kind of biased estimation and thus there is bias between the predicted value and the center of the prediction interval. The bias is given as

$$\hat{B}(\hat{m}(x)) = L(x)^T \hat{m} - \hat{m}(x) \tag{3.5}$$

where $\hat{m} = (\hat{m}(X_1), ..., \hat{m}(X_n))^T$. The variance of the prediction at x is given as

$$\hat{V}(\hat{m}(x)) = L(x)^T \hat{\Sigma}^2 L(x) \tag{3.6}$$

where $\hat{\Sigma}^2 = diag(\hat{\sigma}^2(X_1), ..., \hat{\sigma}^2(X_n))$ and

$$\hat{\sigma}^2(x) = \frac{L(x)^T diag(\hat{\epsilon}\hat{\epsilon}^T)}{1 + L(x)^T (SS^T - S - S^T)} \tag{3.7}$$

S denotes the smoother matrix of the initial smooth and $\hat{\epsilon}$ denotes the residuals of $\hat{m}(X_1), ..., \hat{m}(X_n)$.

The KRR based abstraction process is shown in Fig. 3.4. At first, all the digital

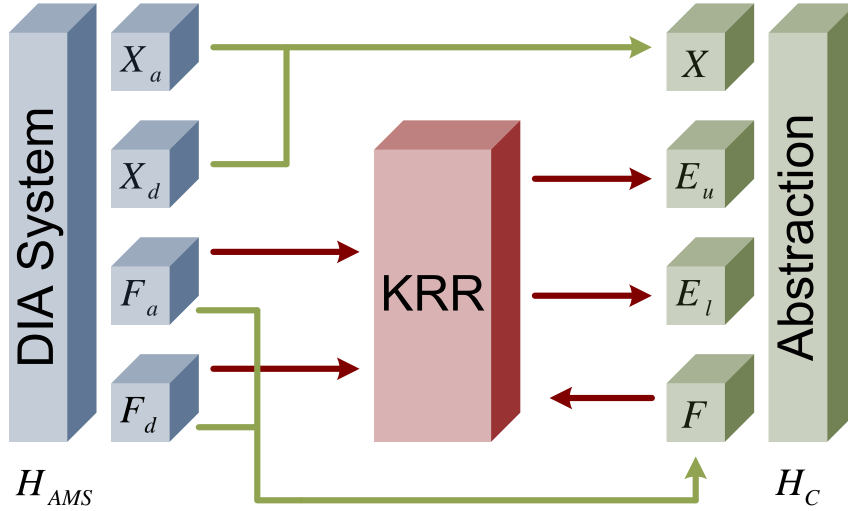


Figure 3.4: Error interval prediction via KRR.

variables are approximated by continuazation and ideal continuous characteristics like the linearized transition of the phase detector are used to approximate digital transitions. Then, for every extracted variables in X, KRR is trained to get the error prediction. After KRR training, the upper and lower bounds are obtained by (3.4) – (3.7).

3.2.4 State Space Representation with Support Functions

Support functions [18] are used to represent convex sets. In reachability analysis, such representations are leveraged in [85] to overapproximate the reachable space with a tight convex space. For a convex set $\Omega \subseteq \mathbb{R}^d$, the support function is defined for any arbitrary vector $l \in \mathbb{R}^d$:

$$\rho_\Omega(l) = \max_{x \in \Omega} l \cdot x \tag{3.8}$$

The support function of any vector l actually defines a space $l \cdot x \leq \rho_\Omega(l)$. The convex set Ω can be represented by the intersection of such spaces:

$$\Omega = \bigcap_{l \in \mathbb{R}^d} \{x \in \mathbb{R}^d | l \cdot x \leq \rho_\Omega(l)\} \tag{3.9}$$

If the support function value $\rho_\Omega(l)$ is computed for any $l \in \mathbb{R}^d$, the support function representation of a convex set is precise. In practice, it is impossible and also unnecessary to build a full-precision convex presentation by storing an infinite number of l vectors with their corresponding support function values. So, more practically, we sample a finite number of l_i from $\mathbb{R}^d$ and record $\rho_\Omega(l_i)$ to approximate the convex set. According to (3.9), the representation of Ω with k vectors is

$$\hat{\Omega} = \bigcap_{1 \leq i \leq k} \{x \in \mathbb{R}^d | l_i \cdot x \leq \rho_\Omega(l_i)\} \tag{3.10}$$

which is a polyhedron overapproximation of Ω. For example, the blue polygon in Fig. 3.5a is the support function representation of the red convex set with eight l_i vectors. If we increase the number of l_i vectors, as shown in Fig. 3.5b, the corresponding support function representation gets closer to the exact convex set.

3.3 Preliminary Approach Towards Efficient Reachability Analysis

This section proposes a preliminary method for the verification of abstracted systems based on a basic idea of divide and conquer. Developed on top of the simulation-assisted SMT framework, our proposed method divides the abstracted system into subsystems, demonstrating the potential advantages of the aforementioned DIA abstraction, the support function state space representations, and the system partitioning.

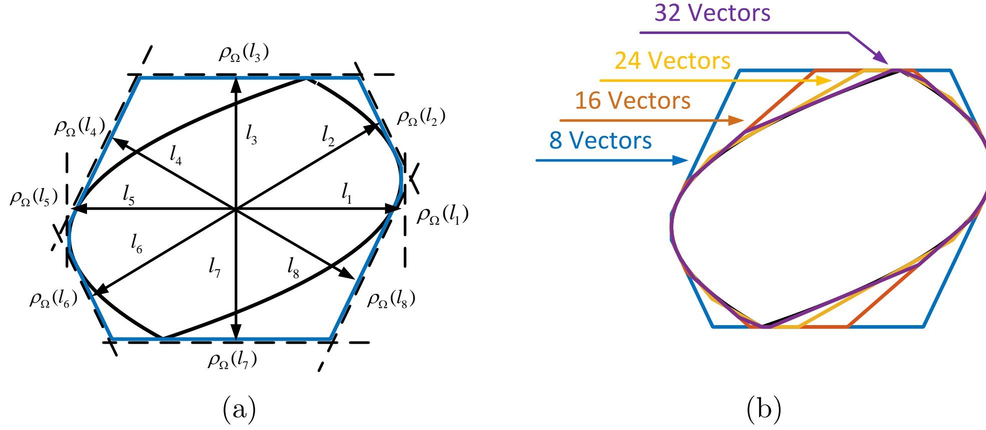


Figure 3.5: Support function representation.

3.3.1 Simulation-assisted SMT Method with Support Function Representations

In the simulation-assisted SMT method [145], there are two stages to compute the reachable state space $\Omega(t+\delta t)$ of the system at time $t+\delta t$ from its initial state space $\Omega(t)$ at time t. In the first stage, the method runs simulations to discover the bulk of the reachable states by discretizing the N-dimensional state space into N-dimensional boxes, namely N-orthotopes or hyper-rectangle, and combining all the boxes hit by simulation trajectories as an approximation $\tilde{\Omega}(t+\delta t)$ of the reachable state space $\Omega(t+\delta t)$.

Once $\tilde{\Omega}(t+\delta t)$ is computed, all those reachable states that fall outside $\tilde{\Omega}(t+\delta t)$ are found and added into $\tilde{\Omega}(t+\delta t)$ to make the approximation conservative during the second stage. The exploration of such reachable states can be converted into a satisfiability problem with the following conjunctions:

$$\mathbb{F} \wedge (X(t) \in \hat{\Omega}(t)) \wedge (X(t+\delta t) \notin \tilde{\Omega}(t+\delta t)) \tag{3.11}$$

where X is the state vector, and $\mathbb{F}$ denotes the transient transition relationship or

characteristics of the circuit model. Solving this satisfiability problem is to search a pair of $X(t)$ and $X(t+\delta t)$ that satisfies (3.11), meaning the existence of a certain trajectory that starts from the previous reachable state space at time t but results in a state outside the current approximation of the reachable state space at time $t+\delta t$. The satisfiability of (3.11) involves continuous state variables and thus is usually handled by SMT [14, 96] which is developed to solve the satisfiability problems with formulas containing both integer and real-valued variables.

Originally, the reachable state space is represented by discrete boxes in [145]. Such representation is flexible in the sense that it can describe an arbitrary state space regardless of whether the state space is convex or non-convex, and whether it is continuous or disjoint. However, such discretization suffers from the curse of dimensionality in its efficiency and its memory usage. Theoretically, the satisfiability problems have been proven to be NP-complete whose worst-case complexity is exponential. In SMT methods, the problem size n is roughly proportional to the number of variables, including both $X(t)$ and $X(t+\delta t)$, and the number of constraints. Using the box representation will blow up the number of constraints in the description of $X(t) \in \hat{\Omega}(t)$ and $X(t+\delta t) \notin \tilde{\Omega}(t+\delta t)$ in (3.11). Moreover, computing the combination of box representations in the first state may also lead to considerable excessive cost.

To remedy the problems posed above, support function representations can be utilized in the simulation-assisted SMT method to replace the box representations. One of the advantages is that the number of the l_i vectors in (3.10) is a trade-off between accuracy and efficiency, which is less vulnerable to the curse of dimensionality compared to the box representations.

Another advantage of the support function representation is its capability of efficient convex hull computation by recognizing the following property of support

functions:

$$\rho_{CH(U,V)}(l) = max(\rho_U(l), \rho_V(l)) \tag{3.12}$$

where U and V are two convex sets that are already represented by support functions, and $CH(U, V)$ denotes the convex hull of U and V. This property can be leveraged in the first stage where simulations are conducted to quickly approximate the bulk of the reachable state space. A reachable state hit by a simulation trajectory can be modeled as a small convex set (e.g. a point, a hypersphere or still a box) but represented by support functions, and the convex hull of all these convex sets can be efficiently computed to roughly approximate $\Omega(t + \delta t)$.

In addition, the number of the constraints occupying the descriptions of $X(t) \in \hat{\Omega}(t)$ and $X(t + \delta t) \notin \tilde{\Omega}(t + \delta t)$ in (3.11) is determined by the number of the l_i vectors. If the full system can be partitioned into multiple subsystems with each having a smaller number of state variables, it may require fewer l_i vectors in their support function representations and consequently lead to more efficient solutions of the corresponding SMT problems.

3.3.2 System Partition

After the DIA system H_{DIA} is converted to the abstracted system H_C, most of the digital components are linearized by the abstraction process and a large portion of F in H_C is linear. Since there exists effective methods for linear systems, partitioning the system into a linear portion and a nonlinear portion is beneficial in two aspects: (1) linear methods are applicable to the linear portion; (2) the number of variables in the nonlinear portion is reduced. This suggests a method to pick up variables from X that are involved in linear transitions to form a subset X_L and variables that are involved in nonlinear transitions to form a subset X_{NL}. If variables in X_L and X_{NL} are independent with each other, then $X_L \cap X_{NL} = \phi$. However, there

often exist variables in X_L correlated to variables in X_{NL}. Therefore, either subset should contain additional correlated variables to cover those correlations. Besides, F is divided into a set of linear transition mapping functions, F_L, and a set of nonlinear transition mapping functions, F_{NL}.

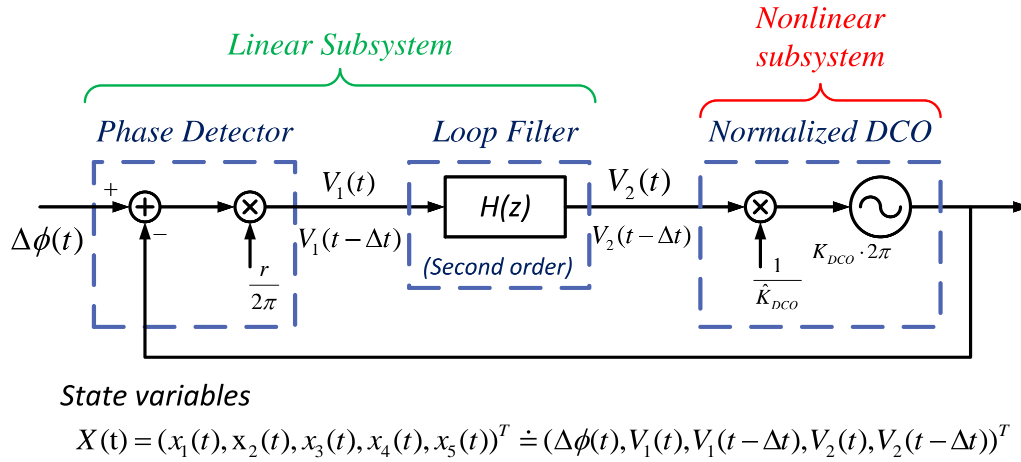


Figure 3.6: System partition of DI-PLL.

In the model of DI-PLL shown in Fig. 3.6, the phase detector and the loop filter are linearized with system abstraction while the DCO remains nonlinear. Assuming that the loop filter is a second order IIR digital filter, at time t, the state of the system is $X(t) = \{x_i(t)\}^T, i = 1, 2, 3, 4, 5$, where $x_1(t) = \Delta\phi(t)$, $x_2(t) = V_1(t)$,$x_3(t) = V_1(t-\Delta t)$,$x_4(t) = V_2(t)$, and $x_5(t) = V_2(t-\Delta t)$. From t to $t+\Delta t$, the transition of the system can be partitioned as a linear transition:

$$X(t) \to (x_2(t+\Delta t), x_3(t+\Delta t), x_4(t+\Delta t), x_5(t+\Delta t))^T \tag{3.13}$$

and a nonlinear transition:

$$(x_1(t), x_4(t))^T \to (x_1(t + \Delta t)) \tag{3.14}$$

After the partition, our method solves the reachable state space of the two subsystems with different methods. For the linear subsystem, once our method gets the constraint state space of t, it can directly compute the 4-dimensional reachable state space by support function space representation and corresponding properties. For the nonlinear subsystem, our method uses an SMT-based formal method to conservatively compute the reachable space. After exploring the reachable space of the two subsystems, our method compute the intersection of the two subspaces to obtain the reachable space of the entire system. Note that $x_1(t + \Delta t)$ is correlated with $x_2(t + \Delta t)$ and $x_5(t + \Delta t)$ since they are all determined by $x_1(t)$ and $x_4(t)$, the nonlinear subsystem reachable space should include x_1, x_2 and x_5, otherwise the intersection computation may cause great overapproximation.

3.3.3 Analysis of the Linear Subsystem

For the linear subsystem, let $X(t) \in \mathbb{R}^n$ denote the state of the whole system at t and $X_L(t) \in \mathbb{R}^m$ denote the state of the linear subsystem at $t + \Delta t$, then the transition can be formulated as:

$$X_L(t + \Delta t) = AX(t) \tag{3.15}$$

where the size of A is $m \times n$. To solve the reachable space of this system, support functions have a useful property to be exploited:

$$\rho_{AU}(l) = \rho_U(A^T l) \tag{3.16}$$

where $U \subseteq \mathbb{R}^d$ and A is an arbitrary matrix.

If the initial state space Ω_0 is represented by an accurate support function formula $\rho_{\Omega_0}(l)$, reachable spaces Ω_t at any moment can be solved by mapping any $l \in \Omega_t$ to $l' \in \Omega_0$ and hence an accurate support function representation is maintained. However, in the linear subsystem of DI-PLL, the formula form of the support function representation is difficult to use since the reachable state space computation contains the intersection operation between the linear and nonlinear reachable state space and vectors in Ω_t can only be mapped to $\Omega_{t-\Delta t}$. Hence, it is impractical to represent the reachable state spaces with symbolic formula of support functions.

Our approach, more practically, uses a finite number of normalized direction vectors l and their corresponding support function values to represent the reachable state space. As mentioned earlier, using a finite number of vectors, l, is actually producing a tight polyhedral over-approximation of Ω. For the convenience of the following computations, our method evenly samples normalized l from a d-dimensional spherical coordinate system.

To represent a reachable space Ω_t, our method stores a list of normalized vectors l_i and their corresponding support function values $\rho(l_i)$ and uses them to compute the support function values of $\Omega_{t+\Delta t}$. The problem is that for a vector $l \in \Omega_{t+\Delta t}$, $l_x = A^T l_x \in \Omega_t$ may not be stored in the representation of Ω_t. Therefore, our method uses the following algorithm to approximate $\rho_{\Omega_t}(l_x)$.

In a 2D space shown in Fig. 3.7, for an unknown vector l_x, our approach can first find two closest vectors from the evenly sampled known direction vector list. Denote the two vectors as l_1 and l_2, and the intersection point of planes $l_1 x = \rho(l_1)$ and $l_1 x = \rho(l_1)$ as v_x, according to the definition of support function, $\rho(l_x) \leq l_x \cdot v_x$. Thus our method can first compute the intersection point v_x and then use $l_x \cdot v_x$ to approximate $\rho(l_x)$.

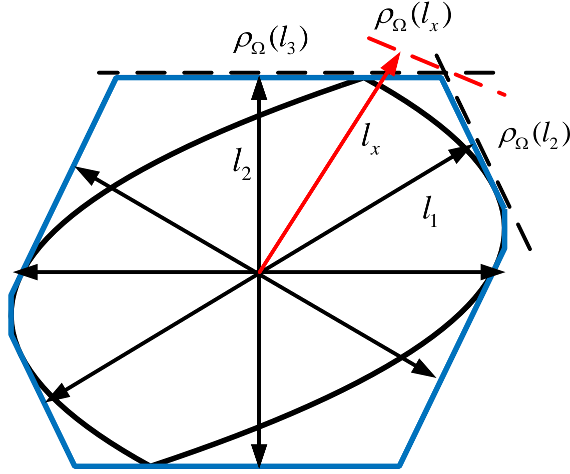


Figure 3.7: Approximate support function value of arbitrary direction l_x.

More generally, for arbitrary $l_x \in \mathbb{R}^d$, to approximate $\rho_\Omega(l_x)$ with a list of stored direction vectors and their corresponding support function values, our method can first select d closest vectors $l_1, ..., l_d$ from the direction vector list, and then compute the intersection point v_x of d corresponding hyperplanes. The selection of $l_1, ..., l_d$ can be easily achieved by mapping l_x into a spherical coordinate system since the stored vectors are sampled evenly from that system. Since v_x lies on the boundary of the corresponding half space of $l_1, ..., l_d$, it can be computed by solving:

$$\begin{bmatrix} l_1^T \\ l_2^T \\ \vdots \\ l_d^T \end{bmatrix} v_x = \begin{bmatrix} \rho_\Omega(l_1) \\ \rho_\Omega(l_2) \\ \vdots \\ \rho_\Omega(l_d) \end{bmatrix} \tag{3.17}$$

After v_x is computed, $\rho_\Omega(l_x)$ can be approximated by $v_x \cdot l_x$. With this algorithm, the reachable state space of the linear subsystem can be solved with a support function representation of finite direction vectors.

To make the reachable state space conservative, error intervals obtained by the KRR-based method should be considered. Since the system transition is linear, the reachable points of the states on the boundary of the current space still appear on the boundary of the reachable space, our method can simply extend the boundary according to error intervals of on-boundary states.

3.3.4 Analysis of the Nonlinear Subsystem

For the nonlinear subsystem, $X_{NL}(t+\Delta t) = F_{NL}(X_{NL}(t))$, our method leverages the SMT-based framework discussed in Section 3.3.1 to explore the reachable state space. At first, simulations are used to quickly explore the bulk of the reachable state space, and then an SMT solver is invoked to find the remaining reachable state space. The flow has been improved by using support functions as the state space representation partially due to the capability of efficient convex hull computation described in (3.12).

In the first stage, each time a reachable state is generated from a simulation, it is enlarged into a small convex set according to its error interval estimation given by the previous KRR-based method. Then, our method computes the convex hull of the current reachable state space and this small convex set to update the reachable space. After a bunch of simulations, the result covers most of the reachable state space. After that, our method uses an SMT solver to check whether the reachable state space is completely covered or not. If not, the solver provides a counterexample that is reachable but still not covered in the current approximation. The counterexample is added into the convex hull and SMT checking is repeated until the solver cannot find any counterexample or, in other words, the satisfiability problem (3.11) is unsatisfiable. The final convex hull conservatively covers the reachable state space.

For a DIA system, the partitioned nonlinear subsystem is expected to have much

fewer state variables compared to the full system since a majority of the digital components will be linearized in the abstraction. As a result, the complexity of applying the simulation-assisted SMT method to the subsystem is much lower than applying the method to the entire system.

3.3.5 Hybrid Reachability Analysis

In the complete flow of the hybrid reachability analysis, after a simplified model is produced using abstraction, the system is partitioned to obtain a linear and a nonlinear subsystem. Then, the support function based method is employed to solve the linear subsystem and the SMT-based method is used to explore the reachable space of the nonlinear subsystem.

After the reachable state space exploration of the two subsystems are completed, the intersection of the two subspaces should be computed to obtain the reachable space of the whole system. At first, our method should extend the two subspaces to the same dimension, which can be performed in a similar way as approximating $\rho(l_x)$ for an unrecorded l_x. After that, for convex sets represented by support functions, the intersection can be achieved by:

$$\rho_{U \cap V} = min(\rho_U(l), \rho_V(l)) \tag{3.18}$$

As discussed in [51], the *min* operation can be used to compute the intersection of the two convex sets, but the result is not a strictly defined support function due to possible redundant halfspaces. Even so, the system partition based method inevitably introduces overapproximation in the reconstruction of the high dimensional state space of the full system. In addition, to maintain the correlations between the linear and the nonlinear subsystems, state variables that interact between the two subsystems are carefully selected and added into both subsystems, which requires

additional manual work and expertise. Nevertheless, this hybrid method shed light on the potentials of tacking the reachability analysis via system partition. An enhanced system decomposition will be introduced in the next section to address the challenges occurred here.

3.4 Proposed System Decomposition

While error accumulation is a critical problem in reachability analysis, the aforementioned overapproximation problem in the state space reconstruction needs to be addressed before a partitioned based method is applied to the reachability analysis framework.

One of the key contributions of this work is a technique that rigorously deals with correlations between the partitioned subsystems and completely removes the undesirable overapproximation in the full system state space reconstruction by operating on extended state variable vectors. By adopting polyhedral convex sets based support function representations [51],[52],[85] to present the reachable state spaces, what is novel in this work is that we show that special forms of support functions can be chosen to ensure the equivalence between the full system state space and the one that is implicitly represented by the support functions of the subsystems. This important development allows us to efficiently reconstruct the full system state space without any overapproximation.

3.4.1 Overapproximation Error

For any circuit with n state variables, we define the state of the system at time t as:

$$X(t) = (x_1(t), x_2(t), ..., x_n(t))^T \in \mathbb{R}^n \tag{3.19}$$

In analog circuits, the state transition towards $X(t)$ can be tracked by using the information of the previous state $X(t-\delta t)$ in a way that corresponds to the application of forward Euler method in circuit simulation:

$$X(t) = F(X(t-\delta t), u(t)) \tag{3.20}$$

The state evolution of (3.20) can be leveraged to explore the reachable state space over the time. It is clear that partitioning the circuit into smaller pieces would speedup the process. However, care must be taken to avoid overapproximation errors when reconstructing the reachable space for the full system.

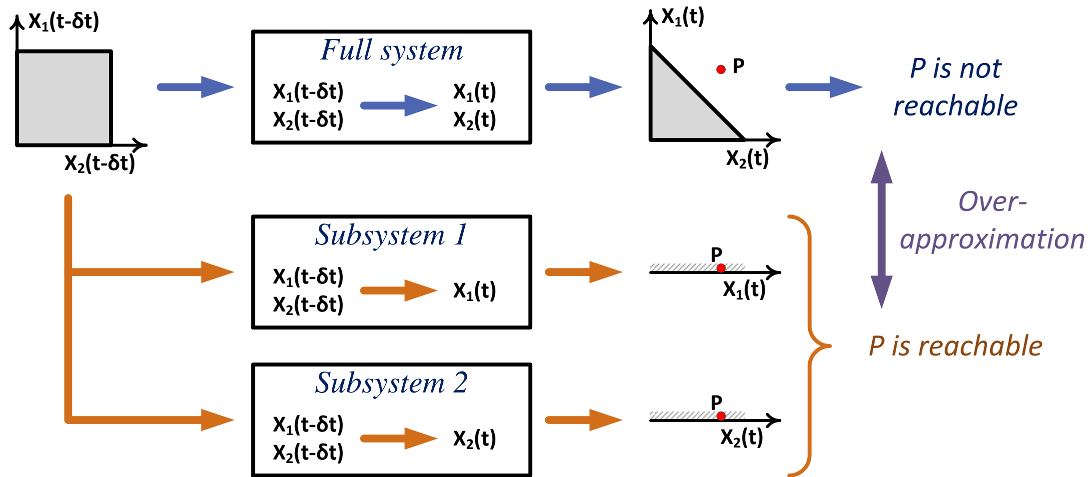


Figure 3.8: Over-approximation of manifest system partition.

Without loss of generality, let us consider partitioning the circuit such that each state variable $x_i(t)$ corresponds to the only state variable of a subsystem. This implies that a system with n state variables will be partitioned into n subsystems. For example, a two-dimensional system is shown in Fig. 3.8 and the original state

transition is a $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ mapping. One may partition this 2-d system into two 1-d subsystems and separately compute the reachable state space for each of the two state variables at time t given the initial state space of the full system at $t - \delta t$. This leads to the two 1-d reachable spaces shown in Fig. 3.8. In contrast with the actual 2-d reachable space of the full system, one can see that an unreachable state P may be mistakenly included in the reachable subspaces, i.e. overapproximation error happens with the system partition. The immediate reason for this error is that the correlation between the two state variables is not properly captured in this manifest system partition method.

3.4.2 Concepts of Extended State Spaces

The above problem is clearly significant. Such error can be easily accumulated over time and grow much larger. A close examination of the correlation problem from a dynamical system point of view reveals that the correlations stem from the fact that current states of different subsystems commonly depend on the previous state variables of the full system. This key observation motivates us to capture such correlations very naturally by building support function state space representations based on extended state variable vectors, i.e., vectors consisting of state variables at two adjacent time points. We start by defining the extended state vector X(t) for the full system of n state variables:

$$\begin{aligned} X_e(t) &= (X(t), X(t-\delta t)) \in \mathbb{R}^{2n} \\ &= (x_1(t), ..., x_n(t), x_1(t-\delta t), ..., x_n(t-\delta t)) \end{aligned} \tag{3.21}$$

Denote the reachable extended state space of the full system by Ω_e. For the k-th subsystem produced by the above partition, its extended state vector is defined as:

$$\begin{aligned} X_{ek}(t) &= (x_k(t), X(t-\delta t)) \in \mathbb{R}^{n+1} \\ &= (x_k(t), x_1(t-\delta t), ..., x_n(t-\delta t)) \end{aligned} \tag{3.22}$$

whose reachable state space is denoted by Ω_{ek}.

Our key idea is to perform reachability analysis in the extended state spaces of the full system and each subsystem so as to address the above overapproximation problem. Considering that at some time $(t-\delta t)$, the reachable extended state space of the full system $\Omega_e(t-\delta t)$ is already computed. From $\Omega_e(t-\delta t)$, we can easily extract out the reachable state space $\Omega(t-\delta t)$ of the full system at time t. As shown in Fig. 3.9, to move onto the next time point t, we perform a separate reachablity analysis for each subsystem using the full system $\Omega(t-\delta t)$ as its initial conditions. However, to track the correlations across all subsystems, the result of each subsystem reachability analysis is cast in its reachable extended state space, i.e. by computing $\Omega_{ek}(t)$. Since each $\Omega_{ek}(t)$ captures the new reachable space for the k-th state variable x_k at time t and its correlation with $\Omega(t-\delta t)$ (common initial conditions for all subsystems), they can be combined to precisely reconstruct the full system reachable extended space $\Omega_e(t)$ at time t. The same process repeats.

Considering that $\Omega(t-\delta t)$ (marked as S in Fig. 3.9) is covered among all the spaces in Fig. 3.9, there exists one-to-one correspondence between the reachable states in $\Omega_e(t)$ and the ones in $\Omega_{ek}(t)$ since they have common corresponding projection in $\Omega(t-\delta t)$. Moreover, it can be easily proved by contradiction that any unreachable state in $\Omega_e(t)$ is also unreachable in $\Omega_{ek}(t)$. Hence the combination of all the $\Omega_{ek}(t)$ is an accurate representation of $\Omega_e(t)$.

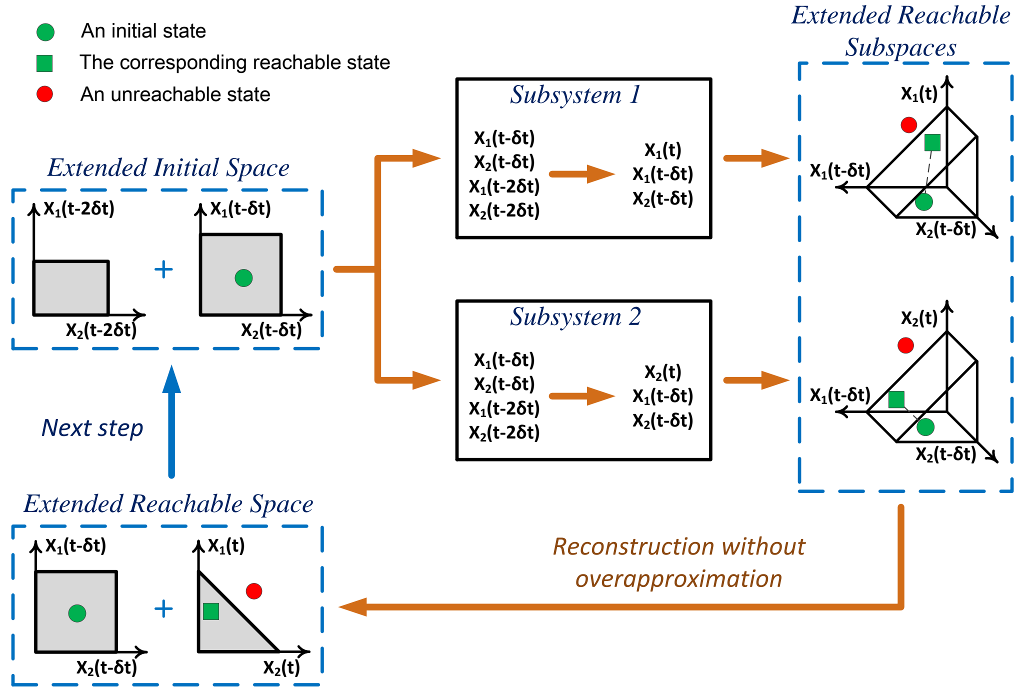


Figure 3.9: Reachability analysis on extended decomposed state spaces for a 2D system.

The actual implementation of the above process involves manipulation of proper support function representations of both full system and subsystems, which is described as follows.

3.4.3 Support Function Representations of Reachable Extended State Spaces and Their Manipulation

We describe the support function representations involved in the above process. The reachable extended state space of the full system $\Omega_e(t-\delta t)$ at $(t-\delta t)$ is represented using a set of vectors of the form:

$$l_{ei}^{(t-\delta t)} = (0, ..., 0, l_{ei,1}^{(t-\delta t)}, ..., l_{ei,n}^{(t-\delta t)}) \in \mathbb{R}^{2n} \tag{3.23}$$

where $l_{ei,k}^{(t-\delta t)}$ denotes the (n+k)-th entry of $l_{ei}^{(t-\delta t)}$ which can be non-zero. This particular representation of $\Omega_e(t-\delta t)$ essentially provides the (un-extended) reachable state space of the full system $\Omega(t-\delta t)$ at time $t-\delta t$. Using $\Omega(t-\delta t)$ as the initial conditions, we perform a reachability analysis of each k-th subsystem and compute the reachable extended state space $\Omega_{ek}(t)$ for the subsystem at t using a support function representation based on a set of vectors $l_{si} \in \mathbb{R}^{n+1}$.

To project $\Omega_{ek}(t)$ to the extended state space of the full system while keeping the corresponding support function value unchanged, we simply extend each vector $l_{si} \in \mathbb{R}^{n+1}$ to a corresponding vector in $\mathbb{R}^{2n}$:

$$l_{esi,k} = (0, ..., l_{si,1}, ..., 0, l_{si,2}, l_{si,3}, ..., l_{si,n+1}) \tag{3.24}$$

where $l_{si,j}$ is the j-th entry of l_{si} and $l_{si,1}$ appears at the k-th entry of $l_{esi,k}$. The key observation here is that these projected subsystems reachable extended spaces $\Omega_{ek}(t)$ collectively provide a support function representation of the full system reachable extended space $\Omega_e(t)$ based on a specific set of vectors of the form of (3.24).

At this point, we have indeed computed the desired $\Omega_e(t)$. However, it is not in the right form to repeat the same process for the next time point. To continue, it is desirable to map $\Omega_e(t)$ from the form in (3.24) to a support function representation based on the following set of vectors:

$$l_{ei}^{(t)} = (l_{ei,1}^{(t)}, ..., l_{ei,n}^{(t)}, 0, ..., 0) \in \mathbb{R}^{2n} \tag{3.25}$$

where $l_{ei,k}^{(t)}$ similarly denotes the k-th non-zero entry of $l_{ei}^{(t)}$. Essentially such mapping is to compute the support function values under $l_{ei}^{(t)}$ of (3.25) from the support function values under $l_{esj,k}, k = 1, ..., n$ of (3.24). According to (3.8), if the subspace

representation of any k-th subsystem Ω_{ek} is constructed under l_{si}, without going in the detail it can be shown that the computation of $\rho_{\Omega_e}(l_{ei}^{(t)})$ can be be formally achieved by solving the *linear programming* (LP) problem:

$$\begin{aligned} &\max_{X_e(t)} l_{ei}^{(t)} \cdot X_e(t) \\ s.t. \quad & l_{esj,k} \cdot X_e(t) \leq \rho_{\Omega_{ek}}(l_{sj}) \end{aligned} \tag{3.26}$$

where $k = 1, ..., n$ and $j = 1, ..., m$ if the number of vectors l_{sj} is m.

Note that (3.26) only provides a formal mechanism to illustrate the transformation of support function representations. There exist more efficient practical workarounds for our reachability analysis flow as described in the next section.

3.5 Proposed Parallel Algorithm

We develop our parallel reachability analysis based upon the simulation-assisted SMT framework discussed in Section 2.2. In this framework, a set of circuit simulation runs are used to discover a bulk part of the reachable state space at a given time. A convex hull using the support function representation is computed to enclose the reachable state space discovered by simulations. Then, a set of SMT runs are launched to possibly find additional points in the reachable state space that are not enclosed by the convex full. Finally, a conservative convex hull representation of the full reachable state space is computed by enclosing the parts of the reachable space found by both simulation and SMT.

To achieve the best possible efficiency, as shown in Fig. 3.10, we speed up both the simulation and SMT checking tasks with two levels of parallelism. The first level of parallelism is a direct result of the proposed system decomposition. At this level, a number of simulations are launched in parallel to quickly approximate the reachable state spaces of the extended full system and all the subsystems. Then,

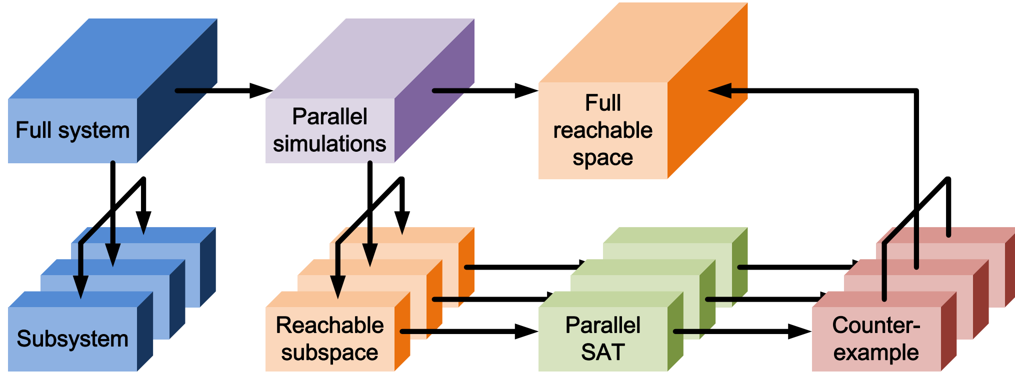


Figure 3.10: The proposed reachability analysis flow with two-level parallelism.

multiple SAT (SMT) checks are invoked for all subsystems simultaneously to search for the missing reachable states. If such states are found, they will be added into the reachable spaces of both their subsystems and the extended full system. The SAT checking process continues until all the reachable states are discovered. At the second level, we solve each SAT problem in parallel to gain additional speedups. These two levels of parallelism are described in detail as follows.

3.5.1 Subsystem-level Parallelism

After the full system is extended and decomposed by the proposed method, it is straightforward to apply the SMT-based reachability analysis to all the subsystems in parallel within one time step. Since such decomposition inherently simplifies the $\mathbb{R}^n \to \mathbb{R}^n$ state transition of the full system into a set of $\mathbb{R}^n \to \mathbb{R}$ state transitions, i.e. the number of variables involved in the SAT problems is reduced from $2n$ to $(n+1)$, the efficiency speedup of the subsystem level parallelism is promising.

At first, our proposed method explores the reachable extended state spaces of all the subsystems simultaneously via simulations. Then, using these simulation points, the convex hull approximations of the reachable extended state spaces of all subsys-

tems and the full system are computed in parallel. Parallel SAT checks are launched to search for any possible missing reachable points which are used to expand the convex hull approximation in the corresponding subsystems and, at the same time, the one of the full system. The final convex hull of the full system is computed based on inexpensive operations in (3.12) instead of costly solutions of the LP problems. Leveraging the equivalence between the support function representations of the extended system and its subsystems, our proposed method dramatically reduce the cost of the full system state space reconstruction.

3.5.2 SAT-level Parallelism

While our SMT checking is applied to the reachable extended state spaces, for ease of discussion, we use the processing of reachable state spaces to illustrate the same idea. According to (3.10), if $\tilde{\Omega}(t)$ and $\tilde{\Omega}(t+\delta t)$ is constructed under the same group of vectors $l_i, i = 1, ..., k$, then $X(t+\delta t) \notin \tilde{\Omega}(t+\delta t)$ is a clause of:

$$\bigvee_i l_i \cdot X(t+\delta t) > \rho_{\tilde{\Omega}(t+\delta t)}(l_i) \tag{3.27}$$

Our method further decomposes each SAT problem into k subproblems. The j-th subproblem can be formulated as:

$$\mathbb{F} \wedge (X(t) \in \hat{\Omega}(t)) \wedge (\bigvee_i l_i \cdot X(t+\delta t) > \rho_{\tilde{\Omega}(t+\delta t)}(l_i)) \tag{3.28}$$

where m is the number of l_i vectors and $i \in [\frac{m(j-1)}{k}, \frac{mj}{k})$. The subproblem reduces the number of literals in the clause, i.e. the number of l_i vectors, from m to m/k, leading to significant efficiency improvements.

Another benefit of solving these subproblems in parallel is the possibility of simultaneously finding multiple reachable states that are missed by the current convex

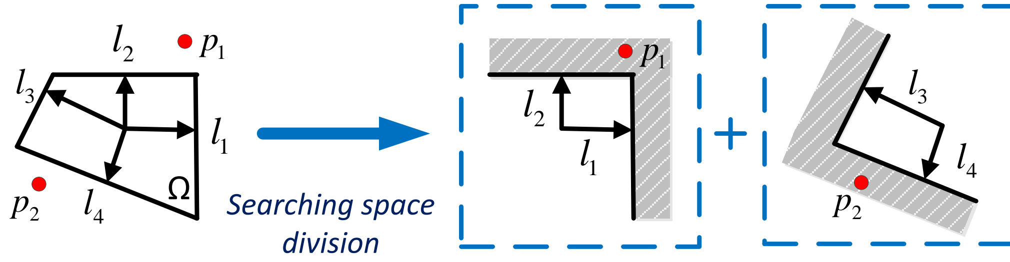


Figure 3.11: Parallel counterexample exploration.

approximation. For example, as shown in Fig. 3.11, Ω is the approximation of the reachable space under support function vectors l_1, l_2, l_3 and l_4. The serial SMT method can only find out one satisfiable solution, i.e. either p_1 or p_2 outside Ω, at a time. If the SAT problem is divided into two subproblems, one under l_1 and l_2 and the other under l_3 and l_4, then the two subproblems explore solutions in different areas (the shaded areas shown in Fig. 3.11). As a result, the parallelized search in multiple subproblems tends to produce multiple solutions at one time. For example, both p_1 and p_2 in Fig. 3.11 can be captured by the two subproblems simultaneously.

3.6 Experimental Results

We demonstrate the efficiency improvement of the proposed parallel algorithm by verifying a PWM DC-DC converter and the nonlinear analog behavioral model of a *digitally-intensive phase-locked loop* (DI-PLL)[85].

3.6.1 PWM DC-DC Converter

A simplified model of a DC-DC converter is shown in Fig. 3.12, with R_L and R_C representing the parasitic resistance of inductor L and capacitor C, respectively. The power switch is controlled by the PWM waveform to keep charging or discharging the LC filter repeatedly, and hence generating both output voltage and inductor current ripples. Assumeing that the switch, the voltage source, the PWM control module,

and the feedback mechanism are all ideal components with perfect characteristics, the state of the system can be represented by two variables: the current i through the inductor and the output voltage v.

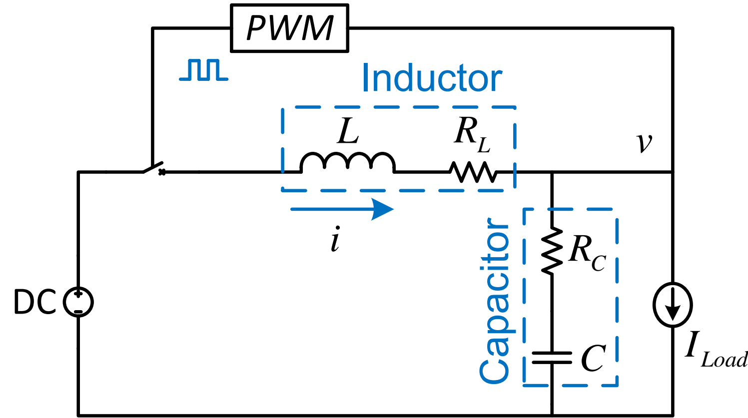


Figure 3.12: Simplified model of a DC-DC converter.

The target of this experiment is to verify the settling time of v after the load current i_{load} is doubled, which reflects one of the key specification of the DC-DC converter. Due to the ripple and other forms of uncertainty associated with the circuit, neither v nor i can be considered as fixed value when the change in i_{load} is applied. Therefore, to conservatively verify the settling time, we start with a set of v and i as the initial state space and track the reachable space of all possible circuit trajectories from this initial state space.

Fig. 3.13a shows the same conservative bounds of v that are computed by both the sequential and parallel implementations of the proposed reachability analysis. The bounds tightly enclose many sets of simulation trajectories, implying a settling time smaller than $10\mu s$. The initial state space before the change of i_{Load} is shown in Fig. 3.13b, where the red convex set is the actual initial space and the blue polygon is

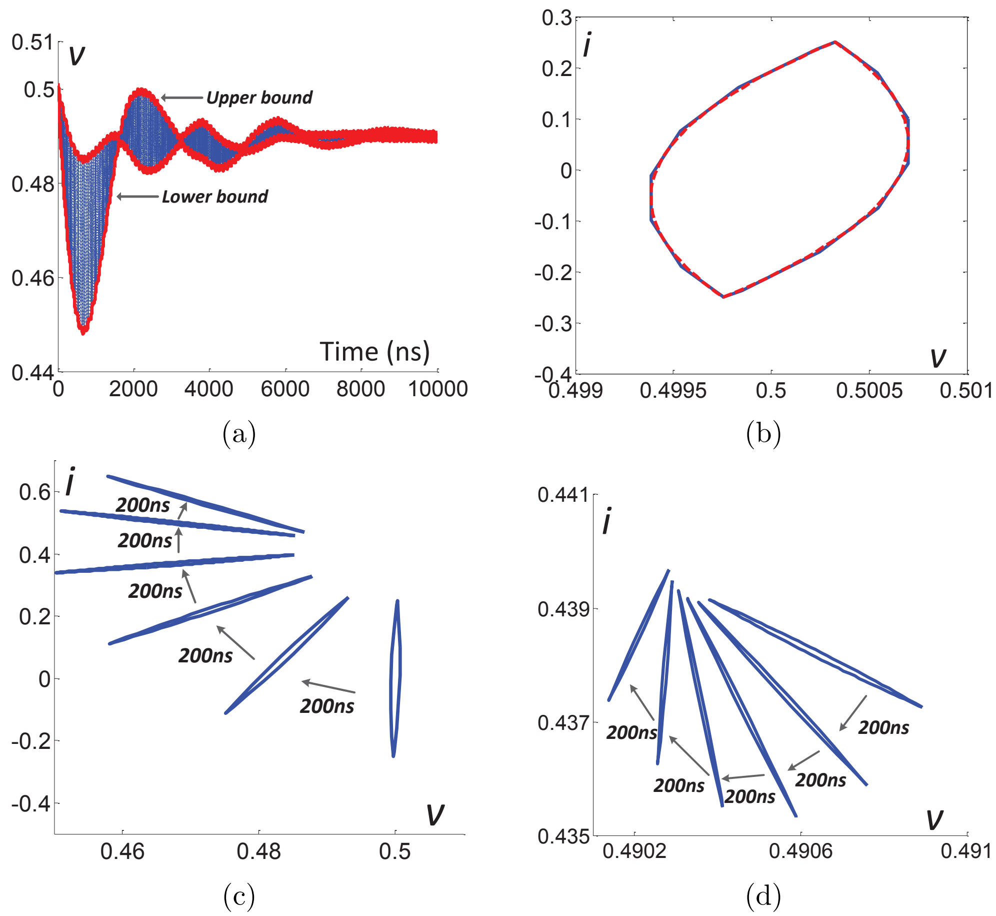


Figure 3.13: Transient reachability analysis: (a) simulation trajectories and the bounds of v; (b) initial state space; (c) reachable space evolution in the first $1\mu s$; (d) reachable space evolution in the last $1\mu s$.[83]

the support function approximation. Fig. 3.13c shows the evolution of the reachable state space in the first $1\mu s$ after the variation of i_{Load} during which the variation on v is greatly increased compared to the ripple of the initial state space. Fig. 3.13d indicates that in the last $1\mu s$ of the transient experiment, the ripple of v become smaller than 0.001 volts, meaning v settles down eventually.

This simple example is only used to demonstrate the basic working the proposed techniques. Next, we consider a much more challenging verification problem.

3.6.2 PLL

Digital PLLs have been a popular choice for clock generation and RF communication applications due to their inherent robustness to process variations [116]. While a DI-PLL may have a significant digital content, it inevitably has an analog feedback path from the *digitally controlled oscillator* (DCO) to the *phase detector* (PD) and its overall loop behavior is similar to analog type (e.g. charge-pump based) PLLs. In order to demonstrate the verification of general analog circuits, we extract an equivalent nonlinear analog model (Fig. 3.14) out of the DI-PLL to test the proposed algorithms.

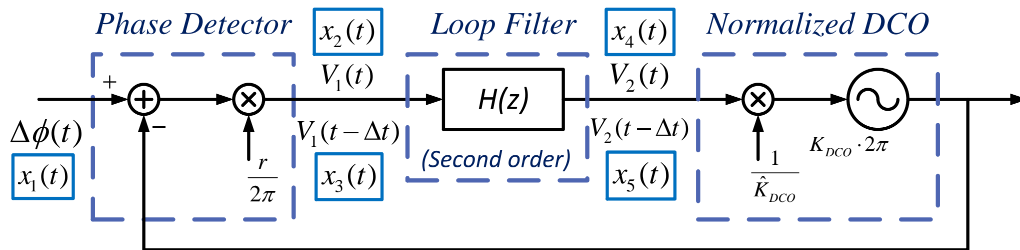


Figure 3.14: The analog behavioral model of PLL.

Assuming that the loop filter is a second order IIR filter and the transition char-

acteristic of all the functional blocks are continuous, the extracted analog behavioral model of the PLL and the state variables needed to describe its state space are shown in Fig. 3.14. The state vector of the model at time t is defined as:

$$\begin{aligned} X(t) &= (x_1(t), x_2(t), x_3(t), x_4(t), x_5(t)) \in \mathbb{R}^5 \\ &= (\Delta\phi(t), V_1(t), V_1(t-\Delta t), V_2(t), V_2(t-\Delta t)) \end{aligned} \tag{3.29}$$

The goal of the experiment is to verify if the PLL can reach a locked state within a given lock-time specification if started from an arbitrary state from a given initial state set, where the initial phase difference $x_1(0)$ may vary from $-\pi$ to π. Verification of the lock-in time of a nonlinear PLL under a wide range of initial condition uncertainty is a computationally demanding task as the nonlinear circuit trajectories must be tracked over a large number of time points. While this circuit has five continuous state variables, it is not a trivial target for many existing formal techniques. For the simulation-assisted SMT-based reachability analysis proposed in [145], it takes hours to verify a charge-pump PLL with 3 continuous variables, by discretizing the state space with a resolution of 100^3 cubes and performing SMT checking for $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ mappings. In Fig. 3.14, the model has a higher dimensional state space which requires 100^5 cubes to achieve similar resolution and performs more complex SMT checking for $\mathbb{R}^5 \rightarrow \mathbb{R}^5$. With the above setups, this lock-in time verification is extremely computationally challenging and becomes a good test for the proposed techniques

In this experiment, several different configurations of SMT-based verification are implemented in C/C++ and run on a Linux server with 24 AMD Opteron(TM) 2.2GHz processors to analyze the speedups of the proposed techniques. The multi-threaded parallel programs are implemented in OpenMP and iSAT [48] is invoked as

the SMT solver.

When solving a single SAT problem, as mentioned previously, the speedup of the system decomposition mainly comes from the reduction of state variables involved in each SAT problem. To focus on the effect of the state variable reduction, we compare the speedups of solving the conjunctions of $\mathbb{F} \wedge (X(t) \in \hat{\Omega}(t))$, which is a chunk in (3.11) for easy manipulation, when different number of variables is reduced from $\mathbb{F}$. For the SAT-level parallelism, the speedup comes from the reduction of the number of literals in (3.27). We divide the SAT problem of the entire PLL model into different number of subproblems and compare their speedup below.

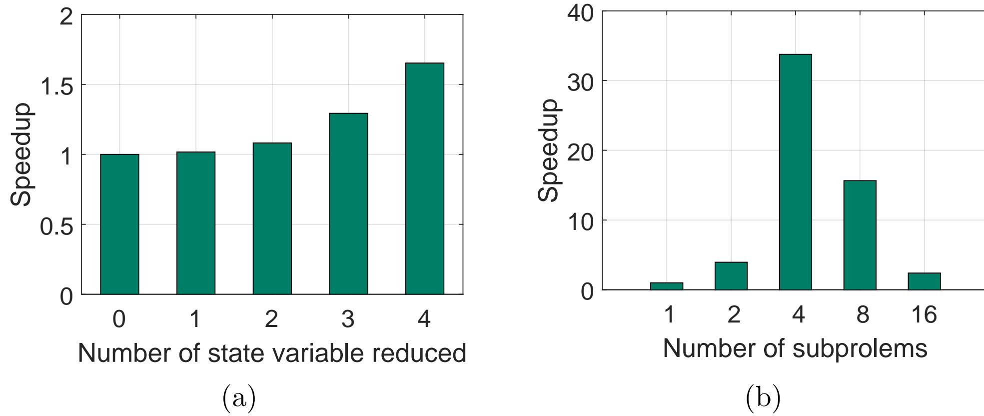


Figure 3.15: Speedup in solving a single SAT problem: (a) system decomposition; (b) SAT parallelism.

As shown in Fig. 3.15, the speedup in solving a single SAT problem by system decomposition tends to grow superlinearly when the number of involved SAT variables is reduced. And the SAT-level parallelism also shows a significant speedup by dividing the SAT problem into four subproblems. Note that the speedup may be worse for deep parallelized subproblem which is likely to be unsatisfiable and hence

takes the SAT solver more time to reach the solution. However, even if the subproblem takes the same run time as the original SAT problem, our framework can still benefit from its capability of producing multiple satisfiable solutions simultaneously.

Now we combine the two strategies and analyze the speedup of the entire parallel flow using various numbers of threads. Since the serial reachability analysis flow for the PLL model shown in Fig. 3.14 cannot run to a completion within one week, it is impractical to compare the total runtime of the serial flow and the parallel flow. Therefore, we compute the speedup by comparing the runtime of the first five time steps of different flows. For the serial flow, it takes 30 hours to perform reachability analysis for the first five time steps.

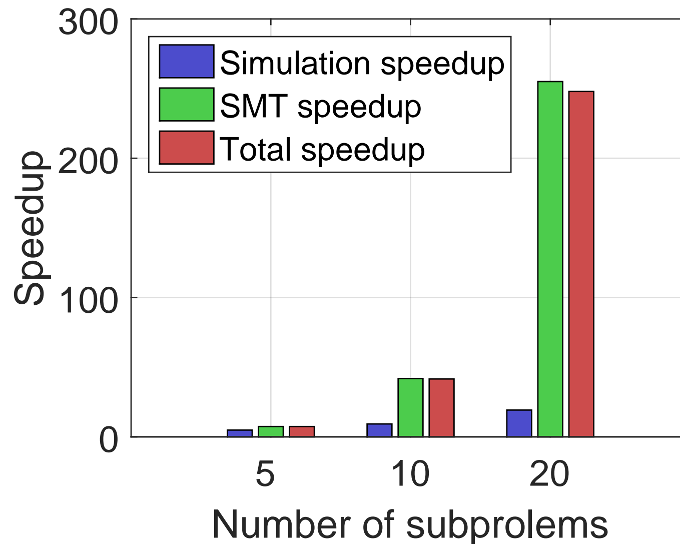


Figure 3.16: Speedup of the entire flow for various numbers of threads [83].

Fig. 3.16 shows the speedups of different parallel implementations. Here we decompose the system into five subsystems via the proposed methodology. For the first column with five threads, each subsystem is solved by one thread, meaning that only

the subsystem level parallelism is adopted, which speed up the process by 7.4 times. In the second column in Fig. 3.16, two threads are allocated to each subsystem, which means the SAT problem of each subsystem is divided into two subproblems to be solved in parallel. The system decomposition and the two levels of parallelism all contribute to the efficiency improvement and together produce a significant speedup of 42X. Similarly, in the third column in Fig. 3.16, each subsystem is solved with four threads, achieving a speedup of 248X.

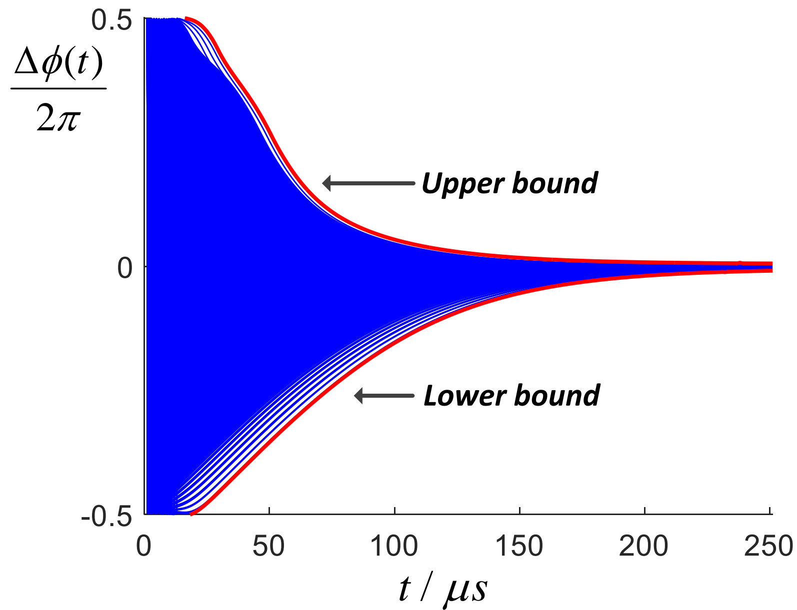


Figure 3.17: Convergence of the reachability analysis on $\Delta\phi(t)$ and the PLL locktime verification.

We use the parallel implementation with 20 processors to verify the lock-time specification of the extracted analog model of the PLL. The results of the reachability analysis on one of the state variables $\Delta\phi(t)$ are shown in Fig. 3.17 where the red curves represent the upper and lower bound of the phase difference and the dense blue

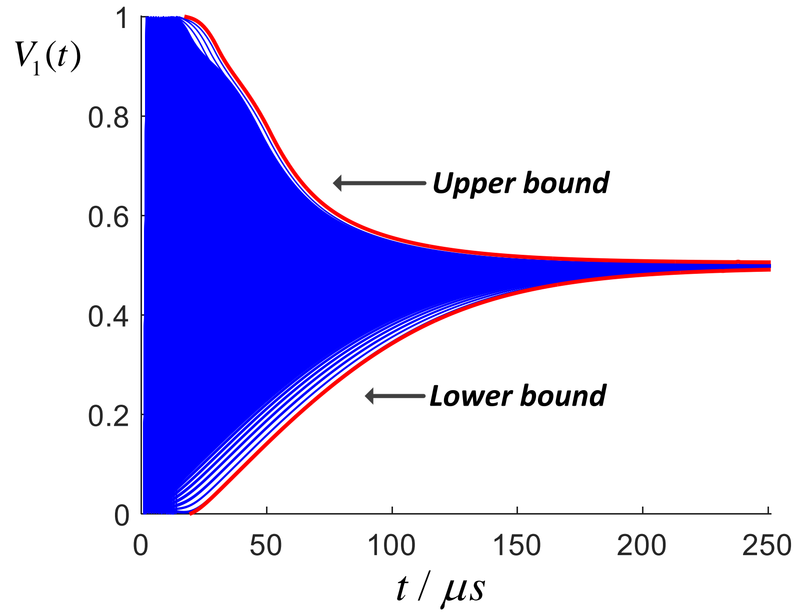


Figure 3.18: Convergence of the reachability analysis on $V_1(t)$.

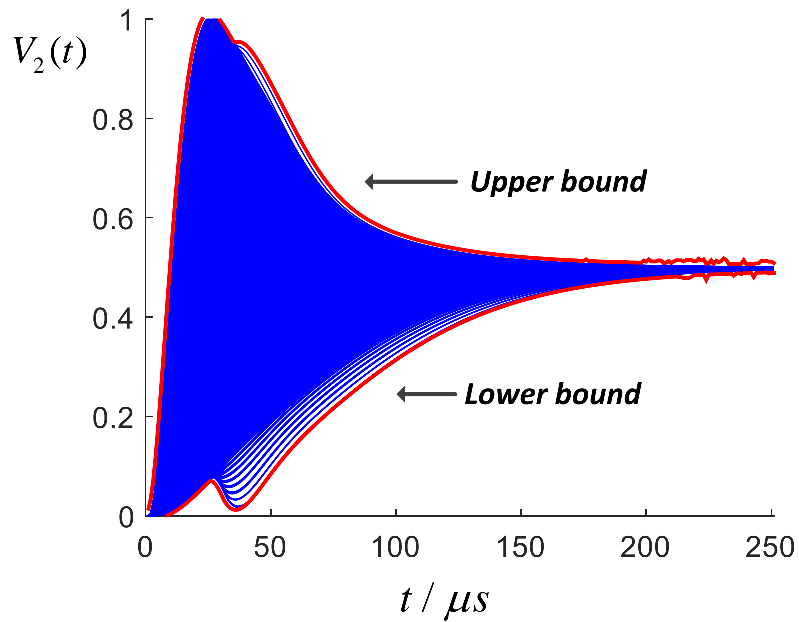


Figure 3.19: Convergence of the reachability analysis on $V_2(t)$.

traces are the simulation trajectories, which are bounded by the two bounds tightly. It is indicated that the PLL can lock within 0.25 μs with an arbitrary initial phase difference based on the convergence of the phase difference $\Delta\phi(t)$. The convergence of $V_1(t)$ illustrated in Fig. 3.18 and the convergence of $V_2(t)$ demonstrated in Fig. 3.19 also lead to the same conclusion.

3.7 Summary

This section proposes a parallel hierarchical reachability analysis method to significantly improve the efficiency of analog verification. The general system decomposition method is proposed to decompose an analog system into less complex subsystems without any reconstruction overapproximation. In addition, a two-level parallel algorithm is implemented to decrease the complexity of the SAT problems and further speedup the verification by finding out multiple satisfiable solutions simultaneously.

4. ACTIVE LEARNING GUIDED SUPPORT VECTOR MACHINE BASED CIRCUIT VERIFICATION *

Even though the efficiency of the reachablity analysis is greatly improved by system decomposition and parallelization in Section 3, complex nonlinear AMS systems are still often beyond its capability. Since leveraging machine learning has been proven as a promising avenue for addressing many practical simulation-based circuit challenges, developing simulation-based AMS verification under the context of machine learning may produce promising performance. This section demonstrates a novel active learning guided machine learning approach for circuit performance characterization and verification. When employed under the context of support vector machines, the proposed probabilistically weighted active learning approach is able to dramatically reduce the size of the training data, leading to significant reduction of the overall training cost. The proposed active learning approach is extended to the training of asymmetric support vector machine classifiers, which is further sped up by a global acceleration scheme. The excellent performance of the proposed techniques is demonstrated by three case studies: DC/DC converter ripple noise analysis, PLL lock-time verification, and prediction of chip peak temperature using a limited number of on-chip temperature sensors.

4.1 Introduction

Understanding circuit performances' dependencies on key design, process and operating condition parameters is a central issue in many phases of IC development. For

*Reprinted with permission from "Circuit Performance Classification With Active Learning Guided Sampling for Support Vector Machines" by Honghuang Lin and Peng Li, 2015. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 34, Page 1467-1480, ©2015 IEEE.

instance, design equations or circuit performance models that capture specifications' dependency on design parameters are essential for guiding design optimization.

However, as a by-product of design complexity increase and technology scaling, design, process parameters and operating conditions interact with design performance and operation in an increasingly complex manner. Optimizing design performance and safeguarding design robustness becomes a significant challenge.

In this light, the ability in characterizing circuit performance in the complex design/technology/operating parameter space is key to design, verification and test. In practice, performance characterization process often entails collecting and processing large volumes of simulation or measurement data, which can be extremely costly and time consuming. To address the above challenge, we leverage a specific type of machine learning techniques, support vector machines (SVMs) [30, 129], for efficient characterization of design performance.

Machine learning has been adopted in EDA research in the past. Voting based methods are used in [88] and [33] to train analog performance models parameterized in design parameters. In [33, 41], one-class SVM is adopted to represent analog circuit performance and perform outlier analysis for cost reduction of delay test. In [91, 15], regression techniques are used to analyze circuit reliability and rank design features that contribute to unmodeled systematic timing effects. Focusing on a somewhat different problem, the authors of [110] combine the extreme value theory and machine learning (e.g. support vector machine) for yield estimation of SRAMs. The same problem is approached by several other authors through fast Monte-Carlo importance sampling techniques or boundary searching based non-Monte-Carlo methods (see, for example [40, 69, 142]).

SVM is well-known for its capability of handling nonlinear problems. Compared to other classification techniques in the machine learning domain, SVM tends to

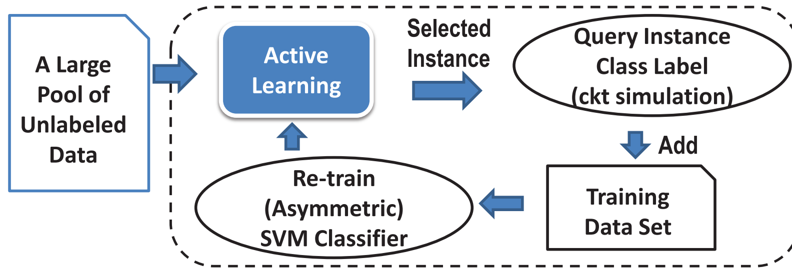


Figure 4.1: Active-learning guided SVM.

generate a sparser solution from an evenly sampled training data set. As a useful toolbox in the EDA community, leveraging such sparsity to reduce the intense need in sampling has been largely left untouched, which is the key focus of the present paper. We propose an optimized probabilistic active learning guided SVM approach to train high-quality circuit performance classifiers with significantly reduced cost. As shown in Fig. 4.1, active learning is a class of powerful techniques that can be leveraged to build "intelligence" into the sampling step of classifier training [79, 127, 105]: only promising instances that are expected to improve the performance of the classifier are selected for query (e.g. circuit simulation).

To select the optimal query instance at a time, we rank the candidate instances according to a probabilistically weighted goodness metric that is computed by rigorously evaluating potential reduction of *version space* if the instance were queried. As such, our active learning scheme intelligently selects query instances which are expected to act as support vectors throughout the iterative classifier training process, hence avoids committing wasteful queries and significantly reduces the required training data and the overall training cost.

For many binary classification problems in circuit design, certain degree of asymmetry exists between the two classes. For instance, in design verification, it is practically relevant to answer the following question: how to ensure designs passing the

verification are very likely to be actually acceptable, while allowing some of the acceptable designs to be rejected? To ensure the conservativeness in verification, the error on predictions of acceptance shall be more heavily minimized to avoid overly optimistic prediction of design robustness. To this end, we extend the proposed probabilistic active learning to the case of asymmetric SVMs by generalizing the concept of version space reduction and evaluating the biased safety level. Finally, we present a global acceleration scheme to further improve the convergence of asymmetric SVMs.

The presented ideas are rather general. We demonstrate their application by conducting four classification case studies: ripple noise analysis of LC based DC/DC converters , lock-time verification of charge-pump PLLs, reliability analysis of ring oscillators, and prediction of peak chip temperature based on a limited number of on-chip temperature sensors. For these applications, our techniques have led to one order of magnitude of efficiency improvement.

4.2 SVM

Support vector machine (SVM) [30, 129] is a useful supervised learning algorithm in solving binary classification problems. Given a set of training samples, the technique constructs a discriminant function as a classifying hyperplane in the input space. Its training process can be solved as a quadratic programming optimization problem. The objective of the optimization is to find the structural optimal hyperplane that separates the training data with largest margin. Such objective is called structural risk minimization and provides promising performance in many situations.

In practice, training samples are often not linearly separable, especially in circuit related applications. To achieve a nonlinear classifier, SVM projects the input data into a higher dimensional space, which is called feature space, and then tries to linearly separate the samples in the feature space. The higher dimensional hyperplane

may have a nonlinear projection in the input space, serving as a nonlinear classifier for the input data.

Denote the i-th sample in the training data set as

$$(x_i, y_i), y_i \in \{-1, +1\}, i = 1, 2, 3, ..., n \tag{4.1}$$

which consists of the input vector x_i and the corresponding class label y_i. Let $\phi(x)$ be the mapping function that maps any input vector x from $\mathbb{R}^n$ to $\mathbb{R}^m$. SVM defines the discriminant function of the classifier as

$$f(x) = w \cdot \phi(x) + b \tag{4.2}$$

where w is a vector with m entries. An unlabeled x will be classified as a positive instance or a negative instance if $f(x) > 0$ or $f(x) < 0$, respectively. The separating hyperplane in the feature space is defined by $f(x) = 0$. Instances closest to the hyperplane are defined as support vectors. The distance from any support vector to the hyperplane is called margin, which should be maximized during the training process.

The primal form of SVM is defined as follows:

$$\min_{w,b} \frac{\|w\|^2}{2} \tag{4.3}$$

subject to

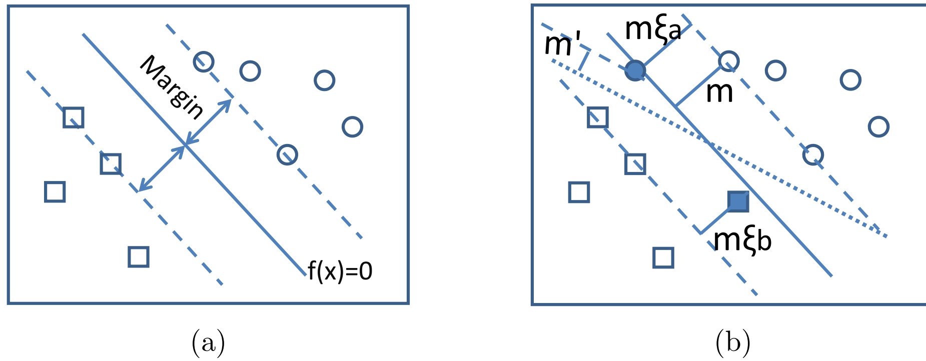
$$y_i(w \cdot \phi(x_i) + b) \geq 1. \tag{4.4}$$

Equality of (4.4) holds when x_i is a support vector. Based on the definition and the

constraints described in (4.4), the margin can be computed by:

$$m = \frac{1}{\|w\|} \tag{4.5}$$

and thus minimizing the objective function is actually maximizing the margin (see Fig. 4.2a).



(a) (b)

Figure 4.2: (a) Hard margin SVM. Circles and squares are instances of the two classes. $f(x) = 0$ is the separating hyperplane; (b) Differences between soft margin and hard margin SVM. The dot line is the hyperplane of hard margin SVM with margin m'. The solid line is the hyperplane of soft margin SVM with larger margin m. The solid circle and square violate the margin m with slack variable ξ_a and ξ_b respectively.

To solve the equivalent optimization problem of the SVM model, Lagrange multipliers are often employed, and the dual form of the SVM problem is derived as:

$$\min_{\alpha_i} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \cdot \phi(x_i) \cdot \phi(x_j) \tag{4.6}$$

subject to

$$\alpha_i \geq 0 \tag{4.7}$$

and

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{4.8}$$

where Lagrange multipliers α_i are the new model parameters. The support vectors can then be defined as training vectors whose corresponding α_i is nonzero. And the original w in (4.2) can be expressed as:

$$w = \sum_{i=1}^{n} \alpha_i y_i \cdot \phi(x_i). \tag{4.9}$$

From the dual problem, we can find that only inner products of vectors in the feature space are involved in computation. Thus we can simply define a kernel function $K(x_1, x_2) = \phi(x_1) \cdot \phi(x_2)$ to implicitly describe the mapping towards the feature space while solving the optimization problem instead of finding an exact mapping function $\phi(x)$.

One of the widely used kernel functions is called *Radial Basis Functions* [129] (RBF, a.k.a. Gaussian kernel):

$$K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}. \tag{4.10}$$

The usage of kernel functions, the form of the discriminant function, and the structural risk minimization make SVM distinguished from other classification techniques, since the optimal solution may be sparse in the context that it can be represented by just a small portion of the samples, i.e. support vectors.

Leveraging the kernel (4.10), the biasing term b in the discriminant function is often set to be zero for simplicity. With $b = 0$, the constraint (4.8) will be freed, leaving (4.7) be the only constraints of the dual problem. In this work, we use (4.10) as the default kernel and adopt the unbiased hyperplane by setting $b = 0$ in order to

facilitate active learning[127] (details in later sections).

Overfitting is one of the common obstacles in supervised learning techniques. The fact is that the "strictly learned" discriminant function from a large amount of training data is often outperformed by well regularized discriminant functions with some extent of relaxation. To balance the overfitting and underfitting, a modified SVM called soft margin SVM is proposed in [30] and [129].

Rather than satisfying all the constraints defined by the training data set, soft margin SVM uses slack variables and a cost factor to trade-off the training error and the margin. The primal form of soft margin SVM can be posed as follows:

$$\min_{w,\xi} \frac{\|w\|^2}{2} + C\sum_{i=1}^{n} \xi_i \tag{4.11}$$

subject to

$$y_i(w \cdot \phi(x_i)) \geq 1 - \xi_i, \xi_i \geq 0 \tag{4.12}$$

where C is the constant cost coefficient defined before the training and ξ_i is the slack variable representing the margin violation of the i-th training example. And the margin (as shown in Fig. 4.2b) is defined as

$$m = \min_{i} \frac{y_i f(x_i) + \xi_i}{\|w\|} = \frac{1}{\|w\|}. \tag{4.13}$$

Similarly, by applying Lagrange multipliers to the primal problem, one can derive the dual problem with exactly the same objective function as (4.6) but with different constraints:

$$0 \leq \alpha_i \leq C. \tag{4.14}$$

In this case, support vectors can still be defined as vectors with $\alpha_i > 0$. For those

vectors with $\alpha_i < C$, they locate right on the boundaries of the margin. For those vectors with $\alpha_i = C$, they are outliers that violates the margin, sometimes may even be misclassified samples.

4.3 Active Learning Guided SVM

The goal of circuit performance classification is to predict if a circuit with uncertain parameters (i.e. design parameters, process variations, or working conditions) will meet the performance specification or not. Given sufficient training data, an SVM classifier can be trained for accurate prediction. However, expensive circuit simulation usually makes it infeasible to obtain a large volume of samples to train an accurate SVM.

To reduce the intense need of training data, a group of techniques called semi-supervised active learning [73] is developed by leveraging the spatial information (e.g. clustering based in [63]) or the sampling history (e.g. significance space construction in [99]). It only works for problems with finite candidates and thus is not applicable to analog circuits whose sampling space is usually infinite. Supervised active learning uses a clearer strategy to select samples: hypothesis space reduction (e.g. agnostic active learning [19]) or SVM oriented version space reduction [127, 43]. In this domain, the concern lies on the efficiency of the algorithm. However, in circuit application, since simulations can be extremely expensive, we need to optimize the quality of the selected sample as well. Therefore, in this section, we propose active learning for high quality and fast convergence. Another active learning scheme for conservative prediction is proposed in the next section.

4.3.1 Version Space

Pool-based active learning [79] is a method that starts with an initial training data set and a pool of unlabeled data. The learner can query the class labels of

instances in the pool and add them to the training data set.

Different active learning strategies have been developed in different scenarios for the selection of candidates from the pool . In the EDA domain, for example, [33] employs active learning to generate a balanced, in terms of the number of different types of samples, training data set. Another method proposed in [91] develops active learning strategy to reduce the need of expensive aging simulations in building regression models for circuit reliability analysis.

A querying strategy based on version space introduced by [127] is an effective method for active learning with SVM. With this strategy, active learner would start with a set of all the possible separating hyperplanes, i.e. the version space, and then choose the next query to shrink this set as much as possible until the size of the set is small enough. Then the optimal hyperplane of this small set could be used as an accurate approximation of the real separating hyperplane (more details in the next subsection).

Version space V is defined as the set of all the hyperplanes in the form of $f(x) = w \cdot \phi(x) = 0$ that completely separate the training data in the feature space. Since the normalization of w

$$\hat{w} = \frac{w}{\|w\|} \tag{4.15}$$

will not change the hyperplane, we can define the version space as:

$$V = \{\hat{w} \in W | \|\hat{w}\| = 1, y_i(\hat{w} \cdot \phi(x_i)) > 0, \forall i\} \tag{4.16}$$

where the parameter space W has the same dimension of the feature space F. According to the duality between the parameter space W and feature space F [129], every training sample x_i has a corresponding hyperplane $\phi(x_i) \cdot w = 0$ in W that

cuts off a part of the hypersphere $\|\hat{w}\| = 1$. An example in a 2-D parameter space is shown in Fig. 4.3a. Each training data is actually a constraint in the optimization problem, partially defining the set of all the feasible solutions. As a result, all the infeasible $\hat{w}$ will be cut off by any training sample they violate and the surface on the hypersphere that remains at last will be the version space.

Due to the structural risk minimization in the SVM model, the corresponding $\hat{w}^* \in V$ of the optimal classifier produced by the hard margin SVM will be the center of a sphere which is tangent to the cutting hyperplanes in W associated with support vectors (see Fig. 4.3a):

$$\hat{w}^* = \arg\max_{\hat{w} \in W} \min_i \{y_i(\hat{w} \cdot \phi(x_i))\}. \tag{4.17}$$

Here $\|\hat{w}\|$ is normalized instead of the normalization of $|f(x_i)|$ for support vectors in the hard margin SVM problem (4.3)(4.4). Therefore $\hat{w}^*$ is the normalization of the optimal separating hyperplane represented by w^*, and the distance between support vectors and this hyperplane is the margin m in (4.5).

Note that (4.16) is defined under the condition that all the training data are linearly separable in the feature space. However, since most circuit variables(i.e. voltage, current etc.) are continuous and the characteristics are often nonlinear, samples in circuit application are usually not completely separable. Thus we employ soft margin SVMs instead of hard margin SVMs in our applications and propose another definition of version space V for the soft margin SVM:

$$V = \{\hat{w} \in W | \|\hat{w}\| = 1, y_i(\hat{w} \cdot \phi(x_i)) > -\frac{\xi_i}{\|w\|}, \forall i\} \tag{4.18}$$

where w and ξ_i are constants produced by the training process of soft margin SVM

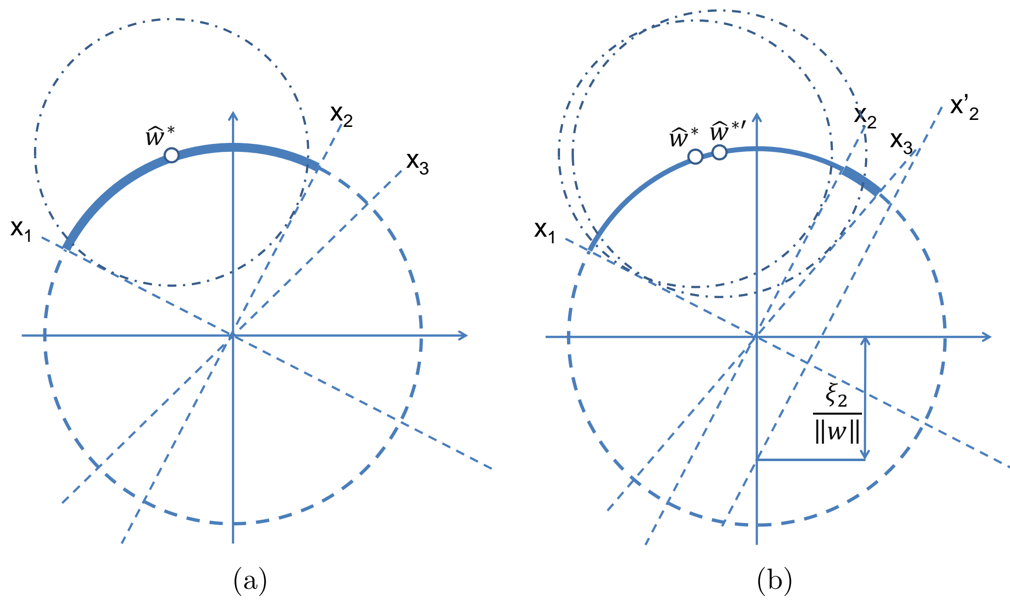


Figure 4.3: Version Space in 2-dimensional W Space: (a) x_1,x_2 and x_3 are cutting hyperplane associated with training data. x_1 and x_2 are support vectors. the solid arc is the version space. (b) $\hat{w}^*$ and version space between x_1 and x_2 correspond to hard margin SVM. In soft margin SVM, hyperplane of x_2 is shift to x_2' with a vertical offset of $\xi_2/\|w\|$. the version space is enlarged by the arc between x_2 and x_3. $\hat{w}^{*\prime}$ corresponds to the optimal classifier of soft margin SVM. x_3 becomes the support vector as well.

(4.11)(4.12) with given cost coefficient C. As mentioned in the previous section, the cost coefficient C provide a trade-off mechanism between the structural risk and the constraint slackness. In practice, such mechanism often provides much better regularization, hence it's reasonable to assume that the slackness ξ_i can also make the version space more resistant to the overfitting problem.

As shown in Fig. 4.3b, a non zero slack variable ξ_i moves the corresponding cutting hyperplane vertically, relaxing the constraint defined by x_i. Therefore, compared to the version space of hard margin with the same training data, soft margin SVM enlarges the size of the version space as well as the margin of the classifier.

Alternatively, from the perspective of the dual problem and the kernel method, if we define $\hat{\alpha}_i = \alpha_i y_i$, by substituting (4.9) into (4.2), the form of all the possible discriminant function can be defined as:

$$f_d(x) = \sum_{i=1}^{n} \hat{\alpha}_i \cdot K(x_i, x). \tag{4.19}$$

Then the equivalent hard margin version space can be redefined as:

$$V_d = \{\hat{\alpha} \in \mathbb{R}^n | \|\hat{\alpha}\| = \|\alpha^*\|, \sum_{i=1}^{n} \hat{\alpha}_i \cdot K(x_i, x_j) > 0, \forall j\} \tag{4.20}$$

where n denotes the total number of training samples x_j and $\hat{\alpha}$ is an n-entry vector. The α^* corresponds to the output of the SVM training, constraining the version space to a hypersphere again.

Given a distinct form of discriminant functions, the cardinality of V_d is different from V . However, the optimal member in both version spaces that corresponds to the solution of the SVM should be the same due to the duality.

Similarly, for the dual problem of the soft margin SVM, if we follow the same

adjustment performed in (4.18), i.e. using a soft margin SVM with given cost factor C to determine the relaxation ξ_i on all the training data x_i, the proposed soft margin version space can be redefined as:

$$V_d = \{\hat{\alpha} \in [-C, C]^n | \|\hat{\alpha}\| = \|\alpha^*\|, \sum_{i=1}^{n} \hat{\alpha}_i \cdot K(x_i, x_j) > -\xi_j, \forall j\}. \tag{4.21}$$

Compared to the hard margin version space, the soft margin one is constrained within a smaller space $[-C, C]^n$ instead of the whole n dimensional space $\mathbb{R}^n$. If $C = \infty$, all the ξ_j will be zero and thus the soft margin version space should be equivalent to the hard margin one.

4.3.2 Proposed Accelerated Active Learning with Probabilistic Version Space

Suppose that there is a pool of unlabeled data with a size of n, V_n is the resulting version space if we query all the instances in the pool and $\hat{w}_n^*$ is the corresponding optimal separating hyperplane, then $\hat{w}_n^*$ lies in version space V_i after i queries.

In case that the pool is of a huge size, it is too expensive to query the labels for all the instances in the pool. Since $\hat{w}_n^*$ is determined only by support vectors, a limited number of queries that include all the support vectors will also produce the same optimal hyperplane $\hat{w}_n^*$. Thus $\hat{w}_n^*$ can be approximated with much lower expense.

Version space based active learning tends to find a smart way of querying and finally has an accurate approximation of $\hat{w}_n^*$ with limited number of queries. For hard margin SVM, according to the definition of version space, $V_i \supseteq V_{i+1}$ for $i = 0, 1, 2, ...$, which means the size of the version space is reduced as more and more instances are queried. For soft margin SVM, as every query adds a cutting hyperplane in W, the version space would be reduced by more queries as well. If the size of V_k is small enough, any $\hat{w} \in V_k$ is close to $\hat{w}_n^*$ and we can use $\hat{w}_k^*$ to approximate the optimal classifier of the whole pool. The upper error bound of this approximation will be

smaller if V_k is shrinking. Therefore, to make $\hat{w}_k^*$ converge to $\hat{w}_n^*$ as fast as possible, we can select query that shrink the version space as much as possible in every step.

Let $Area(V_k)$ denote the surface area of the version space V_k which is defined by k queried samples. Since the shape of the version space could be very complicate in high dimensional space, it is impractical to compute $Area(V_k)$ explicitly. In general, $\hat{w}_k^*$ may be located near the center of the version space, thus the distance between $\hat{w}_k^*$ and support vectors, which corresponds to the margin of the unnormalized w_k^* in feature space, can be used to represent the surface area of the version space. We assume that $Area(V_k)$ is proportional to the margin of w_k^* and, for simplicity, the constant of proportionality is 1:

$$Area(V) = \frac{1}{\|w_k^*\|}. \tag{4.22}$$

In [127], the authors tried to halve the version space with every query as far as possible. They select such an instance in the pool in every step that has equal or most approximate size of version space no matter which class it is labeled. This method requires twice retraining of SVM for every instance in the pool, which is very expensive when it comes to cases that the pool is often of a huge size. And the reduction of version space in every step is likely to be no more than one half.

In this paper, we propose an accelerated active learning method with probabilistic version space. It reduces the retraining cost by pool size shrinking and tends to have a more aggressive reduction in version space.

In active learning process, version space described in the previous subsection is used to measure the resulting benefit of any candidate query. To compute the resulting reduction in version space of any candidate instance, we need to get its label and retrain an SVM by adding it into the training data set. We avoid querying

the real label of the candidate instance by assuming its label as +1 and -1 and perform SVM retraining respectively (no simulation is committed yet at this point). In every single step of active learning, if the pool has n instances, we should train $2n$ SVMs to find the optimal query. It is too expensive especially when it comes to our applications of circuit performance classifications.

Since most circuit parameters are of analog value, the pool consists of infinite unlabeled instances, which makes it impossible to calculate the expected size of resulting version space for every instance. One simple way is to randomly sample a certain number of instances in the input space to form a pool of finite size in every step. However, for a high dimensional input space, it requires a huge number of samples for the algorithm to reach certain accuracy. It would be far too expensive to find an optimal query out of the pool.

In the SVM model, based on the definition of the constraints (4.4) and (4.12), for samples x_i right on the margin (i.e. support vectors in the hard margin SVM or support vectors with $\xi_i = 0$), we have $|f(x_i)| = 1$. According to (4.17), a new query of x will reduce the size of version space if and only if

$$|f(x)| < 1 \tag{4.23}$$

which means x should be closer to the hyperplane than support vectors with zero ξ_i. Therefore, in every step of active learning, we can sample a certain number of instances that satisfy (4.23) to form a pool with acceptable size instead of sampling in the whole input space and then perform previously mentioned active learning algorithm after that. Whichever instance in the pool is chosen to be queried, part of the version space will be cut off, and hence we can guarantee that the version space is reduced in every step.

In addition to the above acceleration, in every step of active learning, we try to find such an instance x in the pool that has the largest expectation of size reduction in version space. In other words, after we add x into the training data set, the expectation of the resulting version space should be the smallest. Let V_i denotes the version space after i queries and x_{i+1} denotes the $(i+1)$th query, define:

$$V_i^+ = V_i \cap \{\hat{w} \in W| + \hat{w} \cdot \phi(x_{i+1}) > -\frac{\xi_i}{\|w\|}\}; \tag{4.24}$$

$$V_i^- = V_i \cap \{\hat{w} \in W| - \hat{w} \cdot \phi(x_{i+1}) > -\frac{\xi_i}{\|w\|}\}. \tag{4.25}$$

Obviously V_i^+ and V_i^- denote the version spaces that x_{i+1} is labeled as +1 or -1 respectively. A more aggressive strategy is to find an instance x_{i+1} in the $(i+1)$th step with smallest value of

$$E = P(y=1|x_{i+1})Area(V_i^+) + P(y=-1|x_{i+1})Area(V_i^-) \tag{4.26}$$

where $P(y=1|x_{i+1})$ and $P(y=-1|x_{i+1})$ denote the probability of x_{i+1} being labeled as +1 and -1 respectively.

We use an intuitive function for the conversion between SVM output $f(x)$ and the probability in our algorithm:

$$P(y=1|x) = \frac{1}{1+exp(-\alpha f(x))}. \tag{4.27}$$

With a tuned parameter α, it provides symmetric probability results for computing the expectation of version space.

The algorithm flow of active learning combined by all the above steps is demonstrated in Algorithm.1.

Algorithm 1 Accelerated Active Learning with Probabilistic Version Space

1. Find an initial training data set S which includes both positive and negative examples
2.Train a soft margin SVM classifier $f(x)$ with S
3. Randomly pick up a huge set T from the input space
4. Classify T with $f(x)$
5. Pick up a certain number of instances with small enough $|f(x)|$ in T, form a subset $\hat{T}$
6. For every $x_i \in \hat{T}$,
(a) Assign $P(y = 1|x_i)$ and $P(y = -1|x_i)$ to x_i
(b) Label x_i as positive, add it to S temporarily
(c) Train an SVM to compute its resulting size of version space $Area_+$
(d) Label x_i as negative, add it to S temporarily, and train another SVM to get $Area_-$
(e)$E = P(y = 1|x_i)Area_+ + P(y = -1|x_i)Area_-$
7. Query the real label of the instance with smallest E, add it into S
8. Repeat steps 2 - 7 until E is small enough
9. Train a final SVM with S

It should be aware that the pool is randomly regenerated in each iteration, making it unique to the others in the other iterations. As the process iterates, the granularity of the superposition of all the pools in the previous iterations, which can be viewed as the discretization of the continuous input space, gets finer and finer. This can be used to capture the continuity of analog systems and makes the proposed method distinguished from the traditional active learning algorithms (such as [127, 105, 73]) for finite discrete applications.

4.3.3 Error Bound

An error bound estimation of the SVM classifiers based on leave-one-out cross validation is proposed in [129], providing a method to approximate the upper bound of the misclassification from a given training data set. However, such method is based on the assumption that all the training samples are selected independently. In

the context of active learning, such assumption is always not true since new samples are always evaluated and selected by the current SVM classifier trained from the current training samples.

As mentioned in the previous section, standard SVM models with sufficient training data often produce good classifiers with promising performance. Therefore, rather than analyze the upper bound of the misclassification of the actively learned model, we treat the SVM classifier learned from sufficient data as the desired "golden" classifier and explore the maximum difference between such classifier and any other solutions in the reduced version space.

Due to the duality between the primal form and the dual form of the SVM model, the optimal solution $\hat{w}^* \in V$ is equivalent to the optimal solution $\hat{\alpha}^* \in V_d$. Although V and V_d might have different cardinality, considering the powerful expressive capability of kernel method with (4.10), we make the assumption that $\hat{w} = \sum_{i=1}^{n} \hat{\alpha}_i \phi(x_i)$ for arbitrary $\hat{w}$, which guarantees the trends or convergence of applying active learning version space reduction in both version space definitions should be consistent.

As a result, the error between the golden classifier represented by $\hat{\alpha}^*$ and any solution $\hat{\alpha}$ in the version space can be formulated as:

$$\Delta(\hat{\alpha}) = |f_d^{\hat{\alpha}^*}(x) - f_d^{\hat{\alpha}}(x)| = \sum_{i=1}^{n} |\hat{\alpha}_i^* - \hat{\alpha}_i| K(x_i, x). \tag{4.28}$$

If the Guassian kernel (4.10) is applied here, since $0 \leq K(x, y) \leq 1$, we have $\Delta(\hat{\alpha}) \leq \sum_{i=1}^{n} |\hat{\alpha}_i^* - \hat{\alpha}_i|$. Taking all the training data into account, the upper bound of the difference between the golden classifier and any solution in the version space

is the optimal solution of the following linear programming problem:

$$\max_{\hat{\alpha}} \sum_{i=1}^{n} |\hat{\alpha}_i^* - \hat{\alpha}_i| \tag{4.29}$$

subject to:

$$\sum_{i=1}^{n} \hat{\alpha}_i K(x_i, x_j) > 0, \forall j = 1, 2, ..., n \tag{4.30}$$

where x_i and x_j denote arbitrary training samples.

The procedure of active learning in the context of the error bound defined above is not as explicit as version space reduction. The reason is that adding a new sample x_{x+1} into the training data set might reduce the current set of the feasible solutions defined by (4.30), but a new variable $\hat{\alpha}_{n+1}$ will also be introduced. Nevertheless, since it suffices to consider only the feasible solutions on the boundary to find out the optimal solution (i.e. the error bound) of a linear programming problem, a new sample may help to lower the error bound if it cuts off a part of the boundary of the set of feasible solutions.

4.4 Asymmetric Active Learning

Compared to related active learning methods developed in the machine learning domain [127, 105, 73], the proposed method mainly focuses on continuous problems rather than finite discrete space. In traditional discrete applications, the imbalance between the numbers of positive and negative samples is usually a challenging problem. While in circuit scenarios, since the continuous space can be infinitely discretized, meaning infinite number of different types of samples can be obtained, such problem may not be a concern any more. However, the need of safe prediction occurs as a new challenge in the circuit domain.

For example, in applications like circuit verification, we are much more concerned

about the class of failure than the class of success. Misclassifying acceptable circuits into the class of failure may be allowed, while misclassifying flawed circuits into the class of success should be strictly restricted. In this light, we expect that the prediction on the failure class is conservative while the classifier still has high global accuracy.

Some adaptive sampling techniques such as [59, 25, 134] are developed to generate balanced training set from imbalanced data in the machine learning domain, and similar algorithms [40, 69] designed for circuit problems are adopted in the EDA community to solve the sampling problem of rare events (e.g. SRAM yield analysis). In this work, the objective problem is different. Instead of dealing with imbalanced data density, we adjust our active learning algorithm to meet different safety requirements on the two classes.

4.4.1 Cost Asymmetric SVM

To reduce misclassifications in one class, one frequently used method is to assign a higher cost or penalty coefficient to that class [98][9]. The previous soft margin SVM model will be further modified into the following cost asymmetric SVM model(a.k.a. 2C-SVM):

$$\min_{w,\xi} \frac{\|w\|}{2} + C_+ \sum_{i:y_i=+1}^{n} \xi_i + C_- \sum_{i:y_i=-1}^{n} \xi_i \tag{4.31}$$

subject to

$$y_i(w \cdot \phi(x_i)) \geq 1 - \xi_i, \xi_i \geq 0. \tag{4.32}$$

Without any loss of generality, assume that $C_+ > C_-$. As a result, rather than assigning a non-zero slack variable to a positive training sample, the 2C-SVM model tends to increase the slackness of one or more negative samples. Therefore, after the training process of 2C-SVM, fewer relaxation will be assigned to the positive training

samples compared to the negative samples. The resulting classifier should be more accurate for positive instances.

The dual problem of the 2C-SVM shares the same objective function (4.6) while the constraints will be modified as:

$$0 \le \alpha_i \le \begin{cases} C_+, & y_i > 0; \\ C_-, & y_i < 0. \end{cases} \tag{4.33}$$

By defining the class of failure as positive and the class of success as negative, if $C_+ \gg C_-$, then errors that misclassifying a positive instance as a negative one, a.k.a. false negative, will be much fewer than misclassifications on negative instances, a.k.a. false positive.

4.4.2 Safety Level Evaluation

It is hard to evaluate the safety level of a classifier on positive class explicitly, but apparently, a classifier with few false negatives and large margin on positive class is more conservative than those with more false negatives or smaller margin on positive class.

Hinge loss[131] is the loss function for soft margin SVM. For a training example (x, y), it is defined as:

$$l(x, y) = \max(0, 1 - y \cdot f(x)). \tag{4.34}$$

It is widely used in SVM to penalize if (x, y) violates the margin, which is already implicitly included in the standard soft margin SVM and the 2-C SVM. Recalling the definition of soft margin SVM, $l(x, y)$ is in fact ξ_i in (4.11)(4.12). If $0 < l(x, y) \le 1$, then (x, y) is the instance with distance to the hyperplane smaller than the margin. If $l(x, y) > 1$, then (x, y) is a misclassification.

In order to evaluate the conservation of the positive class, we sum up all the hinge loss in the positive class only:

$$L = \sum_{y_i=+1} l(x, y). \tag{4.35}$$

The smaller L is, the more conservative the classifier is. If $L = 0$, since there is no misclassification or margin violation, we believe that it is the most conservative case.

4.4.3 Asymmetric Active Learning and Performance Optimization

To get a conservative classifier, we employ the 2C-SVM and the newly-defined hinge loss metric in the previous strategy to achieve a conservative active learning algorithm. In every step, we query an instance that may reduce the version space largely and maintain a small L at the same time. For every instance in the pool, we compute its expected loss as

$$Loss = P(y = 1|x_{i+1})L^{+} + P(y = -1|x_{i+1})L^{-} \tag{4.36}$$

where L^{+} and L^{-} denote the resulting hinge loss on the positive class if the instance is labeled as positive or negative respectively. Then we perform the following calculation for every instance

$$D = E + \eta \cdot Loss \tag{4.37}$$

where η is the trade-off coefficient and the instance with smallest D will be selected to be queried. This strategy may not shrink the version space the fastest, but it can make the classifier more conservative.

Since instance with smallest D is less likely to have a smallest E at the same time,

the version space reduction in every step will be smaller and hence the convergence will be slower. In order to improve the efficiency, we propose the following global strategy for improving the convergence. Considering two 2C-SVMs with different cost coefficient but trained by the same training data set:

$$\min_{w,\xi} \frac{\|w\|}{2} + C_{1+} \sum_{i:y_i=+1}^{n} \xi_i + C_{1-} \sum_{i:y_i=-1}^{n} \xi_i; \tag{4.38}$$

$$\min_{w,\xi} \frac{\|w\|}{2} + C_{2+} \sum_{i:y_i=+1}^{n} \xi_i + C_{2-} \sum_{i:y_i=-1}^{n} \xi_i. \tag{4.39}$$

We know that $\|w\|$ and $\sum^n \xi_i$ are conflict notions that represent the margin and the violation of the margin respectively. Thus $\|w\|$ will increase if $\sum^n \xi_i$ decreases and vice versa. For the two 2C-SVMs, if $C_{1\pm} > C_{2\pm}$ respectively, then we will have $\|w_1\| \geq \|w_2\|$. According to (4.22), it means that the size of the resulting version space of (4.38) will be smaller than that of (4.39). Assumes that C_+^* and C_-^* are the two cost coefficients that could achieve a conservative classifier in the final step, we may start the active learning from larger cost coefficients C_+ and C_- and gradually reduce them to C_+^* and C_-^* respectively. By this strategy, we can achieve a faster convergence than the strategy that use C_+^* and C_-^* from the very beginning.

The algorithm flow of the asymmetric active learning is shown in Algorithm. 2.

4.4.4 Dynamic Model Selection

According to the earlier discussions, the selection of the SVM model or, more specifically, the selection of the cost parameters (i.e. the C in the soft margin SVM, the C_+ and C_- in the 2C-SVM) plays an important role in the active learning procedure.

A popular choice of the algorithms on model selection is based on cross validation.

Algorithm 2 Asymmetric Active Learning

1. Find an initial training data set S which includes both positive and negative examples
2.Train a soft margin SVM classifier $f_1(x)$ with S
3. Randomly pick up a huge set T from the input space
4. Classify T with $f_1(x)$
5. Pick up a certain number of instances with small enough $|f_1(x)|$ in T, form a subset $\hat{T}$
6. For every $x_i \in \hat{T}$,
(a) Assign $P(y = 1|x_i)$ and $P(y = -1|x_i)$ to x_i
(b) Label x_i as positive, add it to S temporarily
(c) Train a 2C-SVM with parameter C_+ and C_- to compute its resulting size of version space $Area_+$ and loss L_+
(d) Label x_i as negative, add it to S temporarily, and train another 2C-SVM with parameter C_+ and C_- to get $Area_-$ and loss L_-
(e)$D = P(y = 1|x_i)(Area_+ + \alpha L_+) + P(y = -1|x_i)(Area_- + \alpha L_-)$
7. Query the real label of the instance with smallest D, add it into S
8. If C_+ and C_- are larger than the given C_+^* and C_-^*, reduce C_- and C_+ accordingly
9. Repeat steps 2 - 8 until D is small enough
10. Train a final 2C-SVM with S and the cost coefficients C_+^* and C_-^*

Models with different parameters are trained and tested by cross validation and the one with the best average performance will be picked. However, in active learning, the number of training samples is limited and, again, samples are not independent with each other. Therefore methods based on cross validation are not applicable in the procedure of active learning.

A simple yet practical analytic method for model selection is proposed in [27] to determine the appropriate parameters for the SVM regression. We adopt the similar reasoning here to determine the model parameter in each iteration during the active learning procedure.

Due to the fact that $0 \leq K(x, y) \leq 1$ for Gaussian kernel (4.10) and the dual form definition of the soft margin SVM, we have the following inequality:

$$\begin{aligned} |f(x)| &\leq \sum_{i=1}^{n_{SV}} |\alpha_i y_i K(x_i, x)| \\ &\leq \sum_{i=1}^{n_{SV}} |\alpha_i| \\ &\leq n_{SV} \cdot C \end{aligned} \tag{4.40}$$

where x_i denote the support vectors (i.e. those with non-zero α_i) and n_{SV} denotes the total number of them. Then the model parameter C can be estimated by:

$$C \geq k \cdot \frac{\max_x |f(x)|}{n_{SV}} \tag{4.41}$$

where k is a parameter used to implement the adjustment in C mentioned in the previous subsection. At the beginning of the active learning procedure, we use a large k for fast convergence, and then reduce it gradually towards 1 to get a better regularized solution. Since the input space x is often bounded in the circuit application,

calculating $\max_x |f(x)|$ should be trivial once we obtain the form of $f(x)$.

In the application of active learning, the given initial training data set is often very small, and often easy to be completely separable. Thus applying an SVM model with a large C will produce us the first approximation of the discriminant function $f(x)$ and the first group of support vectors. After that, in the n-th ($n \geq 2$) iteration, we use the $f(x)$ and n_{SV} gained from the $(n-1)$-th iteration to estimate the parameter C.

Another important model parameter is the ratio of C_+/C_- in the asymmetric model. One way to determine such ratio is to use the practical cost for the misclassification on different classes. However, such ratio might be indefinite or indistinct. In such cases, we can determine the ratio from the context of the model.

According to the definition of the 2C-SVM, an outlier x_i in the positive class that violates the margin may have an impact of $C_+K(x_i, x)$ to the learned discriminant function, since its corresponding $\alpha_i = C_+$. If it is in the negative class, then the impact should be $C_-K(x_i, x)$. The essence of the 2C-SVM is to make the impact of the two kinds of outliers different. Therefore, to ensure that outliers in the positive class always have greater impacts on the discriminant function than the negative outliers, we can select the ratio that satisfies:

$$\frac{C_+}{C_-} \geq \frac{\max_{x_i,x_j} K(x_i, x_j)}{\min_{x_i,x_j} K(x_i, x_j)} \tag{4.42}$$

where x_i and x_j are arbitrary vectors in the input space. Again, since the input space in circuit application is often bounded, the maximum and minimum of the kernel function on the input space should be computable.

4.5 Experimental Results

In this section, our proposed method is applied as a general flow to various circuit applications. Since the flow mainly focuses on simulation need reduction, for very high dimensional system, the running of a single simulation can still be expensive. Due to the limited scope, we illustrate the effectiveness of the proposed method in four simplified circuit systems.

Our proposed active learning approach employed in the following experiments is implemented in C++ on a Linux server. Our active learning scheme interface with the base-line SVM package SVM^{Light}[67]. And we use the default $\gamma = 1$ in this tool for SVM training. For the probability estimation in (4.27), the tuning parameter is set to be a constant $\alpha = 3$.

4.5.1 LC Based DC/DC Converter Design Classification

A simplified model of an LC-based DC-DC converter is shown in Fig. 4.4, where R_L and R_C represent the parasitic resistance of the inductor L and capacitor C, respectively. The PWM unit controls the power switch to keep charging or discharging the LC filter repeatedly, and hence generating both output voltage and inductor current ripples. Ripple noise is one of the important specifications of a DC/DC converter and, apparently, choices of design parameters L and C may produce different levels of ripple noise on the output voltage.

Assume that the switch, the voltage source, the PWM control module, and the feedback mechanism are all ideal components with perfect characteristics, and additionally the resistance of R_L and the conductance of R_C are proportional to the value of L and C, respectively. In this case, the state variables of the system can be defined as the current i on the inductor and the output voltage v if the input voltage V_{in} and the load current I_{Load} are given. And the dynamics of the system in

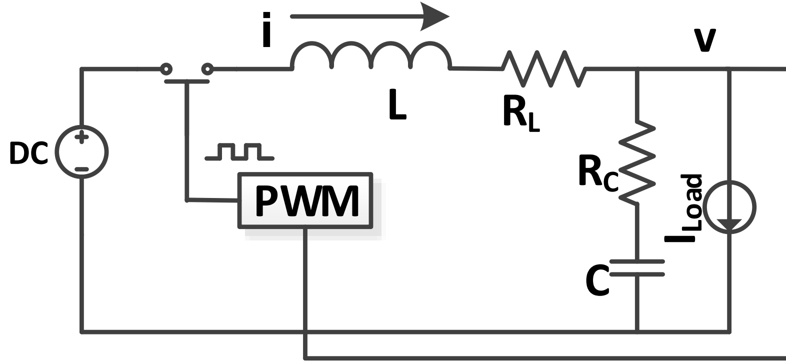


Figure 4.4: A simplified model of an LC-based DC/DC converter.

the s-plane can be described by

$$v = \frac{(1 + sCR_C)[V_{in} - (sL + R_C)I_{Load}]}{1 + sC(R_C + R_L) + s^2LC} \tag{4.43}$$

and

$$i = \frac{sCV_{in} + (1 + sCR_C)I_{Load}}{1 + sC(R_C + R_L) + s^2LC}. \tag{4.44}$$

By assuming that both v and i are initially set to be zero, the simulation is implemented using numerical methods to approximate (4.43) and (4.44). In terms of efficiency, active learning is not superior compared to passive learning due to the inexpensive simulations. This two dimensional system is used as an example to show the intuition of the convergence of active learning and the predominant improvement in reducing the need of samples/simulations.

The goal of our analysis here is to examine the dependencies of the ripple on important design parameters under a fixed working condition (i. e. with fixed input voltage and load current). For this, an SVM model can be extracted by defining the input features as L and C and labeling each instance according to its ripple noise. Given a threshold noise level, if a circuit with a group of design parameters

has a ripple noise that exceeds the threshold, it will be labeled as a positive sample. Otherwise, the negative label meaning that it meets the given threshold will be assigned to it. In this experiment, we set the load current $I_{load} = 0.1A$, the input voltage as $1V$ and the threshold of ripple noise as $0.2mV$.

The objective of the classifier is to accurately predict whether a combination of L and C within a given area will meet the given ripple noise specification or not. Thus the proposed symmetric model is adopted. We evenly select 9 samples out of the input space as the initial training data set and the procedure of active learning is shown in Fig. 4.5. In the query process for an unknown pair of (L, C), a transient simulation of (4.43) and (4.44) is invoked and the ripple noise is measured after the system gets stable, i.e. the current on the inductor and the output voltage settle down.

From Fig. 4.5a to Fig. 4.5d, the convergence of the active learning sampling is clearly demonstrated. New samples tend to be selected around the boundary between the positive and negative classes and gradually cluster the accurate separating hyperplane. The 200 samples selected by the active learning scheme in Fig. 4.5d already shows the outline of the hyperplane. Compared to Fig. 4.5e that shows the hyperplane with a large number of random instances, the approximated hyperplane in Fig. 4.5d is very close.

The learned classifier can be exploited to quickly determine the performance of a certain design. Compared to SVM training based on random sampling, the required number of simulations is much fewer in active learning. Such comparison with respect to the number of simulations is shown in Fig. 4.6 and a large reduction in simulations is achieved by active learning. For example, active learning only needs 10% the total number of simulations of the random sampling to produce a classifier with 97% accuracy.

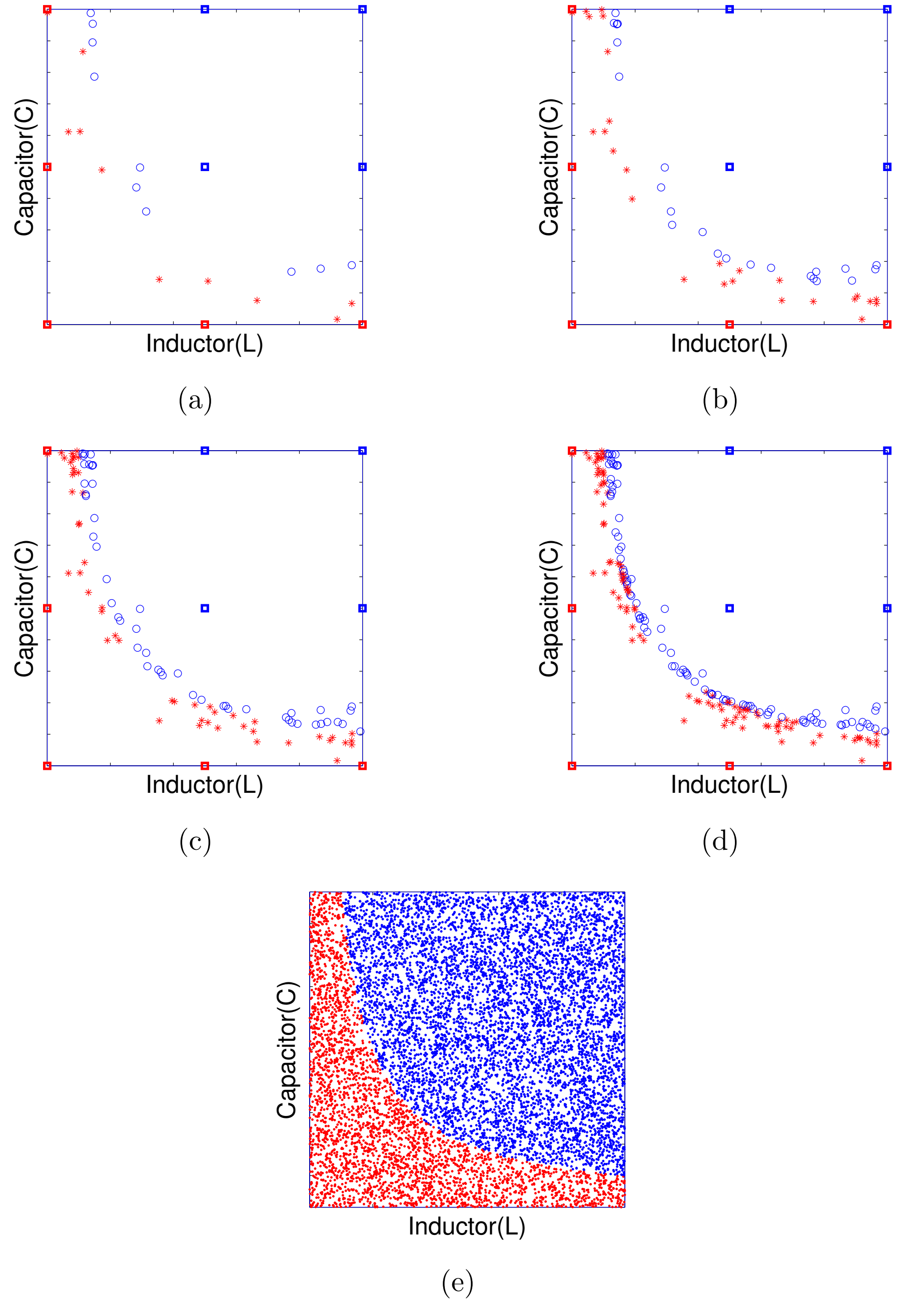


Figure 4.5: The sampling procedure of active learning in the parameter space $L : [1nH, 1uH] \times C : [10nF, 10uF]$ (red star denotes positive instances, blue circle denotes negative instances, square denotes initial training samples): (a) 20 samples selected; (b) 50 samples selected; (c) 100 samples selected; (d) 200 samples selected; (e) 10,000 random instances

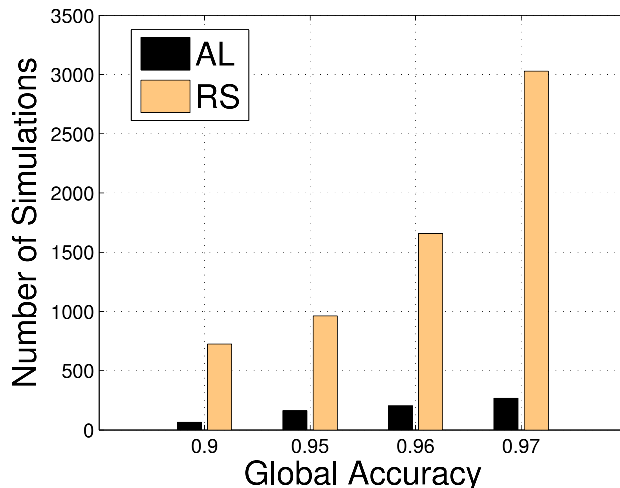


Figure 4.6: Comparison in the required number of simulations (AL short for Active Learning, RS short for Random Sampling) where the global accuracy is defined as the fraction of the correct predictions on 10K random test samples. (e.g. 0.1 means 10%)

In general, the active learning flow can be extended to other circuit design problems. We believe building an accurate classifier for circuits by taking various design parameters as its input features is meaningful for quick performance prediction in circuit design/debug. Once the classifier is given, designers can directly tell whether a design with some new parameters is going to work or the And the training cost to obtain such classifier can be significantly reduced by our proposed active learning scheme.

4.5.2 PLL Verification

The charge pump phase-locked loop being verified is shown in Fig.4.7. If the divided output signal is ahead of the reference clock, the voltage on the node *down* will be logical 1 and *up* be logical 0, controlling the charge pump to discharge the two capacitors in the loop filter to lower the input voltage of VCO, which will lower the frequency and reduce the phase of the output signal, or vice versa. If the phase

difference between the reference clock and the divided output signal is 0, both *up* and *down* are logical 0. Through such negative feedback, the frequency of the output voltage should gradually converge and finally be locked to N times the reference frequency.

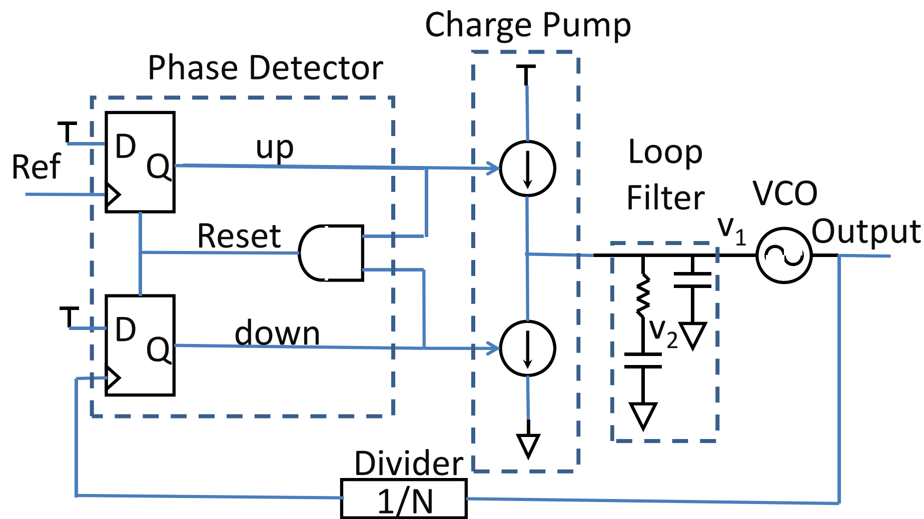


Figure 4.7: Block diagram of PLL

Typically, the *phase detector* (PD) is implemented by two DFFs and one AND gate. The behavior of the PD can be described as: if the divided signal *div* takes the lead, a rising edge will be generated at *down*, followed by a falling edge triggered by the arrival of the rising edge of the reference clock *ref*. A similar pulse will be generated at *up* if reference clock *ref* takes the lead. A pulse at *up* makes the charge pump to charge the loop filter and increase the input voltage of the VCO, while a pulse at *down* reverses such impact. The timing diagram of the phase detector and the resulting output of the charge pump and loop filter is shown in Fig. 4.8.

We use Verilog-A to set up the behaviorial model of the PLL and, similar with

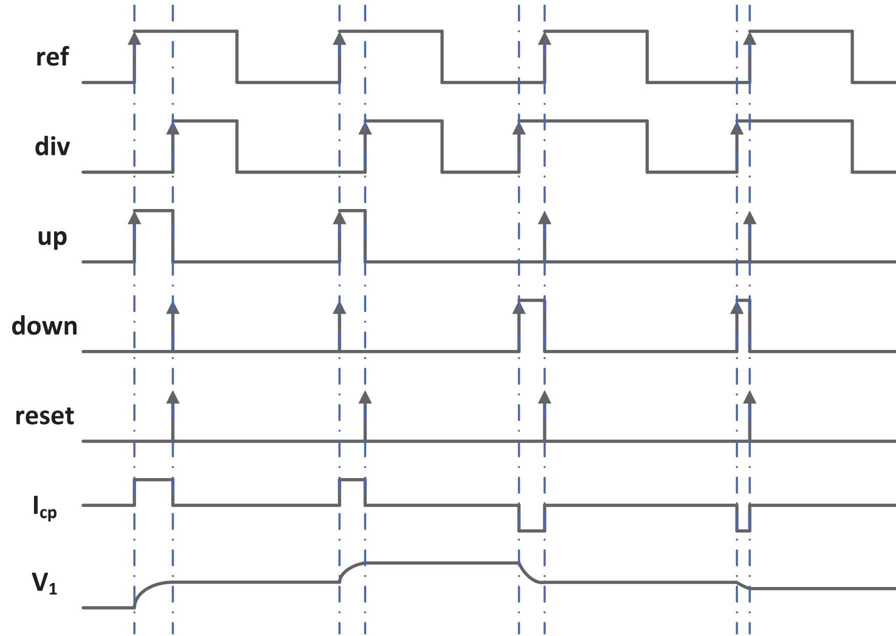


Figure 4.8: Timing diagram of the phase detector and the resulting output of the charge pump and loop filter.

the PLL model built in [146], define the state of the circuit with 5 variables: voltages on two capacitors v_1 and v_2, phase difference ϕ and two digital variables *up* and *down*.

Our goal is to verify that PLL with various size of initial state space will be locked within a certain time or not. To start with some failures for the SVM training, we set a rather large searching space for the state variables that is very likely to break the PLL (which is a similar concept in stress-testing). The searching space for the two initial voltages is the interval [0V, 0.8V] and the phase difference has a range $[0, 2\pi)$. There are only three combinations of *up* and *down*: $\{0, 0\}$, $\{0, 1\}$, $\{1, 0\}$.

In this case we hope to have a conservative verification of PLL lock-time over uncertain initial startup conditions. Thus the asymmetric active learning algorithm is employed. The class with positive label corresponds to meeting the specified

lock time, and the class with negative label corresponds to failing the lock-time specification. We compare the performance of this method with another method that only trains a 2C-SVM with random samples to illustrate the improvement in efficiency of the our method.

We use a set of 27 instances evenly distributed in the whole state space as our initial training data set. To make the comparison more reasonable, we also add this set into the random samples of the uniform sampling. The results of classifying a randomly generated data set with a size of 100,000 by two methods are shown in Fig. 4.9. We can infer that much less simulations and runtime is needed for our active learning method to achieve certain accuracy and prediction safety levels, the latter of which is measured by the classification error on the positive class (i.e. percentage of false negatives over the positive class). Considering achieving same accuracy, the number of simulations is reduced by an average of 70% and a maximum of 92%, the runtime is reduced by an average of 69% and a maximum of 91%. Considering the safety, an average of 89% and a maximum of 96% simulations are saved, with an average of 72% and a maximum of 89% saving in runtime.

By neglecting the two digital state variables, the distributions of instances sampled by active learning is shown in Fig. 4.10a. Again, the result shows that samples selected by active learning tend to cluster the separating hyperplane. A further projection into a 2-dimensional space (V_1, V_2) clearly shows the trend. The projection also implies that the initial states V_1 and V_2 have more impacts on the lock time of the PLL.

Note that the verification problem defined in this experiment is different from what is commonly approached as the problem of formal verification, such as [146, 83] based on reachability analysis and [147, 71] based on global convergence analysis. Their identical concern is to explore the problematic initial states in a fixed initial

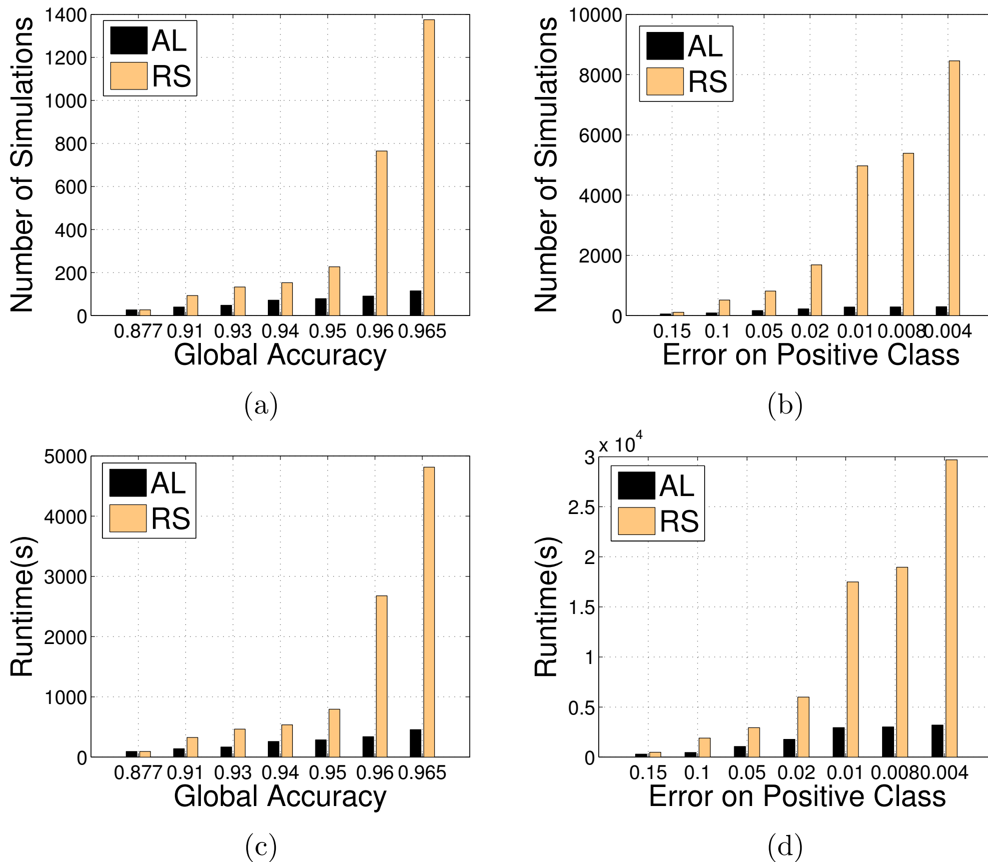


Figure 4.9: Results of PLL experiments (AL short for active learning, RS short for random sampling) where the global accuracy and the error on positive class are defined as the fraction of the correct predictions, with a test set whose size is 100K. (e.g. 0.1 means 10%): (a) number of simulations for certain accuracy; (b) number of simulations for safe prediction; (c) runtime to achieve certain accuracy; (d) runtime to achieve safe prediction

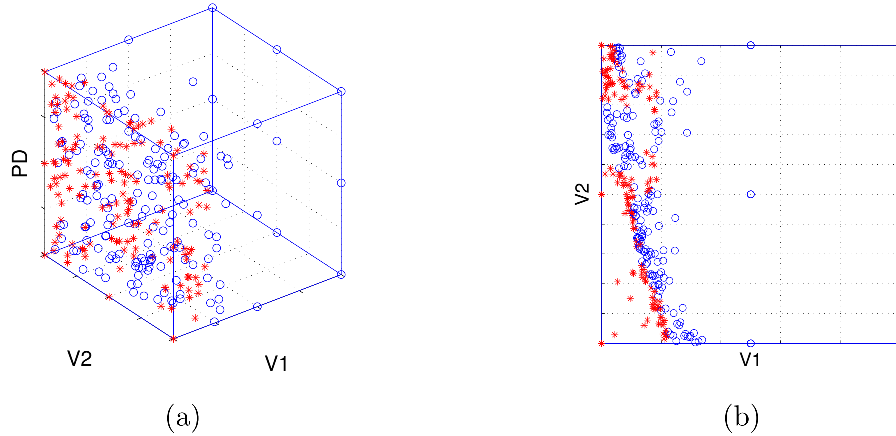


Figure 4.10: Samples selected by active learning: (a) the projection of the selected samples in the continuous 3-D $(V_1, V_2, \Delta\phi)$space, where V_1 and V_2 are the initial voltages on the two capacitors, and $\Delta\phi$ is the initial phase difference; (b) the projection of the selected samples in the (V_1, V_2) space. .

state space. However, in our experiment, we aim at determining proper initial state spaces for a design. Once an SVM classifier is trained for a given PLL, if the actual initial state space is a subset of the class of success, it's confident to accept the design. If not, certain suggestions can be fed back to designers to fix the problem. For example, pre-charging mechanism before or during the power-on process can be added to the PLL to shift the original state space to the desired space. The model and the flow are also reusable if the design is changed in the events like re-designing the circuit, or embedding the circuit into a larger system, and so on.

4.5.3 Prediction of Peak Chip Temperature Using a Limited Number of On-chip Thermal Sensors

In this experiment, finite elements simulations are employed to check whether the temperatures on a chip exceed the highest allowable value or not via limited number of sensors integrated on the chip. Fig. 4.11 shows the functional blocks placement of

a processor, similar to the simplified DEC alpha chip model in [81]. We place five sensors on top 5 blocks with highest possible power density (the positions of the red squares in Fig. 4.11.

Full-chip thermal simulation is performed by running a finite-difference based in-house thermal simulator. The simulator adopts the conjugate gradient iterative method with an incomplete LU (ILU) preconditioner. On a Linux-based desktop, each thermal simulation takes about 65 seconds to complete. This simulation runtime will grow if a finer thermal PDE discretization is adopted or finer details of material inhomogeneity is considered along the lateral dimensions of each material layer on the chip, producing a higher demand for smart learning.

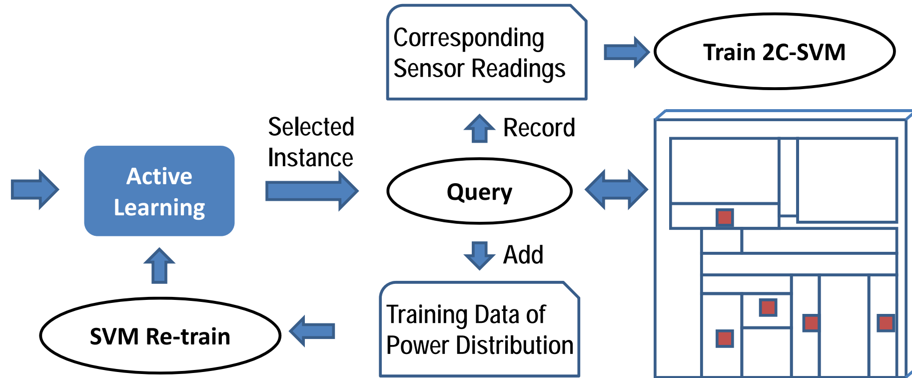


Figure 4.11: Functional blocks placement, sensor positions and experiment flow

For every block, the parameter is the power consumption that varies from 0 to its peak value. The parameters of all the 15 blocks form a 15-dimensional space. Our goal is to train an accurate and conservative SVM classifier that can take the readings from the five sensors and then predict the actual peak chip temperature, which may not take place at any of these sensor locations. To guarantee the safety of the chip, we use asymmetric models in two steps.

In the first step of the flow, we train a 2C-SVM classifier mapping block level power consumption to the actual peak chip temperature. The classifier takes the power consumption on all the blocks as the input features, and the label/prediction is made according to the given temperature threshold. During this step, if any simulation is invoked to query the real label of a certain combination of the 15 parameters, the sensor readings are also recorded and paired with the obtained label. Active learning is invoked in this step to select samples with high quality.

In the second step, we use the recorded sensor readings and their corresponding labels to train another 2C-SVM classifier which makes the prediction of peak temperature directly based on sensor readings. This step assembles the fine samples generated from the last step into a classifier with desired format.

We use a set of 100 random instances as the initial training data set for the active learning. To provide a comparison reference, we repeat the above two-step process based on passive learning, i.e. training on a large set of random samples. Again we add this set into the random samples of the passive learning. The results of classifying a randomly generated data set with a size of 100,000 by two methods are shown in Fig. 4.12. The active learning costs significantly less in terms of simulation and runtime. In terms of accuracy, we save an average of 51% and up to 67% in simulations as well as an average of 38% and up to 54% in runtime. In terms of achieving the same level of safety guarantee (e.g. false negatives), there is an average of 71% and up to 84% reduction in the needed number of simulations together with an average of 63% and up to 87% reductions in runtime.

This experiment shows the potential application of the proposed flow in monitoring complex systems. While implementing a substantial number of sensors in the system is beneficial to the safety, it may be costly for complex and highly integrated systems. In this light, the active learning guided sampling and SVM techniques can

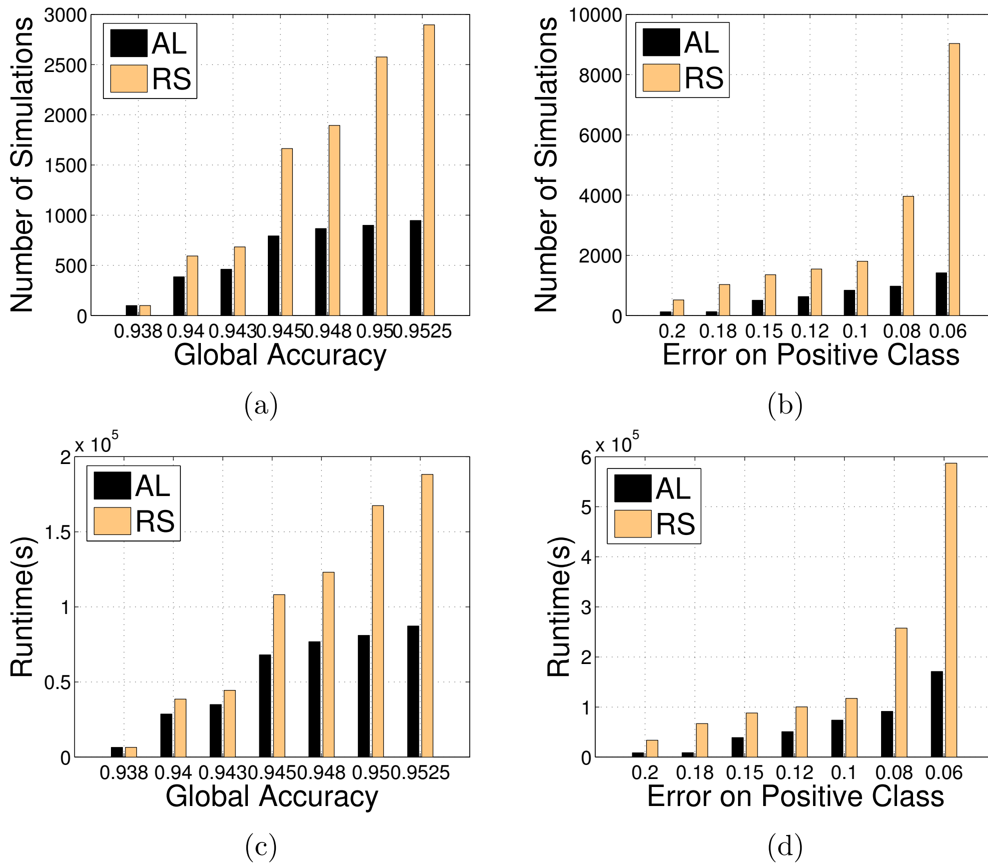


Figure 4.12: Results of chip thermal experiments (AL short for active learning, RS short for random sampling) where the global accuracy and the error on positive class are defined as the fractions of the correct predictions, using a test data set whose size is 100K. (e.g. 0.1 means 10%): (a) number of simulations for certain accuracy; (b) number of simulations for safe prediction; (c) runtime to achieve certain accuracy; (d) runtime to achieve safe prediction

be applied to reduce the required number of sensors while retaining high alertness in the observation of the working conditions of the systems.

4.6 Summary

This section presents an SVM-based active learning method for circuit performance classification and verification. There are two algorithm flows proposed: the symmetric one for classifications only considering the accuracy, and the asymmetric one concerning about both safety and accuracy. The proposed method is applied to DC/DC converter ripple noise analysis, PLL lock-time verification, and chip thermal checking. The proposed algorithms provide significant simulation reduction and efficiency improvement in all the experiments.

5. STATISTICAL AMS VERIFICATION AND PARAMETRIC ANALYSIS VIA BAYESIAN LEARNING

The effectiveness of learning based methods applied to AMS verification problems has been demonstrated in the preceding section. However, even an asymmetric strategy is developed to conform the conservative nature of verification problems, the completeness and reliability of the verification solutions is not guaranteed or preserved to an acceptable extent. As a first step towards a conservative learning based method, it is essential to evaluate the coverage or reliability of the learned models. In this light, statistical learning methods can be a good fit since the evaluation of the coverage or reliability can be achieved in the statistical sense.

The verification tasks can be facilitated with accurate and computationally efficient models that estimate the corresponding circuit performances or specifications based on the given circuit parameters. Besides, both the design phase and the testing phase can benefit from those models if they can reliably capture complex dependencies of circuit performances on essential circuit and device parameters or test signatures. To build such models, this section presents a novel Bayesian learning technique, namely *sparse relevance kernel machine* (SRKM), for characterizing analog circuits with sparse statistical regression models. SRKM not only produces accurate models learned from a moderate amount of simulation or measurement data, but also computes a probabilistically inferred weighting factor quantifying the criticality of each parameter as part of the overall learning framework, hence offering a powerful enabler for variability modeling, failure diagnosis, and test development. Compared to other popular learning-based techniques, the proposed SRKM produces more accurate models, requires less amount of training data, and extracts more re-

liable parametric ranking.

Throughout this section, bold capital letters denote matrices and bold letters in lower case denote vectors. For a matrix $\boldsymbol{X}$, we use $\boldsymbol{X}(i,j)$ to denote its entry at the i-th row and the j-th column. We represent the i-th entry of a vector $\boldsymbol{x}$ by $\boldsymbol{x}(i)$. And $diag(\boldsymbol{d})$ is a diagonal matrix whose diagonal elements are defined by $\boldsymbol{d}$.

5.1 Introduction

For AMS systems, it is essential to characterize the dependencies of circuit performances/specifications on various circuit and device parameters or test signatures for purposes such as design, verification and test optimization. However, doing so is not trivial since the targeted dependencies are usually complex and nonlinear with deep-rooted correlations, making it arduous to reliably quantify the importance of numerous parameters. Particularly, embedding parametric analysis into statistical AMS verification flow is eminently appealing, but faces obstacles from both the complex AMS nature and the feature weighting problem.

5.1.1 Challenges from the Complex AMS Nature

For characterizing sophisticated circuit systems, machine learning techniques based on circuit simulations or measurements have been proven to be effective and produced promising outcomes. For example, the preceding section uses the SVM as a nonlinear classifier to capture the mapping from input parameters to circuit performance. A regression extension to the SVM is employed in [15] to rank circuit parameters based on their correlations with unexpected timing deviations. Additionally, Bayesian inference is often used to build statistical circuit models. For instance, a co-learning Bayesian model is proposed in [133] to efficiently model the performance of AMS circuits.

Input parameters of vastly different amplitudes are often normalized to the same

value range before being fed to a machine learning algorithm. This and other issues may amplify the impact of redundant or noisy parameters in the model and aggravate its vulnerability to noisy training data and/or inclusion of noisy or redundant input features (parameters). Since the sensitivities of circuit performances to various parameters may vary vastly, it is instrumental to extract accurate statistical models and reliable parameter criticality simultaneously. While traditional feature selection or importance ranking techniques may help to identify and select some important parameters out of a large parameter set, building models only with the selected parameters usually degrades the model performance and few of those techniques can guide the model to achieve higher accuracy [61, 26]. These difficulties present important roadblocks to analog/mixed-signal circuit characterization with machine learning techniques.

In addition, to build an accurate predictor, most machine learning techniques require appropriate pre-processing of the features/parameters in training data, which may require domain expertise, especially for scenarios where there is noise and redundancy in the collected samples. Regrading this problem, due to the diversity of AMS circuits, developing an accurate statistical model with automatic parameter handling or processing in training samples is of great demand.

5.1.2 Challenges from the Feature Weighting Problem

Generally, the performance of machine learning techniques, especially classification and regression techniques, is heavily affected by the quality of the training data. To enhance the quality of the training data set before feeding it to a training model, it usually requires great efforts in pre-processing the so-called features, which are basically the circuit parameters in the scenario of AMS verification. Techniques such as feature construction/extraction [56] and feature selection [24, 56, 21] may be em-

ployed to convert the raw data into features that will be finally used to train the model.

Traditionally, feature selection may be performed by combinatorial search [112, 130] to incrementally add or remove features from the selected subset, which is evaluated by the performance of its resulting predictor. Another kind of methods [20, 123, 95] trim the feature space with regularization based methods, like introducing new regularization terms into the cost function, to perform the feature selection. All these techniques are considered as *linear* feature selection methods since they handle or formulate the features in a linear manner, such as combinatorial search or L_1 norm regularization. To capture the nonlinear dependencies among features and their nonlinear "relevancy" to the targets, some other methods [80, 61, 26] switch the roles of features and samples in their learning models and apply the kernel method to the features instead of samples. This kind of methods also belong to the category of regularization based methods since their optimization models using Euclidean inner product as their kernel functions are equivalent to Lasso regression [123] based on the conclusion provided by [80]. A drawback of such methods is that their results usually only improves the regularization of the learning model but not the accuracy, since they work independently as preceding filters of the training process.

Moreover, in AMS verification where features' relevancy may vary in a large range, proper weighting of the features may improve the learning quality if the weighting can reflect the impacts of features on the targeted performance. In this sense, feature selection is a 0-1 binary weighting scheme, whose capability is limited when it comes to AMS verification applications. The task of assigning weights directly to the features and embed the weighted samples into the kernel function [124] is very challenging, since most commonly used kernel functions are nonlinear, making weights difficult to manipulate and costly to optimize.

5.1.3 Overview

To address all the problems mentioned above, this work proposes a novel Bayesian learning framework for characterizing analog circuits in tasks including, but not limited to, AMS verification by consolidating learning models with embedded feature weighting mechanisms to achieve accurate models. Since the SVM is well-known for achieving superb performance in both classification and regression applications, we propose a novel implicit feature weighting scheme in the kernel machine framework adopted by the SVM. Instead of directly manipulating the features in a traditional way, the original kernel function is atomized into the newly defined feature kernels with weighting factors that obliquely reflect the relevancy of the features. Then, the training model of the new kernel machine with feature weighting is developed following the *relevance vector machone* (RVM) [124] framework, which is a sparse Bayesian learning treatment of the SVM, to achieve a sparse model called *sparse relevance kernel machine* (SRKM) for AMS circuit characterization.

The proposed SRKM simultaneously seeks relevant training samples (i.e. samples) and parameters (i.e. features) to derive a sparse model in both the vector and parameter spaces based upon the atomized feature kernel weighting. As a result, the SRKM not only produces accurate models learned from a moderate amount of simulation or measurement data, but also computes a probabilistically inferred weighting factor quantifying the criticality of each parameter as part of the overall learning framework, hence offering a powerful enabler for variability modeling, failure diagnosis, and test development. In addition, an iterative algorithm is developed for efficient training of the proposed SRKM.

Compared to other popular learning-based techniques, the SRKM produces more accurate models, requires less amount of training data, and extracts more reliable

parametric ranking. The effectiveness of SRKM is demonstrated in terms of the statistical variability verification of a *low-dropout regulator* (LDO) and the *built-in self-test* (BIST) development and optimization of a charge-pump *phase-locked loop* (PLL).

5.2 Relevance Kernel Machines

This section introduces the newly defined feature kernel for implicit feature weighting, by reviewing the popular kernel methods in the first subsection and describing the proposed kernel machine in the second and third subsections.

5.2.1 Kernel Methods

SVMs have been widely used in various domains like bioinformatics [23], medical imaging [66], handwriting recognition [87], and circuit applications[84], as a powerful supervised learning toolbox solving classification and regression problems. According to a recent experiment conducted by [45], the comparison of 179 classifiers evaluated on 121 data sets shows that SVM is still among the top methods.

The excellence of SVM mainly relies on two portions of its model: the cost function and the famous “kernel trick”. Similar with some other learning methods, the cost function of SVM is composed of a fitting loss term and a smoothing penalty term. By using different formulas for losses and penalties, several variants of SVM [121, 111, 70, 64] are derived from the original quadratic optimization problem. Such composition of cost functions provides exceptional robustness and regularization [141]. On the other hand, kernel trick or kernel method has shown great success in handling nonlinear problems.

In SVM, the idea of solving nonlinear problems is to map the original input vectors from the *input space*, which are not linearly solvable, into a higher dimensional space, which is referred to as *feature space* and explore a linear solution in that space. For

example, as shown in Fig. 5.1, the five 2-D training samples for classification are not linearly separable in the input space. By using the mapping $\boldsymbol{x} \rightarrow \hat{\boldsymbol{x}}$ described in Fig. 5.1, the five mapped training samples are linearly separable in the 3-D feature space.

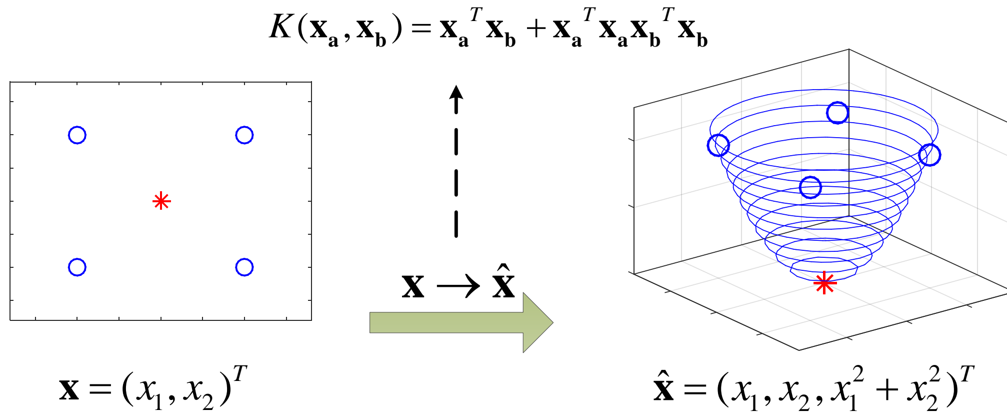


Figure 5.1: A mapping example from a 2-D space to a 3-D space with its corresponding kernel function. Different colors denote different classes.

By using the Lagrange multipliers, the exploration of the optimal separating hyperplane in the mapped higher dimensional space can be expressed as the optimization problem (4.6). It is clearly shown in (4.6) that the optimization problem can be solved by merely knowing the inner product of any pair of the mapped input vectors $\langle \hat{\boldsymbol{x}}_{\boldsymbol{i}}, \hat{\boldsymbol{x}}_{\boldsymbol{j}} \rangle$, without explicitly defining the mapping $\boldsymbol{x} \rightarrow \hat{\boldsymbol{x}}$. Therefore, defining a kernel function K that satisfies or represents $K(\boldsymbol{x}_{\boldsymbol{i}}, \boldsymbol{x}_{\boldsymbol{j}}) = \langle \hat{\boldsymbol{x}}_{\boldsymbol{i}}, \hat{\boldsymbol{x}}_{\boldsymbol{j}} \rangle$ and substituting it into the optimization problem should achieve the same results as what is produced by making the mapping $\boldsymbol{x} \rightarrow \hat{\boldsymbol{x}}$. For example, using the kernel function described in Fig. 5.1 to construct the SVM training model should be equally sufficient as using the mapping shown in Fig. 5.1 to solve the classification problem. As long as the kernel

function K satisfies Mercer's condition [129], there exists a feature space where the inner product is generated by K. In this light, a kernel function can be considered as an implicit definition of a certain mapping.

5.2.2 Feature Kernel Weighting

Although the implicit mapping defined by kernel methods are very powerful in solving nonlinear problems, similar with other learning techniques, it may easily be confused by irrelevant or redundant features. For example, suppose we further sample the 2-D input space in Fig. 5.1 and find that one of the features is actually irrelevant to the target, as Fig. 5.2 shows, using the same kernel and the same resulting mapping, the mapped input vectors can no longer be linearly solved in the 3-D feature space. Alternatively, another mapping $\boldsymbol{x} \rightarrow \tilde{\boldsymbol{x}}$ shown in Fig. 5.2 manages to map the samples in the input space into another space where a linear solution exists. The new mapping is based upon the original mapping by eliminating the irrelevant feature x_1 from the mapping.

A more complex situation is that features may have quite different relevancy as Fig. 5.3 shows. The feature x_2 is the major feature that reflects the class lables of the samples while x_1 has subtle but not negligible relevancy. This example cannot be linearly solved by the mapping $\boldsymbol{x} \rightarrow \hat{\boldsymbol{x}}$, either. A feasible mapping $\boldsymbol{x} \rightarrow \tilde{\boldsymbol{x}}$ add a weighting factor to adjust the impact of the less relevant feature x_2 and successfully reach another feature space where a linear solution is available.

Based on the implicit mapping mechanism, we propose to construct a new kernel by atomizing the existing kernel functions to achieve implicit feature weighting. Considering the example shown in Fig. 5.2 and Fig. 5.3, to get rid of the interference from the redundant or much less relevant feature, the mapping can be further extended into a even higher dimensional space by breaking any dimension involving

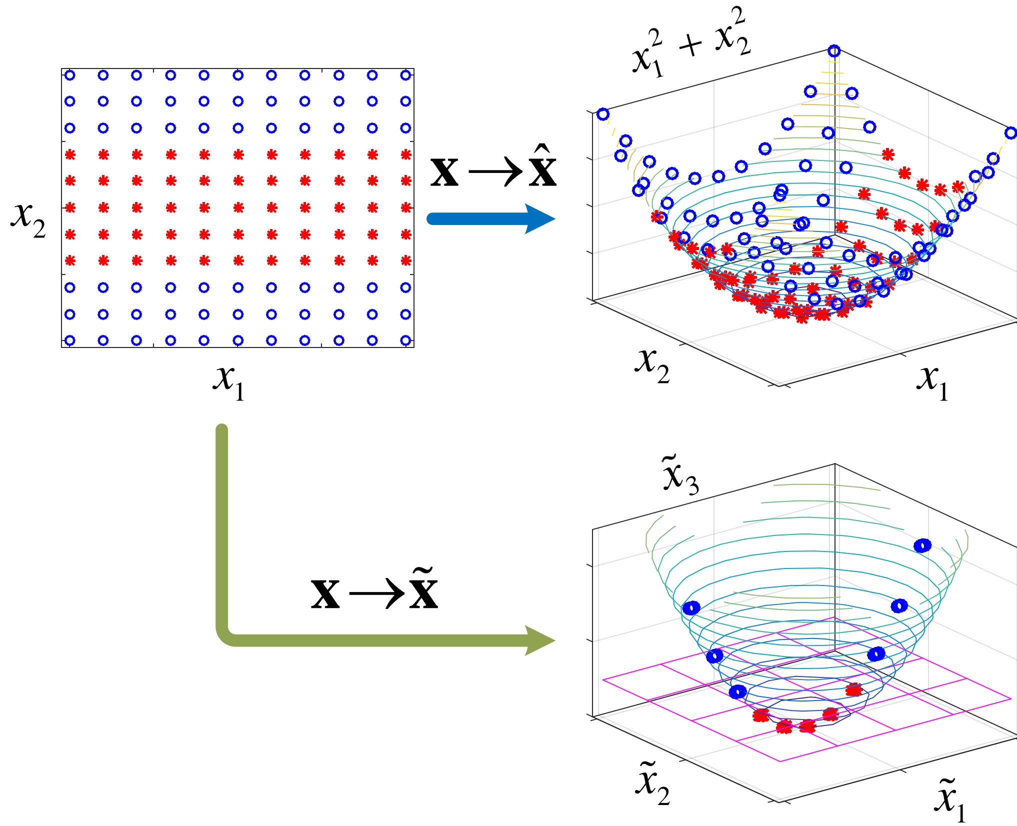


Figure 5.2: An example with an irrelevant feature, where blue circles and red stars denote two different classes respectively. It cannot be linearly solved by the kernel function corresponding to the mapping $\boldsymbol{x} = (x_1, x_2) \rightarrow \hat{\boldsymbol{x}} = (x_1, x_2, x_1^2 + x_2^2)$. The kernel function corresponds to the mapping $\boldsymbol{x} \rightarrow \tilde{\boldsymbol{x}}$ is feasible, with a linear solution denoted by the purple plane. Here $\tilde{x}_1 = x_2$, $\tilde{x}_2 = 0$, and $\tilde{x}_3 = x_2^2$.

multiple features into several dimensions that consists only one feature:

$$\hat{\boldsymbol{x}} = (x_1, x_2, x_1^2, x_2^2)^T,$$

The feasible mappings shown in Fig. 5.2 and Fig. 5.3 are actually linear projections of the above new mapping. Since the mapped samples are linearly solvable in the new 3-D spaces, it means linear solutions exist in spaces which are linearly transformed from the 4-D space defined by $\hat{\boldsymbol{x}}$.

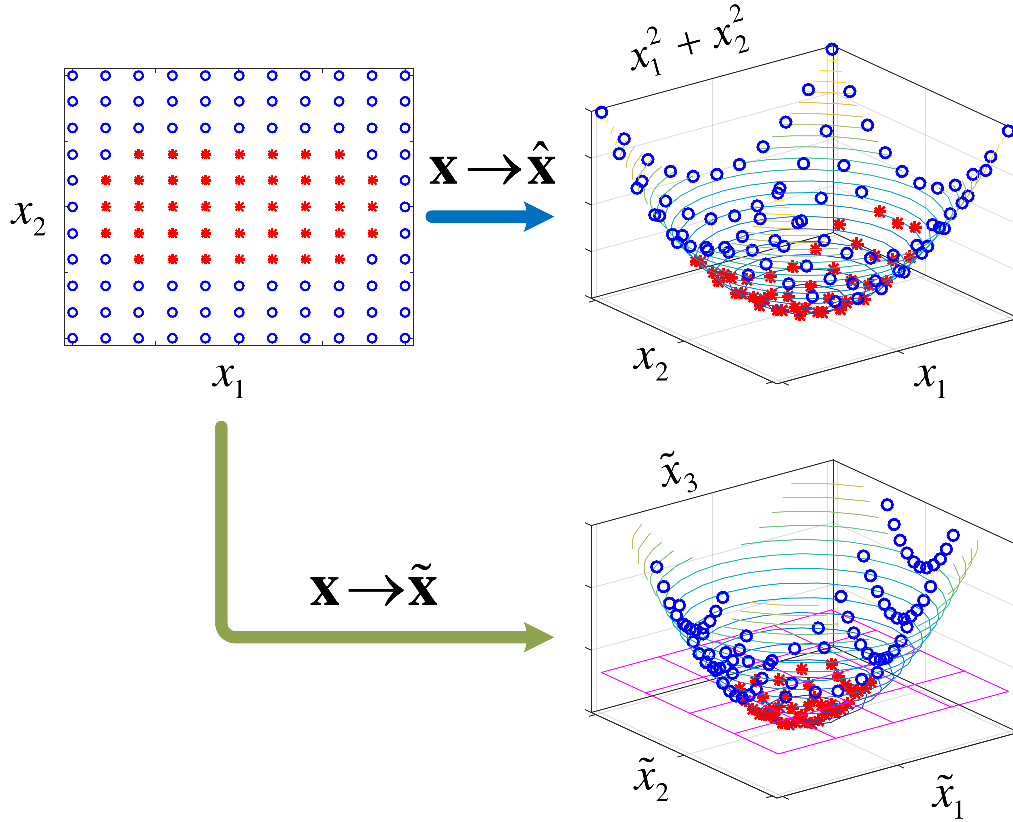


Figure 5.3: An example with irregular feature relevancy, where blue circles and red stars denote two different classes respectively. It cannot be linearly solved by the kernel function corresponding to the mapping $\boldsymbol{x} = (x_1, x_2) \rightarrow \hat{\boldsymbol{x}} = (x_1, x_2, x_1^2 + x_2^2)$. The kernel function corresponds to the mapping $\boldsymbol{x} \rightarrow \tilde{\boldsymbol{x}}$ is feasible, with a linear solution denoted by the purple plane. Here $\tilde{x}_1 = x_1$, $\tilde{x}_2 = \upsilon x_2$, and $\tilde{x}_3 = x_1^2 + \upsilon x_2^2$, where υ is a small positive weight..

While implicit mappings defined by kernel functions are more favorable, for the exact mapping $\boldsymbol{x} \rightarrow \tilde{\boldsymbol{x}}$ defined above, its corresponding kernel function is:

$$\tilde{K}(\boldsymbol{x_a}, \boldsymbol{x_b}) = (\boldsymbol{x_a}^{(1)})^T \boldsymbol{x_b}^{(1)} + (\boldsymbol{x_a}^{(1)})^T \boldsymbol{x_a}^{(1)} (\boldsymbol{x_b}^{(1)})^T \boldsymbol{x_b}^{(1)} + (\boldsymbol{x_a}^{(2)})^T \boldsymbol{x_b}^{(2)} + (\boldsymbol{x_a}^{(2)})^T \boldsymbol{x_a}^{(2)} (\boldsymbol{x_b}^{(2)})^T \boldsymbol{x_b}^{(2)},$$

where $\boldsymbol{x}^{(1)} = diag(1, 0) \cdot \boldsymbol{x}$ and $\boldsymbol{x}^{(2)} = diag(0, 1) \cdot \boldsymbol{x}$. By substituting the original

kernel function K in Fig. 5.1 into the new kernel, it can be re-written as:

$$\tilde{K}(\boldsymbol{x_a}, \boldsymbol{x_b}) = K(diag(1,0) \cdot \boldsymbol{x_a}, diag(1,0) \cdot \boldsymbol{x_b}) + K(diag(0,1) \cdot \boldsymbol{x_a}, diag(0,1) \cdot \boldsymbol{x_b}),$$

where the first term and second term are original kernels with vectors scaled by $diag(1,0)$ and $diag(0,1)$, respectively. In other words, each term eliminates one feature with a zero factor and map $\boldsymbol{x}$ to a portion of the dimensions in $\tilde{\boldsymbol{x}}$ which is isolated from that feature.

More generally, we define a selecting diagonal matrix $\boldsymbol{S_i}$ with:

$$\boldsymbol{S_i} = diag(\boldsymbol{s_i}) \tag{5.1}$$

where $\boldsymbol{s_i}(j) = 1$ for $j = i$, and $\boldsymbol{s_i}(j) = 1$, $\forall j \neq i$. For any existing kernel function $K(\boldsymbol{x_a}, \boldsymbol{x_b})$, we define a new *feature kernel* function as:

$$K_i(\boldsymbol{x_a}, \boldsymbol{x_b}) = K(\boldsymbol{S_i} \cdot \boldsymbol{x_a}, \boldsymbol{S_i} \cdot \boldsymbol{x_b}). \tag{5.2}$$

Such feature kernel only preserves the sensitivity to the i-th feature in both $\boldsymbol{x_a}$ and $\boldsymbol{x_b}$, with other features eliminated by the selecting diagonal matrix $\boldsymbol{S_i}$.

Assuming that kernel K maps samples from an f-dimensional input space to a d-dimensional space, which may be vulnerable to irrelevant or nonlinearly relevant features, we now atomize K into the sum of f weighted feature kernels by assigning one for each feature:

$$\tilde{K}(\boldsymbol{x_a}, \boldsymbol{x_b}) = \sum_{i=1}^{f} v_i K_i(\boldsymbol{x_a}, \boldsymbol{x_b}). \tag{5.3}$$

It will result in a mapping from the f-dimensional input space to another higher but up to $(d \cdot f)$-dimensional space. In this higher dimensional space, we are expecting

that the inner product of any pair of vectors can be expressed or approximated by a linear combination of the feature kernels.

In addition, for the i-th feature kernel K_i, it only represents the information provided by the i-th feature since the influence of other features in the corresponding "axes" of the mapped space is completely dismissed. Therefore, we propose to perform feature weighting via the weighting parameters υ_i as demonstrated in (5.3). Larger $|\upsilon_i|$ means the i-th kernel is more important in the kernel model. For example, in Fig. 5.2, the kernel function corresponding to the projection is $\tilde{K} = 0 \cdot K_1 + K_2$ while in Fig. 5.3, it is $\tilde{K} = K_1 + \upsilon \cdot K_2$ where $\upsilon = 3/8$. These two atomized kernel functions clearly reflect the relevancy of the two features.

One of the advantages of this weighting scheme is that the weighting parameters are much easier to manipulate compared to the schemes that directly apply weighting parameters to the input vectors. Secondly, analogously to the original kernel method, this weighting scheme actually avoids defining explicit feature weighting by instead weighting the linear combination of the feature kernels, which can be considered as an implicitly defined nonlinear weighting scheme.

For an existing kernel function K that satisfies Mercer's condition [129], the newly defined feature kernel K_i should also satisfy Mercer's condition. As a result, for all square integrable $g(\boldsymbol{x})$ we have:

$$\int_{\chi\times\chi} \tilde{K}(\boldsymbol{x_a}, \boldsymbol{x_b}) g(\boldsymbol{x_a}) g(\boldsymbol{x_b}) d\boldsymbol{x_a} d\boldsymbol{x_b} = \sum_{i=1}^{f} \int_{\chi\times\chi} \upsilon_i K_i(\boldsymbol{x_a}, \boldsymbol{x_b}) g(\boldsymbol{x_a}) g(\boldsymbol{x_b}) d\boldsymbol{x_a} d\boldsymbol{x_b} \geq 0,$$

for all $\boldsymbol{x_a}, \boldsymbol{x_b} \in \chi$ as long as $\upsilon_i \geq 0, \forall i$. Furthermore, assuming that all the features are normalized into the same range like, for example, $\chi = [0,1]^f$, and that the original kernel function K is not dependent on the ordering of the features, meaning K is

invariable when the feature ordering in $\boldsymbol{x}$ is changed, we have:

$$\int_{\chi\times\chi} K_i(\boldsymbol{x_a}, \boldsymbol{x_b}) g(\boldsymbol{x_a}) g(\boldsymbol{x_b}) d\boldsymbol{x_a} d\boldsymbol{x_b} = \int_{\chi\times\chi} K_j(\boldsymbol{x_a}, \boldsymbol{x_b}) g(\boldsymbol{x_a}) g(\boldsymbol{x_b}) d\boldsymbol{x_a} d\boldsymbol{x_b},$$

and consequently:

$$\int_{\chi\times\chi} \tilde{K}(\boldsymbol{x_a}, \boldsymbol{x_b}) g(\boldsymbol{x_a}) g(\boldsymbol{x_b}) d\boldsymbol{x_a} d\boldsymbol{x_b} = (\sum_{i=1}^{f} v_i) \int_{\chi\times\chi} K_1(\boldsymbol{x_a}, \boldsymbol{x_b}) g(\boldsymbol{x_a}) g(\boldsymbol{x_b}) d\boldsymbol{x_a} d\boldsymbol{x_b} \geq 0,$$

provided that $\sum_{i=1}^{f} v_i \geq 0$. Hence, the sum of the weighted feature kernels satisfies Mercer's condition if the weighting parameters v_i are properly constrained.

5.2.3 Relevance Kernel Machine

In SVM, the training model is often solved in its dual form and, by leveraging *Karush–Kuhn–Tucker* (KKT) conditions, the prediction model is based upon the following decision function:

$$y(\boldsymbol{x}; \boldsymbol{w}) = \sum_{i=1}^{N} w_i K(\boldsymbol{x}, \boldsymbol{x_i}), \tag{5.4}$$

where $\{\boldsymbol{x_i}\}_{i=1}^{N}$ are the training examples and w_i are actually the Lagrange multipliers referred to as α_i in the SVM. The training process of learning methods using (5.4) as their decision function is to infer all the parameters w_i in (5.4) given the corresponding targets $\{t_i\}_{i=1}^{N}$ of the training examples $\{\boldsymbol{x_i}\}_{i=1}^{N}$.

Taking the error e_i between t_i and $y(\boldsymbol{x_i}; \boldsymbol{w})$ into consideration, the kernel based machine can be written as:

$$\boldsymbol{t} = \boldsymbol{\Phi_w} \cdot \boldsymbol{w} + \boldsymbol{e}, \tag{5.5}$$

where $\boldsymbol{\Phi_w}$ is an $N \times N$ matrix defined by $\boldsymbol{\Phi_w}(i, j) = K(\boldsymbol{x_i}, \boldsymbol{x_j})$, $\boldsymbol{t}$ is simply the target

vector with $\boldsymbol{t}(i) = t_i$ and $\boldsymbol{e}$ is the error vector for the N training samples. Learning methods like SVM aim at minimizing $\boldsymbol{e}$ with fitting loss term and regularizing $\boldsymbol{w}$ with smoothing regularization term simultaneously.

As previously mentioned, most popular kernel functions such as Gaussian kernel (also known as *radial basis function*) and polynomial kernel are vulnerable to features with complicate relevancy. For training data with F features, we define a new *relevance kernel machine* by embedding the sum of weighted feature kernels as a new kernel function into (5.5) to handle feature relevancy:

$$\boldsymbol{t} = \boldsymbol{\Phi_{wv}} \cdot (\boldsymbol{w} \otimes \boldsymbol{v}) + \boldsymbol{e}, \tag{5.6}$$

where $\boldsymbol{\Phi_{wv}}$ is an $N \times (NF)$ matrix defined by $\boldsymbol{\Phi_{wv}}(i, (j-1)F+k) = K_k(\boldsymbol{x_i}, \boldsymbol{x_j})$ with $i, j \in [1, N]$ and $k \in [1, F]$. Besides, $\boldsymbol{w} \otimes \boldsymbol{v}$ in (5.6) is the tensor product of vector $\boldsymbol{w}$ and $\boldsymbol{v}$ which yields an $(NF) \times 1$ row vector with the definition $(\boldsymbol{w} \otimes \boldsymbol{v})((j-1)F+k) = w_j v_k$ where $j \in [1, N]$ and $k \in [1, F]$.

In this new kernel machine, the exploration of $\boldsymbol{w}$ is to figure out relevant samples, which is a notion similar with the support vectors in SVM, while the exploration of $\boldsymbol{v}$ is to weight relevant features by tuning the kernel mapping. Overall the whole vector $(\boldsymbol{w} \otimes \boldsymbol{v})$ weights the atomized feature kernels in the decision function and hence we name the new kernel machine as *relevance kernel machine* (RKM). A learning method that minimize $\boldsymbol{e}$ and regularize $(\boldsymbol{w} \otimes \boldsymbol{v})$ simultaneously should be able to achieve an accurate predictor with built-in feature weighting.

Another useful property of the RKM is that the roles of $\boldsymbol{w}$ and $\boldsymbol{v}$ in the model (5.6) are relatively symmetric and transposable. Model (5.5) can be derived from model (5.6) by moving $\boldsymbol{v}$ from the tensor product to the design matrix defined by $\boldsymbol{\Phi_w}(i, j) = \sum_{k=1}^{F} v_k K_k(\boldsymbol{x_i}, \boldsymbol{x_j})$ with $i, j \in [1, N]$. Similarly, the following model can

be derived from (5.6) by instead moving $\boldsymbol{w}$ to the design matrix:

$$\boldsymbol{t} = \boldsymbol{\Phi_v} \cdot \boldsymbol{v} + \boldsymbol{e}, \tag{5.7}$$

where $\boldsymbol{\Phi_v}$ is an $N \times F$ matrix defined by $\boldsymbol{\Phi_v}(i,k) = \sum_{j=1}^{N} w_j K_k(\boldsymbol{x_i}, \boldsymbol{x_j})$ with $i \in [1, N]$ and $k \in [1, F]$. This property indicates that the exploration process of $\boldsymbol{w}$ and $\boldsymbol{v}$ may be unified, which will be discussed in details in the next section.

5.3 Sparse Relevance Kernel Machine for AMS Characterization

Assuming that there are F circuit parameters of interest with which the circuit is described by a parameter (feature) vector $\boldsymbol{x}$. A sample of the circuit is defined by a pair $\{\boldsymbol{x}_i, t_i\}$ where t_i is the circuit performance under the configuration $\boldsymbol{x}_i$. By collecting a number of N samples, the objective of regression is to capture the mapping $\Psi : \boldsymbol{x} \to t$ with a function $y(\boldsymbol{x})$ whose output can be used as a prediction of the performance t. The set $\{\boldsymbol{x_i}\}_{i=1}^{N}$ along with its corresponding $\{t_i\}_{i=1}^{N}$ is usually referred to as the training data set.

For analog/mixed-signal circuits, the objective of the proposed method is to capture the mapping Ψ from the parameters to the performance and to perform parametric analysis evaluating the relevancy of circuit parameters simultaneously. For this, the RKM proposed in the preceding section is a good fit, with both vector weights w and feature weights v contributing to the accuracy of the regression model, and feature weights v reflecting parameter relevancy additionally.

Despite the exceptional functionality provided by the RKM, for complex AMS circuits, there may be a large number of circuit parameters which may severely inflate the searching space of the solution. A sparse treatment of the RKM is highly appealing since the sparsity may help to reduce the complexity. And in the scenario of parametric analysis, it may help to filter out irrelevant or redundant circuit

parameters as much as possible if $\boldsymbol{v}$ tends to be sparse. Since Bayesian learning frameworks with sparse prior [136, 124, 137] is capable of producing highly sparse models, we develop a sparse relevance kernel machine under the Bayesian learning framework in this section.

5.3.1 Relevance Vector Machine

The *relevance vector machines* (RVM) [124] is a sparse Bayesian model providing a viable probabilistic framework for regression. The Bayesian network model of the RVM is shown in Fig. 5.4.

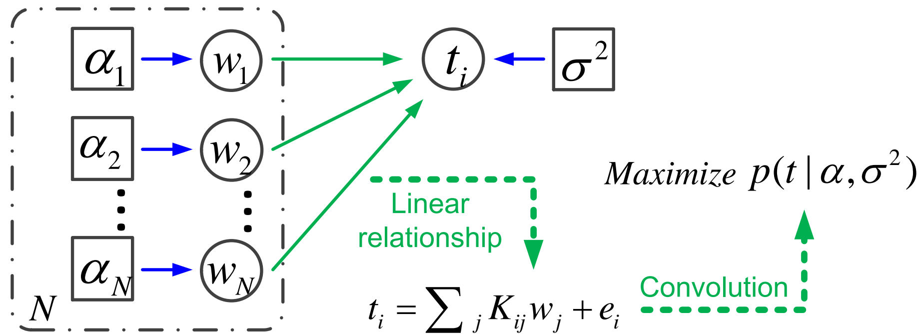


Figure 5.4: The network model of the RVM where $K_{ij} = K(x_i, x_j)$. Circles denote random variables and squares denote deterministic model parameters.

Given the input vectors $\{\boldsymbol{x_n}\}_{n=1}^{N}$ and their corresponding targets $\{t_n\}_{n=1}^{N}$, the RVM is used to probabilistically determine the model (5.5). The given target values are modeled as random variables by assuming every entry of the additive noise $\boldsymbol{e}$ is a zero-mean Gaussian random variable with variance σ^2. Further assuming the independence of t_n gives the following probability distribution

$$p(\boldsymbol{t}|\boldsymbol{w}, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}(y(\boldsymbol{x_n}; \boldsymbol{w}), \sigma^2). \tag{5.8}$$

Different from deterministic learning models (e.g. the SVM) that compute $\boldsymbol{w}$ directly, the RVM defines the prior distribution of $\boldsymbol{w}$ as independent zero-mean Gaussian random variables with variance $\boldsymbol{\alpha}$, and compute $\boldsymbol{\alpha}$ instead of $\boldsymbol{w}$ in the training process:

$$p(\boldsymbol{w}|\boldsymbol{\alpha}) = \prod_{n=1}^{N} \mathcal{N}(0, \alpha_n^{-1}). \tag{5.9}$$

If $\alpha_i < \infty$, w_i is called a *relevance vector* since it has a variance greater than zero, allowing x_i making contributions to the decision function. Note that the RVM performs prediction with the posterior probability of the internal variables (i.e. the weights). Via convolution of Gaussian distributions, the covariance and mean of the posterior $p(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{\alpha}, \sigma^2)$ can be shown to be respectively:

$$\boldsymbol{\Sigma} = (\sigma^{-2}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{A})^{-1}, \tag{5.10}$$

$$\boldsymbol{\mu} = \sigma^{-2}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{t}, \tag{5.11}$$

where $\boldsymbol{A} = diag(\boldsymbol{\alpha})$.

The objective of the Bayesian network is to find the most probable model parameters with the given training samples, i.e., to maximize the posterior $p(\boldsymbol{w}, \boldsymbol{\alpha}, \sigma^2|\boldsymbol{t})$. The objective can be decomposed as:

$$p(\boldsymbol{w}, \boldsymbol{\alpha}, \sigma^2|\boldsymbol{t}) = p(\boldsymbol{w}|\boldsymbol{\alpha}, \sigma^2, \boldsymbol{t}) \cdot p(\boldsymbol{\alpha}, \sigma^2|\boldsymbol{t}).$$

Based on the assumption provided in [124] that the most probable values of the parameters $\boldsymbol{\alpha}$ and σ^2 from $p(\boldsymbol{w}, \boldsymbol{\alpha}, \sigma^2|\boldsymbol{t})$ are identical to those sampled from $p(\boldsymbol{\alpha}, \sigma^2|\boldsymbol{t})$, since $p(\boldsymbol{w}|\boldsymbol{\alpha}, \sigma^2, \boldsymbol{t})$ is Gaussian, the optimal $\boldsymbol{w}$ that maximizes $p(\boldsymbol{w}, \boldsymbol{\alpha}, \sigma^2|\boldsymbol{t})$ can be determined once the optimal $\boldsymbol{\alpha}$ and σ^2 are obtained, and thus maximizing $p(\boldsymbol{w}, \boldsymbol{\alpha}, \sigma^2|\boldsymbol{t})$

with respect to $\boldsymbol{w}$, $\boldsymbol{\alpha}$ and σ^2 is actually maximizing $p(\boldsymbol{\alpha}, \sigma^2|\boldsymbol{t})$ with respect to only $\boldsymbol{\alpha}$ and σ^2. Furthermore, based on the Bayes' rule, such objective is equivalent to:

$$\arg\max_{\boldsymbol{\alpha},\sigma^2} p(\boldsymbol{t}|\boldsymbol{\alpha}, \sigma). \tag{5.12}$$

Once the optimal $\boldsymbol{\alpha}$ and σ^2 are computed by the training process, for any unknown feature vector $\hat{\boldsymbol{x}}$, the expectation of its predicted target is:

$$\hat{y} = \boldsymbol{\mu}^T \boldsymbol{\phi}(\hat{\boldsymbol{x}}),$$

and the variance is:

$$\hat{\sigma} = \sigma^2 + \boldsymbol{\phi}(\hat{\boldsymbol{x}})^T \boldsymbol{\Sigma} \boldsymbol{\phi}(\hat{\boldsymbol{x}}),$$

where $\boldsymbol{\phi}(\hat{\boldsymbol{x}})$ is a vector of size N whose i-th entry is defined by $\boldsymbol{\phi}(\hat{\boldsymbol{x}})(i) = K(\hat{\boldsymbol{x}}, \boldsymbol{x_i})$. The predicted variance consists of the noise estimation σ^2 on the training data and the uncertainty from the weights.

Compared to SVM, the training cost of RVM models is much higher, for which a fast iterative training algorithm is proposed by [125]. The learning process is formulated as the maximization of the marginal likelihood, or its logarithm $\mathcal{L}(\boldsymbol{\alpha})$, with respect to $\boldsymbol{\alpha}$:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\alpha}) &= \log p(\boldsymbol{t}|\boldsymbol{\alpha}, \sigma^2) \\ &= -\frac{N \log 2\pi + \log|\boldsymbol{C}| + \boldsymbol{t}^T \boldsymbol{C}^{-1} \boldsymbol{t}}{2},\end{aligned} \tag{5.13}$$

where $\boldsymbol{C} = \sigma^2 \boldsymbol{I} + \boldsymbol{\Phi} \boldsymbol{A}^{-1} \boldsymbol{\Phi}^T$. Based on the analysis of [44], in each iteration of the updating flow, the objective function (5.13) has a unique maximum with respect to $\boldsymbol{\alpha}$, which leads to the efficient algorithm by [125] that quickly approximates $\boldsymbol{\alpha}$ and

reduces the sizes of the the related matrices (for example $\boldsymbol{\Phi}$ and $\boldsymbol{\Sigma}$) and vectors (for instance $\boldsymbol{\mu}$ and $\boldsymbol{\alpha}$) by discarding entries of non-relevance vectors in each iteration.

5.3.2 Sparse Relevance Kernel Machine

The RVM is a learning model based on the decision function (5.4) focusing on producing a sparse $\boldsymbol{w}$. Under the same framework, a feature selection technique called *relevance feature vector machine*(RFVM) [26] is proposed to exchange the roles of samples and features by defining the "feature vector" $\boldsymbol{f}_i = (\boldsymbol{x}_1(i), \boldsymbol{x}_2(i), ..., \boldsymbol{x}_F(i))^T$ and a new feature kernel K' to solve the following feature weighting model:

$$\boldsymbol{t}' = \boldsymbol{\Phi}' \cdot \boldsymbol{v} + \boldsymbol{e} \tag{5.14}$$

where $\boldsymbol{t}'(i) = K'(\boldsymbol{t}, \boldsymbol{f}_i)$ and $\boldsymbol{v}$ is the weights of all features. This is a regularization based feature selection method, which focuses on building a filter method [24] by ranking the features.

To achieve high model accuracy and high quality feature weighting with the RKM, we propose a *sparse relevance kernel machine* (SRKM) whose conceptual structure is shown in Fig. 5.5.

Without using the RKM model, learning method with embedded feature weighting is often realized by directly assigning weights $\boldsymbol{v}$ to the parameters. However, as we discussed previously, since most kernel functions are nonlinear, it is extremely difficult to develop the models in the Bayesian learning contexts. For example, by assigning Gaussian prior to $\boldsymbol{v}$ in a similar way $p(\boldsymbol{v}|\boldsymbol{\beta}) = \prod_{n=1}^{F} \mathcal{N}(0, \beta_n^{-1})$, the Bayesian network described in Fig. 5.4 is extended to derive the new model in the upper left corner of Fig. 5.6. Assuming Gaussian kernel with linear direct feature weighting

$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = e^{-\gamma \| diag(\boldsymbol{v}) \cdot (\boldsymbol{x_i} - \boldsymbol{x_j}) \|^2},$$

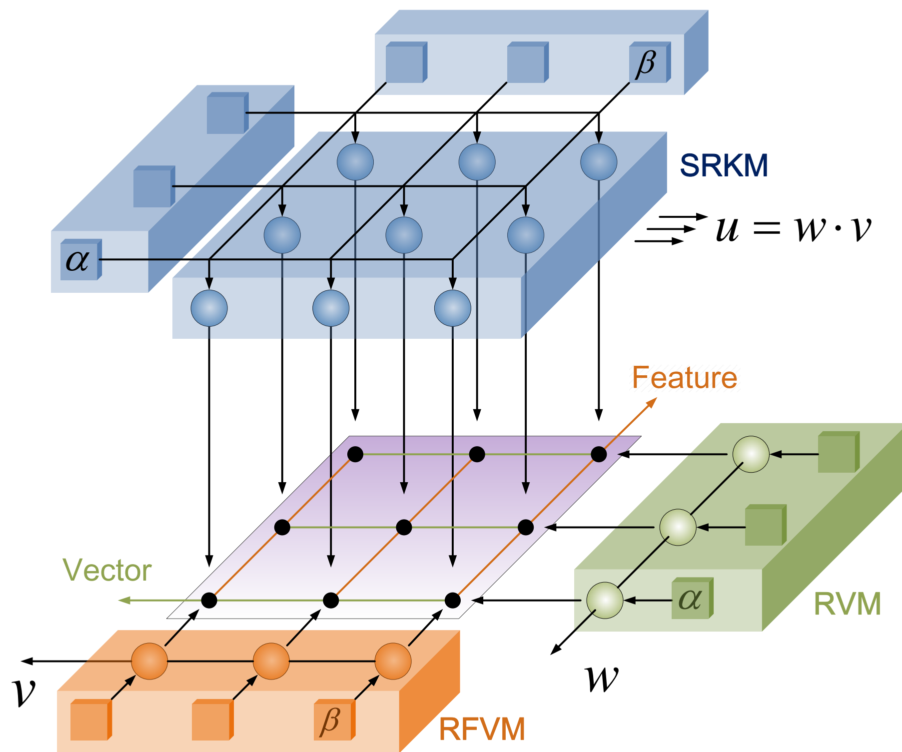


Figure 5.5: The conceptual SRKM model. Small black dots denote training data, with each row representing a vector and each column representing a feature. Circles denote random variables and squares denote deterministic model parameters.

is employed, the deterministic relationship from $\boldsymbol{w}$ and $\boldsymbol{v}$ to t is highly nonlinear. As a result, the following optimization objective of the Bayesian training process is not analytically computable and hence hinders the optimization based training process:

$$p(\boldsymbol{t}|\boldsymbol{\alpha},\boldsymbol{\beta},\sigma^2) = \iint p(\boldsymbol{t}|\boldsymbol{w},\boldsymbol{v},\sigma^2) \cdot p(\boldsymbol{w}|\boldsymbol{\alpha}) \cdot p(\boldsymbol{v}|\boldsymbol{\beta}) d\boldsymbol{w} d\boldsymbol{v}. \tag{5.15}$$

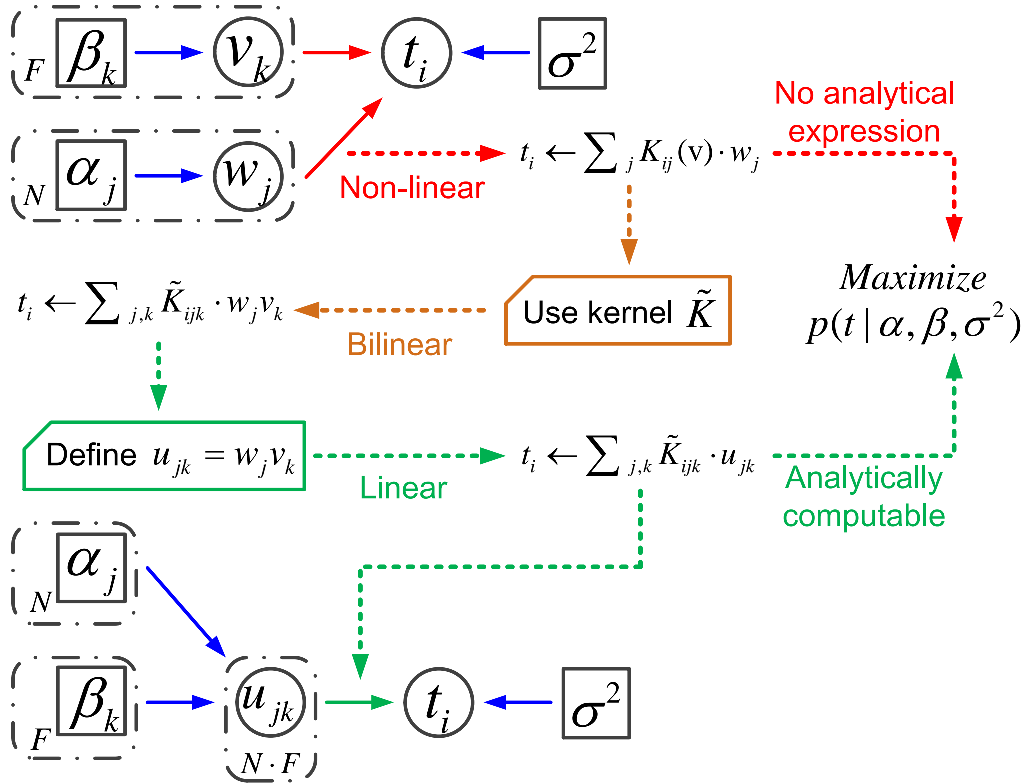


Figure 5.6: Development of SRKM Bayesian network where $K_{ijk} = K_k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Circles denote random variables and squares denote deterministic model parameters.

As the first step towards developing the SRKM, we embed the RKM into the Bayesian learning framework to handle the feature weighting problem. Leveraging the relevance kernel machine in the Bayesian network helps to simplify the highly

nonlinear deterministic relationships into a bilinear form. As described in (5.6), $\boldsymbol{t}$ can be expressed as linear combinations of a series of $\boldsymbol{w}(j)\boldsymbol{v}(k)$ terms.

However, inference in Bayesian networks with nonlinear or even bilinear deterministic relationships requires great effort like piece-wise linearization [29] or dynamic discretization [94] to deal with the nonliearity, which may greatly boost the complexity of the learning model. To address this problem, instead of defining $\boldsymbol{w}$ and $\boldsymbol{v}$ as separate Gaussian random variables, we replace the term $\boldsymbol{w}(j)\boldsymbol{v}(k)$ with a single random variable u_{jk} for all $j \in [1, N]$ and $k \in [1, F]$, which results in linear deterministic relationships from u_{jk} to $\boldsymbol{t}$. If we define a new vector $\boldsymbol{u}$ of size $(N \cdot F)$ whose entry $\boldsymbol{u}((j-1)N + k) = u_{jk} = \boldsymbol{w}(j) \cdot \boldsymbol{v}(j)$, the model becomes:

$$\boldsymbol{t} = \boldsymbol{\Phi}_{\boldsymbol{u}} \cdot \boldsymbol{u} + \boldsymbol{e}, \tag{5.16}$$

where the design matrix $\boldsymbol{\Phi}_{\boldsymbol{u}}$ is identical to $\boldsymbol{\Phi}_{\boldsymbol{wv}}$ in (5.6).

In the RVM, the zero-mean Gaussian prior distribution of $\boldsymbol{w}$ tends to help the model converge to a sparse $\boldsymbol{w}$ since the resulting marginal prior distribution over $\boldsymbol{w}$ is the product of *Student-t* distributions. Similarly, to achieve sparsity in $\boldsymbol{u}$, we also define their prior distributions as independent zero-mean Gaussian distributions. Considering the nature of $\boldsymbol{u}$, if $u_{j,k}$ is irrelevant, meaning the distribution of $u_{j,k}$ is infinitely peaked at zero, either the i-th sample $(\{u_{j,k}\}_{k=1}^{F})$ or the j-th parameter $(\{u_{j,k}\}_{j=1}^{N})$ should be irrelevant as well. To reflect this, we define a proper prior for $\boldsymbol{u}$ as:

$$p(\boldsymbol{u}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = \prod_{j=1}^{N} \prod_{k=1}^{F} \mathcal{N}(0, \alpha_j^{-1} \beta_k^{-1}), \tag{5.17}$$

which leads to our proposed computable linear Bayesian network shown in the bottom left corner of Fig. 5.6 and our conceptual model described in Fig. 5.5. An infinite

$\alpha_j\beta_k = \infty$ means $u_{jk} = 0$ and the corresponding feature kernel is irrelevant to the final decision function. In addition, if $\alpha_j \to \infty$, all the $\{u_{j,k}\}_{k=1}^F$ are zero, meaning the j-th sample is discarded from the set of relevance vectors. Likewise, if $\beta_k < \infty$, the k-th parameter is relevant and there should be at least one non-zero $u_{i,j}$ for $i \in [1, N]$.

Under the same Bayesian inference framework, the posterior covariance and mean of $p(\boldsymbol{u}|\boldsymbol{t}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \sigma^2)$ in the proposed Bayesian network are found to be:

$$\boldsymbol{\Sigma_u} = (\sigma^{-2}(\boldsymbol{\Phi_u})^T\boldsymbol{\Phi_u} + \boldsymbol{A_u})^{-1}, \tag{5.18}$$

$$\boldsymbol{\mu_u} = \sigma^{-2}\boldsymbol{\Sigma_u}(\boldsymbol{\Phi_u})^T\boldsymbol{t}, \tag{5.19}$$

where $\boldsymbol{A_u} = diag(\boldsymbol{\alpha} \otimes \boldsymbol{\beta})$ and $\boldsymbol{\Phi_u}$ is the new design matrix defined in (5.16). The formulas (5.18) and (5.19) are in the same form as the posterior covariance and mean of w in the RVM, and consequently solvable with the existing RVM algorithms.

5.3.3 Efficient Training Algorithm

The marginal likelihood maximization [125] required in training the RVM model is solved in an iterative process similar to the well-known *expectation maximization* (EM) algorithm. Due to the required matrix operations, the worst case computational complexity in each iteration is $O(N^2F + N^2M)$ [72] if there are M relevance vectors in that iteration and F features in total. By performing one-time pre-computation of the full $N \times N$ design matrix Φ with the complexity of $O(N^2F)$ and pre-computing $(\Phi^T\Phi)$, the complexity of each iteration can be reduced to $O(NM^2)$ by using $O(N^2)$ memory instead of $O(NM)$.

For the SRKM, by solving (5.16) with this algorithm, since the size of the vector $\boldsymbol{u}$ is $(N \cdot F)$, a large number of features F will blow up the worst case computational

complexity for each iteration from $O(NM^2)$ to $O(NFM^2E^2)$ if there are M relevance vectors and E relevance features in that iteration.

To address this computational challenge, our proposed efficient algorithm leverages the property that $\boldsymbol{w}$ and $\boldsymbol{v}$ are interchangeable in the bilinear Bayesian network (5.6), and that either vector can be merged into the design matrix to reduce our model to (5.5) or (5.7). As Fig. 5.7 shows, fixing $\boldsymbol{\alpha}$ and moving the resulting posterior expectation of $\boldsymbol{w}$ (denoted by $\bar{\boldsymbol{w}}$) into the design matrix (i.e. converting Φ_u in (5.16) to Φ_v in (5.7)) will reduce every column in Fig. 5.7 to a single weight v_j with its prior β_j. More specifically, the reduced model can be represented as:

$$\boldsymbol{t} = \boldsymbol{\Phi}_{\boldsymbol{v},\bar{\boldsymbol{w}}} \cdot \boldsymbol{v} + \boldsymbol{e}, \tag{5.20}$$

where $\boldsymbol{\Phi}_{\boldsymbol{v},\bar{\boldsymbol{w}}}$ is an $N \times F$ matrix defined by $\boldsymbol{\Phi}_{\boldsymbol{v},\bar{\boldsymbol{w}}}(i,k) = \sum_{j=1}^{N} \bar{\boldsymbol{w}}(j) K_k(\boldsymbol{x_i}, \boldsymbol{x_j})$ with $i \in [1, N]$ and $k \in [1, F]$. This model explores the optimal $\boldsymbol{\beta}$ with the given $\boldsymbol{\alpha}$ and $\bar{\boldsymbol{w}}$, and produces the posterior mean of $\boldsymbol{v}$ as:

$$\bar{\boldsymbol{v}} = \sigma^{-2} \boldsymbol{\Sigma_v} \boldsymbol{\Phi}_{\boldsymbol{v},\bar{\boldsymbol{w}}}^T \boldsymbol{t}, \tag{5.21}$$

where $\boldsymbol{\Sigma_v} = (\sigma^{-2} \boldsymbol{\Phi}_{\boldsymbol{v},\bar{\boldsymbol{w}}}^T \boldsymbol{\Phi}_{\boldsymbol{v},\bar{\boldsymbol{w}}} + \boldsymbol{B})^{-1}$ and $\boldsymbol{B} = diag(\boldsymbol{\beta})$. The computation of the posterior mean and variance is analogous to the computation provided in (5.10) and (5.11).

Similarly, row-wise reduction by fixing $\boldsymbol{\beta}$ and moving the $\bar{\boldsymbol{v}}$ obtained from (5.21) into the design matrix converts the proposed network to another RVM network with $\boldsymbol{w}$ and $\boldsymbol{\alpha}$:

$$\boldsymbol{t} = \boldsymbol{\Phi}_{\boldsymbol{w},\bar{\boldsymbol{v}}} \cdot \boldsymbol{w} + \boldsymbol{e}, \tag{5.22}$$

where $\boldsymbol{\Phi}_{\boldsymbol{w},\bar{\boldsymbol{v}}}$ is an $N \times N$ matrix defined by $\boldsymbol{\Phi}_{\boldsymbol{w},\bar{\boldsymbol{v}}}(i,j) = \sum_{k=1}^{N} \bar{\boldsymbol{v}}(k) K_k(\boldsymbol{x_i}, \boldsymbol{x_j})$ with

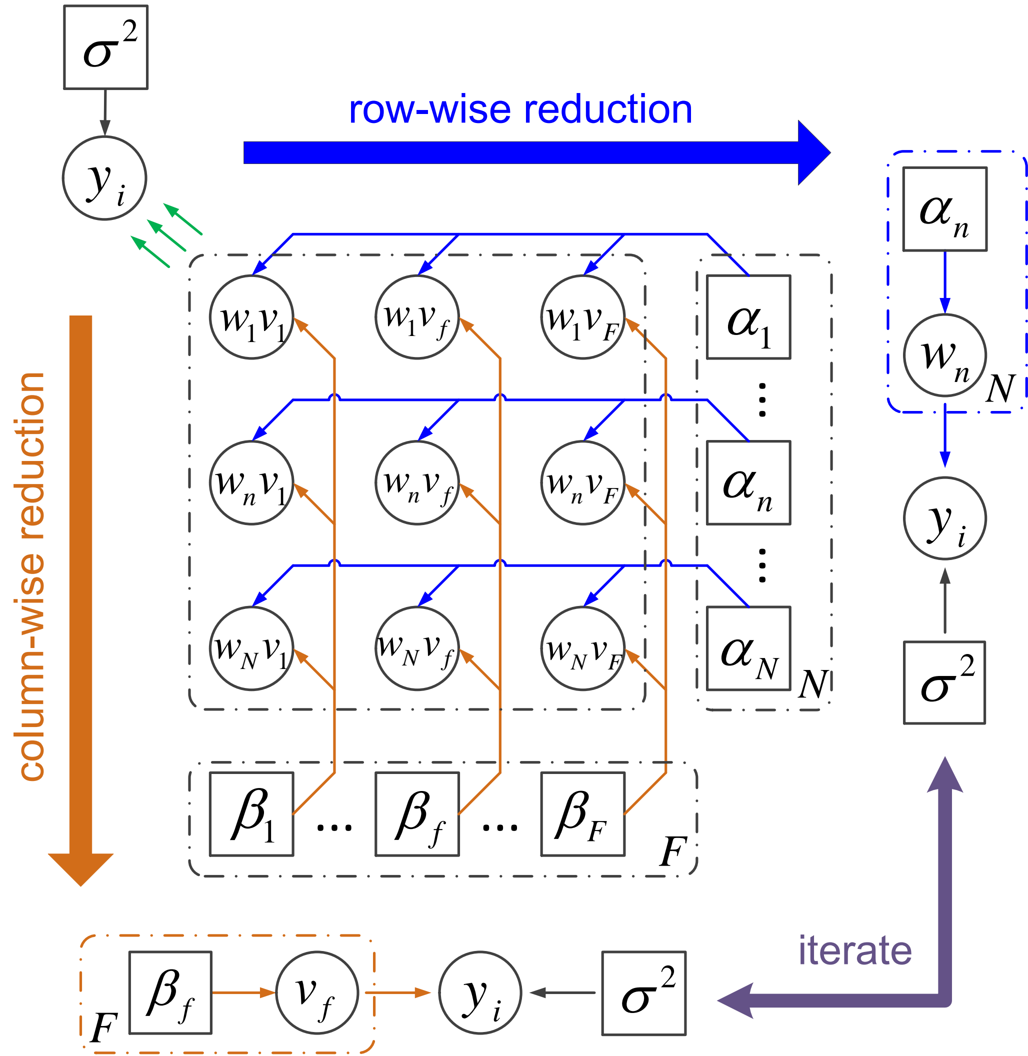


Figure 5.7: Efficient SRKM model with network reduction. Circles denote random variables and squares denote deterministic model parameters.

$i \in [1, N]$ and $j \in [1, N]$. Instead, this model searches the optimal $\boldsymbol{\alpha}$ with the given $\boldsymbol{\beta}$ and $\bar{\boldsymbol{v}}$, and produces the posterior mean of $\boldsymbol{w}$ as:

$$\bar{\boldsymbol{w}} = \sigma^{-2} \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Phi}^T_{\boldsymbol{w},\bar{\boldsymbol{v}}} \boldsymbol{t}, \tag{5.23}$$

where $\boldsymbol{\Sigma}_{\boldsymbol{w}} = (\sigma^{-2} \boldsymbol{\Phi}^T_{\boldsymbol{w},\bar{\boldsymbol{v}}} \boldsymbol{\Phi}_{\boldsymbol{w},\bar{\boldsymbol{v}}} + \boldsymbol{A})^{-1}$ and $\boldsymbol{A} = diag(\boldsymbol{\alpha})$. The computation is again based on (5.10) and (5.11), and the computed $\bar{\boldsymbol{w}}$ can be used to construct the column-wise reduced model (5.20) and compute the $\bar{\boldsymbol{v}}$ in (5.21).

The above discussion suggests an efficient two-level iterative training process. In each iteration of the top level, we reduce the model either row-wise or column-wise, and update $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$ subsequently. In the second level, the original algorithm [125] can be employed to solve either Model (5.22) or Model (5.20). The complexity in each iteration is now reduced to either $O(NM^2)$ or $O(FE^2)$. Due to the nature of this iterative algorithm, the convergence may be affected by the initialization of the parameters. For simplicity, we start the iterative process from the row-wise reduced model (5.22) to firstly update $\boldsymbol{\alpha}$ and $\boldsymbol{w}$ with $\boldsymbol{\beta}(k) = 1$ and $\bar{\boldsymbol{v}}(k) = 1/F$ for all $k \in [1, F]$.

The proposed algorithm is an EM-nested-EM-like process, which has an embedded EM-like subroutine in each iteration of the original algorithm for the RVM. Similar to the original EM-like algorithm for the RVM, the proposed algorithm may not be able to finally reach to the exact global optimal solution. Furthermore, the additionally nested EM-like subroutines may introduce larger errors into the iterative solving process since more approximations have been involved in each iteration. However, analogous to the classic EM algorithm or any EM-like algorithms, it is guaranteed that the objective marginal likelihood will be improved after each iteration.

5.4 Experiments

To demonstrate the superiority of the proposed SRKM, we compare its capability of relevant feature detection with other kernel machine based feature selection methods in the first experiment. In the second experiment, SRKM is applied to verify a realistic *low-dropout regulator* (LDO) design regarding process variations, where performance of the SRKM is compared with popular learning-based techniques including the SVM [128] and the RVM [124]. We also compare the SRKM with the RFVM [26] in terms of parameter (feature) ranking. Since the proposed SRKM is a generic learning model, its potential applicability such as *built-in self-test* BIST scheme optimization is demonstrated in the last experiment.

5.4.0.1 Relevant Feature Detection

Focusing on the feature selection functionality of the SRKM, we conduct the experiment described by [26] to evaluate its performance regarding the detection rate and false alarm rate. Here the training data is generated by the following synthetic function:

$$y(\boldsymbol{x}) = \sum_{i=0}^{20} f_i(x_i) + \sum_{i=21}^{100} 0 \cdot x_i,$$

where the first 20 relevant features are randomly uniformly selected from $[-0.5, 0.5]$ and the remaining 80 irrelevant features are uniformly generated from $[0, 20]$. And

the basis functions $\{f_i\}_{i=0}^{20}$ are randomly chosen from the following eight candidates:

$$
\begin{aligned}
f_{c1}(x) &= 40x, \\
f_{c2}(x) &= 20(1 - x^2), \\
f_{c3}(x) &= 23x^3, \\
f_{c4}(x) &= 20sin(40x - 5), \\
f_{c5}(x) &= 20e^x, \\
f_{c6}(x) &= -log_2(|x|), \\
f_{c7}(x) &= 20\sqrt{1 - x}, \\
f_{c8}(x) &= 20cos(20x - 7).
\end{aligned}
$$

The synthetic function is firstly assembled from the weight candidate basis functions. Then, the SRKM is trained based upon 1000 samples generated by the synthetic function. Features are considered as selected if their corresponding feature kernel weights are nonzero. There are 20 rounds of such process, with the minimum, maximum and average of the detection rates (that is, the number of relevant features selected out of the first 20 features) and the false alarm rates (that is, the number of irrelevant features mis-selected out of the last 80 features) reported in Fig. 5.8. Statistics for other methods including the FVM by [80], the P-SVM by [61] and the RFVM by [26] are collected from the results reported by [26].

Although the SRKM has similar average false alarm rate compared to the other three feature selection techniques, it succeeds in improving the detection rate by capturing roughly 25% more of all the relevant features. Moreover, different from the traditional feature selection methods which in fact performs 0-1 feature weighting, the SRKM assigns more diverse weights to features and such mechanism is able to help the model against misleading irrelevant features. For example, in this experiment,

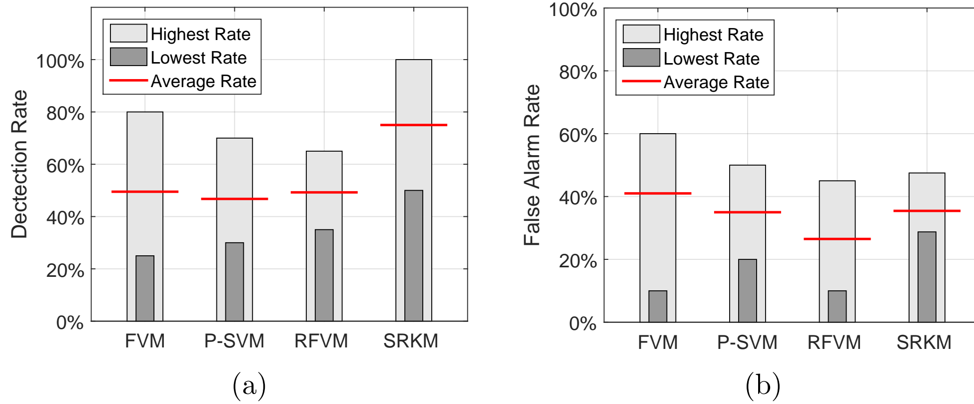


Figure 5.8: Performance comparison in feature selection: (a) detection rate; (b) false alarm rate.

the average weights assigned to the first 20 features are approximately 2.7× larger than the average ones assigned to the last 80 features. It strengthens the importance of the real relevant features and weakens the misleading effects of the false detection.

5.4.1 Variability Analysis of an LDO

Building an accurate regression model for a given analog performance and performing feature ranking among all sorts of process parameters are key to the understanding of the impacts of process variabilities and the verification of AMS systems. Since simulations or measurements are usually expensive, it is of great significance to build an accurate regression model and obtain reliable parameter weighting with a moderate amount of samples, which turns out to be a task well handled by the SRKM.

We investigate the process variations in a realistic *low-dropout regulator* (LDO) design (Fig. 5.9) proposed in [77]. We build SRKMs to analyze the impact of process variations on LDO specifications including its quiescent current, undershoot of the output voltage V_{out} and load regulation. Channel length variations of all transistors in the LDO are modeled at the SPICE level using a commercial 90nm CMOS technology

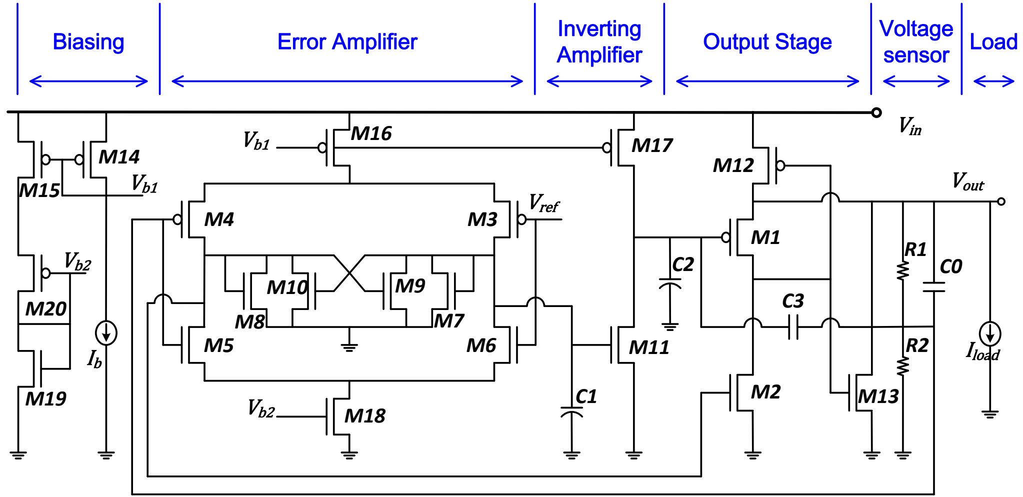


Figure 5.9: Transistor level schematic of an LDO.

design kit. We define the 20 channel length variations as the training features and train a regression model for each specification. We use various numbers of simulation samples to build a regression model relating the model process parameters with each targeted specification and test the accuracies of these models using a testing set of 1,000 simulation samples. The results are shown in Fig. 5.10.

In this experiment, *normalized mean square error* (NMSE) is used as the metric to evaluate the performance of the predictors trained with different techniques. The NMSE is defined as:

$$NMSE = \frac{1}{N}\sum_{i=1}^{N}\frac{(y_i - t_i)^2}{\bar{y}\cdot\bar{t}}$$

where N denotes the number of testing samples, y_i and t_i denote the prediction and target of the i-th testing sample, and $\bar{y}$ and $\bar{t}$ denote the mean of all the N predictions and targets respectively. As Fig. 5.10 shows, the SRKM out-performs the popular SVM and RVM in all cases by achieving one-order of magnitude lower NMSEs. In addition, the trends illustrated in Fig. 5.10 imply that the SRKM can even achieve higher accuracy with fewer samples. For example, in all three cases, the

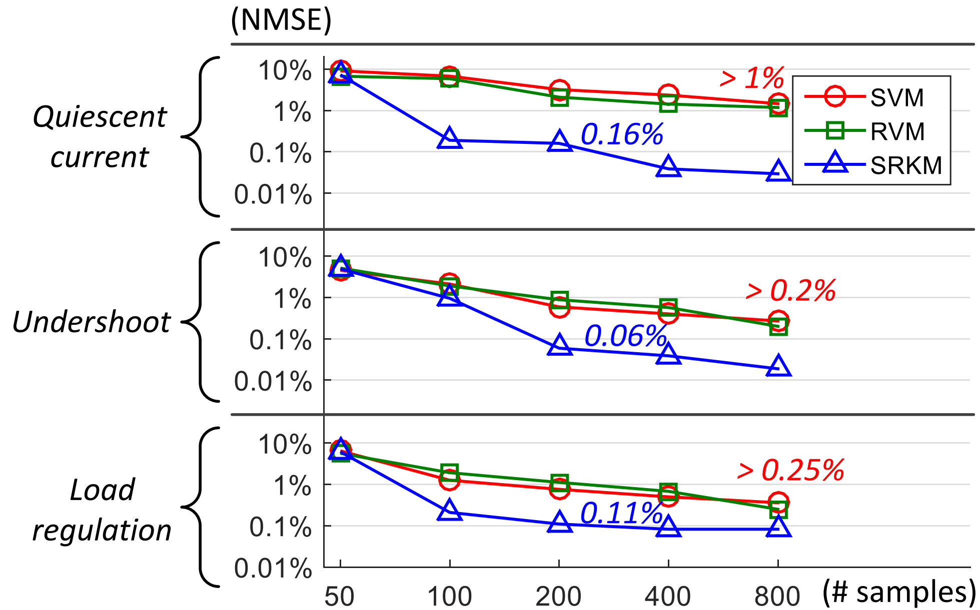
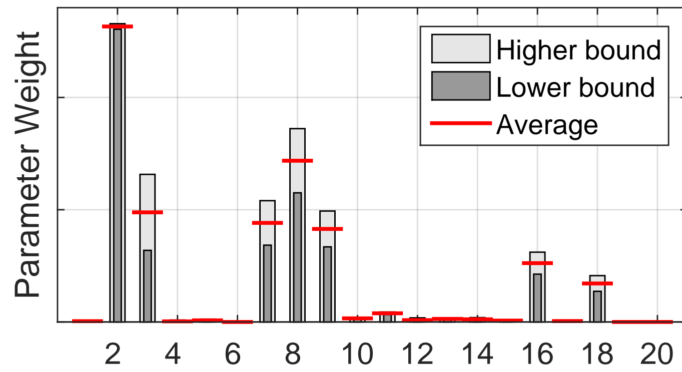


Figure 5.10: Regression performance comparison.

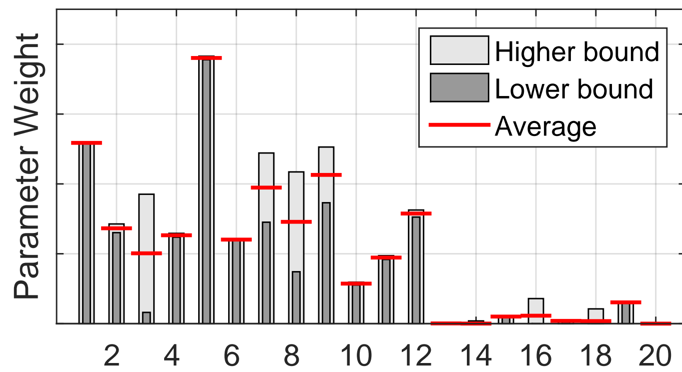
SRKM trained with 200 samples outperforms the SVM and the RVM trained with 800 samples.

With the embedded feature weighing mechanism, the SRKM is able to produce probabilistic circuit parameter weighting during the training process. The weights are represented by normal distributions and the obtained mean and variance of each weights can be used to calculate the confidence interval of each parameter weight shown in Fig. 5.11. The average training time to collect such results is 2.78 seconds.

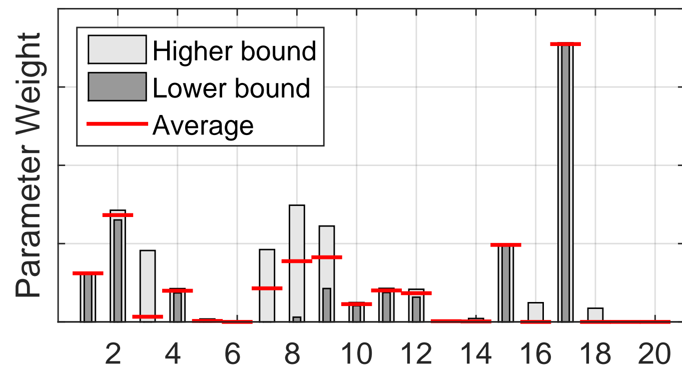
We use design knowledge to provide insights and validation for the parameter weighting of the 20 channel length variations computed by the SRKM. For example, based on the analysis in [76], the multi-loop *flipped voltage follower* (FVF) based LDO is designed as such that a majority portion of its quiescent current is consumed by the fastest two loops in the output stage and hence the variation on $M2$ has direct impact on the majority portion of the quiescent current. Due to the high gain of



(a)



(b)



(c)

Figure 5.11: Weights of transistor's channel length variation in the model of: (a) quiescent current; (b) undershoot; (c) load regulation. Red lines represent the means of the probabilistic weights while the higher and lower bounds represent the 95% confidence bound estimated by SRKM.

the error amplifier, variations on transistors like $M3$, $M7$, $M8$ and $M9$ may lead to mismatches and considerable changes at the two output nodes of the amplifier, one of which is V_g of $M2$. For the quiescent current of the error amplifier, it is also partially defined by the tail current sources ($M16$ and $M18$). This complies with the ranking illustrated in Fig. 5.11a.

Moreover, according to [76], the undershoot of the LDO is mainly determined by the load capacitor and the loop bandwidth, which is further determined by the error amplifier (involving $M3 \sim M10$), the fast loop in the output stage ($M12$), and the in-band zeros locations defined as:

$$\omega_{LCZ} \approx \sqrt{\frac{g_{m1} g_{m11} g_a}{g_{m2} C_{C2} (C_{C1} + C_{C3})}}, \tag{5.24}$$

where g_{mi} is the transconductance of the $i - th$ transistor and g_a is the output admittance of the error amplifier defined by the g_m of $M7 \sim M10$. The ranking of the SRKM in Fig. 5.11b is reliable since it captures all these relevant variations.

Load regulation of the LDO is mainly determined by the DC loop gain, which is given in [76] as the product of the gains of all stages in the loop. The gain of the EA stage is inversely proportional to the g_m of $M7 \sim M10$ and the second stage is comprised of $M17$ and $M11$. Again, the ranking of the SRKM as shown in Fig. 5.11c successfully identifies all these important variations.

To illustrate the capability of handling high dimensional applications, we additionally assign random variations in the range of $\pm 3\sigma$ on the oxide thickness and threshold voltage to each transistor, making a total of 60 features in the regression model. We use 200 SPICE simulations to train the SRKMs for the three specifications. Fig. 5.12 illustrates the average of the normalized weights for various process variations in 20 repetitions of the SRKM training, with an average training time of

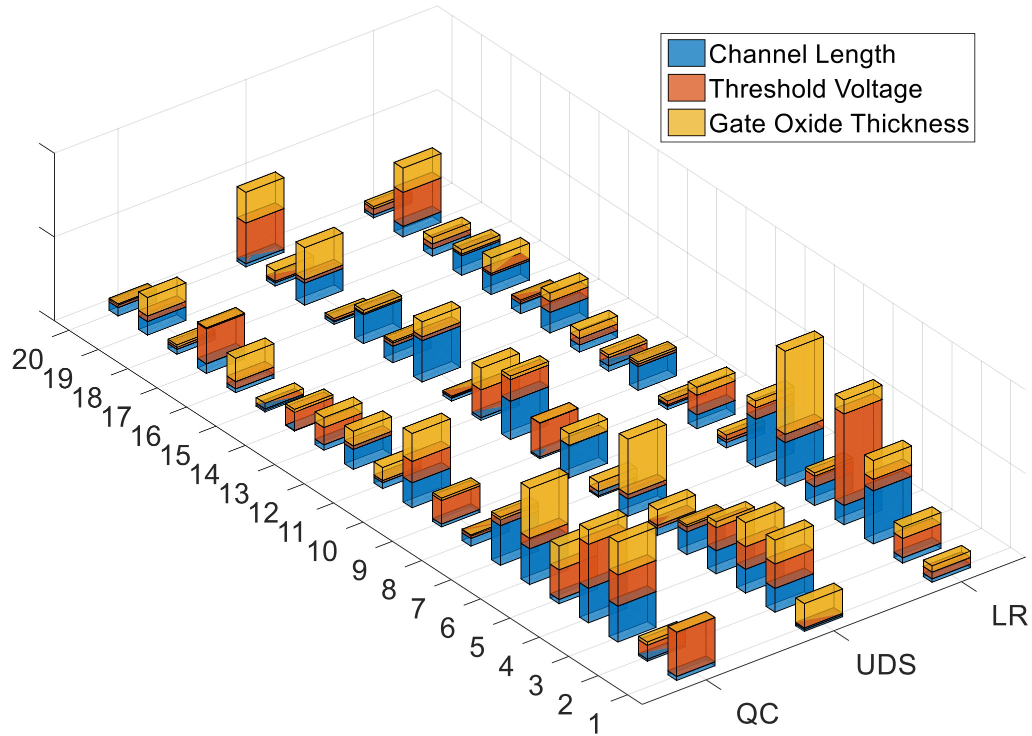


Figure 5.12: Weights for process variations.

4.36 seconds for each repetition. Since transistors $M3 \sim M6$ act as an error amplifier in Fig. 5.9, the results clearly show that the specifications of the LDO are more sensitive or vulnerable to process variations in the error amplifier.

Note that the parametric relevancy and redundancy revealed by the feature weighting of the SRKM is purely inferred from the given training data. In the experiments whose results are reported in Fig. 5.11 and Fig. 5.12, it is assumed that all process variations are independent, thus they are sampled uniformly and independently. As a result, the SRKM only captures the relevancy between the parameters and the target performance based on the evenly collected training data.

Such relevancy or redundancy can be considered as purely circuit performance model oriented parametric analysis without considering any correlations among the

parameters. However, in reality, the observed process variations usually shows a variety of correlations. For example, parameters may be more or less correlated based on their relative spatial locations. Such correlations are referred to as spatial correlations [89] and sometimes are leveraged in layout techniques to minimize mismatches. Taking matching properties [100]into consideration, in the experiment that only involves 20 channel length variations, strong correlation of the variations in each of the four transistor pairs ($M3 \sim M10$) of the error amplifier is assumed. To reflect this, the difference between the two variations in each pair is restricted by an interval of $[-5\%, +5\%]$ in a new uniform sampling process.

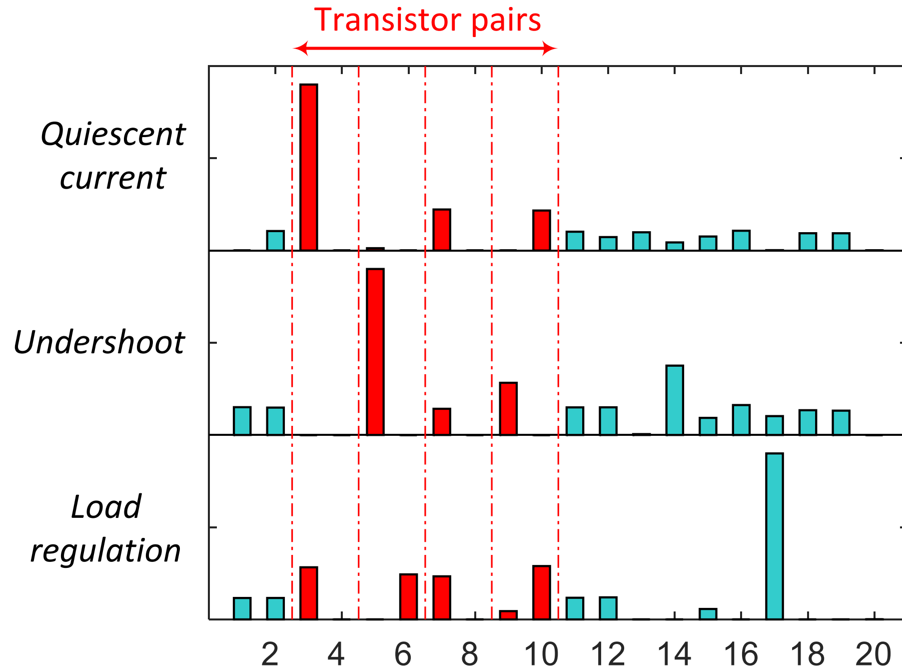


Figure 5.13: Weights for transistors' channel length variations with strong correlations and uniform sampling.

The SRKM trained with the newly sampled training data is shown in Fig. 5.13.

The parameter weighting is quite different compared to the results in Fig. 5.11 since there is one parameter significantly outweights the other in each transistor pair (denoted by red bars in Fig. 5.13). This complies with the strong correlations utilized in the sampling process, indicating that the SRKM can also capture relevancy and redundancy due to correlations reflected among parameters in the training data.

In reality, however, process data reveals more complex correlations than the strong spatial correlation utilized in Fig. 5.13. For example, histograms of process variations reported in [17, 2, 31] suggest that characterizing process variations as Gaussian distributions is more realistic compared to uniform distributions. And for the spatial correlation between any two process variations l_1 and l_2, it is defined in [140] as:

$$\rho_v = \frac{cov(l_1, l_2)}{\sigma l_1 \cdot \sigma l_2}$$

where $cov(l_1, l_2)$ denotes the covariance between l_1 and l_2 and $\sigma l_1 \cdot \sigma l_2$ denotes the product of the standard deviations of l_1 and l_2. Based on [140], ρ_v is 0.8 for minimal spatial distance between l_1 and l_2.

To reflect more realistic matching effects, we utilize the corresponding covariance of $\rho_v = 0.8$ instead of the aforementioned strong correlations and collect new samples with *Latin hypercube sampling* (LHS) from Gaussian distributions [117]. The new parameter weighting produced by the SRKM is shown in Fig. 5.14. The parameter weighting for the quiescent current and the load regulation is more sparse, figuring out only one significant parameter for each case. Since the training data sampled from Gaussian distributions tends to accumulate around mean values rather than uniformly spreading out, it places various implicit emphasis across the sample space and produces results reflecting such emphasis. Therefore, the results indicate that the quiescent current and the load regulation are actually significantly relevant to

only one parameter if the distribution of the process data is taken into consideration.

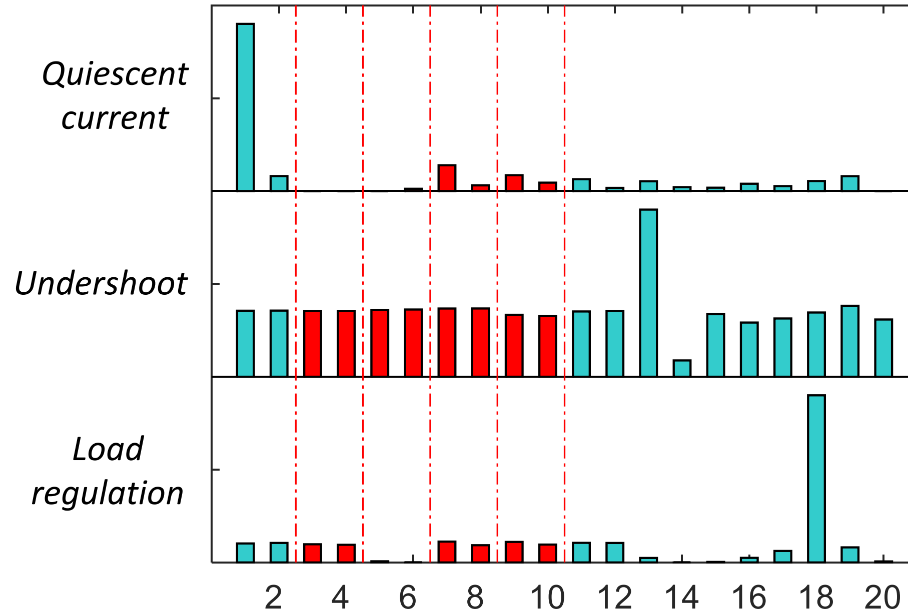


Figure 5.14: Weights for transistors' channel length variations with more realistic spatial correlations.

An interesting observation is that the two parameters in each transistor pairs are assigned notable and similar weights in the undershoot model. Compared to the results in Fig. 5.13 with strong correlations, this indicates that undershoot is actually more sensitive to the mismatch (determined by uncorrelated randomness of the two parameters) rather than the common sizing variation (the correlated portion) in each transistor pair in the error amplifier. Therefore, the weighting results produced by the SRKM of the correlated parameters may reveal the actual relevant portions in those parameters.

Moreover, correlations across various types of process variations also exist. In the experiment that involves 60 process variations, the channel length variation and the

gate oxide thickness variation of each transistor can be considered as independent parameters since they are two fundamental physical parameters. Consequently, only spatial correlation is utilized for these two types of parameters. For the threshold voltage variation, it is an electrical parameter that correlate with the other two parameters [74, 138]. In addition, it may correlate with other variations such as *line-edge-roughness* (LER) [102, 143] and random dopant fluctuation [143, 138].

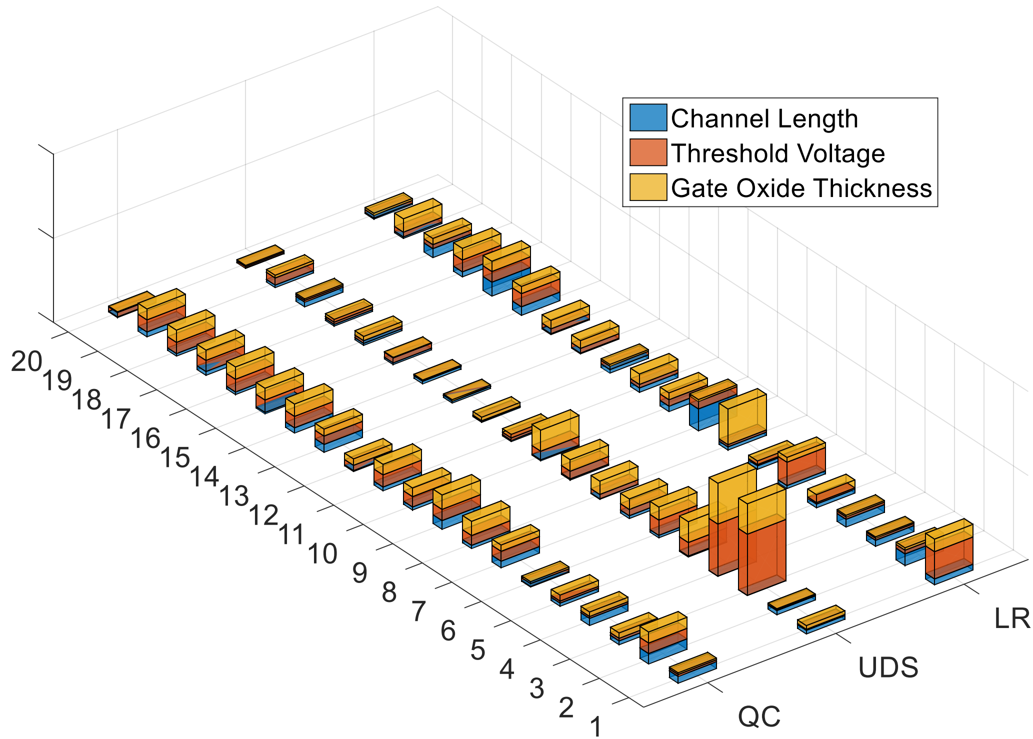


Figure 5.15: Weights for all the variations with correlations across different types of variations.

To reflect the more complex but more realistic correlations, for each type of parameters, we define the correlation $\rho_v = 0.8$ for each pair of transistors and the inter-chip global variation correlation $\rho_v = 0.2$ [140] for other transistors. And for

each transistor, its threshold voltage variation is correlated with both its gate oxide thickness and its channel length variation with $\rho_v = 0.1$. New samples are collected by LHS from Gaussian distributions again and the SRKM trained with the new samples produces feature weighting results shown in Fig. 5.15. The weighting results are more sparse compared to those reported in Fig. 5.12, which makes sensitive parameters more definite under the context of the pre-defined process data.

5.4.2 PLL BIST Scheme Optimization

Built-in-self-test (BIST) is very effective in detecting operational failures of deployed analog/mixed-signal circuits. Base on the concept of alternative test, efficient BIST solutions can be formed by collecting low-cost test signatures and relating the signatures to targeted performance specifications using statistical prediction models. The effectiveness of BIST heavily depends on the quality of the selected signatures and the tradeoffs between accuracy, overhead, and test time. We apply the SRKM to the BIST of a charge-pump PLL targeting three key specifications: *lock-time*(LT), *frequency overshoot*(OVS), and *jitter*(JT).

Fig. 5.16 shows the PLL along with three BIST schemes using various test signatures. Jitter, frequency overshoot and lock-time are important specifications but cannot be easily measured directly on the chip. To capture failures in those specifications, the first candidate BIST scheme [148] collects the readouts of the counter in the divider as its test signature, while the second scheme [62] collects the accumulated *up* and *dn* phase detector outputs via integrators and *time-to-digital converters* (TDCs). The third scheme is an example of IDDQ testing, measuring the quiescent currents of the *charge pump* (CP) and the *voltage control oscillator* (VCO) as test signatures similar to the approach of [90].

The first two schemes operate in a special test mode which instead of feeding

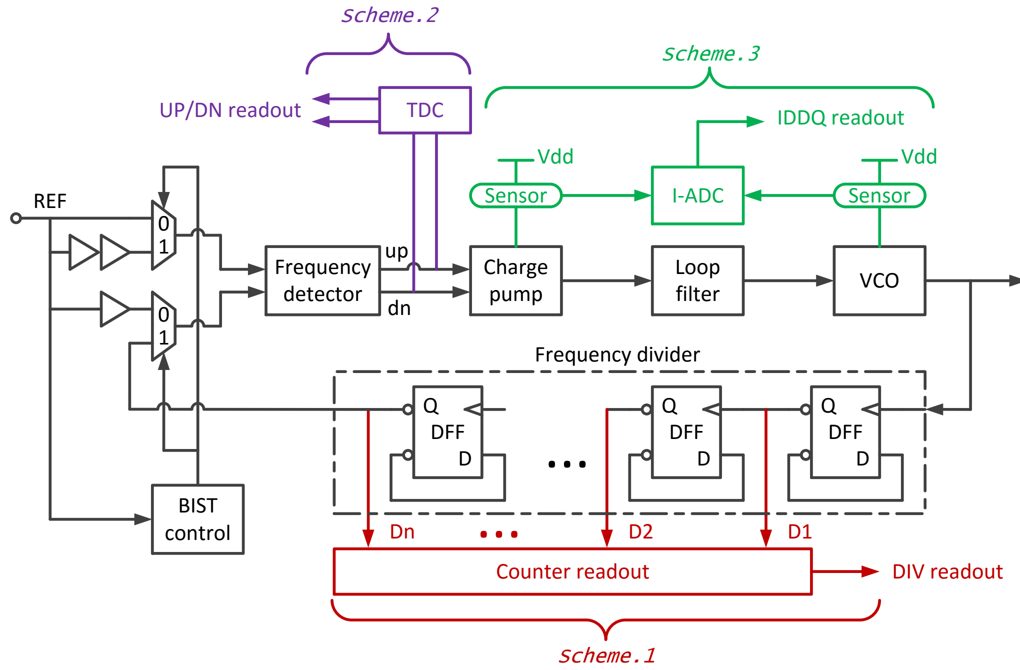


Figure 5.16: A PLL and three BIST schemes.

back the divider output, it first feeds the one-buffer delayed reference clock to the phase detector for 8 reference cycles with a cycle time of 0.1 us. Then, the reference clock input is replaced by the double delayed reference clock for another 8 cycles. Each cycle generates one signature for Scheme 1 and two for Scheme 2, making a total of 16 and 32 signatures for Scheme 1 and 2, respectively. Scheme 3 reads out two signatures, i.e. the CP and VCO quiescent currents, in the quiescent mode.

Recently, learning-based classifiers like the SVM have been trained to perform the failure detection in BIST [148, 22]. To make better usage of the collected test signatures, we apply the proposed SRKM in each scheme. We fit the target specification into a sigmoid function before we employ the SRKM as a classier for failure detection. Three classification techniques, the SVM, RVM, and SRKM, are trained with 200 simulation samples and tested with 4,000 samples. The classifying errors

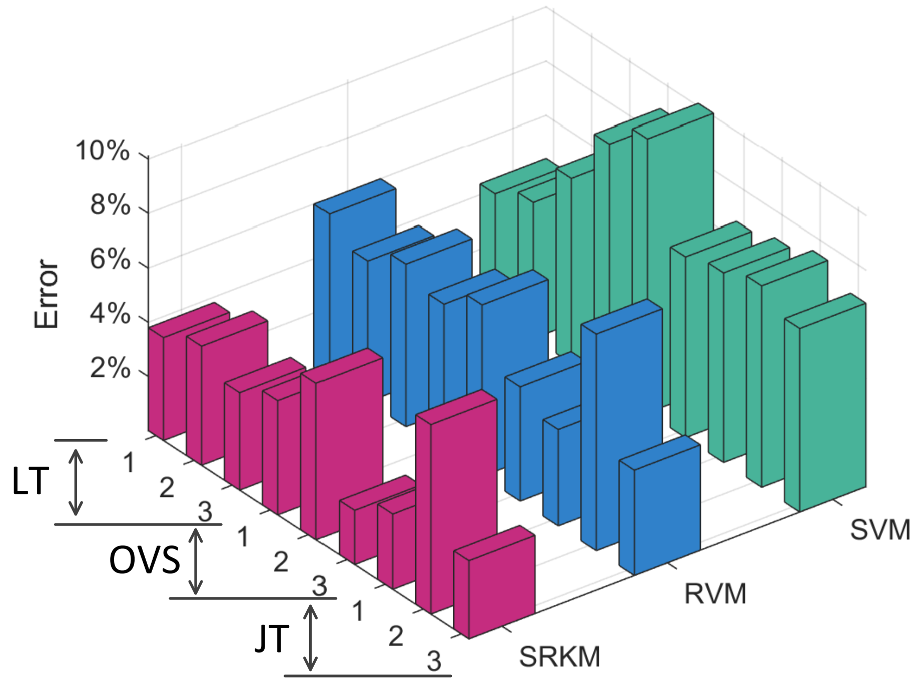


Figure 5.17: Classifier performance comparison.

are compared in Fig. 5.17 which shows the superior BIST classifier accuracy of the proposed SRKM.

In addition, the SRKM also produces reliable ranking among test signatures, which can be further leveraged to improve the efficiency of BIST schemes. For example, the SRKM ranks the 16 test signatures in Scheme 1 as shown in Fig. 5.18 when building the classifier for jitter failure detection. The tenth signature is the last one with a significant weight. After that, the remaining 6 signatures are of little importance and can be considered as redundant. Using the same procedure, we reduce the test time for each of the three specifications for Scheme 1 as reported in Table 5.1.

Assuming that realizing all three schemes on-chip does not lead to significant overhead, we seek to improve BIST accuracy by leveraging the signatures of all the

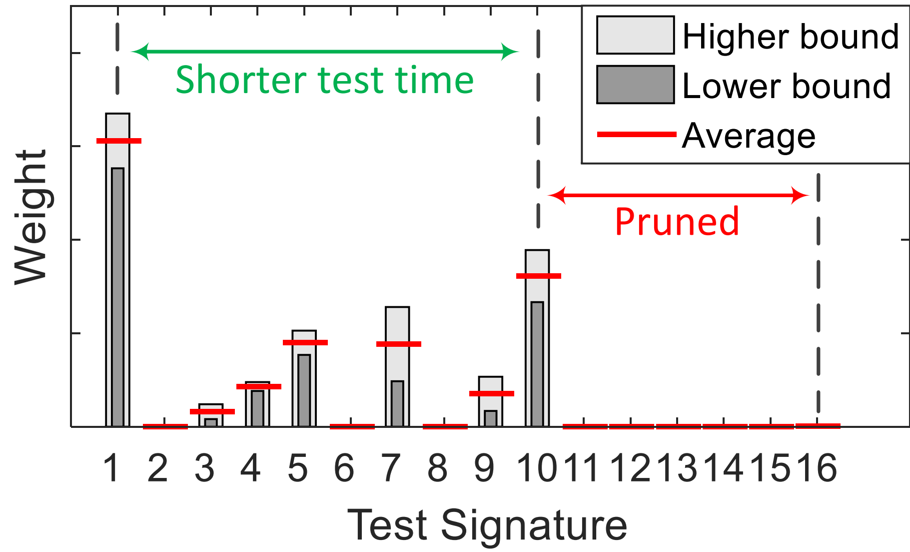


Figure 5.18: Signature ranking for jitter prediction with Scheme 1.

Table 5.1: Test time optimization of Scheme 1

Spec.	Original Accuracy	Selected Readouts	Resulting Accuracy	Test Time Reduction
JT	97.22%	1 - 10	96.20%	37.5%
OVS	95.78%	1 - 12	94.89%	25.0%
LT	96.20%	1 - 6	97.00%	62.5%

schemes. While combing all the signatures can offer the best accuracy, it may not be completely efficient due to the existence of redundant signatures. For this, we train an SRKM on all the signatures across the three schemes to predict the jitter. The average training time to produce signature ranking is 5.18 seconds. Based on the signature ranking shown in Fig. 5.19, we collect the first three signatures in Scheme 1 and the first signature in Scheme 2. Although the third last signature in Scheme 2 also possesses a notable weight, collecting such signature is not cost-effective in terms of test time, and thus it is discarded. For Scheme 3, only the quiescent current of VCO is selected, which can be measured in $0.6us$ according to [90].

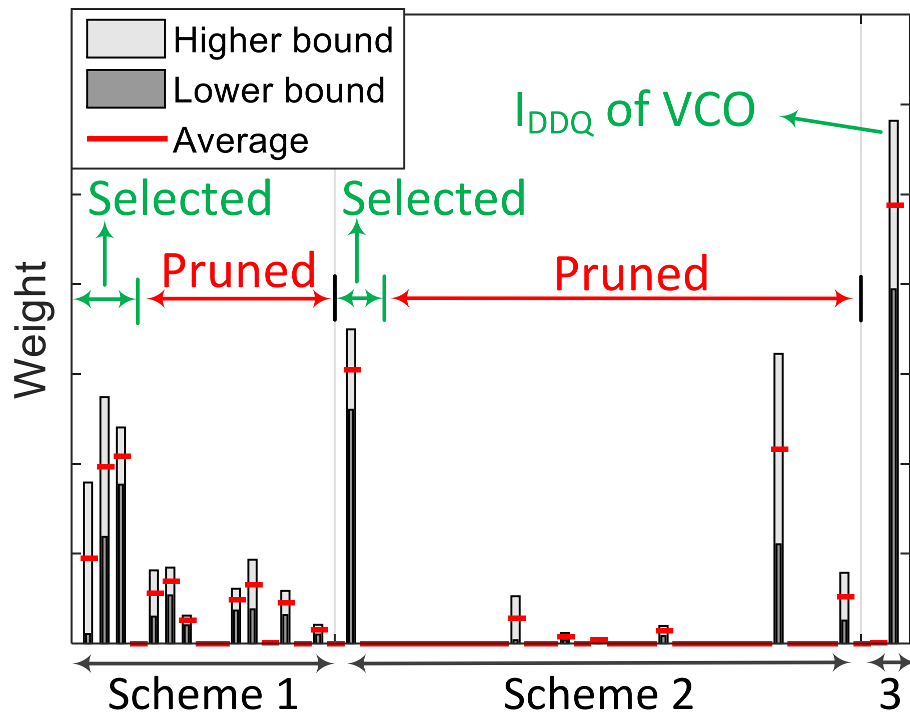


Figure 5.19: Signature ranking for jitter prediction with all three schemes.

Based on these five selected signatures, we synthesize an optimized combined

Table 5.2: BIST scheme synthesis

Spec.	Original Best Accuracy	Synthesized Signatures			Test Time (us)			Test Time Reduction	Optimized Accuracy
		Sch.1	Sch.2	Sch.3	Sch.1&2	Sch.3	Total		
JT	97.22%	1 - 3	1	VCO	0.3	0.6	0.9	43.75%	99.98%
OVS	98.00%	1 - 3	1 - 7	CP	0.4	0.6	1.0	37.50%	99.88%
LT	96.40%	1 - 4	1 - 2	VCO	0.4	0.6	1.0	37.50%	99.95%

BIST scheme for each specification and show the results in Table 5.2. As can be been, by using the proposed SRKM, the BIST accuracy can be boosted to over 99.88% with a test time reduction of about 40%.

5.5 Summary

This paper proposes a novel sparse Bayesian learning framework named sparse relevance kernel machine to capture circuit characteristics and analyze circuit performance dependencies on assorted parameters or signatures via a statistical regression model. The advantages of the proposed framework are demonstrated by variability analysis of an LDO and BIST optimization for a charge-pump PLL.

6. CONCLUSION AND FUTURE WORK

This section concludes with a review of the algorithms for AMS verification proposed in this dissertation, and discusses potential revenues towards the complete road map of the hierarchical and hybrid AMS verification framework utilizing assorted verification algorithms.

6.1 Conclusion

The tremendous and perpetual growth in both importance and complexity of the AMS systems poses great challenges to AMS verification. Existing formal methods suffer from their limitation in efficiency and scalability while traditional simulation based methods rarely provide acceptable completeness or coverage. Moreover, the escalating need of shorter time to market for AMS chips requires close collaboration among design, verification and test, resulting in emerging challenges for design and test analysis in the verification phase.

To enhance the efficiency and scalability of SMT based reachability analysis, the dissertation proposes in Section 3 a new paradigm for AMS system abstraction, which is especially effective for DIA systems, to unify AMS content into a pure analog representation. Based on the paradigm, a system decomposition with fine granularity is adopted to divide the SMT problem into subproblems with lower complexity and fewer constraints to be solved in parallel. Larger AMS circuits can be formally verified more efficiently due to the lower complexity resulted by system decomposition and the parallelization.

For complex AMS systems beyond the capability of formal methods, the dissertation proposes two simulation based methods with different emphasis. The one described in Section 4 converts verification problems to binary classifications and

uses SVMs as the classifiers. Active learning strategy based on probabilistic version space reduction is employed under this context to reduce the required samples. An asymmetric expansion of the active learning strategy is built to conform the conservative nature of verification. The algorithm is efficient and reusable classifiers can be produced for quick validation of re-designs or test plans.

Another simulation-based method proposed in Section 5 builds accurate characterization models for AMS systems with statistical regression techniques and implants a feature weighting mechanism for concurrent parametric analysis. Developed in the Bayesian learning framework, the algorithm is able to achieve sparsity in the sample weighting and the feature weighting and produce probabilistic predictions enabling statistical circuit analysis including verification with confidence level. Besides, the provided probabilistic feature weighting implicitly quantifies the influence of circuit parameters on a certain specification, offering favorable assistance to design analysis and test optimization.

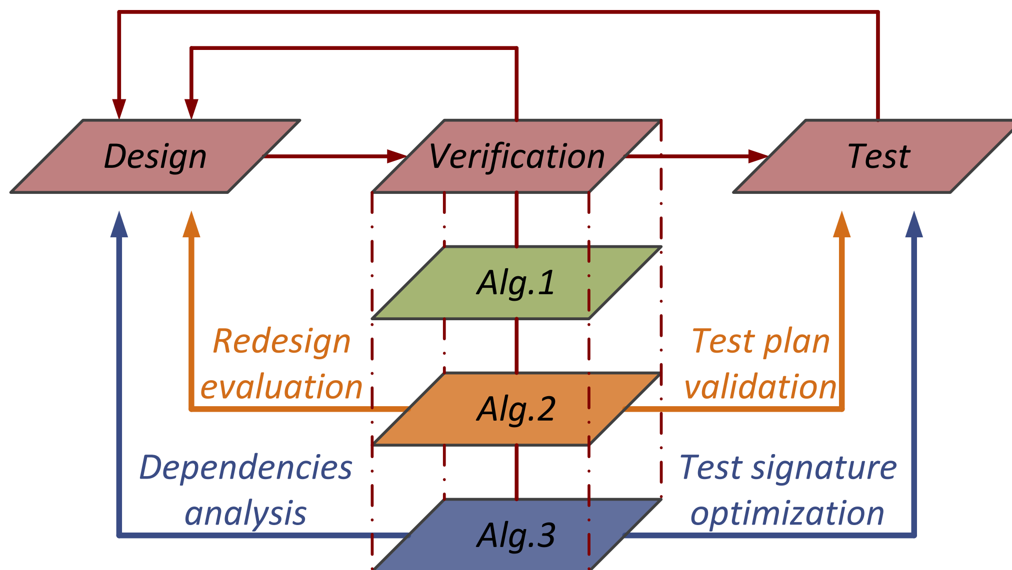


Figure 6.1: Roles of the proposed algorithms in the design flow.

Each of the three algorithms can be invoked in the design flow as an AMS verification solution independently, but plays a distinctive role as shown in Fig. 6.1. The first algorithm is a formal method. If the design passes the process of formal verification, since it is rigorously proven to meet the given specifications, there is no need for re-designing and sometimes even for testing [68] either. However, if the design fails the formal verification, it is difficult to gather useful information from the verification process to aid design debugging or testing. Similar with legacy simulation based methods, the second algorithm lacks coverage or completeness assurance as a verification solution. Nevertheless, it's the most efficient method among the three algorithms and its produced model is highly reusable, enabling instantaneous evaluation of re-designs and validation of test plans. The third algorithm developed under the statistical framework possesses the auspicious efficiency and scalability as a data-driven method and approaches the completeness and coverage problem in a probabilistic sense. Its ability of simultaneous parametric analysis during the training process also provides analysis of performance dependencies to aid design analysis and test signature relevancy to aid test scheme optimization.

6.2 Future Work

There are multiple potential directions that lead to further exploration and expansion of the functionality and efficiency of the proposed algorithms. For example, the proposed system decomposition method for reachability analysis can be migrated to other SMT-based applications like model checking or fault detection for higher efficiency. The active learning strategy can be applied to capture test signatures near the separating hyperplane defined by the given performance threshold, which may lead to more efficient design for test. Moreover, the statistical Bayesian learning method with concurrent parameter weighting can be treated as a general purpose

learning model, which is favorable to any applications with noisy data or redundant parameters. Nonetheless, this section focus on the potential development of AMS verification framework by consolidating the proposed algorithms.

6.2.1 Semi-Formal Verification with Statistical Models

The simulation-based statistical method proposed in Section 5 focuses on building accurate characterization models shown in Fig. 6.2 rather than describing the whole verification flow. Typically, a regression model will be trained to estimate the mean and variance of each performance or specification of the AMS system. Leveraging the fact that performing prediction is computationally efficient, once the models are trained, the verification can be addressed in a straightforward manner by oversampling x from the parameter space Φ and computing the performances.

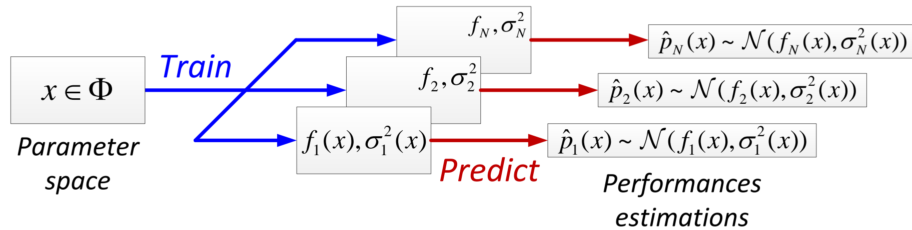


Figure 6.2: Characterization model built by the statistical algorithm.

Considering the completeness and reliability offered by rigorous formal methods, it is hypothesized that implanting the statistical model into a formal framework should produce a very promising semi-formal verification framework. For example, assuming each performance or specification p_i are required to fall into an interval I_i, the statistical model based verification problem can be formulated as the following

satisfiability problem:

$$(x \in \Phi) \wedge (\bigwedge_i [f_i(x) \pm k \cdot \sigma_i(x)] \in I_i) \tag{6.1}$$

where k is a constant parameter that controls the confidence level of the prediction and those $[f_i(x) \pm k \cdot \sigma_i(x)] \in I_i$ terms in (6.1) represent the performance or specification checking of the worst cases.

Analogous to the solution of the satisfiability problems in Section 3, the problem (6.1) can be solved by SMT techniques. Auspiciously, unlike the process in Section 3, iterative SMT constructions and computations are unnecessary while solving the problem (6.1) once should be sufficient to directly verify the system.

By embedding the statistical models into the SMT framework, it is envisaged that this new method will achieve a certain level of formality with the SMT framework while its informality will be well controlled by the confidence level of the statistical models. For example, by using $k = 3$, the problem (6.1) actually checks whether the worst cases of the statistical predictions with 99.7% confidence level meet the performance requirements. This results in a semi-formal method with better efficiency and scalability as well as reliability.

6.2.2 Road Map Towards Hierarchical and Hybrid AMS Verification

The roles of the three proposed algorithms have been illustrated in Fig. 6.1 and the preceding section. Compared to improving each algorithm independently, the avenue to assemble and consolidate all the algorithms into the hierarchical and hybrid AMS verification framework discussed in Section 1.2 is of greater significance, since it is a systematized and multi-level oriented blueprint approaching elimination of the design and verification gap.

Considering a two-level hierarchy that involves a top level and a block level for

AMS systems, as shown in Fig. 6.3, the three proposed algorithms can all be invoked into the block level. Due to their diverse functionality, a hybrid flow is more favorable. For example, for complex blocks beyond the capability of formal methods, the statistical method (Algorithm 3) can be utilized to perform verification with a certain level of confidence. If the block passes the verification, Algorithm 3 provides a statistical model to the verification at the top level. Otherwise, Algorithm 2 can be applied before the procedure goes back to the design phase and the produced model along with the parametric analysis from Algorithm 3 can help to fix the design and perform quick preliminary evaluation of any design modifications. In addition, for blocks with considerations of process variations, since process data is usually statistical in nature, applying Algorithm 3 to those blocks and injecting process data into the model can assist the variability and yield analysis.

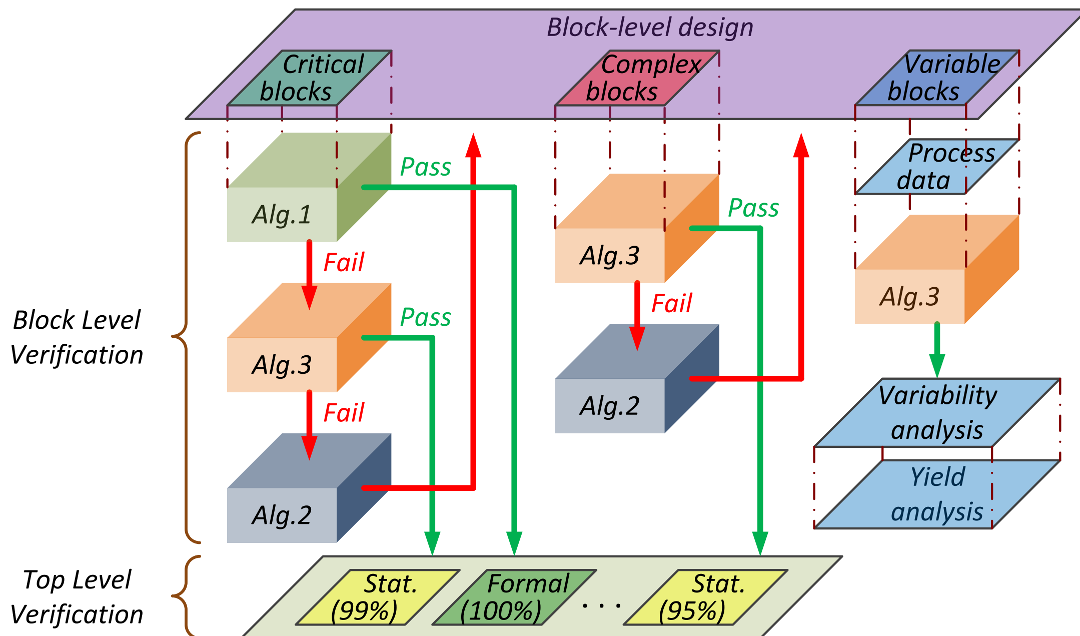


Figure 6.3: Road map towards hierarchical and hybrid AMS verification.

On the other hand, for critical blocks with moderate complexity, Algorithm 1 can be utilized for formal verification. An ideal case is that the block under verification passes the formal verification, providing a formal model with 100% completeness to the top level verification. However, if the block fails, it does not necessarily mean that the design violates the given specifications since formal methods tend to follow stringent criteria which may reject marginal designs that are actually acceptable. Therefore, to complement the flow, if a block fails in Algorithm 1, Algorithm 3 can be applied to the block for further investigation. If the block passes with an acceptable confidence level, Algorithm 3 produces a statistical model with high confidence to the top level. Otherwise the same flow for complex blocks can be repeated again by employing Algorithm 2 and the generated parametric analysis by Algorithm 3 to aid the revision of the design.

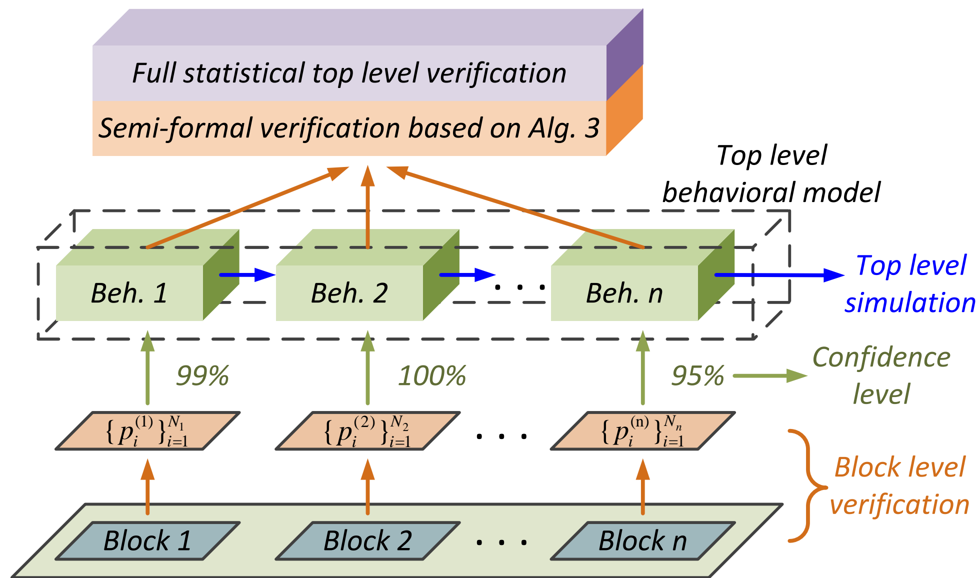


Figure 6.4: Envisioned top level full statistical verification developed from block level models.

The last piece of the road map shown in Fig. 6.3 is the construction of the top level verification that involves a series of formal models and statistical models with various confidence levels. In this light, statistical models can be further applied to the top level to chain together all the block level models and perform top level probabilistic verification. As illustrated in Fig. 6.4, after the specifications $\{p_i\}_{i=1}^{N}$ of each block is verified in the block level verification, based on the specifications, behavioral models for all the blocks can be extracted with a certain confidence level for the purpose of top level simulations. Then, the top level verification can resort to the potential semi-formal verification framework based on Algorithm 3 described in Section 6.2.1, leading to full statistical top level verification. This finalizes the entire road map shown in Fig. 6.3 and embraces the functions and advantages of assorted algorithms. Such framework is envisioned to be applicable to AMS systems with high complexity in a viable and flexible way.

REFERENCES

[1] Parosh Aziz Abdulla, Per Bjesse, and Niklas Eén. Symbolic reachability analysis based on SAT-solvers. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 411–425. Springer, 2000.

[2] Kanak Agarwal and Sani Nassif. Characterizing process variation in nanometer cmos. In *Proceedings of the 44th Annual Design Automation Conference*, pages 396–399. ACM, 2007.

[3] Ghiath Al Sammane, Mohamed H Zaki, Zhi Jie Dong, and Sofiene Tahar. Towards assertion based verification of analog and mixed signal designs using PSL. In *Proceedings of Forum on Specification & Design Language*, pages 293–298, 2007.

[4] Ghiath Al-Sammane, Mohamed H Zaki, and Sofiène Tahar. A symbolic methodology for the verification of analog and mixed signal designs. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 249–254. EDA Consortium, 2007.

[5] Samantha Alt, Malgorzata Marek-Sadowska, and Li C Wang. Circuit partitioning for behavioral full chip simulation modeling of analog and mixed signal circuits. *Int. J. Modeling and Optimization*, 4(1):74–80, 2014.

[6] M. Althoff, A. Rajhans, B.H. Krogh, S. Yaldiz, X. Li, and L. Pileggi. Formal verification of phase-locked loops using reachability analysis and continuization. In *Proceedings of the International Conference on Computer-Aided Design*, pages 659–666. IEEE Press, 2010.

[7] Matthias Althoff, Akshay Rajhans, Bruce H Krogh, Soner Yaldiz, Xin Li, and Larry Pileggi. Formal verification of phase-locked loops using reachability analysis and continuization. *Communications of the ACM*, 56(10):97–104, 2013.

[8] Matthias Althoff, Olaf Stursberg, and Martin Buss. Reachability analysis of linear systems with uncertain parameters and inputs. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 726–732. IEEE, 2007.

[9] F.R. Bach, D. Heckerman, and E. Horvitz. Considering cost asymmetry in learning classifiers. *Journal of Machine Learning Research*, 7(2):1713, 2007.

[10] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT press Cambridge, 2008.

[11] Ashok Balivada, Yatin Hoskote, and Jacob A Abraham. Verification of transient response of linear analog circuits. In *Proceedings of the 13th IEEE VLSI Test Symposium*, pages 42–47. IEEE, 1995.

[12] Erich Barke, Darius Grabowski, Helmut Graeb, Lars Hedrich, Stefan Heinen, Ralf Popp, Sebastian Steinhorst, and Yifan Wang. Formal approaches to analog circuit verification. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 724–729. European Design and Automation Association, 2009.

[13] Clark Barrett, Christopher L Conway, Morgan Deters, Liana Hadarean, Dejan Jovanović, Tim King, Andrew Reynolds, and Cesare Tinelli. CVC4. In *Computer Aided Verification*, pages 171–177. Springer, 2011.

[14] Clark W Barrett, David L Dill, and Aaron Stump. Checking satisfiability of first-order formulas by incremental translation to SAT. In *Computer Aided Verification*, pages 236–249. Springer, 2002.

[15] Pouria Bastani, Nicholas Callegari, Li-C Wang, and Magdy S Abadir. Statistical diagnosis of unmodeled systematic timing effects. In *Proceedings of the 45th annual Design Automation Conference*, pages 355–360. ACM, 2008.

[16] Mordechai Ben-Ari. *Mathematical Logic for Computer Science*. Springer Science & Business Media, 2012.

[17] Kerry Bernstein, David J Frank, Anne E Gattiker, Wilfried Haensch, Brian L Ji, Sani R Nassif, Edward J Nowak, Dale J Pearson, and Norman J Rohrer. High-performance CMOS variability in the 65-nm regime and beyond. *IBM Journal of Research and Development*, 50(4.5):433–449, 2006.

[18] D.P. Bertsekas, A. Nedi, and A.E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, 2003.

[19] Alina Beygelzimer, John Langford, Zhang Tong, and Daniel J Hsu. Agnostic active learning without constraints. In *Advances in Neural Information Processing Systems*, pages 199–207, 2010.

[20] Christopher M Bishop. *Neural Networks for Pattern Recognition*. Oxford university press, 1995.

[21] Avrim L Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1):245–271, 1997.

[22] Ahcene Bounceur, Belkacem Brahmi, Kamel Beznia, and Reinhardt Euler. Accurate analog/RF BIST evaluation based on SVM classification of the process parameters. In *Proceedings of the 9th International Design & Test Symposium*, pages 55–60. IEEE, 2014.

[23] Evgeny Byvatov and G Schneider. Support vector machine applications in bioinformatics. *Applied Bioinformatics*, 2(2):67–77, 2002.

[24] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[25] Sheng Chen, Haibo He, and Edwardo A Garcia. Ramoboost: Ranked minority oversampling in boosting. *IEEE Transactions on Neural Networks*, 21(10):1624–1642, 2010.

[26] Haibin Cheng, Haifeng Chen, Guofei Jiang, and Kenji Yoshihira. Nonlinear feature selection by relevance feature vector machine. *Machine Learning and Data Mining in Pattern Recognition*, pages 144–159, 2007.

[27] Vladimir Cherkassky and Yunqian Ma. Practical selection of SVM parameters and noise estimation for SVM regression. *Neural Networks*, 17(1):113–126, 2004.

[28] Edmund M Clarke, Orna Grumberg, and Doron Peled. *Model Checking*. MIT press, 1999.

[29] Barry R Cobb and Prakash P Shenoy. Nonlinear deterministic relationships in Bayesian networks. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 27–38. Springer, 2005.

[30] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[31] Nattapol Damrongplasit, Luis Zamudio, Tsu-Jae King Liu, and Sriram Balasubramanian. Threshold voltage and DIBL variability modeling based on forward and reverse measurements for SRAM and analog MOSFETs. *IEEE Transactions on Electron Devices*, 62(4):1119–1126, 2015.

[32] Tathagato Rai Dastidar and PP Chakrabarti. A verification system for transient response of analog circuits. *ACM Transactions on Design Automation of*

Electronic Systems, 12(3):31, 2007.

[33] Fernando De Bernardinis, Michael I Jordan, and A SangiovanniVincentelli. Support vector machines for analog circuit performance representation. In *Proceedings of the 2003 Design Automation Conference*, pages 964–969. IEEE, 2003.

[34] K. De Brabanter, J. De Brabanter, J.A.K. Suykens, and B. De Moor. Approximate confidence and prediction intervals for least squares support vector regression. *IEEE Transactions on Neural Networks*, 22(1):110–120, 2011.

[35] Dimitri De Jonghe, Dirk Deschrijver, Tom Dhaene, and Georges Gielen. Extracting analytical nonlinear models from analog circuits by recursive vector fitting of transfer function trajectories. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1448–1453. EDA Consortium, 2013.

[36] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[37] Sabyasachi Deyati, Barry John Muldrey, Adrish Banerjee, and Avhishek Chatterjee. Atomic model learning: A machine learning paradigm for post silicon debug of RF/analog circuits. In *Proceedings of the 32nd VLSI Test Symposium*, pages 1–6. IEEE, 2014.

[38] Tom Dhaene and Dirk Deschrijver. Stable parametric macromodeling using a recursive implementation of the vector fitting algorithm. *IEEE Microwave and Wireless Components Letters*, 19(2):59, 2009.

[39] Mengmeng Ding and R.I. Vemur. An active learning scheme using support vector machines for analog circuit feasibility classification. In *Proceedings of the 18th International Conference on VLSI Design*, pages 528–534. IEEE, 2005.

[40] Changdao Dong and Xin Li. Efficient SRAM failure rate prediction via gibbs sampling. In *Proceedings of the 48th Design Automation Conference*, pages 200–205. ACM, 2011.

[41] D Drmanac, Brendon Bolin, Li C Wang, and Magdy S Abadir. Minimizing outlier delay test cost in the presence of systematic variability. In *Proceedings of the 2009 International Test Conference*, pages 1–10. IEEE, 2009.

[42] Bruno Dutertre and Leonardo De Moura. A fast linear-arithmetic solver for DPLL (T). In *Computer Aided Verification*, pages 81–94. Springer, 2006.

[43] Seyda Ertekin, Jian Huang, Leon Bottou, and Lee Giles. Learning on the border: active learning in imbalanced data classification. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management*, pages 127–136. ACM, 2007.

[44] Anita C. Faul and Michael E. Tipping. Analysis of sparse Bayesian learning. *Advances in Neural Information Processing Systems*, 14:383–389, 2002.

[45] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, 15(1):3133–3181, 2014.

[46] Dan FitzPatrick and Ira Miller. *Analog behavioral modeling with the Verilog–A language*. Springer Science & Business Media, 2007.

[47] Harry D Foster. Trends in functional verification: A 2014 industry study. In *Proceedings of the 52nd Design Automation Conference*, pages 1–6. IEEE,

2015.

[48] Martin Franzle, Christian Herde, Tino Teige, Stefan Ratschan, and Tobias Schubert. Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. *Journal on Satisfiability, Boolean Modeling and Computation*, 1:209–236, 2007.

[49] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1, 2010.

[50] Abhijit Ghosh and Ranga Vemuri. Formal verification of synthesized analog designs. In *Proceedings of the 1999 International Conference on Computer Design*, pages 40–45. IEEE, 1999.

[51] P.K. Ghosh and K.V. Kumar. Support function representation of convex bodies, its application in geometric computing, and some related representations. *Computer Vision and Image Understanding*, 72(3):379–403, 1998.

[52] Antoine Girard and Colas Le Guernic. Efficient reachability analysis for linear systems using support functions. In *Proceedings of the 17th World Congress of the International Federation of Automatic Control*, pages 8966–8971, 2008.

[53] Mark R Greenstreet and Ian Mitchell. Reachability analysis using polygonal projections. In *Hybrid Systems: Computation and Control*, pages 103–116. Springer, 1999.

[54] Christoph Grimm and Carna Radojicic. Verification and validation of AMS systems: Towards coverage of uncertainties. In *Proceedings of the 20th International Mixed-Signal Testing Workshop*, pages 1–6. IEEE, 2015.

[55] Chenjie Gu and Jaijeet Roychowdhury. FSM model abstraction for analog/mixed-signal circuits by learning from I/O trajectories. In *Proceedings of the 16th Asia and South Pacific Design Automation Conference*, pages 7–12. IEEE Press, 2011.

[56] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti A Zadeh. *Feature Extraction: Foundations and Applications*. Springer, 2008.

[57] Walter Hartong, Lars Hedrich, and Erich Barke. On discrete modeling and model checking for nonlinear analog systems. In *Computer Aided Verification*, pages 401–414. Springer, 2002.

[58] John Havlicek and Scott Little. Realtime regular expressions for analog and mixed-signal assertions. In *Proceedings of the International Conference on Formal Methods in Computer-Aided Design*, pages 155–162. FMCAD Inc, 2011.

[59] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *Proceedings of the 2008 IEEE International Joint Conference on Neural Networks*, pages 1322–1328. IEEE, 2008.

[60] Lars Hedrich and Erich Barke. A formal approach to nonlinear analog circuit verification. In *Proceedings of the 1995 International Conference on Computer-Aided Design*, pages 123–127. IEEE, 1995.

[61] Sepp Hochreiter and Klaus Obermayer. Nonlinear feature selection with the potential support vector machine. In *Feature Extraction*, pages 419–438. Springer, 2006.

[62] Sen-Wen Hsiao, Nicholas Tzou, and Avhishek Chatterjee. A programmable BIST design for PLL static phase offset estimation and clock duty cycle de-

tection. In *Proceedings of the 31st VLSI Test Symposium*, pages 1–6. IEEE, 2013.

[63] Sheng-Jun Huang, Rong Jin, and Zhi-Hua Zhou. Active learning by querying informative and representative examples. In *Advances in Neural Information Processing Systems*, pages 892–900, 2010.

[64] Xiaolin Huang, Lei Shi, and Johan AK Suykens. Ramp loss linear programming support vector machine. *The Journal of Machine Learning Research*, 15(1):2185–2211, 2014.

[65] IEEE-Commission. *IEEE standard for property specification language (PSL).* IEEE Std 1850-2005, 2005.

[66] Alex Pappachen James and Belur V Dasarathy. Medical image fusion: A survey of the state of the art. *Information Fusion*, 19:4–19, 2014.

[67] Thorsten Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods*, pages 169–184. MIT Press, 1999.

[68] Roope Kaivola, Rajnish Ghughal, Naren Narasimhan, Amber Telfer, Jesse Whittemore, et al. Replacing testing with formal verification in Intel CoreTM i7 processor execution engine validation. In *Computer Aided Verification*, pages 414–429. Springer, 2009.

[69] 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 *Proceedings of the 43rd Design Automation Conference*, pages 69 –72, 2006.

[70] S. Sathiya Keerthi, Olivier Chapelle, and Dennis DeCoste. Building support vector machines with reduced classifier complexity. *The Journal of Machine*

Learning Research, 7:1493–1515, 2006.

[71] Taehwan Kim, Do-Gyoon Song, Sangho Youn, Jaejin Park, Hojin Park, and Jaeha Kim. Verifying start-up failures in coupled ring oscillators in presence of variability using predictive global optimization. In *Proceedings of the 2013 International Conference on Computer-Aided Design*, pages 486–493. IEEE Press, 2013.

[72] Davis E King. Dlib-ml: A machine learning toolkit. *The Journal of Machine Learning Research*, 10:1755–1758, 2009.

[73] Jan Kremer, Kim Steenstrup Pedersen, and Christian Igel. Active learning with support vector machines. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(4):313–326, 2014.

[74] Kelin J Kuhn, Martin D Giles, David Becher, Pramod Kolar, Avner Kornfeld, Roza Kotlyar, Sean T Ma, Atul Maheshwari, and Sivakumar Mudanai. Process technology variation. *IEEE Transactions on Electron Devices*, 58(8):2197–2208, 2011.

[75] Ken Kundert and Olaf Zinke. *The Designer's Guide to Verilog-AMS*. Springer Science & Business Media, 2006.

[76] Suming Lai. *Modeling, Design and Optimization of IC Power Delivery with On-Chip Regulation*. Doctoral dissertation, Texas A&M University, 2014.

[77] Suming Lai and Peng Li. A fully on-chip area-efficient CMOS low-dropout regulator with fast load regulation. *Analog Integrated Circuits and Signal Processing*, 72(2):433–450, 2012.

[78] Colas Le Guernic and Antoine Girard. Reachability analysis of hybrid systems using support functions. In *Computer Aided Verification*, pages 540–554.

Springer, 2009.

[79] D.D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 148–156, 1994.

[80] Fan Li, Yiming Yang, and Eric P Xing. From lasso regression to feature vector machine. In *Advances in Neural Information Processing Systems*, pages 779–786, 2005.

[81] Peng Li, Lawrence T Pileggi, Mehdi Asheghi, and Rajit Chandra. IC thermal simulation and modeling via efficient multigrid-based approaches. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(9):1763–1776, 2006.

[82] Xin Li, Peng Li, Yang Xu, and Lawrence T Pileggi. Analog and RF circuit macromodels for system-level analysis. In *Proceedings of the 40th Design Automation Conference*, pages 478–483. ACM, 2003.

[83] Honghuang Lin and Peng Li. Parallel hierarchical reachability analysis for analog verification. In *Proceedings of the 51st Design Automation Conference*, pages 1–6. IEEE, 2014.

[84] Honghuang Lin and Peng Li. Circuit performance classification with active learning guided sampling for support vector machines. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(9):1467–1480, 2015.

[85] Honghuang Lin, Peng Li, and Chris J Myers. Verification of digitally-intensive analog circuits via kernel ridge regression and hybrid reachability analysis. In *Proceedings of the 50th Design Automation Conference*, page 66. ACM, 2013.

[86] Scott Little, David Walter, Nicholas Seegmiller, Chris Myers, and Tomohiro Yoneda. Verification of analog and mixed-signal circuits using timed hybrid Petri nets. In *Automated Technology for Verification and Analysis*, pages 426–440. Springer, 2004.

[87] Cheng-Lin Liu, Kazuki Nakashima, Hiroshi Sako, and Hiromichi Fujisawa. Handwritten digit recognition: Benchmarking of state-of-the-art techniques. *Pattern Recognition*, 36(10):2271–2285, 2003.

[88] Hongzhou Liu, Amit Singhee, Rob A Rutenbar, and L Richard Carley. Remembrance of circuits past: Macromodeling by data mining in large analog design spaces. In *Proceedings of the 39th Design Automation Conference*, pages 437 – 442, 2002.

[89] Pei-Wen Luo, Jwu-E Chen, Chin-Long Wey, Liang-Chia Cheng, Ji-Jan Chen, and Wen-Ching Wu. Impact of capacitance correlation on yield enhancement of mixed-signal/analog integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(11):2097–2101, 2008.

[90] Samed Maltabas, Osman Kubilay Ekekon, Kemal Kulovic, Anne Meixner, and Martin Margala. An IDDQ BIST approach to characterize phase-locked loop parameters. In *Proceedings of the 31st VLSI Test Symposium*, pages 1–6. IEEE, 2013.

[91] Elie Maricau, Dimitri De Jonghe, and Georges Gielen. Hierarchical analog circuit reliability analysis using multivariate nonlinear regression and active learning sample selection. In *Proceedings of the 2012 Design, Automation & Test in Europe Conference & Exhibition*, pages 745–750. IEEE, 2012.

[92] Trent McConaghy. High-dimensional statistical modeling and analysis of custom integrated circuits. In *Proceedings of the 2011 IEEE Custom Integrated*

Circuits Conference, pages 1–8, 2011.

[93] B. Murmann. A/D converter trends: Power dissipation, scaling and digitally assisted architectures. In *Proceedings of the 2008 IEEE Custom Integrated Circuits Conference*, pages 105–112. IEEE, 2008.

[94] Martin Neil, Manesh Tailor, and David Marquez. Inference in hybrid Bayesian networks using dynamic discretization. *Statistics and Computing*, 17(3):219–233, 2007.

[95] Julia Neumann, Christoph Schnörr, and Gabriele Steidl. Combined SVM-based feature selection and classification. *Machine learning*, 61(1-3):129–150, 2005.

[96] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL (T). *Journal of the ACM (JACM)*, 53(6):937–977, 2006.

[97] D. J. O'Riordan and P. K. Bhattacharya. PSL/SVA assertions in SPICE. In *Proceedings of the Design and Verification Conference and Exhibition*, 2012.

[98] Edgar Elias Osuna. *Support Vector Machines: Training and Applications*. PhD thesis, Massachusetts Institute of Technology, 1998.

[99] Edoardo Pasolli, Farid Melgani, and Yakoub Bazi. Support vector machine active learning through significance space construction. *Geoscience and Remote Sensing Letters, IEEE*, 8(3):431–435, 2011.

[100] Marcel J. M. Pelgrom, Aad C. J. Duinmaijer, and Anton P. G. Welbers. Matching properties of MOS transistors. *IEEE Journal of Solid-State Circuits*, 24(5):1433–1439, 1989.

[101] Jean-Pierre Raskin, Andrew R Brown, Butrus T Khuri-Yakub, and Gabriel M Rebeiz. A novel parametric-effect MEMS amplifier. *Journal of Microelectromechanical Systems*, 9(4):528–537, 2000.

[102] Dave Reid, Campbell Millar, Scott Roy, and Asen Asenov. Understanding LER-induced MOSFET variability – Part I: Three-dimensional simulation of large statistical samples. *IEEE Transactions on Electron Devices*, 57(11):2801–2807, 2010.

[103] C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning*, pages 515–521, 1998.

[104] Suresh Seshadri and Jacob A Abraham. Frequency response verification of analog circuits using global optimization techniques. *Journal of Electronic Testing*, 17(5):395–408, 2001.

[105] Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

[106] Natarajan Shankar, Sam Owre, and John M. Rushby. The PVS proof checker: A reference manual. *Technical Report, Computer Science Laboratory, SRI International, Menlo Park, CA*, 1993.

[107] Guoyong Shi. A survey on binary decision diagram approaches to symbolic analysis of analog integrated circuits. *Analog Integrated Circuits and Signal Processing*, 74(2):331–343, 2013.

[108] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[109] Amandeep Singh and Peng Li. On behavioral model equivalence checking for large analog/mixed signal systems. In *Proceedings of the International Conference on Computer-Aided Design*, pages 55–61. IEEE Press, 2010.

[110] 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, aug. 2009.

[111] Alex Smola, Bernhard Scholkopf, and Gunnar Ratsch. Linear programs for automatic accuracy control in regression. In *Proceedings of the 9th International Conference on Artificial Neural Networks*, volume 2, pages 575–580. IET, 1999.

[112] Petr Somol, Pavel Pudil, Jana Novovičová, and Pavel Paclık. Adaptive floating search methods in feature selection. *Pattern Recognition Letters*, 20(11):1157–1163, 1999.

[113] Yang Song, Hao Yu, Sai Manoj Pudukotai Dinakarrao, and Guoyong Shi. SRAM dynamic stability verification by reachability analysis with consideration of threshold voltage variation. In *Proceedings of the 2013 ACM International Symposium on Physical Design*, pages 43–49. ACM, 2013.

[114] Chris Spear. *SystemVerilog for Verification: A Guide to Learning the Testbench Language Features*. Springer Science & Business Media, 2008.

[115] R. B. Staszewski, K. Muhammad, D. Leipold, C. M. Hung, Y. C. Ho, et al. All-digital TX frequency synthesizer and discrete-time receiver for Bluetooth radio in 130-nm CMOS. *IEEE Journal of Solid-State Circuits*, 39(12):2278–2291, 2004.

[116] Robert B Staszewski and Poras T Balsara. *All-digital Frequency Synthesizer in Deep-submicron CMOS*. Wiley-Interscience, 2006.

[117] Michael Stein. Large sample properties of simulations using Latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987.

[118] Sebastian Steinhorst and Lars Hedrich. Advanced methods for equivalence checking of analog circuits with strong nonlinearities. *Formal Methods in System Design*, 36(2):131–147, 2010.

[119] Sebastian Steinhorst, Alexander Jesser, and Lars Hedrich. Advanced property specification for model checking of analog systems. In *Analog 2006: 9. ITG/GMM-Fachtagung Entwicklung von Analogschaltungen mit CAE-Methoden*, pages 63–68, 2006.

[120] J.A.K. Suykens, J. De Brabanter, L. Lukas, and J. Vandewalle. Weighted least squares support vector machines: robustness and sparse approximation. *Neurocomputing*, 48(1):85–105, 2002.

[121] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[122] Donald Thomas and Philip Moorby. *The Verilog Hardware Description Language*. Springer Science & Business Media, 2008.

[123] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B*, 10(1):267–288, 1996.

[124] Michael E Tipping. Sparse Bayesian learning and the relevance vector machine. *The Journal of Machine Learning Research*, 1:211–244, 2001.

[125] Michael E Tipping and Anita C Faul. Fast marginal likelihood maximisation for sparse Bayesian models. In *C. M. Bishop and B. J. Frey (Eds.), Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics*, 2003.

[126] Saurabh K Tiwary, Anubhav Gupta, Joel R Phillips, Claudio Pinello, and Radu Zlatanovici. First steps towards SAT-based formal analog verification. In *Proceedings of the 2009 International Conference on Computer-Aided Design*, pages 1–8. ACM, 2009.

[127] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.

[128] V. Vapnik, S.E. Golowich, and A. Smola. Support vector method for function approximation, regression estimation, and signal processing. *Advances in Neural Information Processing Systems*, pages 281–287, 1997.

[129] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[130] Dimitrios Ververidis and Constantine Kotropoulos. Fast and accurate sequential floating forward feature selection with the Bayes classifier applied to speech emotion recognition. *Signal Processing*, 88(12):2956–2970, 2008.

[131] Grace Wahba. Support vector machines, reproducing kernel hilbert spaces and the randomized GACV. *Advances in Kernel Methods-Support Vector Learning*, 6:69–87, 1999.

[132] David Walter, Scott Little, and Chris Myers. Bounded model checking of analog and mixed-signal circuits using an SMT solver. In *Automated Technology for Verification and Analysis*, pages 66–81. Springer, 2007.

[133] Fa Wang, Manzil Zaheer, Xin Li, Jean-Olivier Plouchart, and Alberto Valdes-Garcia. Co-learning Bayesian model fusion: Efficient performance modeling of analog and mixed-signal circuits using side information. In *Proceedings of the 2015 International Conference on Computer-Aided Design*, pages 575–582. IEEE, 2015.

[134] Senzhang Wang, Zhoujun Li, Wenhan Chao, and Qinghua Cao. Applying adaptive over-sampling technique based on data density and cost-sensitive SVM to imbalanced learning. In *Proceedings of the 2012 International Joint Conference on Neural Networks*, pages 1–8. IEEE, 2012.

[135] Shiping Wen, Tingwen Huang, Zhigang Zeng, Yiran Chen, and Peng Li. Circuit design and exponential stabilization of memristive neural networks. *Neural Networks*, 63:48–56, 2015.

[136] Peter M Williams. Bayesian regularization and pruning using a laplace prior. *Neural computation*, 7(1):117–143, 1995.

[137] David P Wipf and Bhaskar D Rao. An empirical Bayesian strategy for solving the simultaneous sparse approximation problem. *IEEE Transactions on Signal Processing*, 55(7):3704–3716, 2007.

[138] Martin Wirnshofer. Sources of variation. In *Variation-Aware Adaptive Voltage Scaling for Digital CMOS Circuits*, pages 5–14. Springer, 2013.

[139] Stephen Wolfram. *Mathematica: A System for Doing Mathematics by Computer*. Addison Wesley Longman Publishing Co., Inc., 1991.

[140] Jinjun Xiong, Vladimir Zolotov, and Lei He. Robust extraction of spatial correlation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(4):619–631, 2007.

[141] Huan Xu, Constantine Caramanis, and Shie Mannor. Robustness and regularization of support vector machines. *The Journal of Machine Learning Research*, 10:1485–1510, 2009.

[142] Jian Yao, Zuochang Ye, and Yan Wang. Importance boundary sampling for SRAM yield analysis with multiple failure regions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(3):384–396, 2014.

[143] Yun Ye, Frank Liu, Min Chen, Sani Nassif, and Yu Cao. Statistical modeling and simulation of threshold variation under random dopant fluctuations and line-edge roughness. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 19(6):987–996, 2011.

[144] Zhipeng Ye, Honghuang Lin, and Asad Khan. Functional coverage collection for analog circuits – enabling seamless collaboration between design and verification. In *Proceedings of The Design and Verification Conference and Exhibition*, 2016.

[145] Leyi Yin, Yue Deng, and Peng Li. Verifying dynamic properties of nonlinear mixed-signal circuits via efficient smt-based techniques. In *Proceedings of the International Conference on Computer-Aided Design*, pages 436–442. ACM, 2012.

[146] Leyi Yin, Yue Deng, and Peng Li. Simulation-assisted formal verification of nonlinear mixed-signal circuits with Bayesian inference guidance. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(7):977–990, 2013.

[147] Sangho Youn, Jaeha Kim, and Mark Horowitz. Global convergence analysis of mixed-signal systems. In *Proceedings of the 48th Design Automation Confer-*

ence, pages 498–503. ACM, 2011.

[148] Guo Yu and Peng Li. A methodology for systematic built-in self-test of phase-locked loops targeting at parametric failures. In *Proceedings of the 2007 IEEE International Test Conference*, pages 1–10. IEEE, 2007.

[149] Mohamed H Zaki, Sofiène Tahar, and Guy Bois. Formal verification of analog and mixed signal designs: A survey. *Microelectronics Journal*, 39(12):1395–1404, 2008.

[150] Yan Zhang, Sriram Sankaranarayanan, and Fabio Somenzi. Piecewise linear modeling of nonlinear devices for formal verification of analog circuits. In *Proceedings of the 2012 Conference on Formal Methods in Computer-Aided Design*, pages 196–203. IEEE, 2012.