HYBRID VERIFICATION FOR ANALOG AND MIXED-SIGNAL CIRCUITS

A Thesis

by

QINGRAN ZHENG

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

MASTER OF SCIENCE

Chair of Committee,	Peng Li
Committee Members,	Gwan S. Choi
	Sebastian Hoyos
	Jianhua Huang
Head of Department,	Miroslav M. Begovic

December 2017

Major Subject: Electrical and Computer Engineering

Copyright 2017 Qingran Zheng

ABSTRACT

With increasing design complexity and reliability requirements, analog and mixed-signal (AMS) verification manifests itself as a key bottleneck. While formal methods and machine learning have been proposed for AMS verification, these two types of techniques suffer from their own limitations, with the former being specifically limited by scalability and the latter by inherent errors in learning-based models.

We present a new direction in AMS verification by proposing a hybrid formal/machine-learning-based verification technique (HFMV) to combine the best of the two worlds. HFMV builds formalism on the top of a machine learning model to verify AMS circuits efficiently while meeting a user-specified confidence level. Guided by formal checks, HFMV intelligently explores the high-dimensional parameter space of a given design by iteratively improving the machine learning model. As a result, it leads to accurate failure prediction in the case of a failing circuit or a reliable pass decision in the case of a good circuit. Our experimental results demonstrate that the proposed HFMV approach is capable of identifying hard-to-find failures which are completely missed by a huge number of random simulation samples while significantly cutting down training sample size and verification cycle time.

ACKNOWLEDGMENTS

I would like to first thank my advisor, Prof. Peng Li, for his guidance and supports throughout my M.S. stage in Texas A&M University. I would like to express my profound gratitude and sincere thanks to him not only for his insightful suggestions to my research, but also for his valuable tutoring of being a researcher.

Besides, I acknowledge my special thanks to Dr. Honghuang Lin, a previous Ph.D. student in our research group for his helpful guidance and constant encouragement when I was a fresh M.S. student. I also wish to express my heartfelt thanks to Hanbin Hu, a current Ph.D. student and my co-worker, for his technical contributions in research and untiring help in my student life.

I would also like to record my sincere thanks to Prof. Gwan S. Choi, Prof. Sebastian Hoyos, and Prof. Jianhua Huang for serving as my committee members.

Last but not the least, I would like to express the deepest gratitude to my parents, who give me deep-rooted trust, unconditional support, and eternal love, with my best wishes for their happiness and health.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by my advisor, Prof. Peng Li of the Department of Electrical and Computer Engineering.

The work described in Section 6 was completed by me independently. All other work conducted for the thesis was completed by me and my co-worker, Hanbin Hu.

Funding Sources

This thesis paper is based on work supported by the Semiconductor Research Corporation through Texas Analog Center of Excellence (Task 2712.004).

TABLE OF CONTENTS

Page

ABSTRACT ii

ACKNOWLEDGMENTS iii

CONTRIBUTORS AND FUNDING SOURCES iv

TABLE OF CONTENTS v

LIST OF FIGURES vii

LIST OF TABLES ix

1. INTRODUCTION 1

- 1.1 AMS Verification 1
- 1.2 Previous Research Works on AMS Verification 4
  - 1.2.1 Formal Verification 4
    - 1.2.1.1 Equivalence Checking 4
    - 1.2.1.2 Proof-based Symbolic Methods 6
    - 1.2.1.3 Model Checking 6
    - 1.2.1.4 Limitation of Formal Verification 8
  - 1.2.2 Machine-learning-driven Verification 8
- 1.3 Hybrid Formal/Machine-learning-based Verification Method 10
- 1.4 Thesis Organizations 10

2. OVERVIEW OF HFMV 12

3. HYBRID VERIFICATION 15

- 3.1 Statistical ML Methods 15
- 3.2 HFMV Problem Formulation 16

4. HFMV VERIFICATION FLOW 20

- 4.1 Fail Case 21
  - 4.1.1 The Probability for A Point to Fail 21
  - 4.1.2 HFMV Flow for Fail Case 22
- 4.2 Pass Case 24

4.3 Unifying Pass/Fail Verification Flow 25
4.4 Inconclusive Case 26

5. THE STATISTICAL ML AlGORITHM IN HFMV 28

5.1 A Basic Kernel Machine in ML 28
5.2 RVM Overview 29
5.3 RVFM Overview 32

6. EFFICIENT SMT-BASED FORMAL CHECKING 34

6.1 An Unsolvable SMT Problem 34
6.2 An Efficient SMT Method Based on the RVFM Framework 35
6.2.1 A Solvable SMT Method 36
6.2.2 An Efficient SMT Method 37
6.3 Speed Comparison Between Two SMT Methods 38
6.4 Deductions for General Statistical ML Models 39

7. EXPERIMENTAL RESULTS 42

7.1 A Differential Amplifier 44
7.2 A DC-DC Converter 45
7.3 A Low-dropout Regulator 47
7.4 Comparison on Runtime 48
7.5 Verification with Varying Specified Targets 49
7.6 Verification with Varying Kernel Parameters in ML Model 50

8. SUMMARY AND CONCLUSION 52

REFERENCES 53

LIST OF FIGURES

FIGURE Page

1.1 The AMS circuits' design-to-production flow. 2

1.2 Verification history in industry. 3

1.3 An overview of the two categories of AMS verification techniques. 8

2.1 Proposed HFMV methodology. 13

3.1 An example of prediction from statistical ML models. 16

3.2 The formulation of hybrid verification. 17

3.3 Interpretations of probabilistic inferences: (a) Fail case, and (b) Pass case. 17

4.1 An overall summary of the verification flow. 20

4.2 The HFMV flow for a failing circuit. 22

4.3 A detailed summary of the unified verification flow. 26

5.1 An overview of the Bayesian model of RVM. 31

5.2 An efficient RFVM training method. 33

6.1 A masking methodology for SMT solvers. 34

6.2 The simplification of SMT solution. 35

6.3 An efficient SMT flow. 39

7.1 The detailed HFMV flow in experiments: (a) Top-level outer flow (b) Inner flow to do formal checks, adjust γ, and run simulations, and (c) Resampling flow. 43

7.2 A differential amplifier. 44

7.3 A DC-DC converter. 45

7.4 Comparison of HFMV and Monte Carlo Simulation on failure identification: (a) differential amplifier, (b) DC-DC converter, and (c) LDO. The worst-case performance found by each method is normalized with respect to the specified target. 46

7.5 LDO. 46

7.6 Comparison of CPU time between HFMV and Monte Carlo simulations. 48

7.7 HFMV verification of the quiescent current of the LDO. 49

7.8 HFMV verification with varying kernel parameter γ_k. 50

LIST OF TABLES

TABLE Page

4.1 Termination conditions for HFMV.. 26

6.1 Speed comparison between two SMT methods. 39

7.1 HFMV effectiveness... 47

1. INTRODUCTION

Over the past few decades, a tremendous growth in the complexity of very-large-scale integration (VLSI) designs has been witnessed in the semiconductor industry. Due to the increasing complexity of circuits and systems, more and more arduous challenges have been brought up at an unprecedented pace. Verification of analog and mixed-signal (AMS) systems is one of these tough tasks.

1.1 AMS Verification

Nowadays, AMS systems play a key role in all sorts of applications, including computers, personal phones, wireless sensors, wearable and portable devices, robots, automotive electronics, and some fast-developing systems, e.g. Internet of Things and Cyber Physical Systems. The importance of AMS systems continues to increase due to their irreplaceable responsibility as interfaces between the real world and inner systems. However, with increasing design and integration complexity, decreasing time to market, and rising reliability requirements across broad ranges of chip designs, AMS verification manifests itself as a key bottleneck [1].

AMS verification is a methodology that checks the correctness of properties of an AMS design and investigates whether it conforms to some design specifications. As shown in Fig. 1.1, the AMS circuits' design-to-production flow starts from design, followed by verification. These two phases have a close interaction with each other and are also called pre-silicon stages. The next step comes to fabrication, which actually transforms the design into silicon and is very expensive. The stages after fabrication are called post-silicon stages, including testing, yield analysis and finally production. Usually, bugs or failures identified in post-silicon testing are much more expensive compared to the ones captured in pre-silicon verification. As a result, the pre-silicon design-verification iterations are

expected to capture and fix as many failures as possible, and the role of verification has become more and more important.

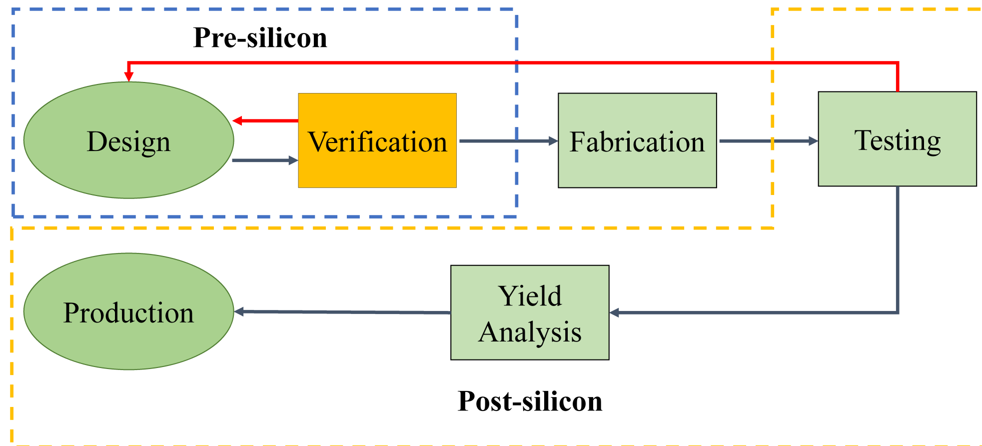


Figure 1.1: The AMS circuits' design-to-production flow.

As design complexity has increased over the decades, verification in the digital world has been greatly advanced as shown in Fig. 1.2. In old times, digital verification mainly focused on simulation-based techniques (e.g. simulations on test benches) in transistor/gate level or Register-transfer level (RTL) using Hardware Description Language (HDL), e.g. Verilog [2]. Nowadays, Universal Verification Methodology (UVM) [3], which emphasizes the automotive generation of test benches and improves the interoperability of verification components, has become a mainstream in the industry. Currently, some new techniques, like reusable verification Intellectual Property (IP) and FPGA emulation/prototyping, have also been developed for increasingly complicated digital systems.

However, AMS verification is still immature compared to the digital side mostly due to the natural complexity of AMS systems. One key reason for this is that design parameters (e.g. amplitude, frequency, and phase) and operation conditions of AMS systems are continuous variables, which makes design spaces or searching spaces of verification

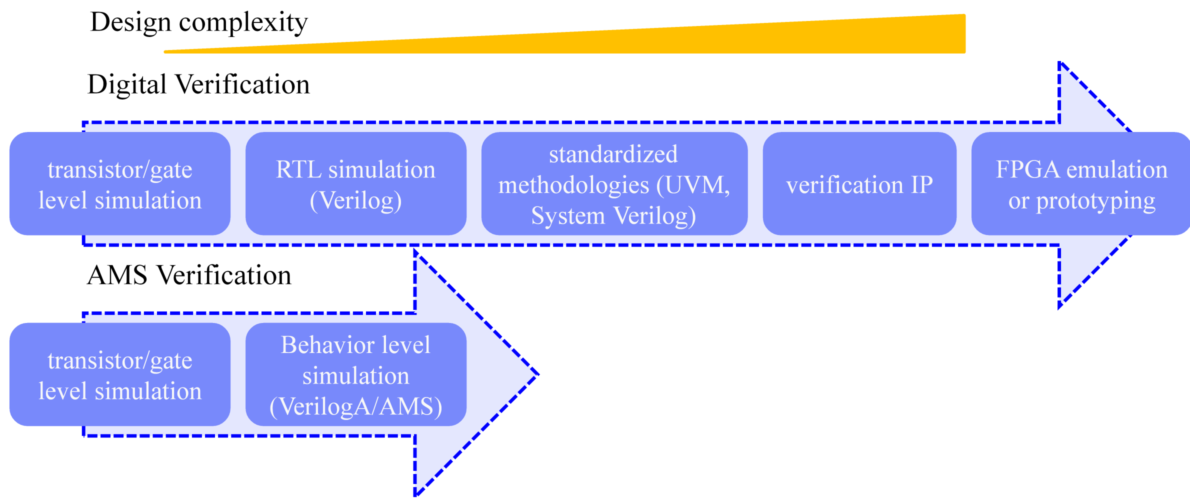


Figure 1.2: Verification history in industry.

infinitely large and would easily lead to the curse of dimensionality. Another reason is the intrinsic nonlinearity of AMS systems. Unfortunately, the functionality of AMS circuits cannot be atomized to a series of basic logic operations like digital circuits. Instead, it's described by some continuous and usually nonlinear mathematical functions. This nature also results in expensive simulation cost. Besides, most AMS systems still heavily depend on customized designs, making them impractical to follow some general and standardized rules or methodologies like digital systems.

As shown in Figure 1.2, for AMS verification, simulation-based methods, no matter in transistor/gate level or behavior level with VerilogA/AMS [4], still prevail in the industry. Similar to old-time digital verification methods, those AMS verification methods verify AMS designs by running simulations across "corners" in design spaces and then checking if the outputs of simulations pass or fail some given specifications. To be more detailed, simulations are usually done with different permutations of parameters, such as initial conditions, process variations and design parameters, varying in given ranges. The selected permutations of parameters (usually minimal, nominal and maximal values) are so-called

"corners". As illustrated before, the design space or searching space of AMS verification is infinitely large due to the inherent continuity of AMS parameters. Hence, it is impossible to give a complete coverage only with these "corners". In addition, the selection of these "corners" also heavily depends on engineers' experience and knowledge, which makes it hard to be generalized. It's obvious that simulation-based methods can only demonstrate the presence of failures but cannot demonstrate their absence, i.e. they cannot give an absolute pass or fail answer to AMS designs.

1.2 Previous Research Works on AMS Verification

While traditional simulation-based verification still dominates in industrial AMS verification, many efforts have been put in this area by researchers. For example, formal verification techniques [5, 6, 7, 8] have been attracting more and more interests from researchers. On the other side, some people are also trying to make innovations by employing machine learning (ML) techniques to extract performance models based on simulations of AMS circuits [9, 10, 11].

1.2.1 Formal Verification

Formal verification is appealing as it builds rigorous models of design under verification (DUV) and mathematically proves or disproves the correctness of DUV, with a coverage of 100%. Through formal techniques, we are able to give a provable or absolute "pass/fail" answer with respect to some specifications. Generally, formal methods can be divided into three groups: equivalence checking, proof-based symbolic methods, and model checking. Here we will give some brief introductions for these three methods.

1.2.1.1 Equivalence Checking

As illustrated in Fig. 1.2, AMS systems are usually modeled on different levels of abstraction, such as transistor/gate level (e.g. schematics and SPICE [12] netlists), behav-

ioral models [13], macromodels [14], etc. Thus, it's necessary to guarantee that models in different levels for the same AMS design are functionally equal or at least similar. Equivalence checking is used to perform such tasks, proving or disproving the functional equivalence or similarity between two models of DUV. This method is also widely used in digital verification, e.g. checking the equivalence between synthesized netlists and their corresponding HDL descriptions. The mathematical description of equivalence checking is shown as follows:

Given an input space Φ_I and parameter space Φ_P and assuming that now we have two different models M_1 and M_2 for DUV, the goal is to prove or disprove the following formula with respect to any input x and parameter p in input and parameter spaces.

$$M_1(x,p) = M_2(x,p), \forall x \in \Phi_I, \forall p \in \Phi_P, \tag{1.1}$$

or

$$M_1(x,p) \approx M_2(x,p), \forall x \in \Phi_I, \forall p \in \Phi_P. \tag{1.2}$$

.

Several works have been done in this area. For example, for linear AMS systems that can be described by transient functions, [15] verified the correctness of transient behaviors by performing equivalence checking between transfer function models and the implementation. Other works have also been done for nonlinear AMS systems [16, 17], which implies that ordinary differential equations (ODEs) or differential algebraic equations (DAEs) can be used to build behavior-level models for AMS systems. Besides, aiming at verifying functional similarity instead of equivalence, authors in [13] proposed a way to consider the equivalence checking problem as an optimization problem with a goal to minimize the errors between a behavioral model and its corresponding implementation.

1.2.1.2 Proof-based Symbolic Methods

Generally, proof-based symbolic verification methods are theorem proving methods based on formal deductions of properties of DUV. To be more specific, properties of DUV can be imaged as a set of formulas and conditions, namely axioms, and a given specification T is the theorem that we want to prove. The mathematical description of these methods is shown below.

Given a set of rules of inference R_{infer} and a specification T, the proof can be denoted as $H_{proof} = (A_1, A_2, ..., A_n, P)$, in which A_i is either a direct axiom of DUV or an inference from its predecessors $(A_1, A_2, ..., A_{i-1})$ following rules R_{infer}.

A practical way to solve this proof is to use theorem provers. For example, [18] utilized a high order logic proof checker PVS [19] to perform equivalence checking between an approximated linear model of a synthesized netlist and its corresponding behavioral model, and [20] used Mathematica [21] to prove the correctness of properties of AMS systems with normalized recurrence equations. In addition to making use of theorem provers, other attempts have been done to develop some binary decision diagram (BDD) [22, 23, 24] based data frameworks to manipulate Boolean based symbolic representations.

1.2.1.3 Model Checking

Model checking is a group of methods for determining whether a system satisfies a formal specification that is usually expressed in temporal logical formulas. Model checking was first introduced to verify the correctness of properties of some discrete finite-state systems [25]. For AMS verification, DUV is first transferred to a model, and then a model checker is employed to determine if a given specification T is satisfied by completely searching the entire design or state space [26].

For a specification T, we want to build a model M of DUV and check if T is satisfied or not, by analyzing all the reachable state spaces of M. This typically reduces to "reach-

ability problems" on M, and it is called reachability analysis [27, 8, 26]. The definition of reachability analysis is shown as follows:

For an initial state space Φ_0, we can compute a sequence of temporal reachable state spaces $(\Phi_1, \Phi_2, ..., \Phi_n)$ for M until T is satisfied or violated in Φ_n.

The next reachable state can be calculated with a transition function Tr:

$$\Phi_{i+1} = Tr(x), \forall x \in \Phi_i. \tag{1.3}$$

For linear AMS systems, Equation 1.3 can be accurately and efficiently computed [28, 29]. However, for nonlinear AMS systems, an overapproximated state space $\hat{\Phi}_i$ is adopted in practice, due to the arduousness in computing the exact reachable state space.

$$\hat{\Phi}_{i+1} \supseteq Tr(x), \forall x \in \hat{\Phi}_i. \tag{1.4}$$

The side effect from the above equation is that the accumulation of overapproximation may harm the convergence of reachability analysis. To address this problem, many research works have been developed to tighten the overapproximation, e.g. by using hypercubes [30, 31], convex polygons [32, 33], support functions [34, 35], zonotopes [36], etc.

In recent years, satisfiability modulo theories (SMT) [37, 38, 39], especially the theory of real numbers, have been greatly developed to prove the satisfiability of formulas. By employing the power of SMT, [33, 40, 7, 8] proposed methods that convert reachability analysis problems into SMT problems and check reachable state spaces using different kinds of SMT solvers (e.g. iSAT3 [41], Z3 [42], Yices [43], etc.).

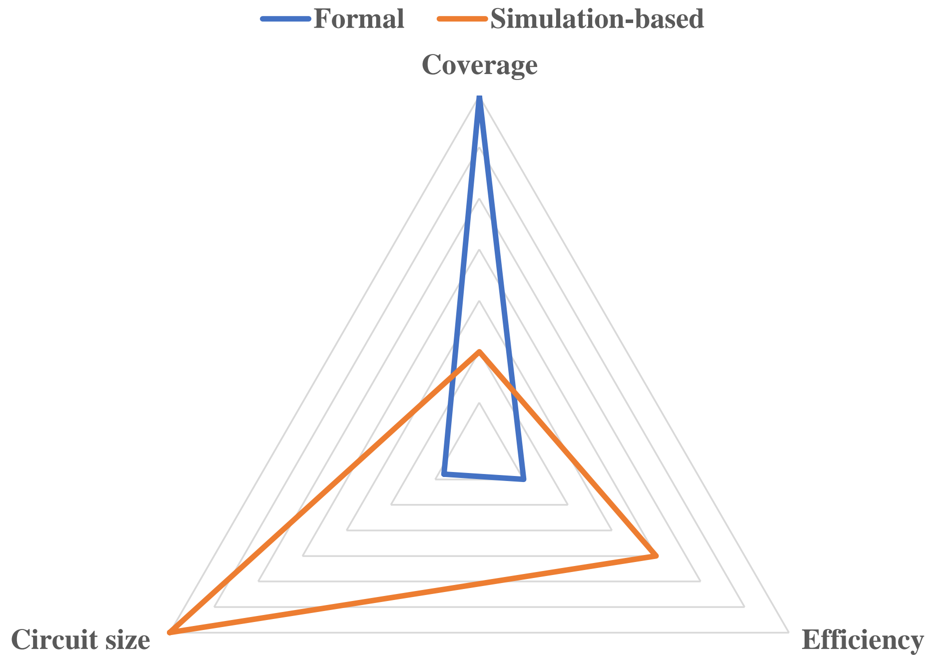


Figure 1.3: An overview of the two categories of AMS verification techniques.

1.2.1.4 Limitation of Formal Verification

Whereas in the past decades, we have witnessed considerable development in academic research of AMS formal verification, the efficiency of AMS formal verification is still an obstacle, and the applicability is limited to circuits with small sizes [5, 6]. Consequently, although a noticeable growth of formal methods has been observed in the industry of digital integrated circuits [44], the industrial availability of such techniques in AMS verification is still minimal. Fig. 1.3 has a good overview of formal verification, compared to simulation-based verification.

1.2.2 Machine-learning-driven Verification

ML techniques keep being fast developed in recent years and have been widely used in various domains, like imaging processing, computer vision, bioinformatics, information retrieval, big data, and so on, showing their great power for performing tasks of classifica-

tion, regression, clustering, and decision making. With the benefits of being data-driven, incremental, and scalable, recent researchers leverage ML as powerful techniques to solve assorted circuit problems, including AMS verification.

For a circuit verification task, naturally, there are two types of outcomes: either pass or fail with respect to given specifications. This can be considered as a binary classification problem. [45, 46] employed the Support Vector Machine (SVM) [47] algorithm to train classifiers for the feasibility and performance of AMS circuits.

A different approach towards ML-driven verification is to describe or characterize AMS circuits with regression models, which reduces the arduousness in analyzing and extracting specific models for AMS systems. For example, authors of [48] proposed a method that makes use of recursive vector fitting (RVF) techniques [49] to model the time-domain response from a nonlinear circuit and hence automatically extracts the circuit's analytical behavior-level model. By utilizing statistical ML algorithms for training regression models of circuits' performance, [9] developed a Co-Learning Bayesian Model Fusion (CL-BMF) ML algorithm, which can take advantage of the performance side information collected from simulation/measurement, and [11] implemented the Gaussian Process [50] algorithm with an artificially composed kernel.

Despite the goodness of improvements in efficiency and scalability, exploiting ML techniques in AMS verification also introduces new challenges, not only from the nature of ML, like over-fitting problems and inherent model uncertainty, but also from the problems in the AMS verification perspective, such as the limited availability of training data due to expensive simulation cost. Hence, integrating ML techniques into the AMS verification procedure is a challenging task which requires innovative works from both domains.

1.3 Hybrid Formal/Machine-learning-based Verification Method

Motivated by the above AMS verification challenges, this work presents a new perspective in AMS verification with a hybrid formal/machine-learning-based verification technique (HFMV) framework that simultaneously exploits SMT-based formal checking [33, 40, 7, 8] and statistical ML techniques. This framework has the best of the two worlds: it adds a degree of formalism on top of ML models by utilizing satisfiability modulo theories (SMT) based formal techniques; and it is much more scalable than pure formal techniques at the same time. Furthermore, to circumvent the inherent model uncertainty of ML techniques, the proposed framework “formally" bounds learning model uncertainty and practically verifies design properties over high-dimensional spaces of pure design uncertainty, for which only bounds of parameter values are assumed.

The proposed HFMV makes the following key contribution to AMS verification: **1)** narrowing the gap between design complexity and the scalability of current verification approaches by developing a novel hybrid technique, and **2)** building a degree of formalism into ML-based verification methods while boosting scalability.

Our experimental results have demonstrated that HFMV can provide reliable verification of AMS design's specifications in high-dimensional parameter spaces for which time-consuming Monte Carlo simulations, however, produce misleading results. At the same time, HFMV reduces the overall verification runtime from hundreds of CPU hours to tens of CPU hours.

1.4 Thesis Organizations

In this thesis, Section 2 introduces a top level of proposed HFMV methodology, which includes two core parts: a formal check block working on the design space and a ML engine performing data acquiring and ML model training. A brief introduction about the inputs and outputs ("Pass", "Fail", "Inconclusive") of HFMV is provided.

Section 3 focuses on illustrating the "formal" problem we want to verify, which is built on statistical ML models. A detailed deduction is done to bring up the formalism in probability. In addition, the basic concept of statistical ML models is introduced.

Section 4 covers a detailed flow of HFMV, which includes an iteratively active learning process with a sampling technique guided by formal checking. This flow is first demonstrated in the "Fail" case and then unified to other two cases (i.e. "Pass" and "Inconclusive"). A summary of conditions of these three cases is also listed at the end of this Section.

Section 5 talks about the statistical ML algorithm that employed by HFMV. A brief introduction of Relevance Vector Machine (RVM) [51] and its successor Relevance Vector and Feature Machine (RVFM) [52] is given.

In Section 6, an efficient SMT-based formal check technique is proposed. It covers necessary mathematical deductions for the efficient formal check technique, based on the framework of RVFM. In the end, extended deductions have been done to demonstrate this technique can also be applied to a general statistical ML algorithm.

Section 7 shows the experimental results of HFMV, which prove its superiority in both capability to identify failures and running time. Experiments are done on three circuits: a differential amplifier, a DC-DC converter [53] and a Low-dropout regulator (LDO) [54].

This thesis is concluded in Section 8, with a brief summary of properties of HFMV.

2. OVERVIEW OF HFMV

HFMV “formally" checks a given circuit against a targeted specification over a space of uncertain factors, e.g. the uncertainty of design parameters and operating conditions, with a user-specified confidence level. The proposed hybrid verification is performed by conducing formal checks on a statistical ML model trained using simulation/measurement data of a circuit and predicts a targeted performance over a high-dimension design/uncertainty space. The fundamental objective of HFMV is to determine if the circuit meets a given specification over the entire design/uncertainty space with the user-specified probability or confidence level.

As shown in Fig. 2.1, three types of outputs can be produced by HFMV: “Pass", “Fail" and “Inconclusive". A “Pass" conclusion gives an evident “yes" answer to the circuit's validity and is only produced with the user-specified high probability/confidence. Otherwise, any identified failure, i.e. a point in the design/uncertainty space at which the targeted specification is not met, would mark the circuit as “Fail". In this case, HFMV takes one step further to generate an accurate failure prediction model, which tells the designer where the circuit fails in the design/uncertainty space to provide insights for debugging. An “Inconclusive" answer is made when HFMV does not find any true failure and a high-confidence answer on the circuit's validity cannot be reached based on available training data (i.e. with a probability greater than the user-specified confidence level).

For a given circuit, the user provides the following inputs to the HFMV engine in Fig. 2.1:

- The circuit's description (e.g. SPICE-level netlists and behavior description using VerilogA/AMS).
- Ranges of variation for design parameters or working conditions of the circuit.

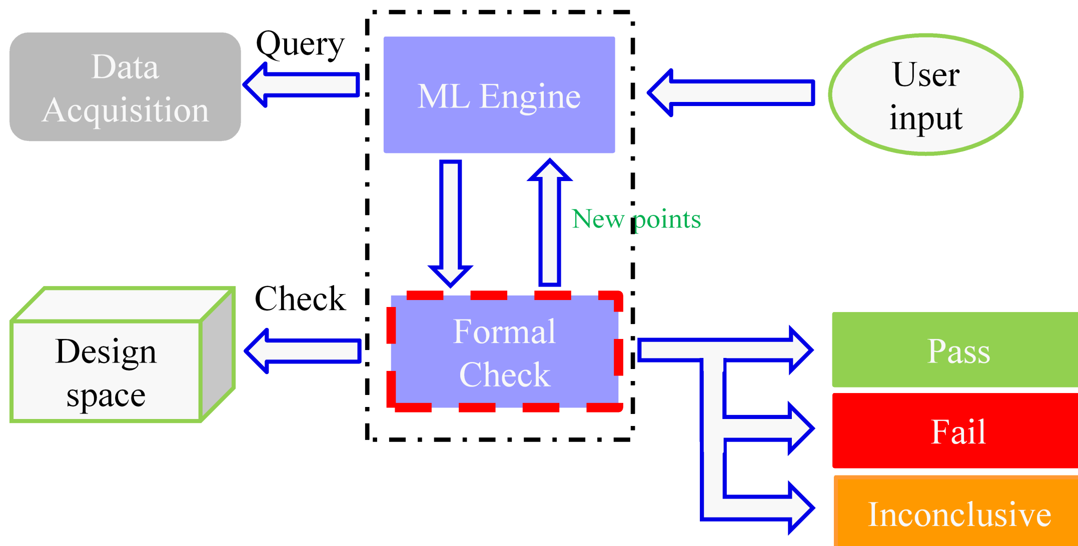


Figure 2.1: Proposed HFMV methodology.

These ranges represent the design uncertainty that is used to determine the design space we want to work in.

- A small number of initial sample data of the circuit. To start with, the HFMV engine must have some initial data, not necessary to be very big, to build an initial ML model.

- A targeted specification T. Without loss of generality, this paper assumes that a greater performance value corresponds to a worse specification. A point where the circuit's performance exceeds T is regarded as a failure.

- User-specified confidence P_0 for "Pass". HFMV produces a "Pass" answer if and only if the specification T is estimated to be satisfied at any point with a probability greater than P_0 in the entire design/uncertainty space.

- Maximum sample size S_{max}. It specifies an upper limit for the amount of training data to be acquired in practice.

With the above inputs, the HFMV engine builds models trained by statistical ML algorithms based on the initial data and calls formal checks over the entire design space, while interacting with the data acquisition block, e.g. a circuit simulator, which will provide new training data. In addition, a smart resampling strategy guided by SMT-based formal check techniques is implemented to feed new points back to the ML training process and iteratively improve the reliability of statistical ML models.

3. HYBRID VERIFICATION

The key novelty of proposed hybrid approach lies in the fact that it leverages statistical ML for scalability and SMT-based formula checking for a degree of formalism on top of ML models.

3.1 Statistical ML Methods

There exists a large body of statistical ML algorithms where each inference is probabilistic, e.g. Gaussian process [50], Bayesian additive regression trees [55], and RVM [51]. Typically, an inference regression prediction is assumed to be Gaussian with a mean $\hat{y}_{est}$ and a variance $\hat{\sigma}_{est}$, latter of which quantifies the confidence of prediction. Fig. 3.1 gives an example of the prediction from statistical ML algorithms, in which the red dots are training samples, the green line stands for the predicted mean value $\hat{y}_{est}$, and the yellow lines are the bounds of the 95% confidence interval of the prediction ($[\hat{y}_{est} - 1.96\hat{\sigma}_{est}, \hat{y}_{est} + 1.96\hat{\sigma}_{est}]$).

HFMV offers a generic hybrid verification framework and is agnostic about the specific choice of statistic ML models. For demonstration purpose, this work adopts RVFM [52], which is an extension to RVM [51] and offers improved accuracy and appealing probabilistic feature weighting capability, as the built-in statistical ML model of HFMV. A circuit's model trained by RVFM predicts the circuit's performance at a given point $\boldsymbol{x}$ by computing the mean and variance as follows:

$$\hat{y}_{est}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}), \tag{3.1}$$

$$\hat{\sigma}^2_{est}(\boldsymbol{x}) = \sigma^2 + \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\Sigma} \boldsymbol{\phi}(\boldsymbol{x}), \tag{3.2}$$

where σ^2 is the estimated noise, $\boldsymbol{w}$ and $\boldsymbol{\Sigma}$ are the posterior mean and covariance matrix of samples' weights, respectively, and $\boldsymbol{\phi}(\boldsymbol{x})$ is the kernel vector based on a chosen kernel

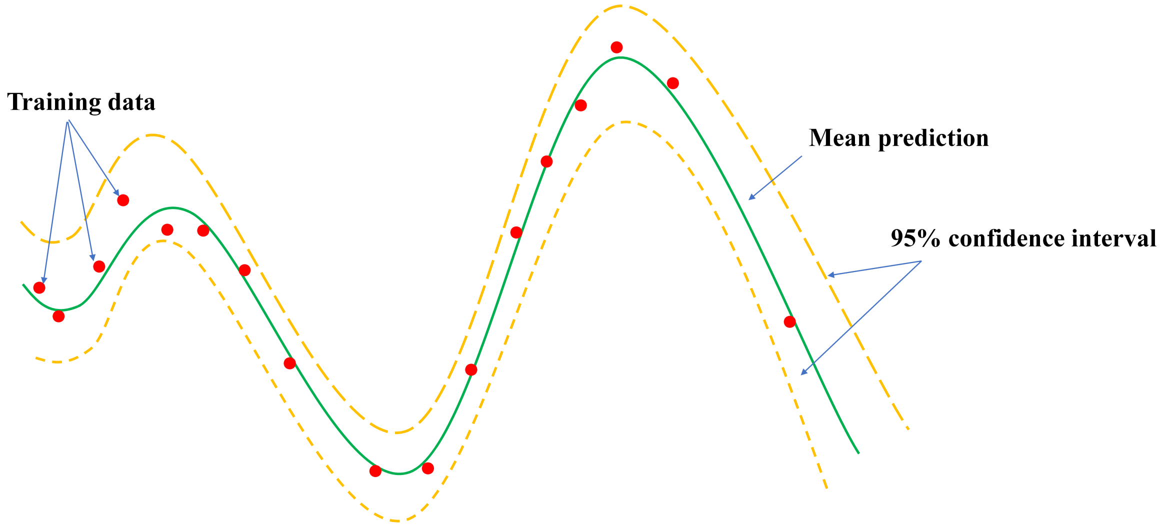


Figure 3.1: An example of prediction from statistical ML models.

function. The detailed expressions for the above prediction can be found from [52].

3.2 HFMV Problem Formulation

To circumvent the inherent model uncertainty of ML methods, we add a degree of formalism on top of models by formulating formal checks of a given specification. This allows us to “formally" bound the uncertainty of learning models and practically verify design properties over a high-dimensional space of design/uncertainty space. In this work, we assume only the bounds of AMS parameters are available. Extension to a combination of the bounded uncertainty with statistically characterized parametric variations is possible, however, is out of the scope of present thesis.

Now, on top of a statistical ML model, we pose a formal verification problem, which basically checks if a targeted specification T can be always met with a probability greater than P_0 over a design/parameter space $\Omega_{\boldsymbol{x}}$, as shown in Fig. 3.2. This boils down to employing an SMT solver (e.g. [37, 42]) to check the following formula:

$$Prob\left\{y\left(\boldsymbol{x}\right) \text{ satisfies spec. } T\right\} > P_0, \forall \boldsymbol{x} \in \Omega_{\boldsymbol{x}}, \tag{3.3}$$

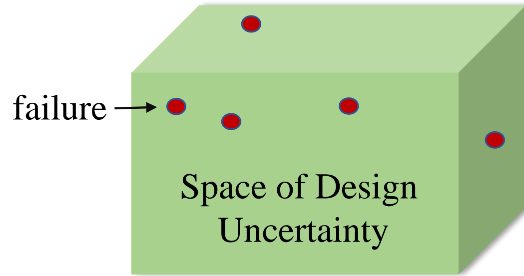


Figure 3.2: The formulation of hybrid verification.

Or,

$$Prob\left\{y\left(\boldsymbol{x}\right)\text{ satisfies spec. }T\right\} = Prob\left\{y\left(\boldsymbol{x}\right) \leq T\right\} > P_0, \forall \boldsymbol{x} \in \Omega_{\boldsymbol{x}}. \tag{3.4}$$

Assuming each prediction from the statistical ML model is Gaussian, if the following formula

$$\hat{y}_{est}\left(\boldsymbol{x}\right) + \gamma_p \hat{\sigma}_{est}\left(\boldsymbol{x}\right) \leq T, \forall \boldsymbol{x} \in \Omega_{\boldsymbol{x}} \tag{3.5}$$

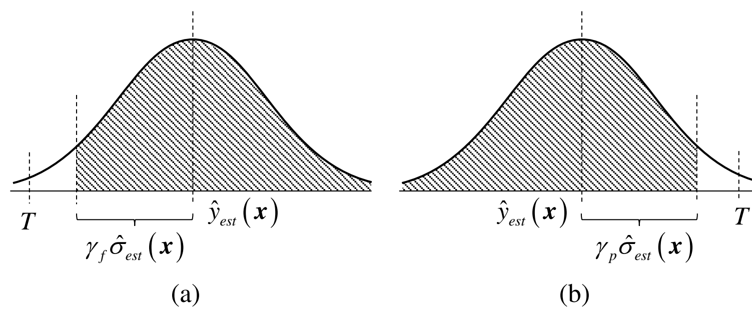


Figure 3.3: Interpretations of probabilistic inferences: (a) Fail case, and (b) Pass case.

is satisfied, with Fig. 3.3(b), we can get

$$\begin{aligned} Prob\left\{y\left(\boldsymbol{x}\right)\leq T\right\} &> Prob\left\{y\left(\boldsymbol{x}\right)<\hat{y}_{est}\left(\boldsymbol{x}\right)+\gamma_p\hat{\sigma}_{est}\left(\boldsymbol{x}\right)\right\} \\ &= \Phi\left(\gamma_p\right), \end{aligned} \tag{3.6}$$

in which γ_p is a positive confidence-level control parameter. This results in a more specific formal check on Equation 3.5, and the probability can be made sufficiently large, e.g. greater than P_0, by increasing γ_p.

This verification problem is formal in the sense that meeting the specification T is examined across the full space of $\Omega_{\boldsymbol{x}}$. In addition, the ML model's uncertainty is bounded by enforcing that T is achieved with a probability greater than P_0.

Similarly, if the following formula

$$\hat{y}_{est}\left(\boldsymbol{x}\right)-\gamma_f\hat{\sigma}_{est}\left(\boldsymbol{x}\right)\geq T, \exists\boldsymbol{x}\in\Omega_{\boldsymbol{x}} \tag{3.7}$$

is satisfied, we can also get

$$\begin{aligned} Prob\left\{y\left(\boldsymbol{x}\right)\text{ fails spec. }T\right\} &= Prob\left\{y\left(\boldsymbol{x}\right)>T\right\} \\ &> Prob\left\{y\left(\boldsymbol{x}\right)>\hat{y}_{est}\left(\boldsymbol{x}\right)-\gamma_f\hat{\sigma}_{est}\left(\boldsymbol{x}\right)\right\} \\ &= \Phi\left(\gamma_f\right), \end{aligned} \tag{3.8}$$

which implies that for an existed point $\boldsymbol{x}$ that satisfies Formula 3.7, the possibility to be a failure is at least $\Phi\left(\gamma_f\right)$.

Having all the knowledge above, we can combine a statistical ML model into Formula 3.5 or 3.7 and achieve a degree of confidence level (controlled by γ_f or γ_p) that can be proved by formal techniques. However, it is nontrivial to solve a series of problems, e.g. the efficiency and trustworthiness of the proposed hybrid verification. We address these

challenges by proposing several techniques, including adaptive tuning of the confidence-level control parameter γ (a unified form of γ_f and γ_p), iterative refinement of the learning model and formal checks (discussed in Section 4), and smart sampling of training data based on a fast SMT solution (discussed in Section 6).

4. HFMV VERIFICATION FLOW

The trustworthiness of HFMV critically depends on the accuracy of the underlying ML model, which can be improved at the cost of additional training samples. As such, one key objective of HFMV is to verify a given circuit with high confidence using a minimum possible amount of simulation or measurement data for training the ML model. We achieve this goal by starting HFMV with a small amount of initial training data and developing an iterative hybrid verification flow which adaptively adjusts conservativeness and training sample size. Fig. 4.1 shows an overall summary of this verification flow. There's an iteratively active learning process between model training and formal checking in the first phase P1, in which the new samples gained through formal checks with respect to specifications are fed back to refine the ML model. If we cannot find any failure in P1 with the limited training samples available, we come to a "Pass" or "Inconclusive" result based on the confidence level achieved. Otherwise, we give a "Fail" result with a failure prediction model, which is shown in P2 of Fig. 4.1.

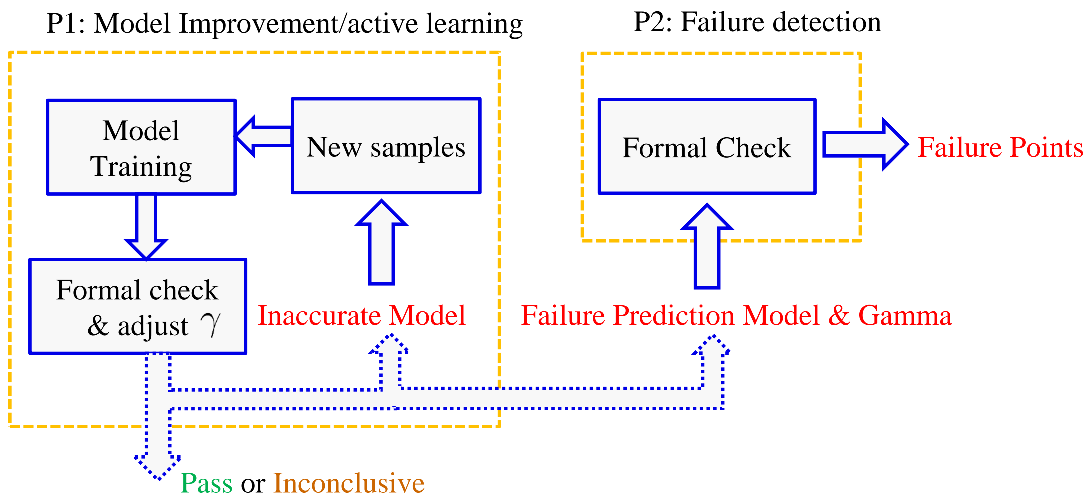


Figure 4.1: An overall summary of the verification flow.

This flow presents a unified solution regardless the type of decisions (i.e. "Fail", "Inconclusive", and "Pass") made at the end of the verification process. We first describe in detail how this verification flow works when a “Fail” decision is made, then extend it to the other two possible decisions.

4.1 Fail Case

4.1.1 The Probability for A Point to Fail

Since the underlying ML model is probabilistic, we examine the probability for circuits to fail at a point $\boldsymbol{x}$ in the parameter space $\Omega_{\boldsymbol{x}}$:

$$Prob\left\{y\left(\boldsymbol{x}\right)\ \text{fails spec.}\ T\right\} = Prob\left\{y\left(\boldsymbol{x}\right) > T\right\}. \tag{4.1}$$

In HFMV, the ML model estimates the true performance $y\left(\boldsymbol{x}\right)$ by $\hat{y}_{est}\left(\boldsymbol{x}\right)$ with a variance of $\hat{\sigma}_{est}\left(\boldsymbol{x}\right)$. Considering the common situation that each inference follows the Gaussian distribution, the probability of $y\left(\boldsymbol{x}\right)$ being in $\left[\hat{y}_{est}\left(\boldsymbol{x}\right) - \gamma_f \hat{\sigma}_{est}\left(\boldsymbol{x}\right), +\infty\right)$ is $\Phi\left(\gamma_f\right)$, where $\Phi\left(\cdot\right)$ is the cumulative distribution function of the normal distribution.

As discussed in Section 3, if a point $\boldsymbol{x}$ satisfies formula

$$\hat{y}_{est}\left(\boldsymbol{x}\right) - \gamma_f \hat{\sigma}_{est}\left(\boldsymbol{x}\right) > T, \tag{4.2}$$

the possibility for it to be a true failure point is at least $\Phi\left(\gamma_f\right)$. This implies that an SMT instance defined by Formula 4.2 or its modified version below:

$$\hat{y}_{est}\left(\boldsymbol{x}\right) > T + \gamma_f \hat{\sigma}_{est}\left(\boldsymbol{x}\right) \tag{4.3}$$

can be formally solved to identify potential failure points. The probability for a point $\boldsymbol{x}$ satisfying (4.2) or (4.3) to be a true failure can be increased by increasing γ_f.

4.1.2 HFMV Flow for Fail Case

In the continuous-valued parameter space $\Omega_{\boldsymbol{x}}$, a failing circuit may fail at an infinite number of points. While identifying one or a list of failures is useful, a more meaningful objective is to provide an accurate failure prediction model to allow designers to identify a large number of failures across the entire parameter space, which offers a good guidance for fixing the circuit.

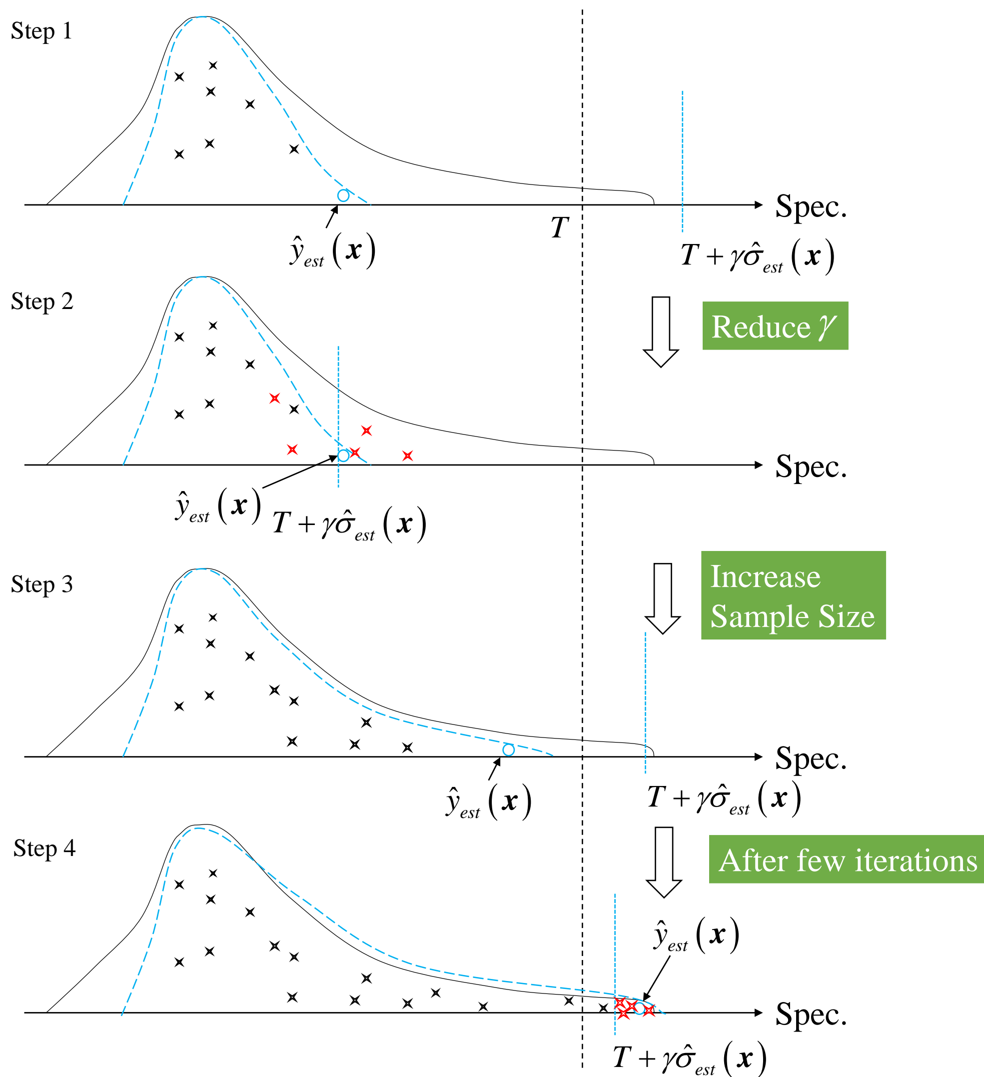


Figure 4.2: The HFMV flow for a failing circuit.

The HFMV flow for a failing circuit is illustrated in Fig. 4.2. The black curve denotes the true distribution of the circuit's performance with a targeted specification T. The two parameters, γ and training sample size, are adjusted adaptively during the entire flow, which is described as follows.

Step 1: Use a positively large γ in Formula 4.3, and solve this SMT instance to find points that might be true failures with a high probability. Due to the small initial training sample size and the model inaccuracy, the SMT solver is unlikely to find any failure by returning an UNSAT answer in the beginning.

Step 2: Lower the conservativeness of finding true failure points by reducing γ value step by step until the formal check of (4.2) or (4.3) finds some failures (marked in red), which may not be all true failures. Measure the failure prediction accuracy by verifying the actual performance of these red points via SPICE simulations. If the failure prediction accuracy R is larger than a preset value R_0, jump to Step 4; otherwise, proceed to Step 3.

Step 3: Perform smart sampling to acquire more training data, by reusing the SPICE data from Step 2 and collecting additional simulation samples at points where the model uncertainty level is high. Retrain a more accurate ML model. Go back to Step 1, and restart the process again with the same positively large γ.

Step 4: Exit with an accurate failure prediction model generated. Here, the failure prediction model is based on checking Formula 4.3 with the gamma value reached in Step 2 using the trained ML model.

In Step 2, the initial ML model may not be accurate enough due to the small training sample size at the very beginning. In this case, γ may need to be adjusted to a negatively large value to satisfy Formula 4.3, which might render a large portion of the failure points

discovered to be false failures. Nevertheless, these points are the best estimates from the current model. After retraining with these additional samples, the model is expected to be more accurate, and some true failure points may be revealed with a positive γ using this improved model. The failure prediction accuracy R, defined as the ratio between true failures and potential failure points found by the SMT solver, is used as an internal measure of model accuracy.

The smart sampling strategy mentioned in Step 3 involves two types of data. The first type includes the failure points predicted by the current model, i.e. ones that satisfy Formula 4.3 with the γ value adjusted in Step 2. These points can be efficiently identified by a fast SMT solution process that will be mentioned in Section 6. The other type of data contains points in the design/uncertainty space where the current ML model has a large prediction variance $\hat{\sigma}_{est}(\boldsymbol{x})$, indicating a low accuracy. In other words, the first data type covers points that are most likely to be true failures while the second helps to make the model accurate across the entire parameter space.

4.2 Pass Case

Recall that a "Pass" decision is only made when the targeted specification T is met with a high confidence level in the entire parameter space. More specifically, the following probability must be sufficiently large:

$$Prob\{y(\boldsymbol{x}) \text{ satisfies spec. } T\} = Prob\{y(\boldsymbol{x}) \le T\}. \tag{4.4}$$

Again, with each prediction being Gaussian, as analyzed in Section 3, if the following formal check

$$\hat{y}_{est}(\boldsymbol{x}) + \gamma_p \hat{\sigma}_{est}(\boldsymbol{x}) \le T \tag{4.5}$$

is true for all $\boldsymbol{x}$ in the parameter space $\Omega_{\boldsymbol{x}}$, the specification T is met at all points in $\Omega_{\boldsymbol{x}}$

with a probability of at least $\Phi(\gamma_p)$.

This probability can be made sufficiently large, e.g. greater than the user-specified level P_0, by increasing γ_p.

4.3 Unifying Pass/Fail Verification Flow

Instead of directly checking the satisfaction of Formula 4.5, HFMV checks the unsatisfactory of Formula 4.6 for the pass case. That's to check if no point in $\Omega_{\boldsymbol{x}}$ satisfies the following formula:

$$\hat{y}_{est}(\boldsymbol{x}) + \gamma_p \hat{\sigma}_{est}(\boldsymbol{x}) > T. \tag{4.6}$$

An interesting fact is that the SMT instances for the fail and pass cases presented in Formula 4.2 and Formula 4.6 respectively, are identical except for the sign before γ_f or γ_p. We combine these two formal checks into one with a unified parameter γ:

$$\hat{y}_{est}(\boldsymbol{x}) - \gamma \hat{\sigma}_{est}(\boldsymbol{x}) > T. \tag{4.7}$$

In the pass case, the probability of the circuit to be good is given by: $\Phi(\gamma_p) = \Phi(-\gamma) = 1 - \Phi(\gamma)$.

The HFMV flow for the fail case described in Section 4.1.2 is applicable to the pass case by only changing Step 4. In this step, instead of exiting after finding an accurate failure prediction model, we exit and draw a "Pass" decision when two conditions are met: 1) The training sample size exceeds S_{max}; this is to maintain the highest possible accuracy for the ML model in HFMV, and 2) the γ value that doesn't satisfy Formula 4.7 is negatively large enough such that the probability for the circuit to be indeed good is greater than the user-specified confidence P_0.

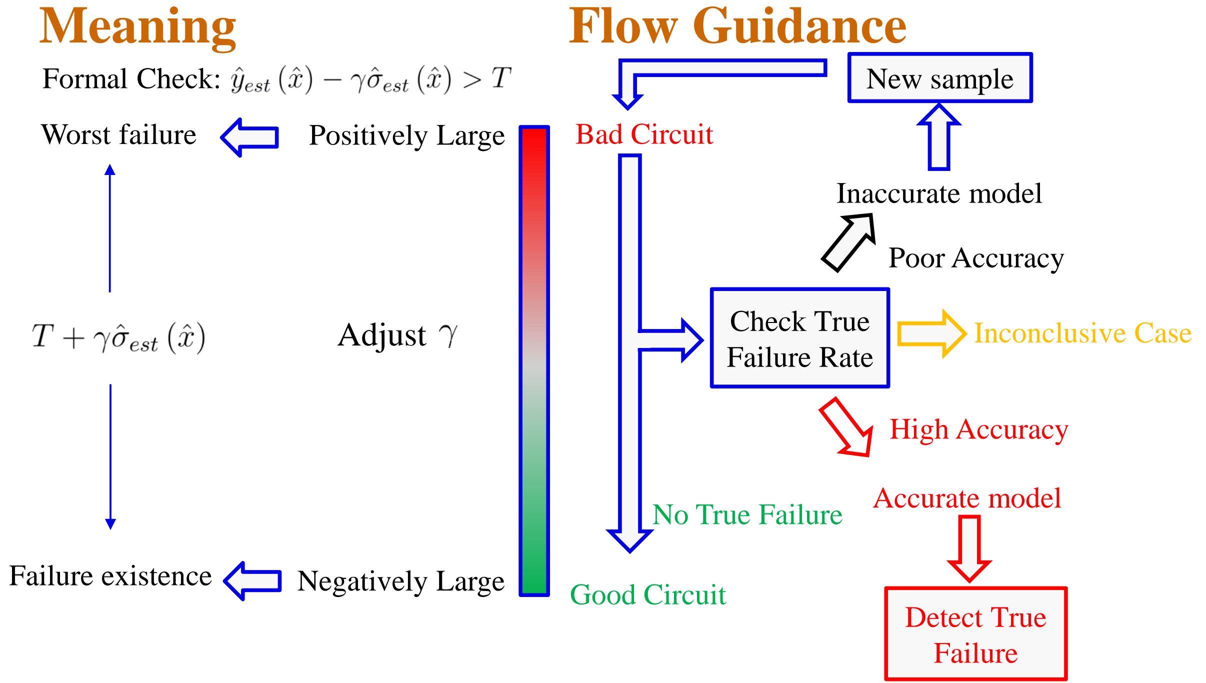


Figure 4.3: A detailed summary of the unified verification flow.

Table 4.1: Termination conditions for HFMV.

Case	# of Samples	γ	R
Pass	$= S_{max}$	$\leq \gamma_0$	$= 0\%$
Inconclusive	$= S_{max}$	$> \gamma_0$	$= 0\%$
Fail_1	$\leq S_{max}$	> 0	$> R_0$
Fail_2	$= S_{max}$	-	$> 0\% \ and \ < R_0$

4.4 Inconclusive Case

Let γ_0 be the γ value corresponding to P_0. That is to say $1 - \Phi(\gamma_0) = P_0$. If the HFMV flow ends up with a negative γ larger than γ_0 and no true failure has been identified in the verification process, an "Inconclusive" decision is drawn since the probability for the circuit to actually pass is $1 - \Phi(\gamma) < P_0$. A more detailed summary of the flow, that unifies these three cases, is shown in Fig. 4.3.

The three possible decisions and their corresponding termination conditions are summarized in Table 4.1. If no true failure is found and the training sample size has reached S_{max}, we draw a pass or inconclusive decision based on the confidence level $1 - \Phi(\gamma)$ for passing the circuit. Table 4.1 lists two different fail cases: "Fail_1" and "Fail_2". "Fail_1" is consistent with the flow illustrated in Fig. 4.2, where an accurate failure prediction model with a positive γ is produced at the very end. In the case of "Fail_2", some true failure points have been found by HFMV. However, no accurate failure prediction model can be learned within the limit of S_{max}.

5. THE STATISTICAL ML AlGORITHM IN HFMV

As illustrated in Section 4, formal check techniques play an important role in the flow of HFMV, and to make HFMV practical, an efficient formal check technique is needed to check or solve Formula 4.7 over the design space $\Omega_{\boldsymbol{x}}$.

It should be mentioned that our HFMV implementation is built on the model of RVFM [52] that is an extension to RVM [51], and the implementation takes advantage of unique properties of RVFM [52] to make formal checking efficient. Hence, before diving into mathematical details of the efficient formal check technique that is discussed later in Section 6, the frameworks of RVM [51] and RVFM [52] are first introduced in this section.

5.1 A Basic Kernel Machine in ML

A basic kernel machine for regression, e.g. SVM [47], is shown in Equation 5.1. In this equation, $\boldsymbol{x}$ stands for an input or a test sample, $\boldsymbol{x}_i$'s are selected training samples (e.g. support vectors in SVM), with a size of N. w_i's are weights assigned to the selected training samples (e.g. Lagrange multipliers in SVM [47]) and $K\left(\boldsymbol{x}, \boldsymbol{x}_i\right)$ is a kernel function which intuitively measures the similarity between $\boldsymbol{x}$ and $\boldsymbol{x_i}$.

To be more general, we add an additional error term into Equation 5.1, and rewrite it into Equation 5.2, in which $\boldsymbol{\Phi}$ is a $N \times N$ kernel matrix with $\boldsymbol{\Phi}\left(i, j\right) = K(\boldsymbol{x_i}, \boldsymbol{x_j})$, $\boldsymbol{t}$ is a symbol of target, and $\boldsymbol{e}$ is the error term.

$$y\left(\boldsymbol{x}; \boldsymbol{w}\right) = \sum_{i=1}^{N} w_i K\left(\boldsymbol{x}, \boldsymbol{x}_i\right). \tag{5.1}$$

$$\boldsymbol{t} = \boldsymbol{\Phi} \cdot \boldsymbol{w} + \boldsymbol{e}. \tag{5.2}$$

5.2 RVM Overview

RVM [51] uses Bayesian inference to determine Equation 5.2. For the training samples $\boldsymbol{x}_i$'s, assuming the error $\boldsymbol{e}$ are zero-mean Gaussian random values with a variance σ^2 and $\boldsymbol{x}_i$ is independent with each other, the following probability distribution can be inferred [51].

$$p\left(\boldsymbol{t}|\boldsymbol{w},\sigma^2\right) = \left(2\pi\sigma^2\right)^{-N/2} \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{t}-\boldsymbol{\Phi}\boldsymbol{w}\|^2\right). \tag{5.3}$$

Different from a normal Bayesian inference process, RVM [51] makes an encoding by making zero-mean Gaussian prior distributions over $\boldsymbol{w}$, with hyperparameters $\boldsymbol{\alpha}$ [51]:

$$p\left(\boldsymbol{w}|\boldsymbol{\alpha}\right) = \prod_{i=1}^{N} \mathcal{N}\left(0,\alpha_i^{-1}\right). \tag{5.4}$$

It's obvious that only when $\alpha_i < \infty$, w_i can have a value greater than zero, which allows corresponding training vector x_i contributes in Equation 5.1. This kind of training vector x_i with $\alpha_i < \infty$, is considered as a relevance vector [51].

Then the probabilistic prediction formula for a new sample t_* can be written as

$$\begin{aligned} p\left(t_*|\boldsymbol{t}\right) &= \int p\left(t_*|\boldsymbol{w},\boldsymbol{\alpha},\sigma^2\right) p\left(\boldsymbol{w},\boldsymbol{\alpha},\sigma^2|\boldsymbol{t}\right) d\boldsymbol{w}\, d\boldsymbol{\alpha}\, d\boldsymbol{\sigma^2} \\ &= \int p\left(t_*|\boldsymbol{w},\boldsymbol{\alpha},\sigma^2\right) p\left(\boldsymbol{w}|\boldsymbol{t},\boldsymbol{\alpha},\sigma^2\right) p\left(\boldsymbol{\alpha},\sigma^2|\boldsymbol{t}\right) d\boldsymbol{w}\, d\boldsymbol{\alpha}\, d\boldsymbol{\sigma^2}. \end{aligned} \tag{5.5}$$

Note that we can actually calculate the middle posterior $p\left(\boldsymbol{w}|\boldsymbol{t},\boldsymbol{\alpha},\sigma^2\right)$ in Equation 5.5 as follows:

$$p\left(\boldsymbol{w}|\boldsymbol{t},\boldsymbol{\alpha},\sigma^2\right) = \frac{p\left(\boldsymbol{t}|\boldsymbol{w},\sigma^2\right) p\left(\boldsymbol{w},\boldsymbol{\alpha}\right)}{p\left(\boldsymbol{t}|\boldsymbol{\alpha},\sigma^2\right)}, \tag{5.6}$$

$$p\left(\boldsymbol{t}|\boldsymbol{\alpha},\sigma^2\right) = \int p\left(\boldsymbol{t}|\boldsymbol{w},\sigma^2\right) p\left(\boldsymbol{w}|\boldsymbol{\alpha}\right) d\boldsymbol{w}. \tag{5.7}$$

Combining Equation 5.3, 5.4, 5.6, 5.7, we can get

$$p\left(\boldsymbol{w}|\boldsymbol{t},\boldsymbol{\alpha},\sigma^2\right) = (2\pi)^{-\frac{N+1}{2}}|\Sigma|^{-0.5}exp\left(-\frac{(\boldsymbol{w}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{w}-\boldsymbol{\mu})}{2}\right), \tag{5.8}$$

with the posterior covariance and mean of $\boldsymbol{w}$ to be [51]:

$$\boldsymbol{\Sigma} = (\sigma^{-2}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{A})^{-1}, \qquad \boldsymbol{A} = diag(\boldsymbol{\alpha}), \tag{5.9}$$

$$\boldsymbol{\mu} = \sigma^{-2}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{t}, \tag{5.10}$$

and

$$p\left(\boldsymbol{t}|\boldsymbol{\alpha},\sigma^2\right) = (2\pi)^{-\frac{N}{2}}|\Omega|^{-0.5}exp\left(-\frac{\boldsymbol{t}^T\Omega^{-1}\boldsymbol{t}}{2}\right), \tag{5.11}$$

$$\boldsymbol{\Omega} = \sigma^2\boldsymbol{I} + \boldsymbol{\Phi}\boldsymbol{A}^{-1}\boldsymbol{\Phi}^T. \tag{5.12}$$

Assuming $(\boldsymbol{\alpha}_{MP},\sigma^2_{MP}) = arg\max_{\boldsymbol{\alpha},\sigma^2} p\left(\boldsymbol{\alpha},\sigma^2|\boldsymbol{t}\right)$, then

$$\begin{aligned}
p\left(t_*|\boldsymbol{t}\right) &= \int p\left(t_*|\boldsymbol{w},\boldsymbol{\alpha},\sigma^2\right)p\left(\boldsymbol{w}|\boldsymbol{t},\boldsymbol{\alpha},\sigma^2\right)p\left(\boldsymbol{\alpha},\sigma^2|\boldsymbol{t}\right)d\boldsymbol{w}d\boldsymbol{\alpha}d\boldsymbol{\sigma^2} \\
&= \int p\left(t_*|\boldsymbol{w},\sigma^2\right)p\left(\boldsymbol{w}|\boldsymbol{t},\boldsymbol{\alpha},\sigma^2\right)p\left(\boldsymbol{\alpha},\sigma^2|\boldsymbol{t}\right)d\boldsymbol{w}d\boldsymbol{\alpha}d\boldsymbol{\sigma^2} \\
&\approx \int p\left(t_*|\boldsymbol{w},\sigma^2\right)p\left(\boldsymbol{w}|\boldsymbol{t},\boldsymbol{\alpha},\sigma^2\right)\delta(\boldsymbol{\alpha}-\boldsymbol{\alpha}_{MP})\delta(\sigma-\sigma_{MP})d\boldsymbol{w}d\boldsymbol{\alpha}d\boldsymbol{\sigma^2} \\
&= \int p\left(t_*|\boldsymbol{w},\sigma^2_{MP}\right)p\left(\boldsymbol{w}|\boldsymbol{t},\boldsymbol{\alpha_{MP}},\sigma^2_{MP}\right)d\boldsymbol{w} \\
&= \mathcal{N}\left(y_*,\sigma^2_*\right),
\end{aligned} \tag{5.13}$$

with

$$y_* = \boldsymbol{\mu}^T\boldsymbol{\phi}(\boldsymbol{x_*}), \tag{5.14}$$

$$\sigma^2_* = \sigma^2_{MP} + \boldsymbol{\phi}(\boldsymbol{x_*})^T\Sigma\boldsymbol{\phi}(\boldsymbol{x_*}). \tag{5.15}$$

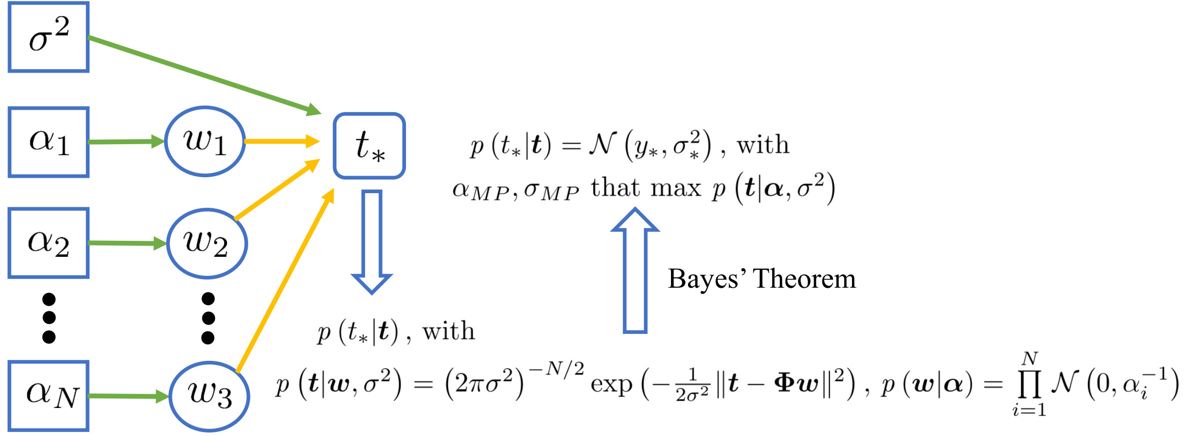


Figure 5.1: An overview of the Bayesian model of RVM.

, where $\boldsymbol{\phi}(\boldsymbol{x}_*)$ is a vector of size N with the i-th entry defined by $\boldsymbol{\phi}(\boldsymbol{x}_*)(i) = K(\boldsymbol{x}_*, \boldsymbol{x}_i)$ [51].

The problem now is how to calculate the value of $\boldsymbol{\alpha}_{MP}$ and σ_{MP}^2. Since $p\left(\boldsymbol{\alpha}, \sigma^2|\boldsymbol{t}\right) \propto p\left(\boldsymbol{t}|\boldsymbol{\alpha}, \sigma^2\right) p(\boldsymbol{\alpha})p(\sigma^2)$, using Maximum a Posteriori (MAP) estimation, we can get

$$(\boldsymbol{\alpha}_{MP}, \sigma_{MP}^2) = arg \max_{\boldsymbol{\alpha}, \sigma^2} p\left(\boldsymbol{t}|\boldsymbol{\alpha}, \sigma^2\right). \tag{5.16}$$

Making the differentiation of Equation 5.11 equal to zero gives [51]:

$$\alpha_i^{new} = \frac{1 - \alpha_i \Sigma_{ii}}{\mu_i^2}, \tag{5.17}$$

$$(\sigma^2)^{new} = \frac{\|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2}{N - \sum_{i=1}^{N}(1 - \alpha_i \Sigma_{ii})}, \tag{5.18}$$

in which Σ_{ii} means the i-th diagonal element of Σ. These two equations can be used to calculate numerical solutions for $\boldsymbol{\alpha}_{MP}$ and σ_{MP}^2 [51].

A brief overview of the Bayesian model of RVM is shown in Fig. 5.1.

5.3 RVFM Overview

RVFM is inspired by the RVM-based feature selection technique proposed in [56], which defines “feature vectors" $\boldsymbol{f_i} = (\boldsymbol{x}_1(i), \boldsymbol{x}_2(i), ..., \boldsymbol{x}_F(i))$ with $i \in [1, F]$, where F is the number of selected features. Generally, they exchange the roles of samples and features and want to solve the following feature weighting model, similar to Model 5.2, with a new feature kernel $K^{'}$:

$$\boldsymbol{t}^{'} = \boldsymbol{\Phi}^{'} \cdot \boldsymbol{v} + \boldsymbol{e}, \tag{5.19}$$

where $\Phi^{'}$ is a new $F \times F$ design matrix with $\Phi^{'}(i, j) = K^{'}(\boldsymbol{f}_i, \boldsymbol{f}_j)$, and $\boldsymbol{v}$ are the weights for feature vectors [56].

After embedding this technique into the flow of RVM, a new model is shown below [52]:

$$\boldsymbol{t} = \boldsymbol{\Phi_{wv}} \cdot \boldsymbol{w} \otimes \boldsymbol{v} + \boldsymbol{e}. \tag{5.20}$$

In Model 5.20, $\boldsymbol{\Phi_{wv}}$ is a $N \times (NF)$ design matrix defined by $\boldsymbol{\Phi_{wv}}(i, (j-1)F + k) = K_k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ that is the kernel function of the i-th and j-th samples on the k-th feature, e.g. $K_k(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp(-\gamma_k(\boldsymbol{x_{ik}} - \boldsymbol{x_{jk}})^2)$ if RBF kernel is adopted [52], with $i, j \in [1, N]$ and $k \in [1, F]$. In addition, $\boldsymbol{w} \otimes \boldsymbol{v}$ is a tensor product of these two vectors and is a $(NF) \times 1$ vector defined by $(\boldsymbol{w} \otimes \boldsymbol{v})((j-1)F + k) = w_j v_k$ [52].

Actually, Model 5.20 can be transferred into Model 5.2 by moving $\boldsymbol{v}$ from $\boldsymbol{w} \otimes \boldsymbol{v}$ into the design matrix $\boldsymbol{\Phi_{wv}}$, forming a new design matrix whose definition is $\boldsymbol{\Phi_w}(i, j) = \sum_{k=1}^{F} v_k K_k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ [52]. Similarly, by moving w into $\boldsymbol{\Phi_{wv}}$, Model 5.19 can be derived, with a new design matrix defined by $\boldsymbol{\Phi_v}(i, k) = \sum_{j=1}^{N} w_j K_k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ [52].

Inspired by this property of Model 5.20, an efficient training method [52] was introduced, which is shown in Fig. 5.2. In each iteration, by either fixing $\boldsymbol{\alpha}$ and moving $\boldsymbol{w}$ into $\boldsymbol{\Phi_{wv}}$, or fixing $\boldsymbol{\beta}$ and moving $\boldsymbol{v}$ into $\boldsymbol{\Phi_{wv}}$, we can reduce the model from (5.20) to either

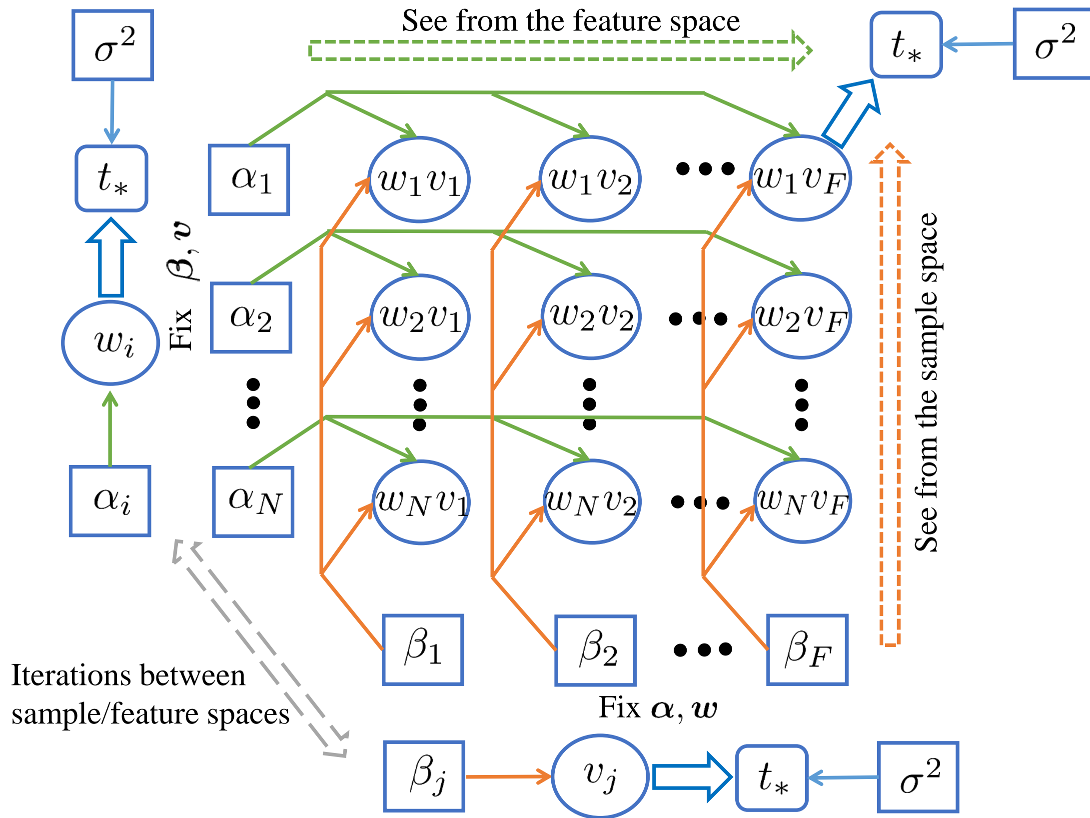


Figure 5.2: An efficient RFVM training method.

(5.19) or (5.2). That's to say for each iteration we are only training for a RVM model either on the sample space or the feature space.

6. EFFICIENT SMT-BASED FORMAL CHECKING

We perform formal checking in HFMV by employing SMT solvers. Generally, SMT solvers take a logic formula f (Equation 4.7 in our case) over a background theory t (the design space $\Omega_{\boldsymbol{x}}$) as an input and output either a SAT answer with a model (a solved point in $\Omega_{\boldsymbol{x}}$) if f is satisfied or an UNSAT answer if f is unsatisfied.

The general flow for finding solutions of Equation 4.7 in $\Omega_{\boldsymbol{x}}$ using a SMT solver is shown in Figure 6.1, with a key idea that adding artificial constraints (i.e. small squares around solved points in Figure 6.1) to mask previous solved points until the SMT solver returns an UNSAT answer or is terminated by the user.

6.1 An Unsolvable SMT Problem

By inserting Equation 3.1 and Equation 3.2 into Equation 4.7, we can easily write the logic formula f as

$$\boldsymbol{w}^T \boldsymbol{\phi}\left(\boldsymbol{x}\right) - \gamma\sqrt{\sigma^2 + \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\Sigma} \boldsymbol{\phi}(\boldsymbol{x})} > T, \tag{6.1}$$

which can be solved formally by SMT solvers. However, it's obvious that this leads to a very complex formula in terms of $\boldsymbol{x}$, particular when the dimension of $\boldsymbol{x}$ is relatively high and some popular nonlinear kernels, e.g. Radial Basis Function Kernel (RBF) which can be represented as $K(\boldsymbol{x_1}, \boldsymbol{x_2}) = exp(-\gamma_k \|\boldsymbol{x_1} - \boldsymbol{x_2}\|^2)$, are chosen for ML methods to guarantee regression performance.

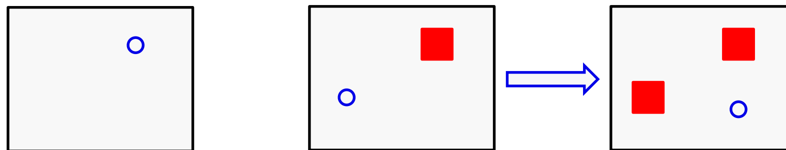


Figure 6.1: A masking methodology for SMT solvers.

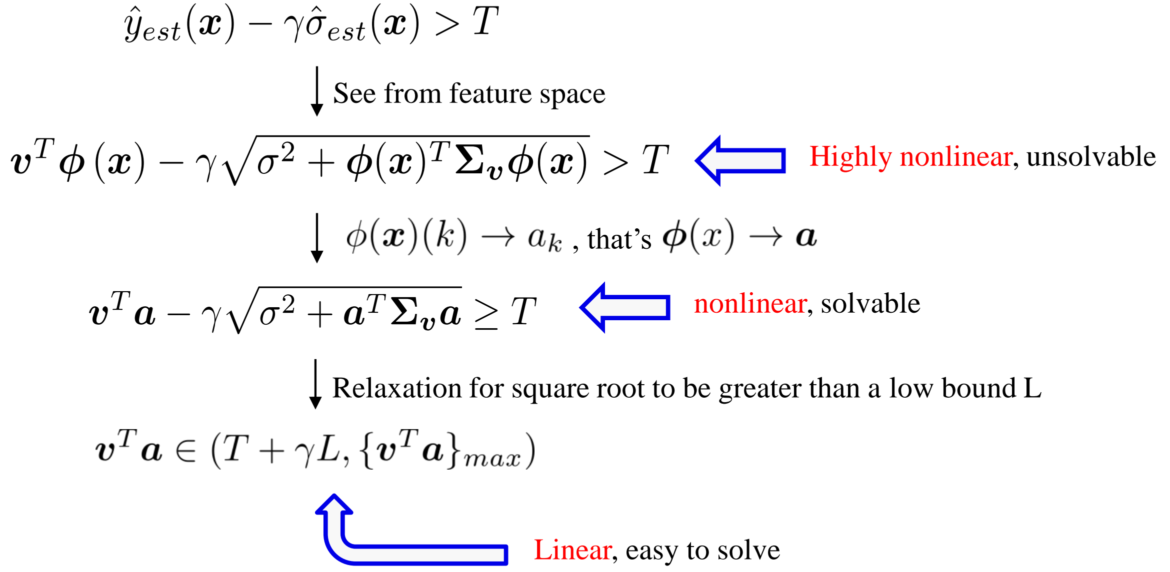


Figure 6.2: The simplification of SMT solution.

My experimental studies have shown that it is infeasible to directly solve such a formula using a nonlinear SMT solver, like iSAT3 [41] (e.g. running for weeks without any result). Therefore, a key bottleneck for HFMV is the significant time taken by formal checking, and an efficient formal check method is needed to make HFMV practical.

6.2 An Efficient SMT Method Based on the RVFM Framework

We develop an efficient SMT method by exploiting unique properties of the RVFM model. The general idea of this solution is to transfer the highly nonlinear and unsolvable formula (i.e. Formula 6.1) into a linear and easy-to-solve formula through variable mapping and approximation, which is shown in Fig. 6.2.

6.2.1 A Solvable SMT Method

Seeing from the feature space in Fig. 5.2, by moving $\boldsymbol{v}$ into the design matrix, we can get the following "decomposed" kernel:

$$K_{decomposed}\left(\boldsymbol{x}, \boldsymbol{x}_j\right) = \sum_{i=1}^{N} w_i \cdot K_k\left(\boldsymbol{x}, \boldsymbol{x}_i\right), \tag{6.2}$$

in which, $k \in [1, F]$, $K_k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = K\left(\boldsymbol{x}_i\left(k\right), \boldsymbol{x}_j\left(k\right)\right)$ is a scalar kernel function involving the k-th feature of input vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and w_i is the weight for $K_k\left(\boldsymbol{x}, \boldsymbol{x}_i\right)$.

Then the expected prediction model from RVFM is:

$$\begin{aligned} \hat{y}_{est}(\boldsymbol{x}) = \boldsymbol{v}^T \boldsymbol{\phi}\left(\boldsymbol{x}\right), \qquad \hat{\sigma}_{est}^2(\boldsymbol{x}) = \sigma^2 + \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\Sigma}_{\boldsymbol{v}} \boldsymbol{\phi}(\boldsymbol{x}), \\ with \quad \boldsymbol{\phi}(\boldsymbol{x})\left(k\right) = \sum_{i=1}^{N} w_i \cdot K_k\left(\boldsymbol{x}, \boldsymbol{x}_i\right), k \in [1, F], \end{aligned} \tag{6.3}$$

where the $\boldsymbol{v}$ and Σ_v are the posterior mean and covariance matrix of features' weights, respectively.

The key observation is that in Equation 6.3, the k-th component $\boldsymbol{\phi}(\boldsymbol{x})\left(k\right) = \sum_{i=1}^{N} w_i \cdot K_k\left(\boldsymbol{x}, \boldsymbol{x}_i\right)$ of $\boldsymbol{\phi}(\boldsymbol{x})$ only depends on the k-th feature of input vector $\boldsymbol{x}$ and sample vectors $\boldsymbol{x}_i$'s. Hence $\boldsymbol{\phi}(\boldsymbol{x})\left(k\right)$ is only a symbolic function of $\boldsymbol{x}\left(k\right)$ and can be written in the form:

$$\boldsymbol{\phi}(\boldsymbol{x})\left(k\right) \equiv a\left(\boldsymbol{x}\left(k\right)\right) \equiv a_k, k \in \left[1, F\right]. \tag{6.4}$$

For example, a is a sum of multiple scalar exponential terms with respect to $\boldsymbol{x}\left(k\right)$ when K_k is chosen to be RBF. Here, we can solve a trivial one-dimension optimization problem to bound each a_k with respect to the range of $\boldsymbol{x}\left(k\right)$:

$$\underline{a}_k \leq a_k \leq \overline{a}_k, \qquad with \quad \underline{a}_k = \min a\left(\boldsymbol{x}\left(k\right)\right), \; \overline{a}_k = \max a\left(\boldsymbol{x}\left(k\right)\right). \tag{6.5}$$

Since all a_k's are independent with each other, Formula 6.1 can be converted into an equivalent form which is only symbolic with respect to $\boldsymbol{a}$:

$$\begin{gathered} \boldsymbol{v}^T\boldsymbol{a} - \gamma\sqrt{\sigma^2 + \boldsymbol{a}^T\boldsymbol{\Sigma}_{\boldsymbol{v}}\boldsymbol{a}} \geq T, \\ with \quad \boldsymbol{a} = [a_1, ..., a_F]^T, \ \underline{a}_k \leq a_k \leq \overline{a}_k, \ k \in [1, F]. \end{gathered} \tag{6.6}$$

If SMT solvers with the input of Formula 6.6 returns an UNSAT answer, then Formula 3.5 is satisfied over the parameter space $\Omega_{\boldsymbol{x}}$.

Note that Formula 6.6 is much simpler than Formula 6.1, which makes it solvable in practice by a nonlinear SMT solver, like iSAT3 [43].

6.2.2 An Efficient SMT Method

Although the method introduced in Section 6.2.1 is solvable in practice, the nonlinearity brought by the variance part $\sqrt{\sigma^2 + \boldsymbol{a}^T\boldsymbol{\Sigma}_{\boldsymbol{v}}\boldsymbol{a}}$ in Formula 6.6 is still unfriendly to SMT solvers, and sometimes it's very expensive to solve a point. Hence, another SMT approach is introduced based on Formula 6.6, and the key idea is to make approximation for the variance part and relax Formula 6.6 into a linear formula.

It's not very hard to find a loose bound for $\sqrt{\sigma^2 + \boldsymbol{a}^T\boldsymbol{\Sigma}_{\boldsymbol{v}}\boldsymbol{a}}$ through mathematic derivations based on the ranges of $\boldsymbol{a}$. Here we take advantage of the fact that $\boldsymbol{\Sigma}_{\boldsymbol{v}}$ is a symmetric matrix, by doing Singular Value Decomposition (SVD), $\boldsymbol{\Sigma}_{\boldsymbol{v}}$ can be decomposed into the following form:

$$\boldsymbol{\Sigma}_{\boldsymbol{v}} = \boldsymbol{U}\boldsymbol{\Sigma}'\boldsymbol{V}^T = \boldsymbol{U}\boldsymbol{\Sigma}'\boldsymbol{U}^T, \tag{6.7}$$

in which U is a $F \times F$ matrix, and Σ' is a $F \times F$ diagonal matrix. With Equation 6.7, we

can get

$$\begin{aligned} \boldsymbol{a}^T\boldsymbol{\Sigma_v}\boldsymbol{a} &= (\boldsymbol{a}\boldsymbol{U}^T)^T\boldsymbol{\Sigma}'(\boldsymbol{a}\boldsymbol{U}^T) \\ &= \sum_{i=1}^{F}\Sigma'_{ii}(\boldsymbol{a}\boldsymbol{U}_i)^2 \\ &= \sum_{i=1}^{F}\Sigma'_{ii}(\sum_{j=1}^{F}a_jU_{ij})^2, \end{aligned} \tag{6.8}$$

where Σ'_{ii} is the i-th diagonal element of $\boldsymbol{\Sigma}'$. Knowing the range of each a_j, it's trivial to derive a low bound for $\boldsymbol{a}^T\boldsymbol{\Sigma_v}\boldsymbol{a}$ and hence for $\sqrt{\sigma^2+\boldsymbol{a}^T\boldsymbol{\Sigma_v}\boldsymbol{a}}$.

Assuming that $\sqrt{\sigma^2+\boldsymbol{a}^T\boldsymbol{\Sigma_v}\boldsymbol{a}} > L$, then Formula 6.6 can be transformed into the following linear form:

$$\boldsymbol{v}^T\boldsymbol{a} \in (T+\gamma L, \{\boldsymbol{v}^T\boldsymbol{a}\}_{max}), \tag{6.9}$$

in which $\{\boldsymbol{v}^T\boldsymbol{a}\}_{max}$ is the max possible value that can be easily calculated by the ranges of $\boldsymbol{a}$.

To prevent the clustering of solved points as much as possible, we rewrite Formula 6.6 into $\boldsymbol{v}^T\boldsymbol{a} = T'$, T' $is\ a\ random\ value\ in$ $(T+\gamma L, \{\boldsymbol{v}^T\boldsymbol{a}\}_{max})$ and flow the implementation shown in Fig. 6.3.

Note that solutions solved from Fig 6.3 are not necessary to be true solutions for the original formula, Formula 6.1. Hence a further verification step is needed to be done on Formula 6.1 to filter false solutions.

6.3 Speed Comparison Between Two SMT Methods

To give you an example of the speedup due to the linear approximation in Formula 6.9, we run a simple test based on a 51-dimension task and compare the CPU time to solve 300 true solutions for Formula 6.1 using the solvable SMT method (illustrated in Section 6.2.1) and the final efficient SMT method (illustrated in Section 6.2.2).

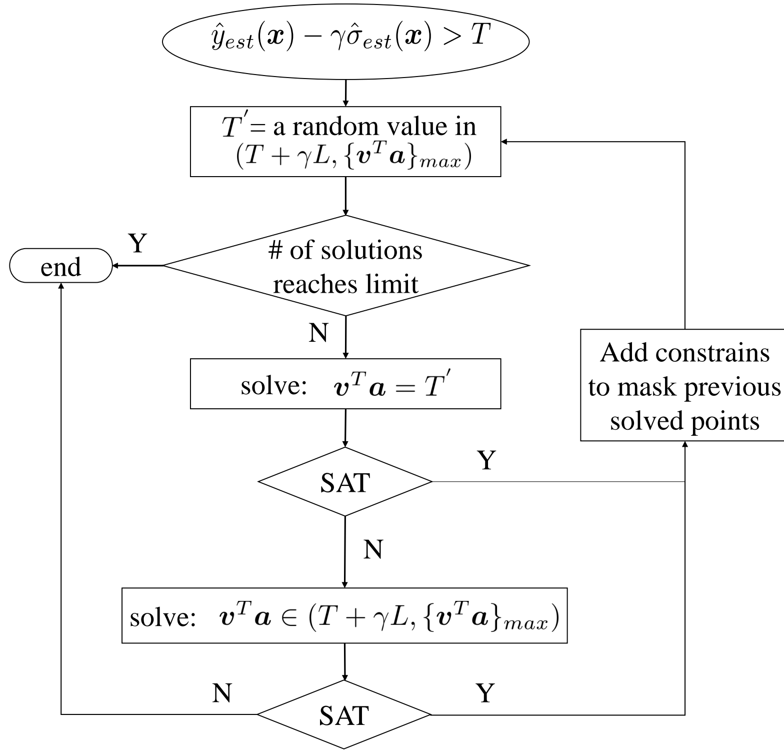


Figure 6.3: An efficient SMT flow.

Table 6.1: Speed comparison between two SMT methods.

Method	SMT Solver Used	CPU Time
The solvable SMT method	iSAT3 [43]	44h57m21s
The efficient SMT method	Z3 [42]	46.91s

The test is done on a workstation with 2.2GHz AMD Opteron 6174 Processors, and the results are shown in Table 6.1, which indicates that the efficient SMT method provides a 3-order-of-magnitude speedup.

6.4 Deductions for General Statistical ML Models

Although the above work is done utilizing unique properties of RFVM, what should be aware is that it's also possible to adopt this method to a general statistic ML model once

the decomposed kernel shown in Equation 6.2 is introduced. For a general statistical ML model without feature selection properties, we can design a similar "decomposed" kernel:

$$K_{decomposed}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \sum_{k=1}^{F} K_k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right). \tag{6.10}$$

In this case, the weight assigned to each feature is fixed to equal one. Let's take the RVM model [51] for illustration purpose, the expected prediction mean and variance can be then expressed as the following equations.

$$\begin{gathered} \hat{y}_{est}(\boldsymbol{x}) = \boldsymbol{u}^T \boldsymbol{\phi}\left(\boldsymbol{x}\right), \qquad \hat{\sigma}^2_{est}(\boldsymbol{x}) = \sigma^2 + \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\Sigma} \boldsymbol{\phi}(\boldsymbol{x}), \\ with \quad \boldsymbol{\phi}(\boldsymbol{x})\left(i\right) = \sum_{k=1}^{F} K_k\left(\boldsymbol{x}, \boldsymbol{x}_i\right), i \in [1, N]. \end{gathered} \tag{6.11}$$

At the first step towards deductions, we notice that the technique illustrated in Section 6.2.1 can still be applied to the mean part:

$$\begin{aligned} \hat{y}_{est}(\boldsymbol{x}) &= \boldsymbol{u}^T \boldsymbol{\phi}\left(\boldsymbol{x}\right) \\ &= \sum_{i=1}^{N} u_i \boldsymbol{\phi}(\boldsymbol{x})\left(i\right) \\ &= \sum_{i=1}^{N} u_i \sum_{k=1}^{F} K_k\left(\boldsymbol{x}, \boldsymbol{x}_i\right) \\ &= \sum_{k=1}^{F} \sum_{i=1}^{N} u_i K_k\left(\boldsymbol{x}, \boldsymbol{x}_i\right) \\ &= \sum_{k=1}^{F} a_k^{'}, \end{aligned} \tag{6.12}$$

in which $a_k^{'}$ is similar to the a_k defined in Equation 6.4.

As to the variance part, we can still utilize the property of Σ being symmetric. Assum-

ing that $\mathbf{\Sigma} = \boldsymbol{U}\mathbf{\Sigma}^{'}\boldsymbol{U}^{T}$, then

$$\begin{aligned}
\boldsymbol{\phi}(\boldsymbol{x})^{T}\mathbf{\Sigma}\boldsymbol{\phi}(\boldsymbol{x}) &= (\boldsymbol{\phi}(\boldsymbol{x})\boldsymbol{U}^{T})^{T}\mathbf{\Sigma}^{'}(\boldsymbol{\phi}(\boldsymbol{x})\boldsymbol{U}^{T}) \\
&= \sum_{i=1}^{N}\Sigma_{ii}^{'}(\boldsymbol{\phi}(\boldsymbol{x})U_{i})^{2} \\
&= \sum_{i=1}^{N}\Sigma_{ii}^{'}(\sum_{j=1}^{N}U_{ij}\boldsymbol{\phi}(\boldsymbol{x})\,(j))^{2} \\
&= \sum_{i=1}^{N}\Sigma_{ii}^{'}(\sum_{j=1}^{N}U_{ij}\sum_{k=1}^{F}K_{k}\left(\boldsymbol{x},\boldsymbol{x}_{j}\right))^{2} \\
&= \sum_{i=1}^{N}\Sigma_{ii}^{'}(\sum_{k=1}^{F}\sum_{j=1}^{N}U_{ij}K_{k}\left(\boldsymbol{x},\boldsymbol{x}_{j}\right))^{2}.
\end{aligned} \tag{6.13}$$

We can observe that $\sum_{j=1}^{N}U_{ij}K_{k}\left(\boldsymbol{x},\boldsymbol{x}_{j}\right)$ only depends on the k-th feature of input vector $\boldsymbol{x}$ and samples $\boldsymbol{x}_{j}$'s. Similarly, denoting $\sum_{j=1}^{N}U_{ij}K_{k}\left(\boldsymbol{x},\boldsymbol{x}_{j}\right) \equiv b_{i}\left(\boldsymbol{x}\left(k\right)\right) \equiv b_{ik},\ i,k \in [1,N]$, we can get

$$\begin{aligned}
\boldsymbol{\phi}(\boldsymbol{x})^{T}\mathbf{\Sigma}\boldsymbol{\phi}(\boldsymbol{x}) &= \sum_{i=1}^{N}\Sigma_{ii}^{'}(\sum_{k=1}^{F}\sum_{j=1}^{N}U_{ij}K_{k}\left(\boldsymbol{x},\boldsymbol{x}_{j}\right))^{2} \\
&= \sum_{i=1}^{N}\Sigma_{ii}^{'}(\sum_{k=1}^{F}b_{ik})^{2}.
\end{aligned} \tag{6.14}$$

Following the same steps in Section 6.2.2, we can come to a linear formula similar to Formula 6.9, which could be fast solved by some linear SMT solvers (e.g. Z3 [42]).

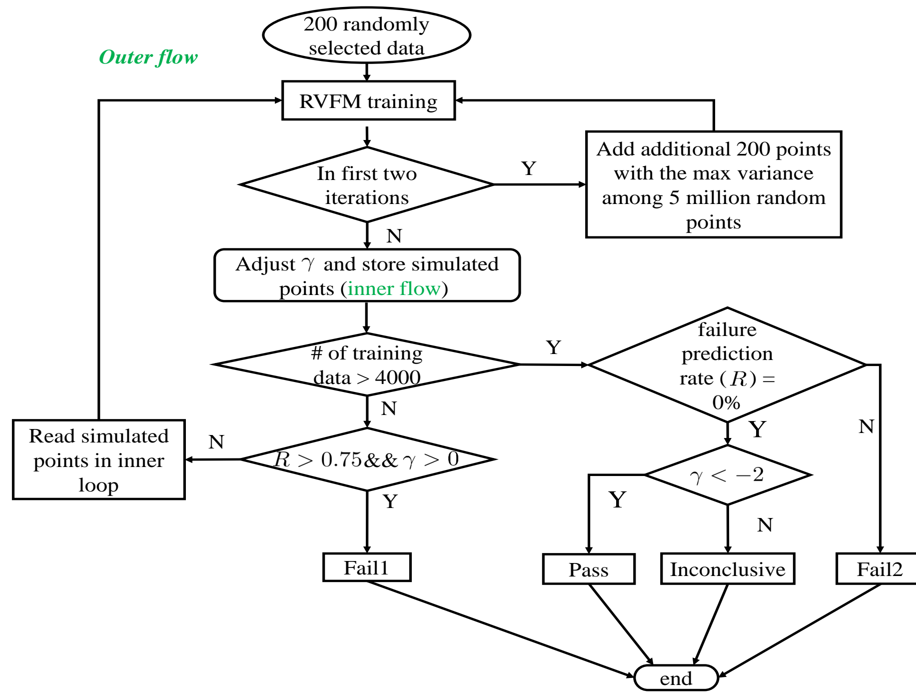
7. EXPERIMENTAL RESULTS

To demonstrate the superiority of proposed HFMV approach, we test it on three analog circuits (i.e. a differential amplifier, a DC-DC converter [53], and a LDO [54]), and compare its performance with the traditional Monte Carlo method. We randomly select 200 simulation samples, following the uniform distribution on each dimension, as the initial training data for HFMV. Much larger number of random samples based on the same sampling scheme is applied to the Monte Carlo method. All simulations for three circuits are done with a commercial 90nm CMOS technology design kit. For the circuit simulators, HSPICE is used for the differential amplifier and the LDO [54] simulations, and Spectre is used for the DC-DC converter [53] simulation.

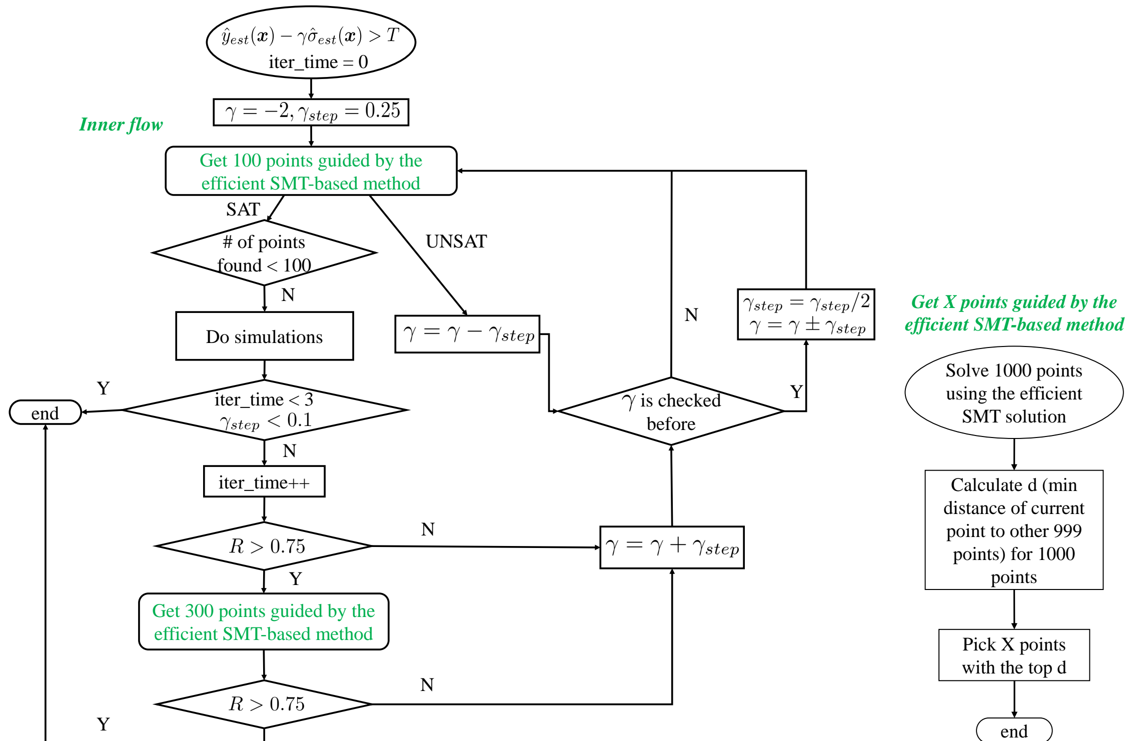
Detailed HFMV flows used in our experiments are shown in Fig. 7.1. In Fig. 7.1, (a) is a top-level flow with $\gamma_0 = -2,\ R_0 = 75\%$ (see Table 4.1), and the maximum sample size $S_{max} = 4000$, and (b) is an inner flow, which performs formal checks, adjusts γ, and runs simulations. These two flows are consistent with Fig. 4.3. Fig. 7.1 (c) is a detailed flow about how to gather resampling points using the knowledge in Section 6. To expand resampled points to the entire space as much as possible, we only pick the top X points, with the greatest minimal distance to other points, from total 1000 points solved following Fig. 6.3.

In addition, the RVFM model is trained employing the 5-folder cross-validation method using RBF kernel, i.e. $K(\boldsymbol{x_1}, \boldsymbol{x_2}) = exp(-\gamma_k \|\boldsymbol{x_1} - \boldsymbol{x_2}\|^2)$, and the range for parameter γ_k in RBF kernel is from 0.1 to 0.9 with a step of 0.1 and from 1 to 6 with a step of 1.

Our HFMV flow is written in C++ and compiled by g++ 4.8.5, and runs on a workstation with 2.2GHz AMD Opteron 6174 Processors.



(a)



(b) (c)

Figure 7.1: The detailed HFMV flow in experiments: (a) Top-level outer flow (b) Inner flow to do formal checks, adjust γ, and run simulations, and (c) Resampling flow.

7.1 A Differential Amplifier

We first investigate the process variations of a one-stage differential amplifier circuit as shown in Fig. 7.2. Three kinds of variations, i.e. channel length, threshold voltage, and thickness of gate oxide, are modeled for each transistor at the SPICE level. The total number of process parameters is 15, which is the dimensionality of the parameter space. Three performances including gain-bandwidth product (GBW), gain, and common mode rejection ratio (CMRR) are verified against the corresponding specifications. For all these three parameters, the smaller the performance value is, the worse the actual performance is.

Figure 7.2: A differential amplifier.

Performance comparison between HFMV and the Monte Carlo method with **600,000** random samples with respect to failure identification is shown in Fig. 7.4(a). In this figure, the worst-case performance found by each method is normalized with respect to the corresponding specified target, and each specified target corresponds to the zero level in the y axis. A positive value in the y axis means that the found performance value is greater than the target, otherwise, it is less. It can be seen that HFMV can find failures that are significantly worse than the targets, confirming the differential amplifier as “Fail". In

comparison, even with 600,000 samples the Monte Carlo method cannot find any failure or even any point close to each target, producing misleading verification conclusions.

7.2 A DC-DC Converter

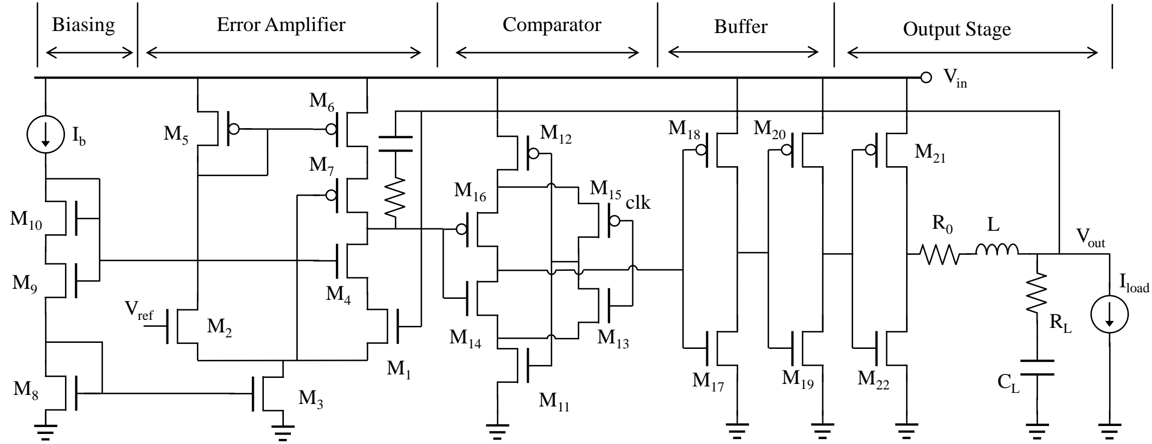


Figure 7.3: A DC-DC converter.

The schematic of DC-DC converter [53] is shown in Fig. 7.3. The DC-DC converter contains 22 transistors, and for each transistor two device-level variations (channel length and width) are considered, leading to a 44-dimensional parameter space. Output accuracy, overshoot, ripple size, and power efficiency are the performances to be verified.

Similarly, we compare HFMV to the Monte Carlo method with **45,000** random samples in Fig. 7.4(b). Note that for the last circuit performance, power efficiency, the smaller the value is, the worse the actual performance is. The opposite is true for the other three performances (output accuracy, overshoot, and ripple size). Again, the Monte Carlo method produces misleading results for each verification task while HFMV is able to find performances worse than the targeted specifications.

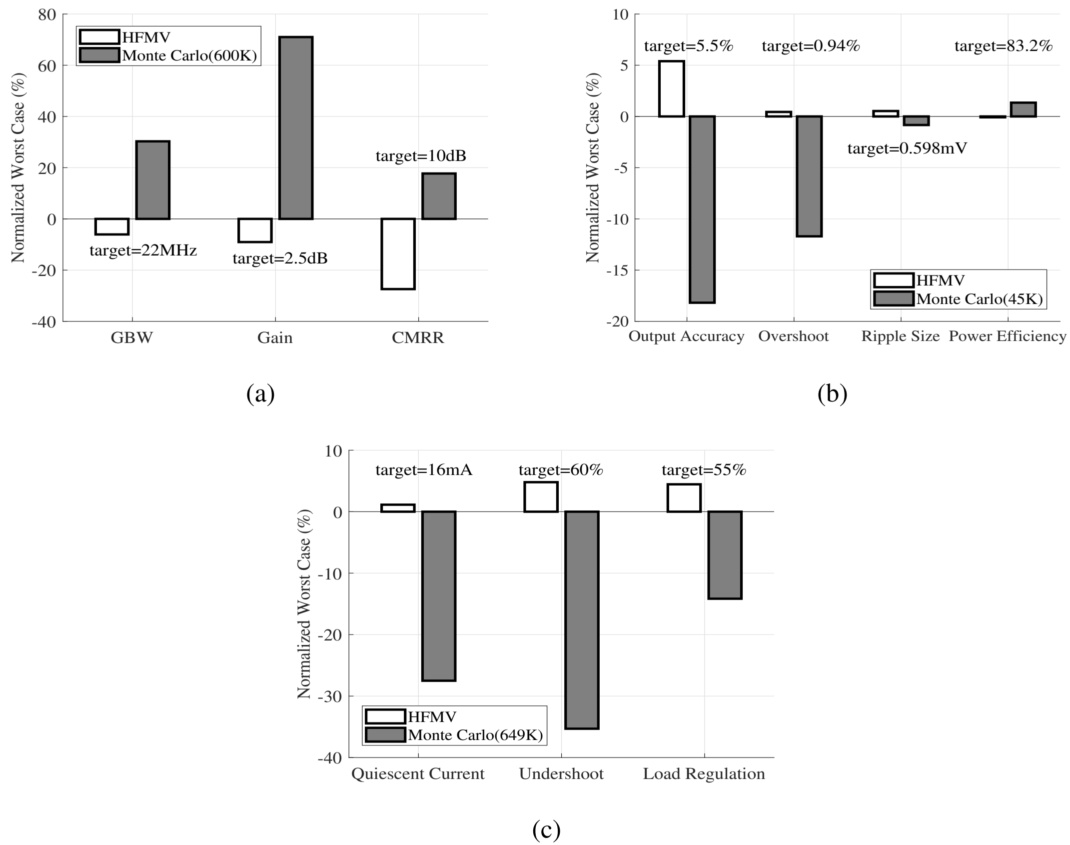


Figure 7.4: Comparison of HFMV and Monte Carlo Simulation on failure identification: (a) differential amplifier, (b) DC-DC converter, and (c) LDO. The worst-case performance found by each method is normalized with respect to the specified target.

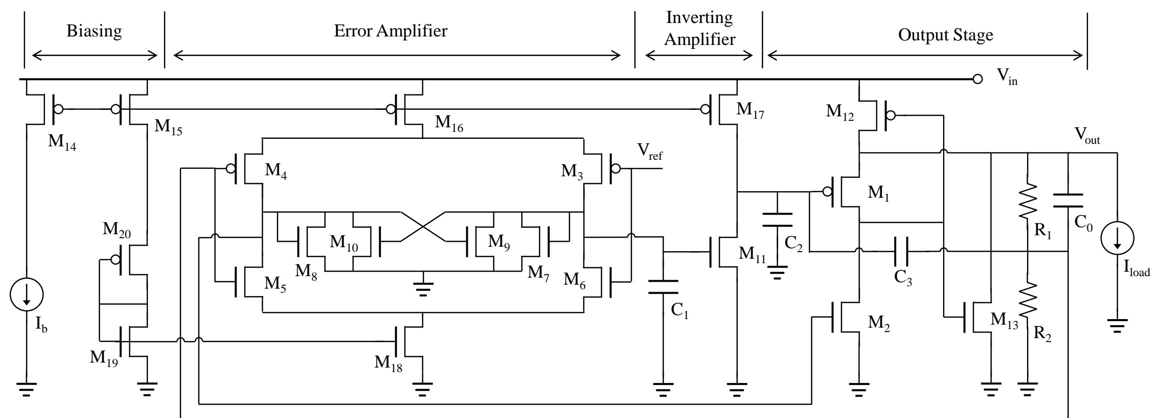


Figure 7.5: LDO.

7.3 A Low-dropout Regulator

We further investigate the low-dropout regulator (LDO) design proposed in [54]. Variations in channel length, threshold voltage, and thickness of gate oxide are considered for all 20 transistors in this design, which results in a 60-dimensional parameter space. Three circuit performances, quiescent current, undershoot, and load regulation, are the specifications of verification. For these three performances, the greater the value is, the worse the actual performance is. About **649,000** random simulation samples are used in the Monte Carlo method, which is compared to HMFV in Fig. 7.4(c). In all cases, HFMV finds performance values worse than the specified targets. The Monte Carlo method, however, is significantly optimistic, again producing misleading answers for verification.

Detail results of HFMV for all three circuits are shown in Table 7.1.

Table 7.1: HFMV effectiveness.

Circuit Specifications		T	N	R
differential amplifier (15 dim.)	GBW	22MHz	1775	100%
	Gain	2.5dB	3595	100%
	CMRR	10dB	1600	100%
DC-DC converter (44 dim.)	Output Accuracy	5.5%	1400	95.0%
	Overshoot	0.94%	1300	99.3%
	Ripple Size	0.598mV	1400	100%
	Power Efficiency	83.2%	1400	100%
LDO (60 dim.)	Quiescent Current	16mA	2496	88.0%
	Undershoot	60%	2200	92.3%
	Load Regulation	55%	2299	82.7%

Notes: The "T" column shows the targets we set for different circuit specifications. The "N" column describes the total number of simulation samples used in the HFMV flow. For each specification in the table, all three circuits are regarded as "Fail_1" and HFMV produces a failure prediction model in each case. The "R" column stands for failure prediction accuracy, the ratio between true failures and potential failure points found by the SMT solver based on 300 predictions.

7.4 Comparison on Runtime

Comparison of CPU time between the two methods is shown in Fig. 7.6. The CPU time of HFMV, the total verification time for all specifications in one circuit, is up to about 11-time smaller than that of the Monte Carlo method for all test cases.

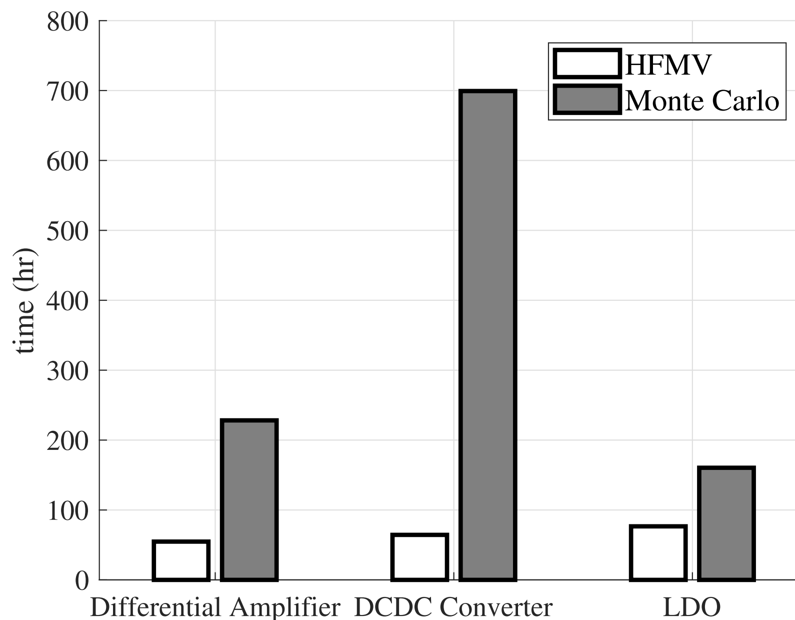


Figure 7.6: Comparison of CPU time between HFMV and Monte Carlo simulations.

It shall be noted that the significantly better efficiency of HFMV is achieved with the fact that the Monte Carlo methods cannot find any failure even by spending hundreds of hours of simulation time. HFMV not only finds true failures but also produces failure models that can efficiently predict lots of failures with a good accuracy as shown in Table 7.1. Collectively, Table 7.1, Fig. 7.4 and Fig. 7.6 demonstrate the superior efficiency and power of HMFV, which utilizes a limited amount of simulation data to achieve highly-accurate

failure prediction in high-dimensional parameter spaces.

7.5 Verification with Varying Specified Targets

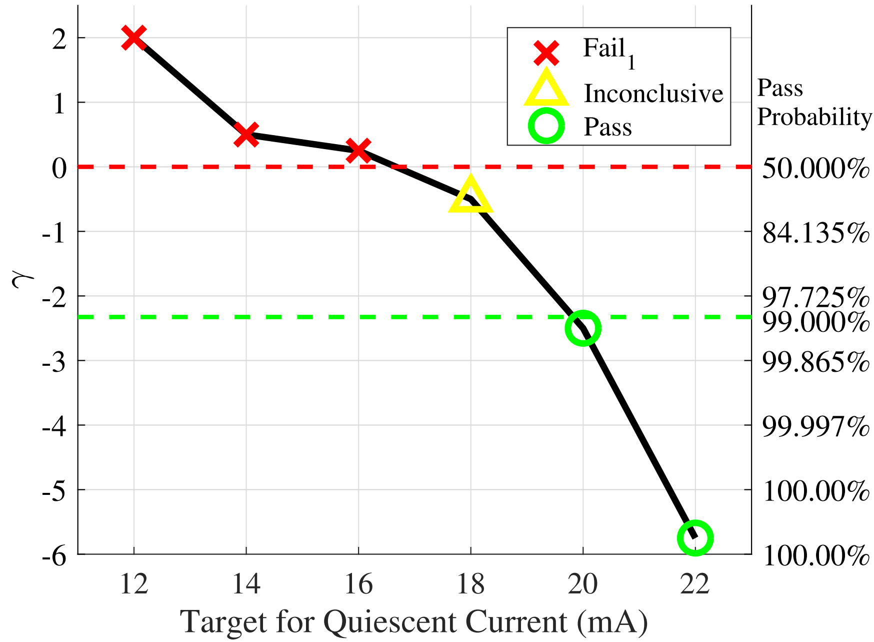


Figure 7.7: HFMV verification of the quiescent current of the LDO.

To provide more insights on how HFMV works, we relax the target for the quiescent current of the LDO by increasing it from 12mA to 22mA, making meeting the specified requirement increasingly easier. The results of HFMV verification are shown in Fig. 7.7, where the values of the internal confidence control parameter γ are also shown. In this series of verification tasks, S_{max} is set to 2,800, and γ_0 is set to -2.3264 (a confidence level of at least 99%). The HFMV flow is terminated when hitting S_{max} if the verification process is not ended earlier with an accurate failure prediction model when the LDO is deemed as "Fail".

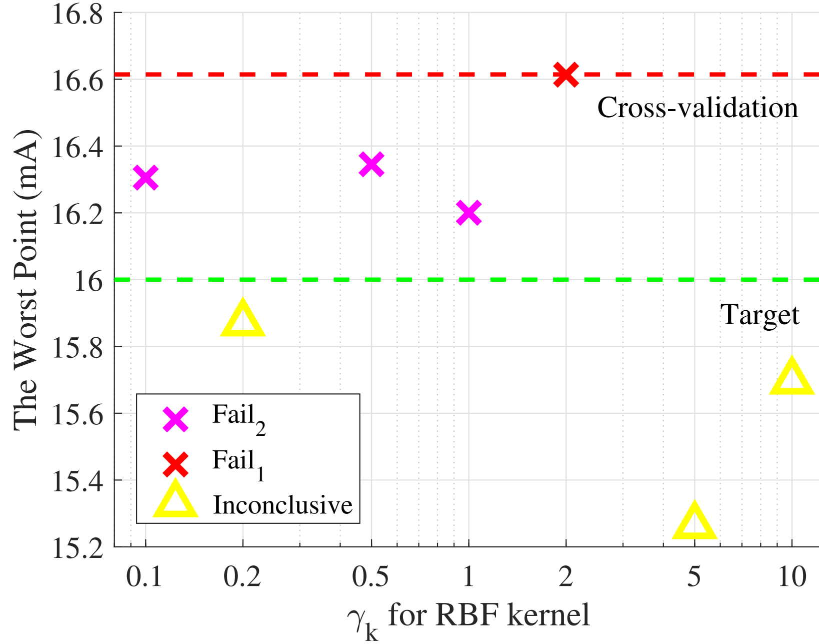


Figure 7.8: HFMV verification with varying kernel parameter γ_k.

In Fig. 7.7, as γ gradually drops from positive to negative, the verification decision changes from "Fail$_1$" to "Inconclusive", and then to "Pass", a well expected outcome since the specified target becomes more relaxed. When γ drops below $\gamma_0 = -2.3264$, since no true failure is identified, we have a high confidence, i.e. a probability of at least 99%, that the LDO is a good passing circuit. In comparison, the Monte Carlo method not only misses true failures but also lacks a built-in mechanism to provide an affirmative "yes/no" answer for the correctness of circuits.

7.6 Verification with Varying Kernel Parameters in ML Model

Generally, ML models are sensitive to inner parameters (e.g. parameters in kernel functions and penalty coefficients in cost functions) and regression performance may vary a lot with different parameters. That's why we implement the cross-validation method for RVFM training in our HFMV flow. However, to measure the sensitivity of our HFMV

flow to the inner parameter of the built-in RVFM model, i.e. γ_k in RBF kernel, we run additional experiments for the quiescent current of the LDO (the target is set to 16mA) by fixing γ_k to a series of values and comparing corresponding performances with the one using the cross-validation method. For all experiments, S_{max} is set to 2800, and the HFMV flow exits **only** when the number of training samples exceeds S_{max}.

Detailed results are shown in Fig. 7.8, and we use the value of the worst point found by HFMV as the metric for performance. In Fig. 7.8, the red line represents the worst failure found using the cross-validation method, and the green line stands for the specified target that is 16mA. For most cases (all except the case that $\gamma_k = 2$), performance descents can be observed when compared to the case employing the cross-validation method, which indicates that our HFMV flow is sensitive to the kernel parameter γ_k to some degree, and model evaluation methods, e.g. the cross-validation method, are suggested to guarantee better capability to identify hard-to-detect failures.

8. SUMMARY AND CONCLUSION

A novel hybrid approach, namely HFMV, has been presented to address the challenges associated with AMS verification. HFMV combines the key benefits of formal verification and ML-based approaches while circumventing their key limitations in terms of scalability and model inaccuracy. It has been demonstrated that HFMV can provide reliable verification of AMS performance in high-dimensional parameter spaces for which much more time-consuming Monte Carlo simulations lead to misleading results.

REFERENCES

[1] H. Chang and K. Kundert, "Verification of complex analog and rf ic designs," *Proceedings of the IEEE*, vol. 95, pp. 622–639, March 2007.

[2] S. Palnitkar, *Verilog®Hdl: A Guide to Digital Design and Synthesis, Second Edition*. Upper Saddle River, NJ, USA: Prentice Hall Press, second ed., 2003.

[3] A. B. Mehta, *UVM (Universal Verification Methodology)*, pp. 17–64. Cham: Springer International Publishing, 2018.

[4] P. Frey and D. O'Riordan, "Verilog-ams: Mixed-signal simulation and cross domain connect modules," in *Proceedings of the 2000 IEEE/ACM International Workshop on Behavioral Modeling and Simulation*, BMAS '00, (Washington, DC, USA), pp. 103–, IEEE Computer Society, 2000.

[5] M. Miller and F. Brewer, "Formal verification of analog circuit parameters across variation utilizing sat," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1442–1447, March 2013.

[6] W. Denman, B. Akbarpour, S. Tahar, M. H. Zaki, and L. C. Paulson, "Formal verification of analog designs using metitarski," in *Formal Methods in Computer-Aided Design (FMCAD)*, pp. 93–100, Nov 2009.

[7] L. Yin, Y. Deng, and P. Li, "Verifying dynamic properties of nonlinear mixed-signal circuits via efficient smt-based techniques," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 436–442, November 2012.

[8] H. Lin, P. Li, and C. J. Myers, "Verification of digitally-intensive analog circuits via kernel ridge regression and hybrid reachability analysis," in *ACM/IEEE Design*

Automation Conference (DAC), pp. 1–6, May 2013.

[9] F. Wang, M. Zaheer, X. Li, J.-O. Plouchart, and A. Valdes-Garcia, "Co-learning bayesian model fusion: Efficient performance modeling of analog and mixed-signal circuits using side information," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 575–582, IEEE Press, 2015.

[10] J. A. Kumar, S. N. Ahmadyan, and S. Vasudevan, "Efficient statistical model checking of hardware circuits with multiple failure regions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, pp. 945–958, June 2014.

[11] M. Rana, R. Canal, J. Han, and B. Cockburn, "Sram memory margin probability failure estimation using gaussian process regression," in *2016 IEEE 34th International Conference on Computer Design (ICCD)*, pp. 448–451, Oct 2016.

[12] L. W. Nagel and D. Pederson, "Spice (simulation program with integrated circuit emphasis)," Tech. Rep. UCB/ERL M382, EECS Department, University of California, Berkeley, Apr 1973.

[13] A. Singh and P. Li, "On behavioral model equivalence checking for large analog/mixed signal systems," in *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 55–61, Nov 2010.

[14] X. Li, P. Li, Y. Xu, and L. T. Pileggi, "Analog and rf circuit macromodels for system-level analysis," in *Proceedings 2003. Design Automation Conference (IEEE Cat. No.03CH37451)*, pp. 478–483, June 2003.

[15] A. Balivada, Y. Hoskote, and J. A. Abraham, "Verification of transient response of linear analog circuits," in *Proceedings 13th IEEE VLSI Test Symposium*, pp. 42–47, Apr 1995.

[16] L. Hedrich and E. Barke, "A formal approach to nonlinear analog circuit verification," in *Proceedings of IEEE International Conference on Computer Aided Design (ICCAD)*, pp. 123–127, Nov 1995.

[17] S. Steinhorst and L. Hedrich, "Advanced methods for equivalence checking of analog circuits with strong nonlinearities," *Formal Methods in System Design*, vol. 36, pp. 131–147, Jun 2010.

[18] A. Ghosh and R. Vemuri, "Formal verification of synthesized analog designs," in *Proceedings 1999 IEEE International Conference on Computer Design: VLSI in Computers and Processors (Cat. No.99CB37040)*, pp. 40–45, 1999.

[19] S. Owre, J. M. Rushby, and N. Shankar, "Pvs: A prototype verification system," in *Proceedings of the 11th International Conference on Automated Deduction: Automated Deduction*, CADE-11, (London, UK, UK), pp. 748–752, Springer-Verlag, 1992.

[20] G. Al-Sammane, M. H. Zaki, and S. Tahar, "A symbolic methodology for the verification of analog and mixed signal designs," in *2007 Design, Automation Test in Europe Conference Exhibition*, pp. 1–6, April 2007.

[21] S. Wolfram, *Mathematica: A System for Doing Mathematics by Computer*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1988.

[22] D. Walter, S. Little, N. Seegmiller, C. J. Myers, and T. Yoneda, "Symbolic model checking of analog/mixed-signal circuits," in *2007 Asia and South Pacific Design Automation Conference*, pp. 316–323, Jan 2007.

[23] S. X. D. Tan, "Symbolic analysis of analog circuits by boolean logic operations," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, pp. 1313–1317, Nov 2006.

[24] G. Shi, "A survey on binary decision diagram approaches to symbolic analysis of analog integrated circuits," *Analog Integrated Circuits and Signal Processing*, vol. 74, pp. 331–343, Feb 2013.

[25] C. Baier and J.-P. Katoen, *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[26] M. H. Zaki, S. Tahar, and G. Bois, "Formal verification of analog and mixed signal designs: Survey and comparison," in *2006 IEEE North-East Workshop on Circuits and Systems*, pp. 281–284, June 2006.

[27] P. A. Abdulla, P. Bjesse, and N. Eén, *Symbolic Reachability Analysis Based on SAT-Solvers*, pp. 411–425. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000.

[28] A. Girard and C. Le Guernic, "Efficient reachability analysis for linear systems using support functions," in *IFAC World Congress*, (Séoul, South Korea), pp. 8966–8971, IFAC, July 2008.

[29] M. Althoff, O. Stursberg, and M. Buss, "Reachability analysis of linear systems with uncertain parameters and inputs," in *2007 46th IEEE Conference on Decision and Control*, pp. 726–732, Dec 2007.

[30] L. Yin, Y. Deng, and P. 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*, vol. 32, pp. 977–990, July 2013.

[31] W. Hartong, L. Hedrich, and E. Barke, *On Discrete Modeling and Model Checking for Nonlinear Analog Systems*, pp. 401–414. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002.

[32] S. Little, D. Walter, N. Seegmiller, C. Myers, and T. Yoneda, *Verification of Analog and Mixed-Signal Circuits Using Timed Hybrid Petri Nets*, pp. 426–440. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.

[33] D. Walter, S. Little, and C. Myers, *Bounded Model Checking of Analog and Mixed-Signal Circuits Using an SMT Solver*, pp. 66–81. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[34] C. Le Guernic and A. Girard, *Reachability Analysis of Hybrid Systems Using Support Functions*, pp. 540–554. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[35] H. Lin and P. Li, "Parallel hierarchical reachability analysis for analog verification," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, June 2014.

[36] M. Althoff, A. Rajhans, B. H. Krogh, S. Yaldiz, X. Li, and L. Pileggi, "Formal verification of phase-locked loops using reachability analysis and continuation," in *in ICCAD, 2011*.

[37] L. de Moura, B. Dutertre, and N. Shankar, "A tutorial on satisfiability modulo theories," in *Proceedings of the International Conference on Computer Aided Verification (CAV)*, pp. 20–36, July 2007.

[38] C. W. Barrett, D. L. Dill, and A. Stump, "Checking satisfiability of first-order formulas by incremental translation to sat," in *Proceedings of the 14th International Conference on Computer Aided Verification*, CAV '02, (London, UK, UK), pp. 236–249, Springer-Verlag, 2002.

[39] R. Nieuwenhuis, A. Oliveras, and C. Tinelli, "Solving sat and sat modulo theories: From an abstract davis–putnam–logemann–loveland procedure to dpll(t)," *J. ACM*, vol. 53, pp. 937–977, Nov. 2006.

[40] S. K. Tiwary, A. Gupta, J. R. Phillips, C. Pinello, and R. Zlatanovici, "First steps towards sat-based formal analog verification," in *2009 IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers*, pp. 1–8, Nov 2009.

[41] M. Fränzle, C. Herde, T. Teige, S. Ratschan, and T. Schubert, "Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure," *JSAT*, vol. 1, pp. 209–236, 2007.

[42] L. D. Moura and N. Bjørner, "Z3: An efficient smt solver," in *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pp. 337–340, April 2008.

[43] B. Dutertre and L. de Moura, *A Fast Linear-Arithmetic Solver for DPLL(T)*, pp. 81–94. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[44] H. D. Foster, "Trends in functional verification: A 2014 industry study," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, June 2015.

[45] M. Ding and R. I. Vemur, "An active learning scheme using support vector machines for analog circuit feasibility classification," in *18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design*, pp. 528–534, Jan 2005.

[46] H. Lin and P. Li, "Circuit performance classification with active learning guided sampling for support vector machines," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, pp. 1467–1480, Sept 2015.

[47] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, Sep 1995.

[48] D. D. Jonghe, D. Deschrijver, T. Dhaene, and G. Gielen, "Extracting analytical nonlinear models from analog circuits by recursive vector fitting of transfer func-

tion trajectories," in *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1448–1453, March 2013.

[49] T. Dhaene and D. Deschrijver, "Stable parametric macromodeling using a recursive implementation of the vector fitting algorithm," *IEEE Microwave and Wireless Components Letters*, vol. 19, pp. 59–61, Feb 2009.

[50] C. E. Rasmussen, "Gaussian processes for machine learning," MIT Press, 2006.

[51] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, June 2001.

[52] H. Lin and P. Li, "Relevance vector and feature machine for statistical analog circuit characterization and built-in self-test optimization," in *ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, June 2016.

[53] Y. Wang, P. Li, and S. Lai, "A unifying and robust method for efficient envelope-following simulation of pwm/pfm dc-dc converters," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 618–625, November 2014.

[54] S. Lai and P. Li, "A fully on-chip area-efficient cmos low-dropout regulator with fast load regulation," *Analog Integrated Circuits and Signal Processing*, vol. 72, pp. 433–450, August 2012.

[55] H. A. Chipman, E. I. George, and R. E. Mcculloch, "Bart: Bayesian additive regression trees," *Annals of Applied Statistics*, vol. 4, no. 1, pp. 266–298, 2010.

[56] H. Cheng, H. Chen, G. Jiang, and K. Yoshihira, *Nonlinear Feature Selection by Relevance Feature Vector Machine*, pp. 144–159. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.