Ph.D. Dissertation

# Probabilistic Validation and Computer-Aided Debugging in Analog/Mixed-Signal Systems:

## Pre-Silicon Global Convergence Property Checking and Post-Silicon Bug Localization

아날로그 혼성신호 시스템에서의 확률적 검증과 디버깅 자동화

2014 년 8 월

서울대학교 대학원

전기 · 컴퓨터공학부

윤 상 호

# Probabilistic Validation and Computer-Aided Debugging in Analog/Mixed-Signal Systems:

## Pre-Silicon Global Convergence Property Checking and Post-Silicon Bug Localization

지도 교수 김재하

이 논문을 공학박사 학위논문으로 제출함

**2014 년 8 월**

서울대학교 대학원

전기컴퓨터공학부

윤상호

윤상호의 공학박사 학위논문을 인준함

**2014 년 8 월**

위 원 장 ______________________ (인)

부위원장 ______________________ (인)

위 원 ______________________ (인)

위 원 ______________________ (인)

위 원 ______________________ (인)

# Abstract

Increasing system complexity, growing uncertainty in semiconductor technology, and demanding requirements in complex specifications pose significant challenges to both pre-silicon design verification and post-silicon chip validation. Thus, this dissertation investigates efficient pre-silicon/post-silicon validation and debugging methodology, especially for analog and mixed-signal (AMS) systems. Principally, validation is formulated as a Bayesian inference problem and analyzed in a probabilistic manner. For instance, *pass*/*fail* property can be checked by Bayesian sampling – the posterior distribution of the unknown failure probability can be measured after many sample validation trials so as to quantify the confidence of *pass* with a given tolerance and model accuracy. This approach is first taken in the pre-silicon verification to check a system's property. In other words, the efficient Monte Carlo-based methods for ensuring global convergence property are proposed using two techniques: fast sample batch verification using cluster analysis and efficient sampling using Gaussian process regression. In addition, a practical design flow for preventing global convergence failure is presented – the notion of indeterminate state $X$ is extended to AMS systems. For the post-silicon validation, in particular, the probabilistic graphical model is proposed as one effective abstraction of AMS systems. Using the probabilistic graphical model and statistical inference, we can compute the probability of each parameter to satisfy a given specification and use it for bug localization and ranking. The proposed model and method are especially useful at the post-silicon validation phase, since they can check and localize bugs in the system under limited observability and controllability.

**Keywords**: Pre-silicon validation, Post-silicon validation, Probabilistic validation, Probabilistic graphical model, Analog and mixed-signal systems

**Student Number**: 2010-30991

# Contents

**Abstract** i

**Contents** ii

**List of Tables** iv

**List of Figures** v

**1 Introduction** 1

**2 Probabilistic Validation and Computer-Aided Debugging in AMS Systems** 5

2.1 Validation as Inference . . . . . . . . . . . . . . . . . . . . . . . . . . 6

2.2 Bayesian Property Checking by Sampling . . . . . . . . . . . . . . . 7

2.3 Probabilistic Graphical Models . . . . . . . . . . . . . . . . . . . . . . 15

**3 Global Convergence Property Checking with Monte Carlo Methods in Pre-Silicon Validation** 20

3.1 Problem Formulation . . . . . . . . . . . . . . . . . . . . . . . . . . . 21

3.2 Fast Sample Batch Verification using Cluster Analysis . . . . . . . . 23

3.2.1 Global convergence failures in state space models . . . . . . . 24

3.2.2 Finding global convergence failures by cluster-split detection 29

3.2.3 Experimental results . . . . . . . . . . . . . . . . . . . . . . . 41

3.3 Efficient Covering and Sampling of Parameter Space . . . . . . . . . 53

3.3.1 Attempt to cover the parameter space – finding transient regions in circuit's state space . . . . . . . . . . . . . . . . . . . 53

3.3.2 Rare-event failure simulation using Gaussian process . . . . . 65

3.4 Preventing Global Convergence Failure via Indeterminate State $X$ Elimination . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 72

3.4.1 Preventing start-up failure by eliminating all indeterminate states . . . . 72
3.4.2 Procedure of eliminating indeterminate states with the extended $X$ for AMS systems . . . . 79
3.4.3 Reducing reset circuits in the $X$ elimination procedure . . . . 82
3.4.4 Experimental results . . . . 90

**4 Bug Localization using Probabilistic Graphical Models in Post-Silicon Validation 98**

4.1 Problem Formulation . . . . 98
4.2 Modeling of AMS Circuits using Probabilistic Graphical Models . . . 100
4.2.1 Probabilistic graphical models . . . . 101
4.2.2 Generating probabilistic graphical models for AMS circuits . 105
4.3 Probabilistic Bug Localization using Probabilistic Graphical Models 106
4.3.1 Posterior estimation using statistical inference . . . . 107
4.3.2 Probabilistic bug localization and ranking . . . . 109
4.3.3 Implementation details . . . . 110
4.4 Experimental Results . . . . 115
4.5 Possible Extensions of Graphical Models – Equivalence Checking . . 122

**5 Conclusion 129**

**Bibliography 132**

# List of Tables

3.1 Coupled ring oscillator analysis results . . . . . . . . . . . . . . . . . . 43

3.2 Differential four-stage dual-delay-path ring oscillator analysis results 46

3.3 DCO analysis results . . . . . . . . . . . . . . . . . . . . . . . . . . . 48

3.4 PLL analysis results . . . . . . . . . . . . . . . . . . . . . . . . . . . 50

3.5 DC-DC converter analysis results . . . . . . . . . . . . . . . . . . . . 51

4.1 Posterior probabilities of the pole/zero parameters of the channel and the CTLE to be within the given spec-range. . . . . . . . . . . . . . . 118

# List of Figures

1.1 Reconvergent paths in validation. . . . . . . . . . . . . . . . . . . . . . 2

1.2 Accelerating validation process. . . . . . . . . . . . . . . . . . . . . . . 3

1.3 Design and validation in product life cycle. . . . . . . . . . . . . . . 4

2.1 Posterior probability of the unknown failure probability ($\mu$) distribution after $N$ number of observations. . . . . . . . . . . . . . . . . . . 9

2.2 Required number of verifying samples ($N$) according to the confidence and the tolerance ($\epsilon$). . . . . . . . . . . . . . . . . . . . . . . . . . . 10

2.3 Simple graphical model of the failure probability $\mu$ and the Bernoulli validation trials $X$s. . . . . . . . . . . . . . . . . . . . . . . . . . 11

2.4 Graphical models of the unknown failure probability $\mu$ that consider model inaccuracy, (left) without simplification and (right) with simplification. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

2.5 Confidence that the failure probability is less than 0.01% when no failure is detected after $N$ validation trials. The red 'x' is when the used model is perfect and the blue 'o' is when it is not. . . . . . . . . 14

2.6 Simple I/O link system. . . . . . . . . . . . . . . . . . . . . . . . . . . 17

2.7 Probabilistic graphical model of the I/O link system in Fig. 2.6. . . 17

2.8 Simple chain graphical model. . . . . . . . . . . . . . . . . . . . . . . 18

3.1 Parallel RLC circuit. . . . . . . . . . . . . . . . . . . . . . . . . . . . 25

3.2 Second-order dynamic systems with an equilibrium (left) and a periodic steady state (right). . . . . . . . . . . . . . . . . . . . . . . . . . 26

3.3 Coupled ring oscillator. . . . . . . . . . . . . . . . . . . . . . . . . . 27

3.4 Illustration of the detection of the global convergence failure in a coupled ring oscillator in Fig. 3.3 using the cluster-split detection (CSD) algorithm. All of the sampled states will form a single cluster throughout convergence into an attractor when there is no global convergence failure (top). Otherwise, a failure can be detected when the sample points, starting from a uniform random distribution of the circuit states (n1, p2, n3), progress over time and form more than one distinct cluster (bottom). . . . . . . . . . . . . . . . . . . . . . . . . 28

3.5 K-nearest neighbor clustering (K=2). . . . . . . . . . . . . . . . . . . 31

3.6 K-nearest neighbor clustering with K-parameter extraction (N=512,K=15). 32

3.7 (a) Running transient simulations with randomized initial conditions, examining distributions at $t_0$ and (b) measuring its entropy to find a non-X node, $A$, and an X node, $B$. . . . . . . . . . . . . . . . . . . . 35

3.8 Proposed merging method on the examples in (a): First, (b) a merging graph is constructed by examining the K-nearest relationships among trajectory samples and the angles between trajectories. Next, (c) a trajectory proximity graph is built and (d) trajectories are merged if they are connected in the merging graph and if removing them does not cause a split in the trajectory proximity graph. . . . . . . . . . . 38

3.9 Number of remaining trajectories after merging at each time step of the CSD in (top) PLL and (bottom) DC-DC converter. . . . . . . . 39

3.10 Visualization of all the possible steady-state solutions in the state space and the final state distributions of the coupled ring oscillator after the CSD algorithm when the W1:W2 ratios are (a, b) 2:1 (c, d) 8:1 (e, f) 1:8 and (g, h) 1:1. . . . . . . . . . . . . . . . . . . . . . . . 42

3.11 Differential four-stage dual-delay-path ring oscillator (bottom) and its unit stage (up). . . . . . . . . . . . . . . . . . . . . . . . . . . . 45

3.12 Two possible oscillation modes, differential oscillation (top) and common-mode oscillation (bottom), of the differential four-stage dual- delay-path ring oscillator. . . . . . . . . . . . . . . . . . . . . . . . . . . 45

3.13 DCO with a global convergence failure. . . . . . . . . . . . . . . . 47

3.14 Normal oscillation mode and false oscillation mode in the DCO. . . . 47

3.15 PLL start-up failures: (a) when the VCO starts at too high a frequency, the divider may fail to output the correct frequency and the PFD drives the loop away from the correct lock (SLOWDIV), and (b) the slow operation of the PFD generates net zero output at an incorrect frequency (SLOWPFD). . . . . . . . . . . . . . . . . . . . 49

3.16 Step-down DC-DC converter. . . . . . . . . . . . . . . . . . . . . . . . . 51

3.17 Attempt to cover the parameter space by volumes. . . . . . . . . . . 54

3.18 Decreasing effective dimensionality of state distribution versus time and increasing effective resolution of finite number of samples versus time. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55

3.19 (a) Example X graph and (b) clique graph. . . . . . . . . . . . . . . 57

3.20 Effective dimensionality of the state distribution at various time instants, measured by the proposed entropy based method and PCA, of (a) BGR, (b) DCO and (c) PLL. . . . . . . . . . . . . . . . . . . . 58

3.21 (a) Dividing the parameter region and checking each region's transitions, (b) building the transition diagram and finding transient regions. 61

3.22 (a) Three-stage ring oscillator and (b) resulting transition diagram after the experiment when the space was divided into 87 regions. . . 62

3.23 Detected transient region's volume ratio versus the number of divided regions in the 3-stage ring oscillator example. . . . . . . . . . . . . 63

3.24 (a) Mean squared error of the GP's prediction for the samples in the 73th sphere; (b) the actual value of one element of the next state vector function and the predicted value by the GP regression. The red points are sample predictions, the red regions are 95% confidence interval of the GP predictions, and the black points are the actual values. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 63

3.25 Importance sampling by using a smooth guidance metric and a regression. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 66

3.26 Ranking samples according to the GP predictions. . . . . . . . . . . 69

3.27 Experimental results (right) of the proposed method to the coupled ring oscillator (left). . . . . . . . . . . . . . . . . . . . . . . . . . . 71

3.28 Experimental results (right) of the proposed method to the coupled ring oscillator (left). . . . . . . . . . . . . . . . . . . . . . . . . . 71
3.29 Finite state machine without reset (left) and with reset (right). . . . 73
3.30 Extended Boolean algebra including the indeterminate value X. . . . 74
3.31 Example of the four-valued simulation in a digital system. . . . . . . 75
3.32 State space with two equilibrium points. . . . . . . . . . . . . . . . . 76
3.33 Indeterminate state X in mixed-signal systems. . . . . . . . . . . . . 77
3.34 Quantization of a continuous random variable. . . . . . . . . . . . . 78
3.35 X elimination procedure. . . . . . . . . . . . . . . . . . . . . . . . . 79
3.36 Number of samples (N) versus the number of detected X's in a digitally controlled oscillator, a phase-locked loop circuit and a switching-mode power supply. . . . . . . . . . . . . . . . . . . . . . . . . . 81
3.37 Isolated X in (a) a NAND gate with fixed zero inputs and in (b) a flip-flop with inactive clocks. . . . . . . . . . . . . . . . . . . . . . . 86
3.38 Finding isolated X in (a) a NAND gate by (b) constructing a circuit graph, (c) eliminating all non-X nodes and (d) finding a connected component which has no path to the observable outputs. . . . . . . . 86
3.39 Finding dependent relationships. . . . . . . . . . . . . . . . . . . . . 88
3.40 Flowchart of the improved X elimination procedure. . . . . . . . . . 90
3.41 DCO experiencing a global convergence failure. . . . . . . . . . . . . 91
3.42 Number of X nodes during the X elimination procedures. . . . . . . 92
3.43 Lists of X nodes in the DCO after the first and second enhanced X elimination iterations. . . . . . . . . . . . . . . . . . . . . . . . . . 93
3.44 Trajectory starting from the problematic initial condition in the original DCO (up) and the trajectory after inserting reset circuits according to the X elimination procedure (bottom). . . . . . . . . . . . . 93
3.45 PLL experiencing a global convergence failure. . . . . . . . . . . . . 95
3.46 Number of X nodes during the X elimination procedures. . . . . . . 95
3.47 Lists of X nodes in the PLL after the first and second X elimination iterations. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 95

3.48 Trajectory starting from the problematic initial condition in the original PLL (up) and the trajectory after inserting reset circuits according to the X elimination procedure (bottom). . . . . . . . . . . . . . . . 96

4.1 Probabilistic graphical model of a I/O link system. . . . . . . . . . . 101

4.2 (a) Continuous-time linear equalizer and (b) its GBN model. . . . . 103

4.3 (a) Decision feedback equalizer and (b) its TBN model. . . . . . . . 104

4.4 Bayesian network model of a four-stage I/O link system with the 100-length input/output waveform evidences. Shaded nodes indicate presence of evidences on the nodes. . . . . . . . . . . . . . . . . . . . . 106

4.5 (a) Estimating $p(A, C, D|B_{ob})$ by Gibbs sampling: (b) initial guess $(A_0, B_{ob}, C_0, D_0)$; (c) sampling A according to $p(A|B_{ob}, C_0, D_0)$; (d) sampling C according to $p(C|A_1, B_{ob}, D_0)$; (e) sampling D according to $p(D|A_1, B_{ob}, C_1)$; (e) a Gibbs sample $(A_1, B_{ob}, C_1, D_1)$. . . . . . . 108

4.6 Proposed bug localization method using probabilistic graphical models.111

4.7 Adding controllability ($\theta_{CTLE1}$ and $\theta_{CTLE2}$) and virtual gain stages ($V_{gain1}$ and $V_{gain2}$) in the probabilistic graphical model. . . . . . . . 113

4.8 Estimated posterior distribution of one of the CTLE parameters. The black spike is the actual value of the parameter. (a) The posterior distribution when no controllability is applied in the experiments. (b) The posterior distribution when we consider controllability of the CTLE. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 116

4.9 Pole-zero plot. Green crosses and circles represent the actual values of poles and zeros, respectively. The blue crosses and circles represent the Gibbs samples of the poles and zeros, respectively. (a) When used a single pair of input/output waveform evidences without controllability and (b) when used two pairs of input/output waveform evidences with controllability. . . . . . . . . . . . . . . . . . . . . . 117

4.10 Posterior distributions of the channel parameters (a,b and c) and the CTLE parameters (d,e and f). The black spike is the expected value of the parameter. . . . . . . . . . . . . . . . . . . . . . . . . . . . 118

4.11 Posterior distributions of the channel's pole/zero parameters when the input of the channel and the output of the DFE are observed. The channel is modeled as 1 zero and 2 poles and samples of their posterior distributions of zero (o) and pole (x) locations are plotted in pole/zero map (blue). The green crosses (x) and a circle (o) are the desired pole/zero locations. From this figure, it can be seen that the posterior distribution does not cover the desired pole/zero location so that we can detect that the channel is buggy in this example. . . . . 119

4.12 Posterior distributions of the CTLE's pole/zero parameters when the input of the channel and the output of the DFE are observed. Each sub-component (i.e., channel and CTLE) has 1 zero and 2 poles and samples of their posterior distributions of zero (o) and pole (x) locations are plotted in pole/zero map (blue). The green crosses (x) and a circle (o) are the desired pole/zero locations. . . . . . . . . . . . . 119

4.13 Posterior distributions of the channel's pole/zero parameters when the input of the channel and the output of the DFE are observed. Each sub-component (i.e., channel and CTLE) has 1 zero and 2 poles and their posterior distributions of real part and imaginary part are displayed. The black spike indicates the desired parameter value. . . 120

4.14 Posterior distributions of the channel's pole/zero parameters when the input of the CTLE and the output of the DFE are observed. The CTLE has 1 zero and 2 poles and their posterior distributions of real part and imaginary part are displayed. The black spike indicates the desired parameter value. . . . . . . . . . . . . . . . . . . . . . . . . 121

4.15 (a) CMOS NAND gate and (b) representing the NAND gate by a Markov Network. (c) Inverter chain, (d) MN representation, (e) reducing the MN by eliminating certain nodes, and (f) reducing the MN by eliminating dependent nodes. . . . . . . . . . . . . . . . . . . 124

4.16 Dimensionality reduction in a NAND gate, a coupled ring oscillator, a digitally controlled oscillator and a phase locked loop using the X analysis and clique decomposition. . . . . . . . . . . . . . . . . . . . 127

4.17 (a) Average JSD distance (top), period difference (middle; normalized), and deviation of transient waveforms with the same initial condition (bottom; normalized) of ring oscillator. (b) Average JSD distance (top), input to output delay difference (middle; normalized), and deviation of transient waveforms with the same initial condition (bottom; normalized) of CMOS NAND gate. . . . . . . . . . . . . . 128

5.1 Validation as inference. . . . . . . . . . . . . . . . . . . . . . . . . . . . 130

5.2 Collecting evidence efficiently using confidence as guidance. . . . . . 130

# Chapter 1
# Introduction

Validation refers to a process of establishing sufficient evidence in order to ensure that a given system accomplishes its intended requirements. In other words, validation procedure checks whether the implemented design indeed satisfies the specifications with a high degree of certainty (Fig. 1.1).[1] For example, a high speed I/O link specification may demand the receiver's jitter tolerance to be greater than 0.5 UI. Then, the validation procedure checks that the implemented receiver circuit meets this specification regardless of possible variations in the system.

However, validation (or verification)[2] is never a trivial task but a major bottleneck in integrated circuit (IC) design process [2–9]. The well-known "70%-rule" says that validation takes up to 70% of the total design effort, while 60% of the validation effort is spent on debugging [8, 9]. Worse, the total amount of efforts needed for validation has been growing fast, making validation tasks even more challenging. This is because system complexity and target performance have been increasing quickly, keeping pace with the exponential rate predicted by Moore. Furthermore, the parameter space that validation should take into account has been expanding – for example, uncertainty in chip fabrication process has increased as the minimum feature size in technology has gone even smaller than the wavelength of light used for lithography.

Verification has traditionally been dependent on computer simulation. However, simulation is orders of magnitude slower than the actual chip and it is impossible to verify today's large system only by computer simulation. The number of possible usage scenarios and parameters in the system has already exploded to bring about scalable challenges.

[1] The implementation or design is a model of the circuit. The specifications are properties that the model must satisfy [1].

[2] In this dissertation, two terms – *verification* and *validation* – will be interchangeably used, assuming both indicate the same procedure to confirm a given implementation.

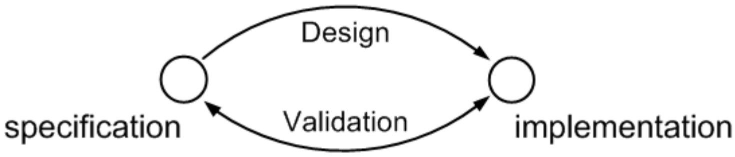


Figure 1.1. Reconvergent paths in validation.

Thus, there have been many different approaches to accelerate verification process so that we can filter out as many bugs as possible in pre-silicon validation. First, there is a simulation acceleration approach. Many different high-speed emulators, which rely on specialized application-specific integrated circuit (ASIC) or field programmable gate array (FPGA), have been used for high-speed simulation. FPGA prototyping has been another popular option – it uses multiple FPGA devices to implement the design and actually runs the system on it. However, there is a drawback with simulation-based approach that it cannot ensure the absence of a bug and only demonstrate one aspect of a bug [3].

Rather than depending on only simulation for the verification, another effective verification methodology – called *formal* verification – has been developed and employed widely. Formal verification refers to a systematic process of ensuring that an implementation satisfies its specifications in exhaustive way [3]. This approach mainly depends on Boolean abstraction and reasoning. For example, we can check equivalence between two Boolean models by using efficient binary decision diagrams (BDDs), automatic test pattern generation (ATPG) or satisfiability (SAT) algorithms. Moreover, there are formal tools called model checker or assertion provers. These tools mathematically prove that an assertion in a given RTL (*i.e.*, Register Transfer Level) design will always hold true [2]. These verification techniques can significantly reduce verification efforts when properly combined with typical simulation technique as depicted in Fig. 1.2.

Yet, this verification methodology is primarily for verifying digital systems and difficult to be used for verifying AMS systems. Boolean expressions that the formal verification methods are based on is not directly usable for analog systems which are essentially continuous dynamic systems.[3] In addition, even though the Boolean-

[3] Moreover, a system is usually non-linear in real-circuits, rather than being pure linear, making

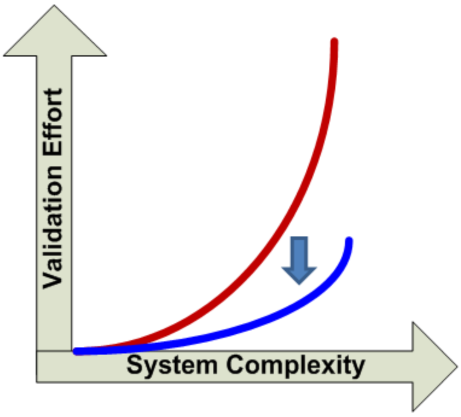


Figure 1.2. Accelerating validation process.

based formal method can efficiently detect logic bugs, it may miss electrical bugs that result from interactions between a design and the electrical state of a system [6].

Thus, this work investigates efficient validation methodology for analog and mixed-signal systems, aiming to achieve a significant reduction in validation efforts similar to the reduction obtained by the digital formal verification methods. In short, *probabilistic validation*, which will be explained in Chapter 2 in detail, is used for both validating and debugging AMS systems. Specifically, validation problem is formulated in a probabilistic framework to take consideration of all the possible uncertainty that arises due to variations in process technology, uncertain environments, lack of information (*e.g.*, due to limited observability and controllability), and even model inaccuracy. Upon the probabilistic base, a degree of assurance that a design meets its specification can be quantified using Bayesian approach.

Chapter 3 presents effective ways to verify a transistor-level AMS design in pre-silicon design validation. In particular, global convergence property checking in AMS system is studied using dynamic system representation (*i.e.*, SPICE models) and sampling-based approaches (*i.e.*, Monte Carlo methods). It is first proposed to accelerate the Monte Carlo-based verification by batch sample verification using cluster analysis. Next, the efficiency of the sampling is further increased by importance sampling using Gaussian process regression. In addition, this work will also

the analysis more difficult.

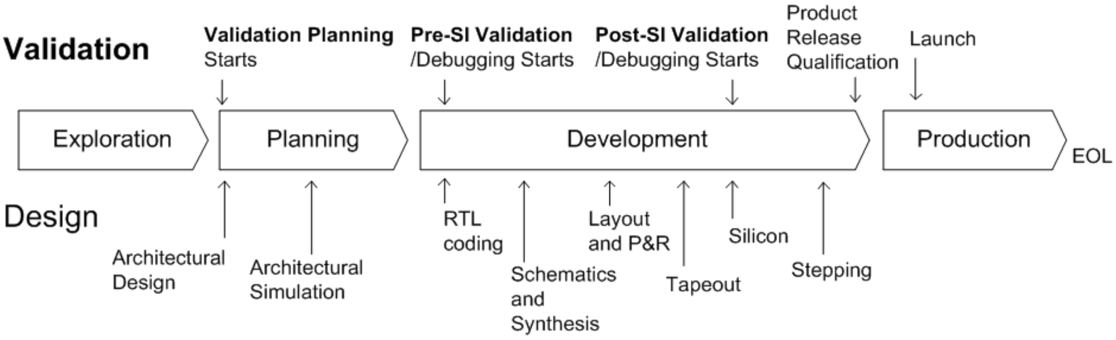


Figure 1.3. Design and validation in product life cycle.

discuss computer-aided debugging methodology since debugging takes a significant portion of total verification efforts [9]. In the last section in Chapter 3, an effective debugging method to prevent global convergence failures in AMS circuits is presented.

Next, a post-silicon bug localization method is discussed in Chapter 4. Since the end goal of validation is to produce a healthy bug-free system on silicon after manufacture, validation involves much more works than the verification before chip fabrication (Fig. 1.3) [5]. In other words, pre-silicon design verification alone cannot detect all the critical bugs in the final system [6]. Possible failures and bugs can easily escape pre-silicon design verification but detected later in post-silicon validation phase, indicating the importance of post-silicon validation. Thus, this dissertation investigates post-silicon validation in addition to the pre-silicon property checking. Specifically, the bug localization method under limited observability and controllability is studied in Chapter 4. Instead of deterministic SPICE models, probabilistic graphical models are used as an effective abstraction and statistical inference is applied to the graphical models so that potential root-causes of the bug are identified and ranked. In addition, the proposed graphical model's effectiveness in the pre-silicon verification stage is additionally investigated in the last section in Chapter 4, implying that the proposed probabilistic graphical model can be an effective abstraction for AMS systems in general.

# Chapter 2
# Probabilistic Validation and Computer-Aided Debugging in AMS Systems

In validation procedure, the design should be *exhaustively* and *systematically* checked. In other words, validation must perform in-depth property checking that computes the attributes of the implementation in many different settings and finally confirms all of them satisfy the desired properties. Not to miss any critical bugs but without impractical overhead, the property checker should systematically explore all the parameter space, meaning that we should maximize the effectiveness of each verification execution without wasting resources.

In digital verification, Boolean abstraction and the coverage definition on finite parameter space have enabled systematic and exhaustive examination [2,3]. In other words, abstraction layer has been well-defined based on the Boolean and we can efficiently check the system. For instance, based on Boolean representation and logic, verification has been facilitated by efficient binary decision diagrams (BDDs), automatic test pattern generation (ATPG) or satisfiability (SAT) algorithms. Furthermore, the coverage can be defined as the ratio of verified (*i.e.*, covered) number of parameters to the total number of parameters since the parameter space is finite. This can guide us to systematically and effectively explore the parameter space.

In contrast, the parameter space of an AMS system is continuous and infinite, and systematic Boolean reasoning would not be directly applicable to AMS systems. Although a continuous space can be quantized and be transformed into Boolean models, it has scalability issues that the number of states to consider would grow exponentially as the system size increases [10].

Another issue in AMS validation is that we should properly consider many different types of unpredictable and uncontrolled factors in the system. For example, circuits may be sensitive to noise and process variations and thus validation method should properly check the circuit's tolerance to this uncertainty.

Last but not least, in post-silicon validation, uncertainty problems are exacerbated by limited observability/controllability and varying chip environment. Although the system under test is considered to be almost deterministic, limited observability and surrounding uncertain factors induce uncertainty to the system. Thus, a validation methodology should be able to properly cope with this type of uncertainty as well.

To resolve these issues and enable systematic validation in AMS systems, this dissertation proposes a *probabilistic* validation approach which offers the following benefits:

- Validation problems can be translated into inference problems and many efficient techniques in machine learning and statistic literatures can be used (Section 2.1).
- A *coverage* concept can be extended to an infinite and uncertain parameter space as *confidence* using Bayesian reasoning (Section 2.2).
- In a probabilistic framework, many different types of uncertainty (for example, variation, noise and model inaccuracy) can be considered (Section 2.2 and Section 2.3).
- The effective abstraction – the probabilistic graphical model – is introduced so that we can model a system in a probabilistic manner with a reasonable complexity (Section 2.3).

## 2.1 Validation as Inference

Validation problems can be translated into inference problems taking Bayesian perspective. The task of computing a certain property of the system (*i.e.*, property checking problem) can be paraphrased as a process that collects enough evidence so that the posterior probability of the unknown property to satisfy a given specification is sufficiently high. In other words, validation is a process that accrues proper evidence ($E$) so that we can assure with a high degree of certainty ($\alpha$) that the property ($A$) satisfies the specification ($A_{spec}$) – it checks if $P(||A - A_{spec}|| < \epsilon|E) > \alpha$. The procedure that computes this posterior probability has been well known as statistical inference in statistics and machine learning literatures [11–14].

Debugging can also be considered to be an inference problem. Typically, we search for a bug root-cause by trial-and-errors that includes three steps: first is to control system properly, feeding different inputs with certain knob settings; second is to collect proper evidences by probing appropriate signals; and finally to check whether the signals are in accordance with our expectation. Likewise, a root-cause of a bug can be detected by (1) controlling a system properly, (2) collecting all the observable data as evidence, and (3) computing each sub-circuit's probability to satisfy a given specification conditioned on the collected evidence. If the computed probability is too low, then the associated sub-circuit is likely to be problematic and need to be further investigated.[1]

## 2.2 Bayesian Property Checking by Sampling

As pointed out in the previous section, validation is the systematic process that exhaustively collects enough evidence so that the confidence of the system's conformance to its specification is sufficiently high. If the evidence is collected by multiple Bernoulli trials – running many validation trials with random parameters and checking each of them whether it satisfies the specification – the confidence after the validation can be computed using a simple formula [15]. That is, the posterior probability to satisfy a given specification conditioned on the observations can be computed.

The posterior probability of the system not to fail (*i.e.*, confidence to satisfy the specification) after many validation trials can be calculated using Bayesian approach. In other words, the relationships among the number of random validation trials with different parameters ($N$), the unknown failure probability ($\mu$) and the confidence ($\alpha$) can be computed by taking a Bayesian approach. Running $N$ validation trials and checking a system's success from them can be considered as consecutive Bernoulli trials with an unknown failure probability $\mu$, which forms a binomial distribution in

[1] More details will be given in Chapter 4

(2.1). By taking a Bayesian approach, the unknown failure probability $\mu$ can also be considered as a random variable and its prior distribution initially assumed to follow the beta distribution as in (2.2) [11]. Initially, both hyper-parameters $a$ and $b$ can be set to 1 and the failure probability $\mu$ could be any value between 0 and 1 with equal probability, as shown in Fig. 2.1 when $N = 0$. This is in accordance with our complete ignorance about the failure probability when we have made no observation.[2] However, after $N$ Bernoulli trials and after observing their results, the distribution of the unknown failure probability (*i.e.*, $P(\mu|D_{\{1\ldots N\}})$) will transform according to the number of observed failures and successes, $m$ and $l$, respectively. The posterior distribution of $\mu$ after the trials, $P(\mu|m,l)$, is the product of the binomial likelihood function in (2.1) and the beta prior in (2.2), and it is still in the form of a beta distribution, as expressed in (2.3) [11].

$$P_{Bin}(m|N,\mu) = \binom{N}{m} \mu^m (1-\mu)^{N-m} \tag{2.1}$$

$$P_{Beta}(\mu|a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \mu^{a-1}(1-\mu)^{b-1} \tag{2.2}$$

$$P(\mu|m,l) = \frac{\Gamma(m+l+2)}{\Gamma(m+1)\Gamma(l+1)} \mu^m (1-\mu)^l \tag{2.3}$$

As the number of observations increases, the uncertainty represented by the posterior distribution in (2.3) will steadily decrease and the uncertainty of the unknown failure probability in this posterior will finally vanish given the limit of an indefinitely large number of observations [11]. For instance, Fig. 2.1 shows the change of the distribution $P(\mu|D_{\{1\ldots N\}})$ as we incrementally make observations when a failure actually occurs with a 10% probability. As the number of observations ($N$) increases, the distribution shapes to have a peak at the correct 0.1 failure probability, as expected.

Using (2.3), we can compute the minimum required number of validation trials ($N$) that bounds the failure probability to be no greater than the tolerance ($\epsilon$) with a certain confidence level ($\alpha$). In other words, when we observe no failure after $N$

[2] However, if there is a prior knowledge about the failure, it can also be reflected by setting hyper-parameters $a$ and $b$ to proper values.

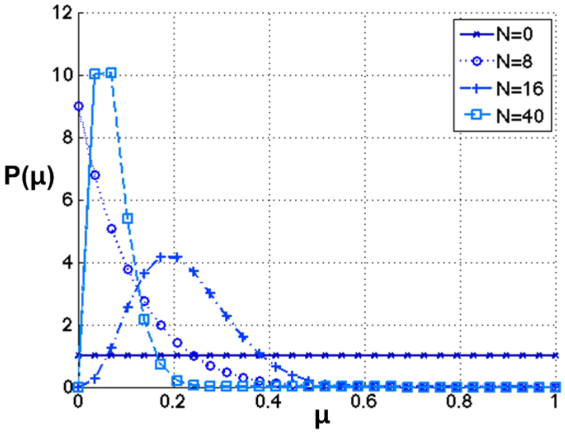


Figure 2.1. Posterior probability of the unknown failure probability ($\mu$) distribution after $N$ number of observations.

trials, the distribution of the unknown failure probability can be computed according to (2.3) with its hyper-parameters $m$ and $l$ being respectively set to 0 and $N$. From this, the probability that the failure probability $\mu$ lies between 0 and the given tolerance $\epsilon$ can be computed as in (2.4), and this probability value corresponds to the confidence that the probability of failure to satisfy a spec is no greater than $\epsilon$.

$$\alpha = \int_0^\epsilon P(\mu)d\mu = \int_0^\epsilon (N+1)(1-\mu)^N d\mu = 1-(1-\epsilon)^{N+1} \tag{2.4}$$

Hence, the required minimum N that bounds the failure probability to be lower than the tolerance $\epsilon$ with the confidence $\alpha$ can be computed as following (2.5):

$$N = \frac{\log(1-\alpha)}{\log(1-\epsilon)} - 1 \tag{2.5}$$

Fig. 2.2 shows the required number of validation trials according to the varying confidence and the tolerance (in other words, the failure probability bound). For example, at least 457 verifying samples are required to confirm that a circuit's failure to satisfy the specification is less than 1% probability with 99% confidence. Or, with 128, 256 and 512 sample checks, the failure probability is bounded to be lower than 2.3%, 1.78% and 1.34% tolerances with 95%, 99% and 99.9% confidence levels, respectively.

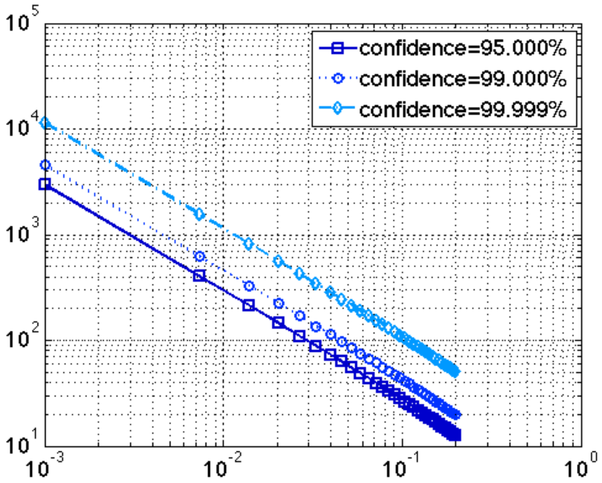


Figure 2.2. Required number of verifying samples ($N$) according to the confidence and the tolerance ($\epsilon$).

The derived equation (2.4) assumes that we take samples from the perfect model so that it will always tell the correct *pass/fail* result for each sample. However, in many settings, we have to depend on an imperfect model to emulate the system in prior to the actual chip fabrication. For example, simplified high-level models such as linear models are often used in the AMS system design for effective design exploration.[3] Thus, the assumption that the used model is perfect may not be held true and the equation (2.4) could not be directly used in many situations.

Nevertheless, we can take consideration of model inaccuracy using a proper graphical model. A graphical model refers to the probabilistic model that uses a graph to denote dependent relationships among random variables [13].[4] The graphical model of the perfect model is shown in Fig. 2.3 and it tells that the model always correctly tells whether a sample $X$ satisfies a given specification or not. Thus, the unknown failure probability $\mu$ can be directly inferred from the observation of $X$s using (2.4). This graphical model can be further modified to properly consider model inaccuracy as Fig. 2.4.

[3] SPICE models are also approximations, even though they can be much more accurate than other approximated models.

[4] More specifically, the graphical model expresses the conditional independence structure

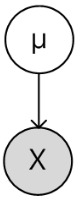


Figure 2.3. Simple graphical model of the failure probability $\mu$ and the Bernoulli validation trials $X$s.

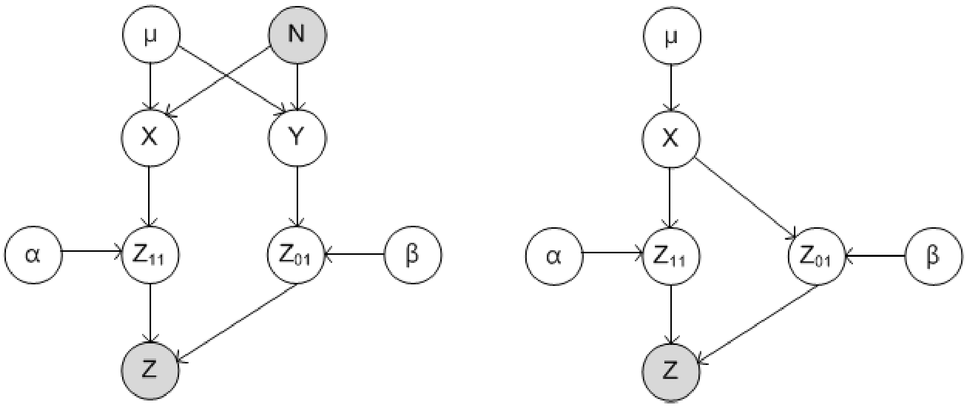


Figure 2.4. Graphical models of the unknown failure probability $\mu$ that consider model inaccuracy, (left) without simplification and (right) with simplification.

The graphical model in Fig. 2.4 illustrates conditional independence relationships among unknown failure probability, sample validation trials and the used model's accuracy: $\mu$ is the unknown failure probability; $X$ is the number of failure samples; $Y$ is the number of success samples; $N$ is the total number of sample validations trials; $\alpha$ is the model's confidence for failure detection when a true failure event happens; $Z_{11}$ is the number of the model's detecting failure samples conditioned on $X$ and $\alpha$; $\beta$ is the model's uncertainty that misclassifies success event as failure; $Z_{01}$ is the number of the model's mispredicted samples (*i.e.*, success samples are classified as failure samples) conditioned on $Y$ and $\beta$. $Z$ is the number of the model's predicted failure samples, which is the sum of $Z_{11}$ and $Z_{01}$. This graphical model can be simplified noting that the sum of $X$ and $Y$ is the number of *pass/fail* check trials $N$. As a result, the graphical model is simplified as shown in the right graph in Fig. 2.4.

Thanks to the structure defined by the graph in Fig. 2.4, the probabilistic relationship among them (*i.e.*, the joint probability distribution) can be factorized to the form $\prod_x p(x|\text{parents}(x))$. Thus, from the graph in Fig. 2.4, the joint distribution is computed as (2.6).

$$\begin{aligned} &P(X, \alpha, \beta, \mu, Z_{11}, Z_{01}, Z) \\ =&P(\mu)P(\alpha)P(\beta)P(X|N,\mu)P(Z_{11}|X,\alpha)P(Z_{01}|N-X,\beta)P(Z|Z_{11},Z_{01}) \end{aligned} \tag{2.6}$$

The probability of each variable conditioned on its parents in the graph forms Binomial distribution since the validation trial is the Bernoulli trial that tells success or failure. Thus, all the conditional probabilities are Binomial distributions except the last term in (2.6): $P(X|\mu) = Bin(N, \mu)$, $P(Z_{11}|X, \alpha) = Bin(X, \alpha)$, $P(Z_{01}|N - X, \beta) = Bin(N - X, \beta)$. The sum of $Z_{11}$ and $Z_{01}$ is always equal to the number of failures predicted by the model ($Z$) and thus $P(Z|Z_{11}, Z_{01}) = \delta(Z - (Z_{11} + Z_{01}))$. The prior distribution of unknown failure probability $P(\mu)$ can be set to be beta distribution. The model's accuracy and inaccuracy, $P(\alpha)$ and $P(\beta)$ can be set using Beta distribution as well. For example, if the model is known to give 8 correct failure predictions out of 10 failure samples than the $a$ and $b$ hyperparameter in $P(\alpha)$ can be set to 8 and 2 respectively.

Using the graphical model in the Fig. 2.4, we can compute the posterior probability distribution of the unknown failure probability $\mu$ after trying $N$ sample validations and observing $Z$ failures among them, using the given imperfect model whose confidence is set by $P(\alpha)$ and $P(\beta)$. Computation of the confidence of the validation result is equivalent to the inference of the posterior probability of the unknown failure probability $P(\mu|Z, N)$ and the integration of this posterior probability, *i.e.*, $P(\mu < tolerance|Z, N)$. However, computing this posterior probability demands integration of the complex conditioned marginal probability and its closed form equation is very difficult to acquire, if not impossible. Instead, an approximated inference technique such as the particle-based approximated inference using Gibbs sampling could be applied to compute the posterior probability [13].

Gibbs sampling, which is a Markov Chain Monte Carlo (MCMC) method, can be employed to estimate the posterior distribution of the unknown failure probability after Bernoulli validation trials – $P(\mu|Z, N)$. The Gibbs sampling is the approximated inference method based on sampling that can be used when the conditional distribution of each variable is known and is easy to sample from.[5] Since all the conditional probability distributions are known, the Gibbs sampling can be used to compute the confidence of the implementation to satisfy a given property after the $N$ number of sample verification trials and $Z$ number of failure observations – $P(\mu|Z, N)$.

For example, Fig. 2.5 shows the probability that the unknown failure (*i.e.*, the chance that the implementation under test fails to satisfy the given specification) is less than 0.01% when no failure is observed after a certain number of validation trials. In the experiment, the model's correct failure prediction (*i.e.*, failure-to-failure) rate $\alpha$ is considered to form the Beta distribution and $P(\alpha)$'s hyperparameters are set to be $a = 80$ and $b = 20$, meaning that the model detects at least 80 out of 100 failures. Similarly, the model's success-to-failure mis-prediction rate $\beta$ is set by the Beta distribution $P(\beta)$ with $a = 20$ and $b = 80$ hyperparameter settings, meaning that 20 out of 100 success samples can be mis-reported as failure samples by the model. Then, the confidence of *pass* after the sample validation trials considering

[5] There are three excellent books in the bibliography [11, 13, 14] that explains Gibbs sampling method well. Moreover, you can find detailed explanation of Gibbs sampling in Section 4.3.1

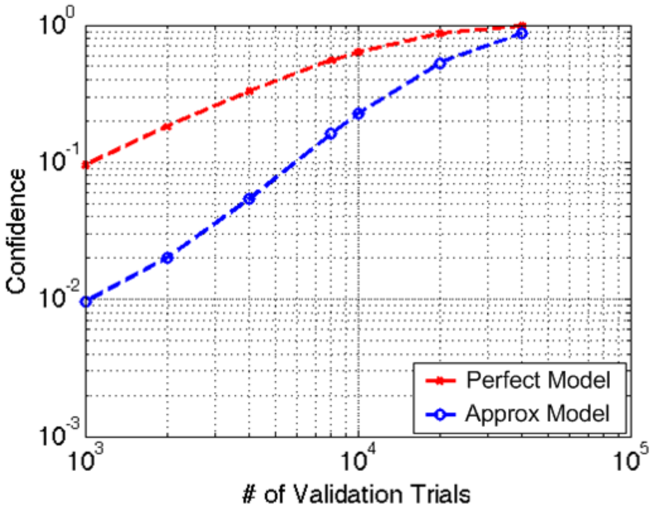


Figure 2.5. Confidence that the failure probability is less than 0.01% when no failure is detected after $N$ validation trials. The red 'x' is when the used model is perfect and the blue 'o' is when it is not.

the model inaccuracy can be computed by the proposed method and the result in the Fig. 2.5 clearly shows the degradation due to the inaccuracy in the model. When 20,000 validation trials are made and no failure is found from the model, the perfect model bound the failure probability to be less than 0.01% with 86% confidence in contrast that the inaccurate model gives only 53% confidence (Fig. 2.4) .

A special case that needs our attention is when the model cannot reveal any hidden failure at all. For example, when a linear model is used for a circuit, it could not say anything about nonlinear effects that come from a transistor's nonlinearity and thus any failure that results from the circuit's nonlinearity would not be predicted by a linear model. In this case, the model's correct failure prediction rate should be set to 0% for the failures come from nonlinearity in the circuit (*i.e.*, hyperparameter $a$ is 0). Then, the above method for computing the confidence of validation will give almost zero-confidence result even though we conduct millions of validation trials, as expected. This tells that the used model should not completely block failures to give meaningful validation result. One way for achieving this is using multiple models so that the chance that all the model block a certain failure becomes very low.

In summary, the confidence of *pass* after the sample validation trials can be computed for a given tolerance and verification payment. For instance, the verification payment is the number of sample verification trials that we can afford to run. The tolerance is a small probability value that bounds the unknown failure probability. Finally, the confidence is the probability that the unknown failure probability is less than the tolerance after paying for a certain number of verification trials. When the used model is near perfect, the relationship among them is in the closed form equation (2.4). On the other hand, if the used model is inaccurate, the confidence can be calculated using the graphical model shown in Fig. 2.4 and performing inference by Gibbs sampling. It should be noted that this quantification allows us to properly quantify the effectiveness of a given validation procedure considering uncertainty so that we can plan validation with a proper balance among the verification cost, tolerance and confidence.

## 2.3 Probabilistic Graphical Models

This work proposes probabilistic graphical models as general and effective templates for the system's internal structure (*i.e.*, abstraction) In other words, the system can be described in a probabilistic manner using graphical models so as to take consideration of a circuit's internal structure information for efficiency.

Analog and mixed-signal systems can be described in a probabilistic manner, rather than in a deterministic way.[6] This probabilistic model is with advantages that it cannot only consider a system's inherent uncertainty such as PVT variations (*i.e.*, process, temperature and voltage variations) but also handle uncertainty that arises due to limited information of the system. For example, most signals in a I/O link circuit in the post-silicon validation would not be measurable and we could only observe its inputs, outputs and a few debugging signals. Consequently, uncertainty can arise due to the unknown signals and parameters in the system even if the system itself is deterministic. However, the probabilistic model is even capable to quantify this uncertainty using a probability distribution, conditioned on observed

[6] Deterministic system can be considered to be a special case of a probabilistic system. When all the probabilistic relationships are without any uncertainty so that underlying probability distributions take the form of dirac-delta distributions, we may call this system as deterministic.

waveforms, rather than simply considering them as unknown. The probabilistic model further allows us to reflect prior knowledge of a hidden parameter as the prior distribution and to improve our inference and validation. This in effect allows us to set a spectrum of belief/weights over possible parameter values in a probability distribution form, instead of a single likely value, so that we can better estimate hidden variables in the system.[7]

Upon probabilistic models, we view a system using joint probability distribution among all the state and parameter variables in the system (*e.g.*, $P(\textit{input}, \textit{output}, \textit{states}, \textit{parameters})$), instead of using deterministic functional relationships (*e.g.*, $\textit{output} = f(\textit{input}, \textit{states}, \textit{parameters})$). This is similar to the view in many statistics and machine learning literatures that treat all the variables in the system as random variables (*e.g.*, Bayesian linear regression [11, 14]).

However, probabilistic models could be limited by scalability issues – it is extremely expensive to characterize high dimensional joint probability distributions. Nonetheless, we can reduce both space and time complexity of probabilistic models from exponential to polynomial by the graphical model [11,13,14].[8] This is because, by the graphical model, the full joint probabilistic relationship among all the signals and the parameters in a system can be decomposed into small factors. For instance, Fig. 2.7 shows a simple example of a probabilistic graphical model (i.e., Bayesian network) that models a system by a graph: nodes are created for signals and the system components' parameters; edges are created for direct dependencies among the signals and the parameters of sub-circuits. Then, the probabilistic distribution among all the signals and the parameters can be factorized into modular components that each of them is a conditional probability density (CPD) of a node and its parents in the graph. That is, $P(IN, A, B, OUT, \theta_{TX}, \theta_{CH}, \theta_{RX})$ in Fig. 2.6 can be factorized to $P(IN)P(A|IN, \theta_{TX})P(B|A, \theta_{CH})P(OUT|B, \theta_{RX})$ using the Bayesian network in Fig. 2.7.

By this factorization, space complexity of the probabilistic model can be reduced

[7] Applying prior distribution is similar to solve a regularized optimization problem, and makes ill-conditioned problems well-conditioned. Posterior distributions (especially their variances) give strong indication whether our estimation is satisfactory or not – parameters that cannot be identified will show a dispersed posterior distribution.

[8] That is, space complexity for storing the joint probability distribution that expresses the system and time complexity of inferences.

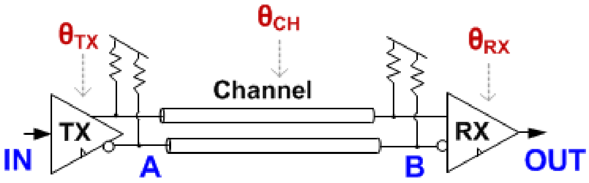


Figure 2.6. Simple I/O link system.

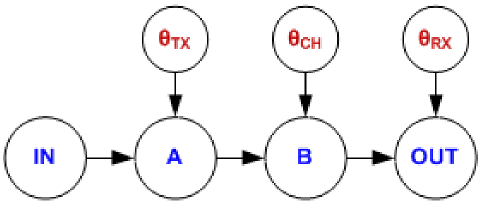


Figure 2.7. Probabilistic graphical model of the I/O link system in Fig. 2.6.

from exponential to polynomial. For example, the number of parameters to save in a simple four-nodes-chain with two quantization levels (Fig. 2.8), is $2^4 - 1$ when no graphical model is available because probabilities of all the possible states should be stored. However, since the distribution can be factorized by the graphical model, it is only required to store distributions of factors and the number of parameters to save is reduced to $3 \times 3 + 1$.

The graphical model can also reduce time complexity of inference from exponential to polynomial.[9] For example, the inference of $OUT$ in Fig. 2.8 (i.e., computing $P(OUT)$) will require $2^3 - 1$ number of additions without a graphical model. However, with the graphical model, the probability $P(IN, A, B, OUT)$ can be factorized and the number of additions can be reduced. In other words, we can move summations inside the factors (i.e., $\Sigma_{IN,A,B}P(IN, A, B, OUT) = \Sigma_B P(OUT|B)\ \Sigma_A P(B|A)$ $\Sigma_{IN}\ P(A|IN)P(IN)$) so that we can reduce the number of required additions to $2\times3$. In general, for a chain example with length $T+1$ and 2 quantization levels, the reduction is from exponential to linear: the number of parameters to save reduced from $2^{T+1} - 1$ to $3T + 1$; the number of additions reduced from $2^T - 1$ to $2T$ [14]. Since it significantly reduces complexity of inference, the graphical model has been

[9] Inference in the probabilistic graphical model refers to computing the marginal distribution of the specific node in the graph.

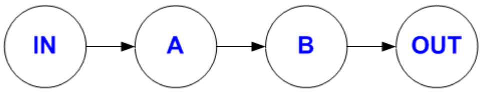


Figure 2.8. Simple chain graphical model.

widely used: forward/backward algorithms in hidden Markov models and Viterbi algorithm for error correction in communication systems are two such examples [11].

In addition to the case where the graphical model is directed and the circuit is feed-forward, it is also possible to extend the model to undirected graphs and circuits with feedback. For example, Phase locked loops (PLLs) and $\Sigma-\Delta$ ADCs are intrinsically a feedback system and the underlying graphical model will be a directed cyclic graph. For another example, the signal at the output of one block depends on the load of the subsequent block – the dependency may be modeled by undirected graphs as the signal flow is less clear. In these cases, the undirected graphical model called Markov network [10] can be used to reduce the cost of probabilistic model [11, 13, 14]. Using Markov network, we can similarly decompose the complex joint distribution into small factors, which is composed of random variables that form a maximal clique in the graph (Section 4.5).

We can generate a probabilistic graphical model of a circuit using topological information in its original model such as a circuit netlist [16, 17]. First, a feed-forward circuit can be modeled by Bayesian network as long as there are explicit input and output ports in the sub-circuits of the system. The Bayesian network model of the circuit is constructed by creating nodes for every input and output signals, connecting its input nodes to its output nodes by edges and finally creating parameter nodes that affects the sub-circuit's operation (e.g. pole/zero parameter of a linear system) and connecting these parameter nodes to the sub-circuits' output nodes. For example, the I/O link circuit shown in Fig. 2.6 can be modeled as a probabilistic graphical model by creating a graph whose nodes correspond to all the signals $IN$, $A$, $B$ and $OUT$ in the system and connecting all the adjacent signal nodes in the circuit topology. Moreover, each signal node that is output of a sub-circuit in the system (e.g. $A$, $B$ and $OUT$) is connected to the parameter node of

[10] Markov network is also called as Markov random field

its corresponding sub-circuit (e.g. $\theta_{TX}$, $\theta_{CH}$, and $\theta_{RX}$) as shown in Fig. 2.7. When it is required to investigate more than one sample for each signal in the system (in other words, when it is necessary to deal with waveform evidences), each $IN$, $A$, $B$ and $OUT$ nodes expand. Similarly, a circuit with feedback can be modeled by Markov Network, creating nodes for all the nodes in the circuit netlist and forming edges between two nodes when they are adjacent in the circuit netlist [16]. Or, both Bayesian network and Markov network may be used together (*i.e.*, hybrid random field) [18] but this dissertation will mainly focus on Bayesian network and use it for post-silicon validation in Chapter 4.

# Chapter 3

# Global Convergence Property Checking with Monte Carlo Methods in Pre-Silicon Validation

In pre-silicon validation phase, it is especially important to filter out fatal bugs, which can cause *functional failures*. Otherwise, resulting failures may render the whole system mal-functional, leading to considerable time and effort in re-spinning the chip.

One such critical failure that is difficult to be caught before chip fabrication is global convergence failures. Global convergence failures (GCFs), accounting for many of the *start-up failures* in analog/mixed-signal (AMS) systems, occur when a starts from a poorly initialized state. That is, a circuit may or may not converge to the correct mode of operation depending on how it is started upon power-on. For example, an oscillator may have multiple oscillation modes; a phase-locked loop (PLL) may not always lock at the correct frequency; and a DC-to-DC switching regulator may not bring up its voltage to the desired level. This phenomenon is due to the non-linearity of the system which can have multiple equilibrium states among which, only one is the desired convergence mode.

This chapter focuses on sampling-based methodologies that check global convergence property of AMS systems, whether there is any possibility of GCFs or not, and finally prevent GCFs.[1] First, efficient and systemic global convergence property checking methods are investigated in first two sections. Basically, the methods are Monte Carlo analysis but they are accelerated by: verifying batch of samples

[1] This sampling approach follows the formulation in Section 2.2

quickly via cluster analysis (Section 3.2); and drawing samples efficiently (Section 3.3.2). Finally, a practical procedure that can guide designers to prevent GCFs is discussed in Section 3.4.

## 3.1 Problem Formulation

Since global convergence failures might not be a probable event, one can run many simulations and never detect them. Thus, some of these failures can easily go unnoticed until the chip is fabricated and tested in the laboratory. Worse, these failures occur intermittently, depending on how the system is started, making them more difficult to analyze and debug. Typical simulations, run to characterize the system performances, are not geared to catch these types of errors. For example, a PLL with a carefully-designed bandwidth and damping factor to ensure its local stability can still suffer from global convergence failures.

It can be rather frustrating when a chip comes back with these types of failures, since the remedies could have been very simple if they were discovered in time. A common fix is to force a part of the initial states to preset values (e.g. to reset the loop filter outputs in case of a PLL). However, once the chip is fabricated, the deadly nature of these failures can render the entire system useless, even preventing from testing other parts of the chip.

Thus, it is important to check global convergence property of a circuit prior to chip fabrication (*i.e.*, in pre-silicon validation). However, this is challenging because the failure modes are unknown and the circuit's correct convergence must be verified from all possible initial states. Obviously, an exhaustive simulation that assesses all possible initial states and checks their individual convergences is not feasible in practice, especially with the continuous and high-dimensional nature of the system states. Nonetheless, the result of GCFs is fatal (*e.g.*, a PLL failing to provide a system clock or a regulator failing to provide the supply voltage to a micro-processor), leading to considerable time and effort in re-spinning the chip.

The fatal nature of the global convergence failures in time-critical IC designs has driven active research in this area, but earlier reported works primarily address false DC convergence states. For instance, if there is non-linearity in only resistive components and if all steady states are the DC equilibrium type, the problem of finding all steady-state solutions can be reduced to a problem of finding all possible DC solutions, and any undesired instances of DC equilibrium can be discovered. In other words, every DC solution of the system could be found by modeling a circuit as piecewise linear resistive circuits [19] or by an interval analysis [20], and the existence or non-existence of a problematic DC solution can be verified. Furthermore, it is possible to detect problematic DC equilibriums in oscillators, where the desired steady state is periodic. For example, Greenstreet et al. [21] proposed a search algorithm to find all of the DC equilibrium points in a ring oscillator and test if a stable point exists. Tiwary et al. [22] described the adoption of a satisfiability (SAT) algorithm in transistor-level circuits using Kirchhoff circuit equations as the underlying theory ultimately to find the problematic initial states from which oscillation does not start. Russo et al. [23] demonstrated the detection of false convergence states by examining the global exponential convergence of each state trajectory via a nonlinear contraction analysis. Stacey et al. [24] back propagated analytic surfaces and approximated the boundary of the asymptotic attraction region of autonomous dynamic system. Stacey's method can be used to check global convergence property by comparing the whole state space with the attraction region.[2]

However, a circuit's steady state that causes a global convergence failure could be other than the DC equilibrium type. For example, two outputs of a differential oscillator may oscillate in phase, exhibiting common-mode oscillation instead of the desired differential-mode oscillation (Fig. 3.11 and Fig. 3.12) [25]. In addition, a PLL may reach a steady-state where its output clock oscillates with an incorrect frequency and no phase alignment [26]. That is, a problematic steady state of a mixed-signal system in general can be periodic, quasi-periodic, or even chaotic [19]. Although periodic and quasi-periodic steady states can be found by a harmonic balance method or a shooting method, finding all periodic/quasi-periodic steady-state solutions remains difficult [27]. Moreover, these methods may not work for some circuits, such as delta-sigma modulators, where a steady-state solution is

[2] Although most works introduced in this paragraph focus on DC convergence states, the last two methods (and possibly Tiwary's method as well) could detect a false steady-state that is not DC equilibrium types.

chaotic [27].

Second issue in verifying global convergence of a circuit is model accuracy and required assumptions. In other words, models used in global convergence property checking should be readily applicable to practical circuits without requiring additional problem formulation. The used model should be accurate enough to give results compatible to results given by traditional SPICE simulations. However, most of previous works require problem formulation at the first stage such as modeling a circuit using analytical equations even though this would increase the chance of having discrepancy between converted model and the original models in SPICE.

Lastly, in global convergence property checking, scalability problem should be addressed as well. Since the initial state space of the circuit needs to be explored for detecting GCFs, the size of parameter space is proportional to the number of circuit nodes, which is usually over hundreds in practical circuits such as PLL. However, experimental results in early prior works in [19–23] are mostly restricted to a small circuit such as oscillators and have not shown its applicability to a large circuits.

Thus, to address three issues mentioned above (*i.e.*, to make the method general, agreeable and scalable), this work takes Monte Carlo-based approach using a SPICE simulator. This is because (1) we can run a SPICE for checking a circuit's convergence to steady-state regardless of its steady-state type for most circuits; (2) A SPICE simulator gives agreeable results with good model accuracy (*i.e.*, circuit designers believe SPICE) and we can avoid over-simplification or supposing unlikely assumptions, which might occur during a problem formulation; (3) lastly, Monte Carlo method (*i.e.*, sampling method) is known to overcome high dimensionality and scalability problem [11].

## 3.2 Fast Sample Batch Verification using Cluster Analysis

This section describes cluster-split detection algorithm (CSD) to detect the global convergence failure problem efficiently. The approach detects a global convergence failure by finding any split among samples of randomly pilot trajectories in the system's state space. Also, it concludes global convergence when all trajectories are expected to follow the same path by an entropy-based uncertainty analysis or by merging any close trajectories during the analysis. For circuits of various sizes – an oscillator, a phase-locked loop and a step-down DC-DC converter – whose number of circuit nodes ranges from 6 to 3,295, it is demonstrated that the proposed approaches effectively detect failures or conclude global convergence within a practical time, which is several minutes for small oscillators and no longer than one day for a large DC-DC converter.

### 3.2.1 Global convergence failures in state space models

A mixed-signal circuit can be represented in a state space model, and its responses can be visualized in the state space as trajectories [19]. For instance, a parallel RLC resonant circuit composed of a resistor, inductor and capacitor (Fig. 3.1) can be described by the differential equation (3.1):

$$C\frac{d^2v}{dt^2} + \frac{1}{R}\frac{dv}{dt} + \frac{1}{L}v = 0 \tag{3.1}$$

$$\begin{pmatrix} \dot{i} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} 0 & -\frac{1}{L} \\ \frac{1}{C} & -\frac{1}{RC} \end{pmatrix} \begin{pmatrix} i \\ v \end{pmatrix} \tag{3.2}$$

Alternatively, a high-order single-variable ordinary differential equation such as that in (3.1) can be expressed as a first-order multivariable system of ordinary differential equations such as that in (3.2). In other words, a high-order ordinary differential equation representation can be substituted by the state-space representation $\dot{X(t)} = F(X(t), t)$. For example, the parallel RLC resonant circuit shown in Fig. 3.1 can also be expressed in state-space representation terms as (3.2), where $X(t) = (X_1(t), X_2(t), ..., X_n(t))$ is the state vector, $X_0$ is the initial condition, and $\dot{X(t)}$ denotes the derivative of $X(t)$ with respect to time. $F(X(t), t) = (F_1(X(t), t), ..., F_n(X(t), t))$ is an n-dimensional map $F : \mathbb{R}^n \times \mathbb{R}^+ \rightarrow \mathbb{R}^n$ , and this is referred to as a vector field because it denotes the direction and speed of a trajectory at every point in the state space and at every instant in time [28]. Then, the possible solutions or responses of the system are all possible trajectories over the vector field in the state space Then, the possible solutions or responses of the system are all

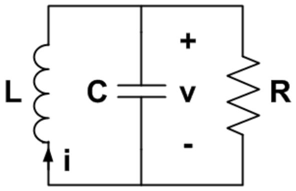


Figure 3.1. Parallel RLC circuit.

possible trajectories over the vector field in the state space.

In this state space model, the problem of detecting global convergence failures in mixed-signal systems is formulated in terms of finding any undesired set of converging trajectories in the state space. In the state space model, a steady state corresponds to a limit set, from which a trajectory does not escape once the trajectory meets any of the points in the set [19]. Moreover, if a limit set attracts nearby trajectories, it is called an attractor (Fig. 3.2), and all state points in the region of attraction (RoA) are called a basin of attraction. Other than attractors, a system may have an unstable limit set which does not attract any trajectories but can easily escape from the limit set by any slight perturbation. However, such an unstable limit set is of no practical interest, as a physical system will never stay in any unstable limit set due to the inherent noise in the system. Thus, the problem of detecting a global convergence failure in the mixed-signal system (*i.e.*, a global convergence analysis) can be formulated under a state space model as a problem of finding any unintended attractor in the state space.

Thus, the existence of global convergence failures can be detected by collecting the multiple circuit simulations, each of which starts from a different initial condition, visualizing the responses as trajectories in a state space, and finding a problematic steady state that attracts some of these trajectories. Although it is generally difficult analytically to construct a phase portrait of a system and to find any split of possible trajectories over this vector field, a simulation trajectory is easy to acquire by a numerical transient simulation. Thus, instead of analytically working on a vector field, we can run a number of transient simulations with randomized initial conditions, periodically take snapshots of trajectories that progress toward steady states, visualizing them in the state space and detect global convergence fail-

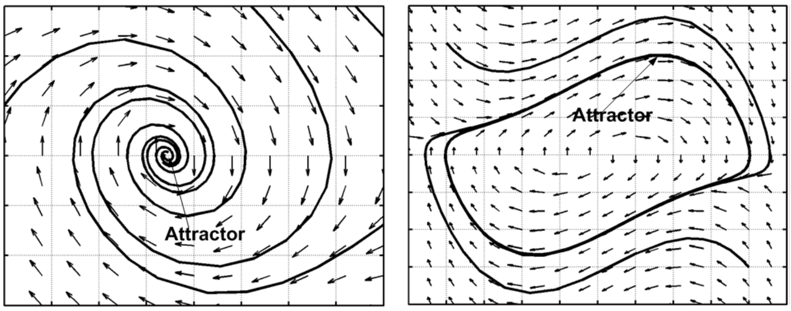


Figure 3.2. Second-order dynamic systems with an equilibrium (left) and a periodic steady state (right).

ures if more than one cluster exists in the state space (*i.e.*, more than one attractor). For example, Fig. 3.4 illustrates the time-progression of the coupled ring oscillator shown in Fig. 3.3, which may have more than one convergence mode depending on the size ratio of its primary inverter W1 and coupling inverter W2. Each point in the figure denotes a unique state of the oscillator consisting of its internal node voltages (n1, p2, and n3, as marked in Fig. 3.3). At time 0, the oscillator starts from an arbitrary initial condition, which is illustrated by the random distribution of the sample points in the state space (Fig. 3.4 (a) and (d)). If the circuit has only one convergence mode, then all points move toward a single steady-state trajectory, which in this case is a closed loop, corresponding to an oscillation (Fig. 3.4 (b) and (c)). However, if the circuit has more than one convergence mode, some points will move towards a different steady-state trajectory, as shown in Fig. 3.4 (e) and (f). In this case, the points form two closed loops, indicating that the circuit has two possible oscillation modes, one of which is undesired. Therefore, global convergence failures can be found by running a large number of sample simulations with randomized initial states and determining if any trajectory is attracted to an undesired region in the state space.

The converging trajectories have disjoint regions of convergence, each of which is a connected set (i.e., a trajectory that never discontinuously jumps or crosses over another region). This assumption is valid for broad classes of circuits that meet the Lipschitz condition. Under this condition, when the system is without a global convergence failure, any trajectory will gradually converge to the same region

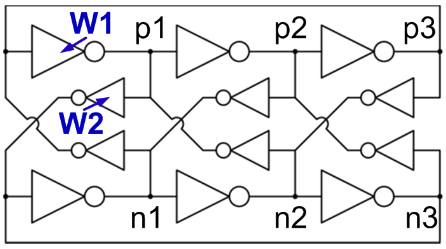


Figure 3.3. Coupled ring oscillator.

where the desired attractor resides. This means that the samples of a number of trajectories that started from random initial conditions should form a single cluster throughout the convergence. Or, when there is more than one attractor in a system, trajectories will increasingly converge towards separate regions in the state space in which different attractors reside. Thus, the samples of the trajectories will finally form more than one cluster in the space. This continuity condition in a region of convergence can be guaranteed under the Lipschitz condition. That is, if vector field $F(X(t), t)$ is piecewise continuous and satisfies (3.3) with some $L \; > 0$ , the system has a unique solution over $[t_0, t_1]$. In such a case, two trajectories cannot cross each other [19], and a trajectory in one RoA is always connected and never crosses any other trajectory that converges to a different steady state. In other words, trajectories in different RoAs will converge to different attractors and the gap between them will usually widen as time goes on (Fig. 3.4). Given that most circuits are considered to have a unique transient solution for every possible initial value, this Lipschitz condition is valid for broad classes of circuits that can be simulated [29].

$$\|F(X,t) - F(Y,t)\| \leq L\|X - Y\|, \forall X, Y \in \mathbb{R}^n, \forall t \in [t_0, t+1] \tag{3.3}$$

This task of finding any undesired attractor in the state space can be simplified further to a problem of detecting a split among sampled states of trajectories in the space. A mixed-signal system is generally supposed to converge into one correct steady state after start-up, meaning that the samples of all trajectories should form a single cluster (i.e., sampled state points stay close to each other) throughout the convergence process (as shown in Fig. 3.4 (a)-(c)). However, if a system has another steady state in its state space, more than one cluster will be formed by different

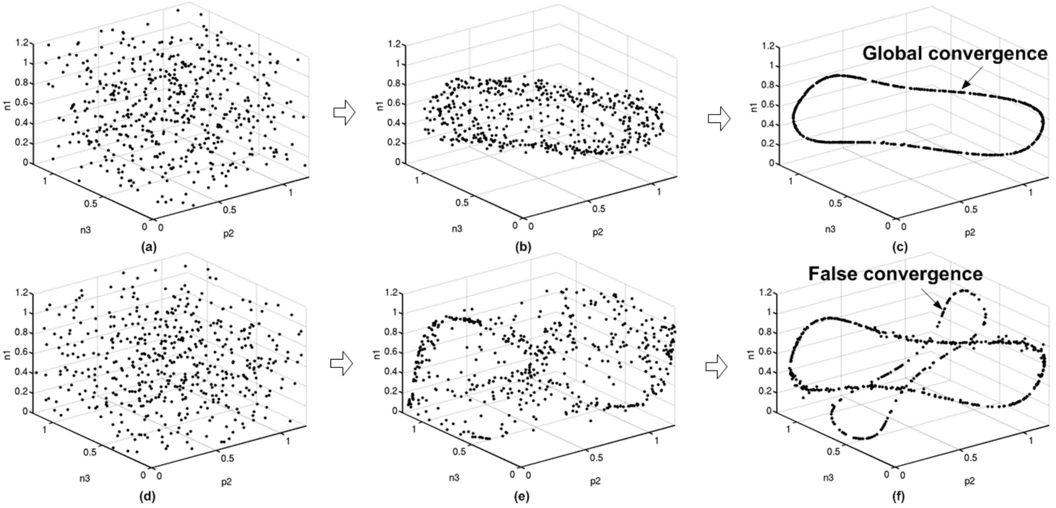


Figure 3.4. Illustration of the detection of the global convergence failure in a coupled ring oscillator in Fig. 3.3 using the cluster-split detection (CSD) algorithm. All of the sampled states will form a single cluster throughout convergence into an attractor when there is no global convergence failure (top). Otherwise, a failure can be detected when the sample points, starting from a uniform random distribution of the circuit states (n1, p2, n3), progress over time and form more than one distinct cluster (bottom).

attractors as trajectories move toward different regions; they will finally split in the state space – samples will form several distant sets of points in the space and there will be more than one cluster in the state space at the moment of the split (Fig. 3.4 (d)-(f)). Thus, we do not need to identify all possible attractors but only to find if there is more than one attractor. We refer to this simplified analysis as cluster-split detection (CSD) in this paper.

However, for the CSD to take multiple samples from each trajectory, the system is required to be autonomous; a systems vector field does not depend on time (i.e., the vector field of a system is described by $\dot{X}(t) = F(X)$ not depending on $t$). Nevertheless, this autonomosity assumption (*i.e.*, a time-invariant vector field) is valid for many circuits. First, as long as the vector field does not change with the external inputs, a circuit can be represented by an autonomous dynamic system. For example, the ring oscillator shown in Fig. 3.3 is an autonomous dynamic system because it can be described by a set of differential equations without any of the coefficients in the differential equations being time-dependent.[3] In contrast, when a system contains any time-variant component such as a time-dependent pulse generator, the system becomes a non-autonomous dynamic system. However, an n-dimensional non-autonomous dynamic system can be transformed to an (n+1)-dimensional autonomous system by appending time as an additional state variable [30]. Furthermore, if the vector field is periodic with period $T$, an n-dimensional non-autonomous system can be converted into an (n+1)-dimensional autonomous system by appending an extra state, $\theta = 2\pi\frac{t}{T}$, which is given by the (3.4):

$$\begin{aligned} \dot{X(t)} &= F(X(t), \frac{\theta T}{2\pi}) \\ X(t_0) &= X_0 \\ \theta &= \frac{2\pi}{T} \\ \theta(t_0) &= \frac{2\pi t_0}{T} \end{aligned} \tag{3.4}$$

For instance, if the reference input clock to a PLL is given as an ideal time-dependent pulse generator, the PLL contains time-dependent components. Therefore, the system becomes non-autonomous. However, when the pulse is periodic with $T$, the PLL can be transformed to an autonomous system by appending the extra state $\theta = 2\pi\frac{t}{T}$. In addition, as the vector field is a function and the state vector is uniquely defined at every state point in the autonomous system, the non-crossing property of a Lipschitz system is preserved in the autonomous system as well [28]. In this work, we regularly take samples from trajectories with the same periodicity of the input pulse (*i.e.*, $t$ in (3.4) is set to integer multiples of $T$), such as the reference clock period in PLL, allowing us to ignore this extra state variable.

### 3.2.2 Finding global convergence failures by cluster-split detection

#### Cluster-split detection by K-nearest neighbor cluster analysis

As explained earlier, a global convergence failure can be detected by finding a split among the sampled states of the trajectories in the space. While this split

[3] Here, we see an oscillator as a nonlinear time-invariant system instead of a linear time-varying system, and thus the oscillator is classified as autonomous [19].

can be easily detected by a human visual inspection for a small-scale circuit whose state space does not span more than three dimensions, an approach relying on the visual inspection quickly becomes infeasible as the circuit size grows. First, the dimensionality of the circuits state space increases, making it difficult to visualize the sample point distribution. Moreover, the computational burden involved in collecting the sample points also increases sharply, as the number of points required and the time it takes for each sample simulation both increase with the circuit size.

Thus, we have proposed the use of data cluster analysis to mitigate these difficulties [26]. Clustering analysis classifies the points distributed in a high-dimensional space into disjointed groups [12]. Hence, the detection of a convergence failure does not have to rely on a visual inspection. In addition, the analysis may be able to detect a failure before it is visually discernible, thus shortening the simulation time. This reduction can be significant because a trajectory tends to converge exponentially to its steady state [19, 23, 31].

Specifically, the K-nearest neighbor clustering algorithm is proposed to detect a split among sample points; the algorithm detects a split by making connections among points according to the nearest neighbor relationships and determining whether every pair of points can reach each other afterwards (Algorithm (1)). The proposed algorithm first constructs a connectivity graph by mapping each point in the state space to a node and forming an edge between every two nodes that consider each other as one of the $i-th$ nearest neighbors ( $i \leq K$ ; called the K-nearest neighbor in this paper). This work uses the Euclidean distance as a similarity measure to determine the nearest neighbors; however, other reasonable measures can also be used. Then, connected components in the graph are found by a graph analysis (*i.e.*, by the depth-first search) and each component is returned as a cluster. If there is more than one connected component, more than one cluster exists and therefore a split is detected. For example, Fig. 3.5 illustrates the flow of the algorithm when it is applied to the example points shown in the figure. That is, it first constructs a graph where each point in the state space is mapped to a node in a graph (Fig. 3.5 (a)), and a directed edge is formed from node A to B if B is one of K number of nearest neighbors of A (Fig. 3.5 (b)). Next, edges with only one direction are removed according to the mutual condition (Fig. 3.5 (c)). Then, connected components in

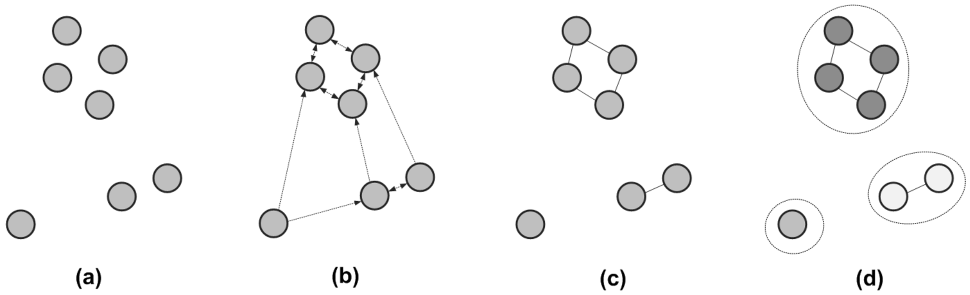


Figure 3.5. K-nearest neighbor clustering (K=2).

the graph are found, and they are reported as clusters (Fig. 3.5 (d)).

In the proposed cluster-split detection process, the K-nearest neighbor clustering algorithm (referred to as KNN-clustering) is performed in units of trajectory by taking multiple samples from each trajectory and connecting samples from the same trajectory in the first phase of the algorithm. This has the effect of reducing the false split detection rate because any temporal deviation of samples would not result in false splits as long as any of the previous samples remain close to any of the other trajectories in the same region of attraction. This would be especially helpful when a trajectory enters a contraction region where the trajectory moves at an exponential rate [31], because the samples from the trajectory would have a much wider gap when the trajectory passes this exponentially converging region. Moreover, this increases a number of available samples and thereby suppresses the probability of concluding a false split due to the small number of samples. For the first ring oscillator example shown in Fig. 3.3, the false split detection rate was as high as 58% when only one sample was used but was lowered to almost 0% when the number of samples was increased to two.

One of the merits of the proposed K-nearest neighbor clustering algorithm is that it can obtain the required parameter $K$ if the number of data objects and their dimensionality are given. Generally, a cluster analysis is not an automatic task that universally works; instead, it typically depends on parameters that need to be iteratively tuned to obtain the best results. For example, a cluster analysis may require the number of expected clusters, a distance function to use or a density threshold, which should be properly given in accordance with the properties of the data set

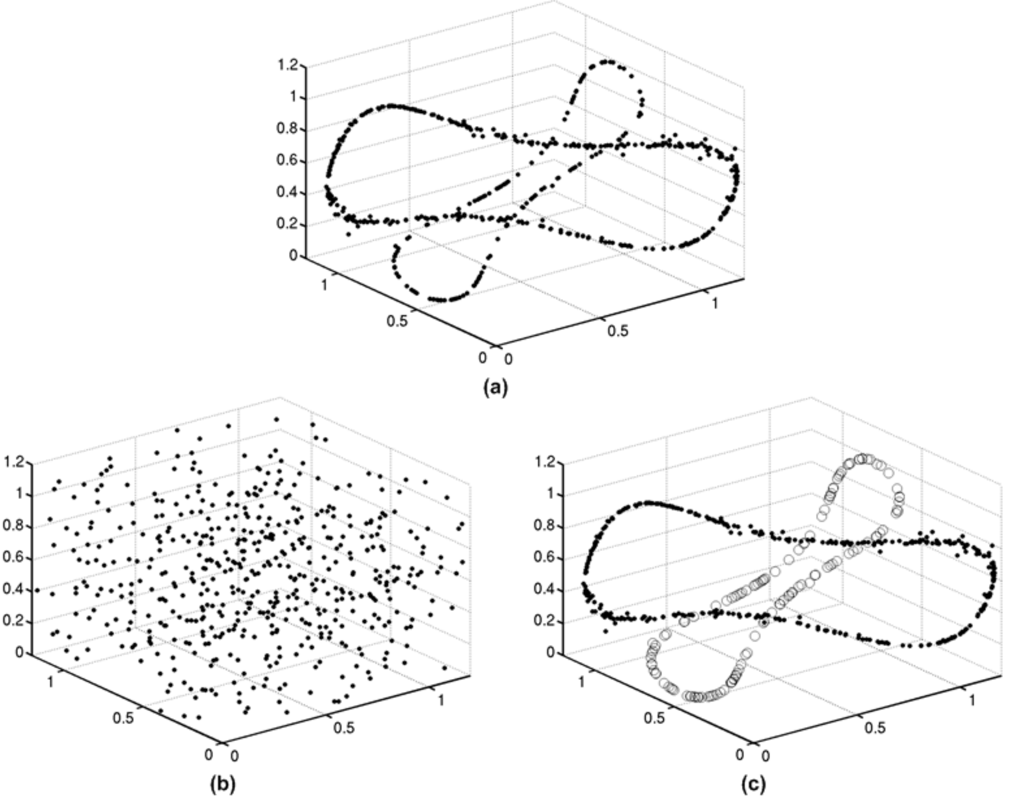


Figure 3.6. K-nearest neighbor clustering with K-parameter extraction (N=512,K=15).

and the purpose of clustering [12]. However, the proposed K-nearest neighbor clustering algorithm can automatically set its required parameter $K$ from the number of data objects and their dimensions [26]. That is, if the dimensionality is $D$ and the number of points is $N$, $K$ can be set to the required minimum to connect uniformly distributed $N$ points in D-dimensional space. If there are two sets of points that are separate enough and if $K$ is derived as above, K-nearest neighbor clustering will classify two sets into different groups because a higher number of neighbor connections would be required to connect all the points into one cluster. Thus, the K-value can be extracted before K-nearest neighbor clustering by (1) pseudo-randomly generating N number of D-dimensional data points and (2) incrementally searching for the minimum $K$ that connects the random points as one cluster. For example,Fig. 3.6 visually illustrates how the $K$ parameter is determined, and K-nearest neighbor clustering is performed on 512 samples (Fig. 3.6 (a)). To acquire the proper $K$ value, 512 pseudo-random samples are first generated in the space, as shown in Fig. 3.6 (b), and the minimum required $K$ to group them as one cluster is found (i.e., 15). Next, 15-nearest neighbor clustering is performed, and this process finally detects distinct clusters in the state space (Fig. 3.6 (c)). However, this incremental search process demands that KNN-clustering be run multiple times, which is overhead. Thus, we pre-generated $K$ while varying $N$ and $D$, with $K$ estimated later by second-order polynomial regression to mitigate the cost of estimating the required parameter $K$. To be specific, when $D$ and $N$ values are given, the pre-generated data with the closest dimensions are selected first; assuming that the relationship between $K$ and $N$ could be approximated by a linear regression model, $a_0 + a_1N + a_2N^2$, the coefficients of this second-order polynomial function are fitted to the selected data by minimizing the sum of squared error function [11]. The $K$ value is then computed by inserting a new $N$ value to this second-order polynomial expression.

### Dimensionality reduction

The required distance to distinguish two distant sets of points in a space by the cluster analysis may grow as the circuit size increases because the size of the state space is proportional to the circuit size. Thus, the CSD's gain from detecting a cluster split early - before all trajectories reaches a steady state - could be reduced

**Algorithm 1** Trajectory-based K-Nearest Neighbor Clustering.

```
Require: samples of trajectories
1: Construct the initial graph by creating a node for each sample of trajectories
   and by forming edges among samples of the same trajectory.
2: Estimate $K$ parameter by second-order polynomial regression on the pre-
   generated $(D, N, K)$ data with the given dimensionality $D$ and the number of
   samples $N$.
3: For every sample, select K-nearest neighbors. When computing the distances
   between samples, the Euclidean distance is used. If both samples consider each
   other as one of their K-nearest neighbors, form an edge between them in the
   graph.
4: Find connected components in the graph.
5: return the connected components
```

in large circuits.

However, the effective dimensionality of states in the space can be much lower than the number of circuit nodes [32] and we propose a reduction in the dimension of the samples by eliminating all of the circuit state variables that have converged to a specific value. In several studies [26,33,34], this determinate circuit node is found by measuring the circuit node's nodal voltage distribution after short-period simulations starting from random initial conditions. If a node has settled to a certain value, such as node A in Fig. 3.7, the node is considered to be determinate. However, if a node has not settled to any value, like node B in Fig. 3.7, the node's state is uncertain and it is said to be in an indeterminate state. To distinguish these indeterminate nodes from determinate nodes and to quantify their degree of uncertainty, the entropy measure in (3.5) can be used. If node $A$ settles to a certain value regardless of its previous condition (i.e., a determinate circuit node), its entropy $H(A)$ would be 0. Otherwise, the entropy would take a positive, non-zero value – the higher it is, the more uncertain the node is. The nodes with zero entropy are referred to as non-X nodes and those with positive entropies are referred to as X nodes.

$$H(A) = \Sigma_{a \in A} p(a) \log \frac{1}{p(a)} \tag{3.5}$$

Thus, before the periodic cluster analysis to detect a split, we computes the entropy of every nodal voltage distribution, finds any non-X circuit nodes, and

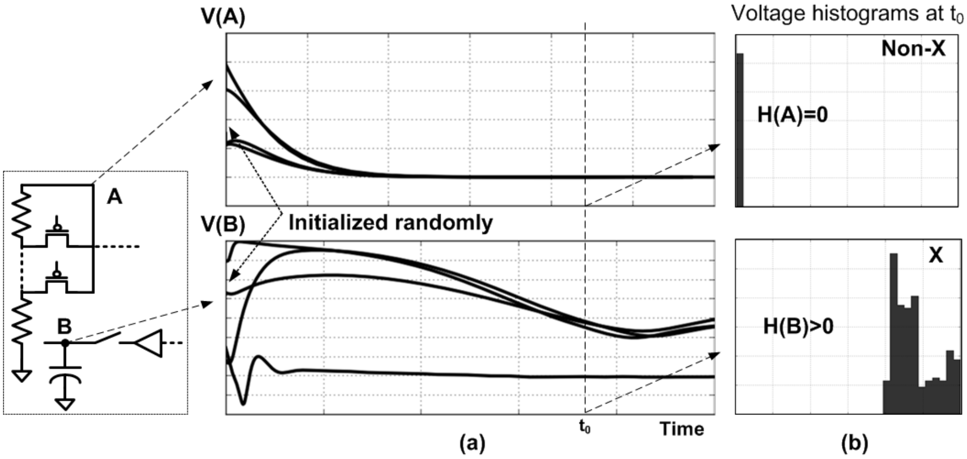


Figure 3.7. (a) Running transient simulations with randomized initial conditions, examining distributions at $t_0$ and (b) measuring its entropy to find a non-X node, $A$, and an X node, $B$.

reduces the dimensionality of the samples by removing all non-X variables from consideration. The axes of these variables in the space can be ignored because they have zero variance in their distributions in the space. The well-known principal component analysis (PCA) [11, 12] would also eliminate the non-X circuit variables from the principal axes, as there is zero variance along the directions of these non-X axes as well. With this reduction, the computational overhead of the KNN-clustering can be reduced and the algorithm can detect a cluster split earlier. That is, the CSD periodically examines the distributions of states by KNN-clustering, and before every cluster analysis, the required parameter $K$ is estimated. If a higher value than the effective dimensionality of the data points is used at this estimation, it unnecessarily increases the $K$ value and thus delays the detection of the cluster split. For example, this entropy-based dimensionality reduction helped the CSD to detect a global convergence failure in the PLL example in Section 4 (Fig. 3.15 (a)) more than 10 times faster.

**Trajectory merging**

The cost of periodically running at least hundreds of parallel circuit simulations is high and thus could limit the scalability of the proposed CSD algorithm. However, it should be noted that not all trajectories that are close to each other need to be simulated. In addition, as the approach relies on covering the continuous state space with a finite number of points, it is better to merge close trajectories to maintain a uniform distance among them. Otherwise, a temporal dense region around close trajectories may cause a false split [26].

Therefore, we propose to merge any two close trajectories during the simulation by determining the K-nearest neighbor relationships and the angles among trajectories (Algorithm (2)). That is, if every sample of two trajectories has at least one K-nearest neighbor in another trajectory, the two trajectories are considered to be very close to each other and the angles between the difference vectors of samples in the trajectories are measured. If all angles are lower than the given threshold (0.2 rad in this work), the two trajectories are classified as a candidate merging pair. After comparing all of the trajectories, a graph is constructed by creating a node for each trajectory and forming an edge between nodes in the candidate merging pairs (Fig. 3.8 (b)). Then, a set of trajectories that can be merged into one trajectory is found by finding the connected components in this graph. Among the nodes in each connected component, the node with the maximum number of neighbors is selected as a merger that represents all of the other trajectories in the component. For instance, in Fig. 3.8, trajectories A and B, B and C, and E and F are grouped as candidate merging pairs and trajectories B and E are chosen as a merger. However, to suppress the probability of a false split due to trajectory merging, another trajectory proximity graph is constructed. This trajectory graph is built by creating a node for every trajectory and forming an edge between two trajectories when any of their samples are K-nearest neighbors to each other (Fig. 3.8 (c)). Then, a merging operation is canceled if eliminating a trajectory causes any division in the trajectory proximity graph (Fig. 3.8 (d)). For example, trajectories E and F in Fig. 3.8 are not merged because removing either of them from the proximity graph disconnects the graph.

The gain from the trajectory merging operation would be substantial because a trajectory tends to converge exponentially to a steady state [19, 23, 30, 31]. Generally, the average convergence rates are governed by Lyapunov exponents [19, 30]. Lyapunov exponents are a generalization of the eigenvalues at an equilibrium point

**Algorithm 2** Trajectory Merging.

```
Require: samples of trajectories
Construct the trajectory proximity graph by creating a node for every trajectory and forming an edge between two trajectories when any of their samples is a K-nearest neighbor to each other.
for every pair of trajectories do
    Two trajectories are considered to be ‘close’ when every sample in one trajectory is a K-nearest neighbor of a sample of another trajectory.
    If two trajectories are ‘close’ and all angles between difference vectors of two trajectory samples are smaller than the given threshold (e.g., 0.2 rad), insert the pair into the list of candidate merging pairs.
end for
Construct the merging graph by creating a node for each trajectory and forming edges between nodes in the candidate merging pairs.
Find connected components in the merging graph.
for every connected component do
    Choose the trajectory with the maximum number of neighbors as a merger.
    for all other trajectories in the component do
        if the removal of the trajectory does not divide the trajectory proximity graph then
            merge the trajectory into the chosen merger and remove mergees from the trajectory proximity graph.
        end if
    end for
end for
```

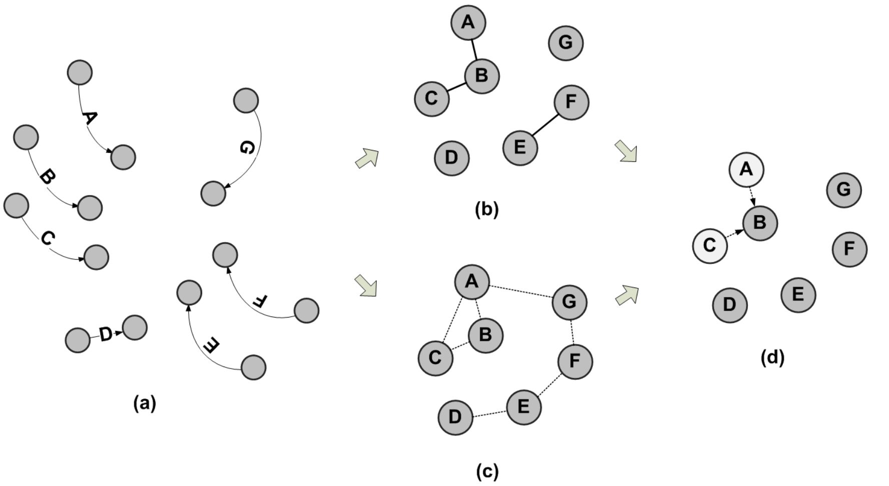


Figure 3.8. Proposed merging method on the examples in (a): First, (b) a merging graph is constructed by examining the K-nearest relationships among trajectory samples and the angles between trajectories. Next, (c) a trajectory proximity graph is built and (d) trajectories are merged if they are connected in the merging graph and if removing them does not cause a split in the trajectory proximity graph.

in a linear system and indicate the rate of contraction along a trajectory. That is, the local behavior of the vector field is determined by the linearized dynamic in (3.6) where $x_0$ is the initial condition of the trajectory and $\Phi_t(X_0)$ is the linear time-varying state transition matrix. Then, the Lyapunov exponents $\lambda_i$ are defined by (3.7) where $\sigma_i(t)$ denotes the singular values of the state transition matrix at time t (*i.e.*, $\Phi_t(X_0)$) and every point in the basin of attraction of an attractor has the same Lyapunov exponents. For DC, periodic and quasi-periodic steady states, it has been shown that Lyapunov exponents are less than or equal to zero [19, 30] Thus, for most circuits whose steady state is not chaotic, trajectories will converge exponentially to limit sets on average [19, 30]. Fig. 3.9 shows the rate of trajectory merging in the PLL and the DC-DC converter examples in Section 3.2.3, indeed showing exponential convergence rates and showing the high level of effectiveness of the proposed method.

$$x(t) = \Phi_t(X_0)x_0 \tag{3.6}$$

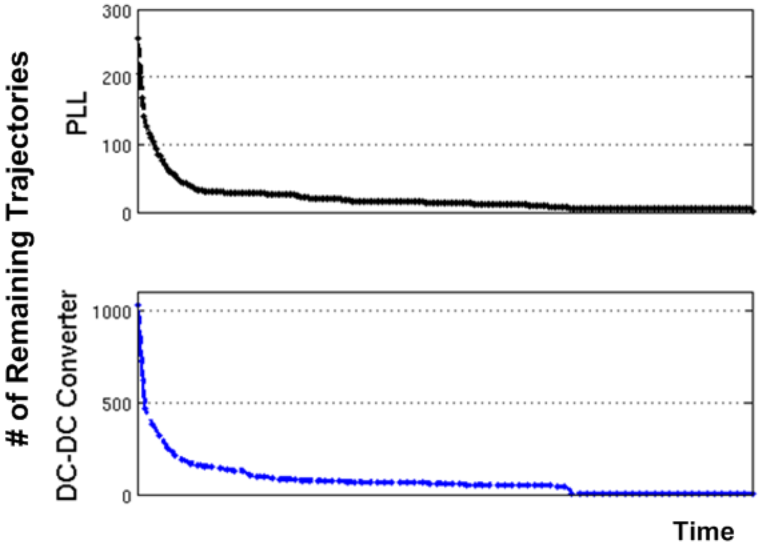


Figure 3.9. Number of remaining trajectories after merging at each time step of the CSD in (top) PLL and (bottom) DC-DC converter.

$$\lambda_i = \lim_{t \to \infty} \frac{1}{t} \ln \sigma_i(t) \tag{3.7}$$

### Termination criteria

For the cluster-split detection process to be complete as a global convergence analysis[4], convergence to a single steady state needs to be detected in parallel to failure detection through a cluster analysis. Although convergence can be detected upon the definition of the limit set [26], this criterion requires all trajectories to converge to a steady state, and it could take a very long time for convergence in some cases, such as in a PLL circuit. Moreover, because this relies on the approximated set of the limit region in the space, it would require an exponentially growing radial distance to approximate the occupied space around the simulated trajectories as the dimensionality of the circuit's state space increases.

Thus, the termination criterion is better to be revised – it concludes the global convergence of a circuit when there remains no uncertainty in the trajectories' vis-

[4] In other words, global convergence property checking.

iting states at the moment. First, the entropy-based uncertainty analyses run for dimensionality reduction is regularly used and global convergence is concluded when there remains no X, as the absence of X means that all trajectories will follow the same path afterwards [26, 33, 34]. Also, a system is deemed as globally converging when all the trajectories merge into one.

**The cluster-split detection algorithm for global convergence property checking**

The overall cluster-split detection algorithm is outlined in Algorithm (3). Basically, the algorithm works on a set of concurrently running multiple trajectories of simulated responses, each starting from a different initial condition. The number of initial conditions to run is initially determined according to (2.5) such that the result can bound the global convergence failure probability to be no greater than the given probability $P_f$ with a certain confidence when no split is detected. Then, the proposed KNN-clustering is periodically applied to detect a split and a global convergence failure is concluded when $M$ number of consecutive splits is detected (in the following experiments, $M$ was set to 2). If no split is found, any close trajectories are merged to reduce the simulation overhead. If every trajectory merges into one or if there remains no X, the system is deemed to be globally converging.

The advantage of the proposed CSD is that the approach is not limited to a certain circuit topology or a class of steady state. The method can detect a global convergence failure as long as any of the problematic attractors is separated enough from the desired steady state in the state space, and it can find a failure as soon as the trajectories move apart to different regions, shortening the time to detect a failure. Additionally, even if the system has no failure, the algorithm reduces the overhead of verifying multiple initial conditions by merging any close trajectories while the analysis runs. Also, the method is not limited to small-scale circuits with a certain topology, such as oscillators, because the approach is applicable to any circuit that can reach a steady state within a practical time through a numerical circuit simulation. Lastly, because the method involves a sample-based analysis, it avoids the curse of dimensionality even when a circuit is large [11]. In other words, the required number of initial conditions in (2.5) only depends on the given confidence level and the failure probability bound regardless of the dimensions of the circuit's state space.

**Algorithm 3** Global Convergence Property Checking by Cluster-Split Detection.

```
1:  Compute the entropy of every circuit nodal voltage distribution after a number of simulations with random initial conditions (e.g., run short transient simulations from 512 random initial states), find the non-X nodes and remove their dimensional axes from the space to project the trajectory samples later at step 6 (i.e., during the KNN-cluster analysis).
2:  Determine the number of initial conditions ($N$) according to (2.5) for the required confidence level and the failure probability bound.
3:  Generate $N$ pseudo-random initial conditions.
4:  Run short transient circuit simulations from the latest states. If it is the first run, start from the previously generated initial conditions
5:  Compute the entropy of every circuit nodal voltage distribution after the simulations and find the X nodes. If there remains no X, conclude global convergence.
6:  Apply the cluster analysis in Algorithm (1).
7:  If more than one cluster is reported $M$ consecutive times, conclude global convergence failure.
8:  Merge nearby trajectories according to Algorithm (2).
9:  If all trajectories are merged into one, conclude global convergence.
10: Run $t_{step}$ duration transient circuit simulations from the last states and go back to  4.
```

### 3.2.3 Experimental results

**Coupled ring oscillator**

The described cluster analysis was applied to a three-stage coupled ring-oscillator implemented in a 130nm CMOS process (Fig. 3.3). This oscillator can have various start-up failures depending on the sizes of the primary buffer ($W1$) and the coupling buffer ($W2$) [26]. Fig. 3.10 illustrates the possible failure modes for different $W1$:$W2$ ratios with the final state distributions. Fig. 3.10 (a) shows that the oscillator has only one mode of oscillation, whereas in Fig. 3.10 (c), it has two. Fig. 3.10 (e) and (g) show that stable DC equilibrium points (i.e., no oscillations) can also exist.

The proposed CSD was applied to all four oscillator examples. That is, the number of random initial conditions ($N_{init}$) to cast initially were set to 128, 256 and 512, which bounded the global convergence failure probability to be no greater than

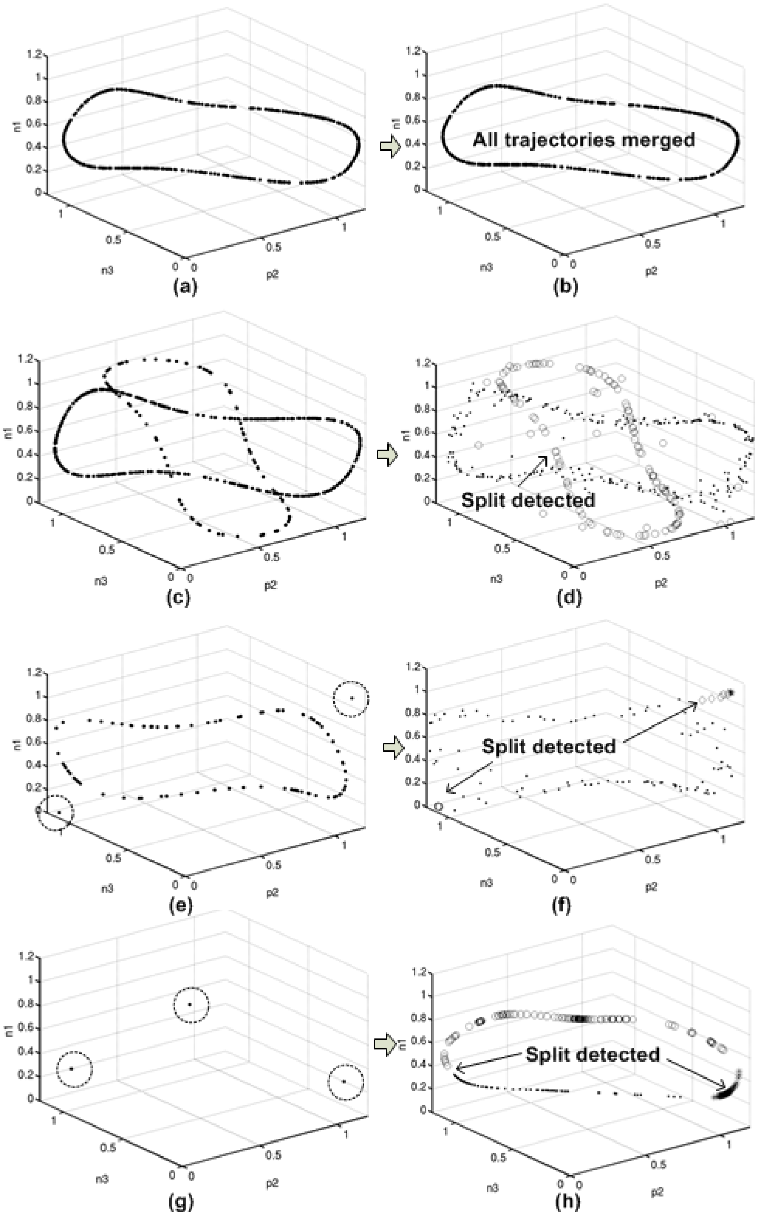


Figure 3.10. Visualization of all the possible steady-state solutions in the state space and the final state distributions of the coupled ring oscillator after the CSD algorithm when the W1:W2 ratios are (a, b) 2:1 (c, d) 8:1 (e, f) 1:8 and (g, h) 1:1.

2.3%, 1.78% and 1.34% with 95%, 99% and 99.9% confidence levels, respectively. Then, $N_{init}$ transient circuit simulations from the different initial conditions were run for 93 *ps* in parallel and KNN-clustering was applied to the final states of the simulated trajectories. As no split was found at the first trial of the cluster analysis in all four cases, the algorithm performed an entropy-based uncertainty analysis and merged any close trajectories. In all four examples, there remained uncertainty (*i.e.*, X) in a circuit and not all trajectories were merged after the first iteration. Thus, further simulations were run for another 93 *ps* from the states the remaining trajectories lastly visited and the above steps were iterated until a split is found (Fig. 3.10 (d), (f), and (h)) or until all trajectories were merged into one (Fig. 3.10 (b)). Fig. 3.10 (d), (f) and (h) illustrate the distribution of the final states at the time of termination, where each cluster is marked by a different shape. (Fig. 3.10 (a) plots the trace of the final remaining trajectory for a while after the termination.) Compared to the distributions shown in Fig. 3.10 (c), (e) and (g), the proposed cluster analysis was able to determine the cluster split before the state points reached the final trajectories.

The experimental results are summarized in Table 3.1. For this simple circuit, our algorithm correctly detected the existence of global convergence failures in all cases, although the number of final clusters ($N_{clt}$) reported was sometimes larger than the correct value. However, this was not problematic because our aim in the global convergence analysis is to detect the presence of a global convergence failure in a circuit.

Although the algorithm takes a certain amount of execution time ($T_{exec}$), as shown in Table 3.1, the proposed cluster-split detection has advantages over the brute-force method of simulating a circuit. First, cluster-split detection provides clear criteria to determine whether a circuit contains a global convergence failure or

Table 3.1. Coupled ring oscillator analysis results

| W1:W2 | 2:1 | | | 8:1 | | | 1:8 | | | 1:1 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # of initial conditions ($N_{init}$) | 128 | 256 | 512 | 128 | 256 | 512 | 128 | 256 | 512 | 128 | 256 | 512 |
| Simulation time ($T_{sim}; ps$) | 930 | 2050 | 2050 | 558 | 651 | 930 | 372 | 372 | 372 | 372 | 465 | 465 |
| # of clusters reported($N_{clt}$) | 1 | 1 | 1 | 2 | 2 | 2 | 15 | 15 | 6 | 3 | 7 | 6 |
| Execution time ($T_{sec}; sec$) | 123 | 291 | 673 | 126 | 243 | 571 | 113 | 299 | 472 | 108 | 225 | 486 |

not. If one depends on circuit simulations to detect global convergence failures, it is uncertain as to when the circuit will finally converge to its steady state; therefore, the minimum transient simulation time becomes difficult to determine. Second, the algorithm detects the failure before it is visually discernible, shortening the simulation time. In this example, the oscillator could take nearly 1 $ns$ to reach a steady state in the worst case but the CSD was able to detect the failure before 1 $ns$, as shown in Table 3.1. The gain from the early detection of a global convergence failure would be significant, especially when a circuit's transient simulation takes more time – in other words when the circuit's size is larger than this simple oscillator example.

### Dual-delay-path ring oscillator

The proposed algorithm was also assessed on a four-stage, dual differential delay-path ring oscillator, which is composed of eight circuit nodes and implemented in a 45nm CMOS process (Fig. 3.11). This oscillator reportedly has a possible false oscillation mode [25]. The existence of such a mode depends on the size ratio of the primary device ($W1$), the auxiliary device ($W2$), and the latch device ($W3$). For instance, if the auxiliary devices are too strong or the latch devices are too weak, common-mode oscillation may occur, where the $A$ and $A_b$ nodes oscillate in phase. However, even for the same oscillator, differential-mode oscillation may still occur depending on the initial states, where the $A$ and $A_b$ nodes oscillate out of phase (Fig. 3.12).

We applied our algorithm to differential four-stage dual-delay-path ring oscillators with different size ratios. With $W1 : W2 : W3$ set to 15:3:1, both common oscillation at 13.1GHz and differential oscillation at 16.8GHz were observed; this was found by detecting more than one cluster (Table 3.2). For the size ratio of 9:4:3, all of the trajectories merged into one trajectory and the algorithm validated that the oscillator can oscillate only in a differential mode. The key results are summarized in Table 3.2.

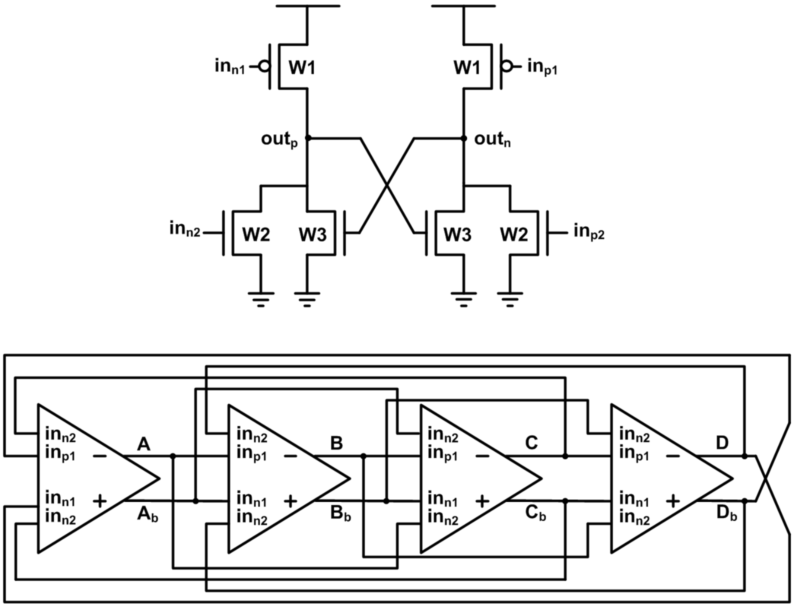


Figure 3.11. Differential four-stage dual-delay-path ring oscillator (bottom) and its unit stage (up).

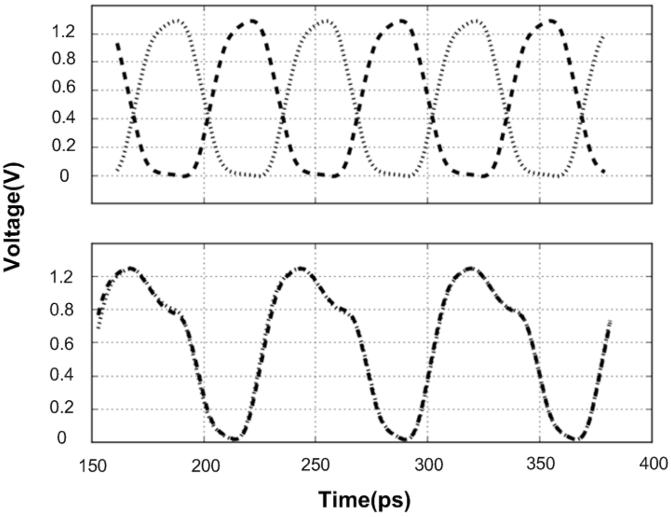


Figure 3.12. Two possible oscillation modes, differential oscillation (top) and common-mode oscillation (bottom), of the differential four-stage dual- delay-path ring oscillator.

### Supply-regulated digitally controlled oscillator

The next example is a supply-regulated digitally controlled oscillator (DCO). It is composed of 145 circuit nodes and implemented in a 45nm CMOS process, and its frequency ranges from 0.6 to 1.9GHz with a five-bit resolution. As shown in Fig. 3.13, the DCO is composed of a digitally controlled resistor (DCR), a replica-based supply regulation loop, and an inverter-based ring oscillator. The replica-based supply regulation loop also acts as a constant-gm biasing loop, which controls the regulated supply voltage *vreg* such that the output resistance of the ring oscillator is equal to the resistance of the DCR. Because the oscillation period varies with the ring oscillator's output resistance, one can tune the DCO's oscillation frequency by adjusting the DCR's resistance through an external digital control method. The differential amplifier used in the supply regulation loop is self-biased (i.e., the bias current level is set by the output of the amplifier itself) and can undergo a start-up failure when the node *vbp* starts from too high and *vbn* starts from too low voltages (Fig. 3.13 and Fig. 3.14)

When our cluster-split detection was applied to this example but the dimensionality of the samples was reduced to 50 after all non-X circuit nodes were removed at the first stage of the CSD algorithm, most of them were intermediate nodes in the resistor ladder (i.e., DCR). The CSD periodically performed KNN-clustering of the trajectory samples in this reduced space and successfully detected the unexpected oscillation mode (Fig. 3.14). In addition, the entropy-based uncertainty analysis constantly reported the problematic node *vbp* as one of the most uncertain X nodes and thereby helped the designer to identify a proper place to add a reset circuitry [34]. After the failure detection process, the DCO was modified to have a proper reset that pulls down this *vbp* at the start, and the algorithm concluded

Table 3.2. Differential four-stage dual-delay-path ring oscillator analysis results

| W1:W2:W3 | 15:3:1 | | | 9:4:3 | | |
|---|---|---|---|---|---|---|
| # of initial conditions ($N_{init}$) | 128 | 256 | 512 | 128 | 256 | 512 |
| Simulation time ($T_{sim}; ps$) | 297 | 396 | 528 | 264 | 297 | 231 |
| # of clusters reported($N_{clt}$) | 2 | 3 | 2 | 1 | 1 | 1 |
| Execution time ($T_{sec}; sec$) | 165 | 353 | 982 | 207 | 284 | 542 |

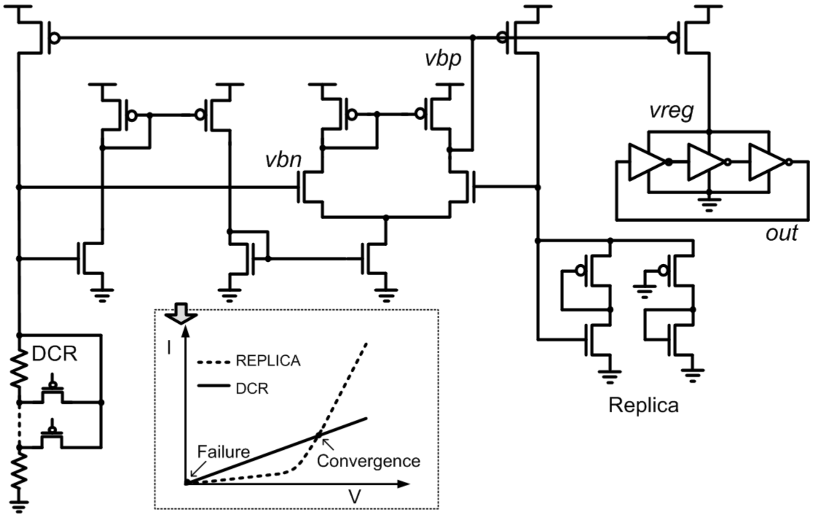


Figure 3.13. DCO with a global convergence failure.

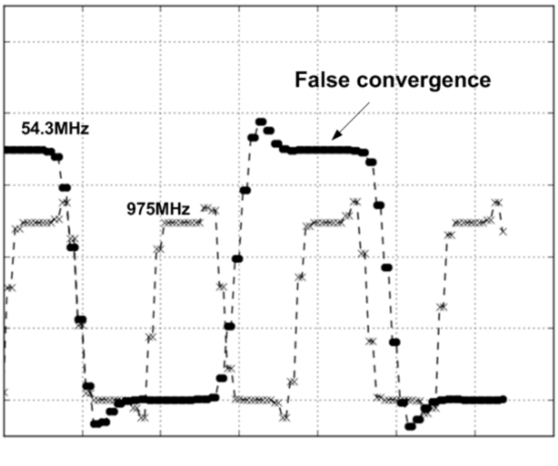


Figure 3.14. Normal oscillation mode and false oscillation mode in the DCO.

Table 3.3. DCO analysis results

| | Without a reset | | | With a reset | | |
|---|---|---|---|---|---|---|
| # of initial conditions ($N_{init}$) | 128 | 256 | 512 | 128 | 256 | 512 |
| Simulation time ($T_{sim}; ns$) | 14.3 | 8.4 | 8.8 | 77.6 | 86.9 | 70.6 |
| # of clusters reported($N_{clt}$) | 2 | 2 | 3 | 1 | 1 | 1 |
| Execution time ($T_{sec}; min$) | 13.8 | 12.8 | 36.1 | 32.9 | 43.2 | 73.2 |

global convergence (Table 3.3).

### Phase-locked loop

The proposed analysis was also applied to PLLs. The PLL circuit implementations were similar to those described in an earlier study [35], and they can have global convergence failures depending on the loop filter type and the maximum speed of the frequency divider. The example PLLs are implemented in a 130nm CMOS process, taking an 833MHz reference clock and generating 0.833-2.44 GHz output clocks with a programmable multiplication ratio.

The first PLL example, denoted as SLOWDIV, is a case with a slow divider (Fig. 3.15 (a)). If the voltage-controlled oscillator (VCO) starts at a high frequency at which the divider cannot operate, the divider may erroneously output a clock with a frequency lower than the reference, after which the phase-frequency detector (PFD) will drive the VCO towards an even higher frequency, failing to acquire a correct lock.

The second PLL example, denoted as SLOWPFD, is illustrated in Fig. 3.15 (b). In this case, when the VCO starts at a high frequency, the divider correctly operates but the phase- frequency detector (PFD) fails to provide a net output directing the VCO towards the desired frequency. This occurs due to the use of a sample-and-reset loop filter (LF) [35] and because of the finite reset period of the PFD. When the clock edge arrives at the PFD while it is resetting, the flip-flop based PFD ignores the edge and can momentarily generate the opposite polarity output. Its interaction with the sample-and-reset LF can result in multiple null points in the effective PFD transfer response, and the PLL may falsely lock at various frequencies other than the desired frequency.

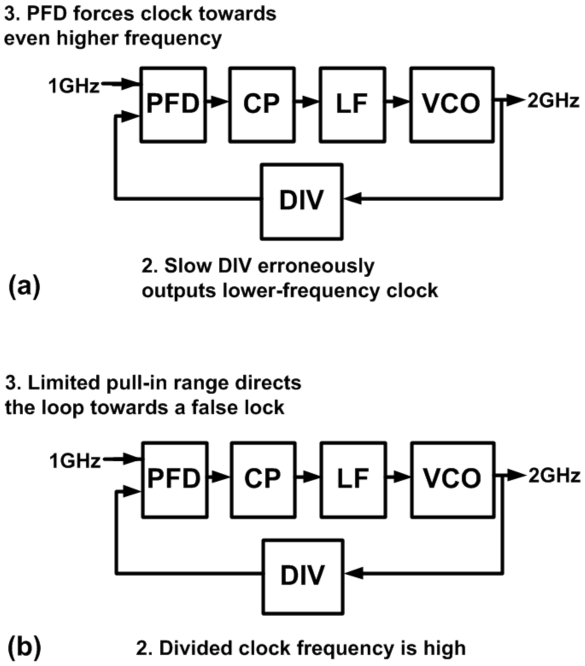

3. PFD forces clock towards even higher frequency
1GHz
PFD
CP
LF
VCO
2GHz
DIV
(a)
2. Slow DIV erroneously outputs lower-frequency clock
3. Limited pull-in range directs the loop towards a false lock
1GHz
PFD
CP
LF
VCO
2GHz
DIV
(b)
2. Divided clock frequency is high

Figure 3.15. PLL start-up failures: (a) when the VCO starts at too high a frequency, the divider may fail to output the correct frequency and the PFD drives the loop away from the correct lock (SLOWDIV), and (b) the slow operation of the PFD generates net zero output at an incorrect frequency (SLOWPFD).

The experimental results with the cluster-split detection algorithms are summarized in Table 3.4. The example PLLs consisted of more than 300 circuit nodes; however, more than 100 circuit nodes were removed after eliminating non-X circuit nodes – most of them the internal nodes of the programmable divider and the CP/LF control logic – and our algorithms correctly detected the existence of global convergence failures in both the SLOWDIV and SLOWPFD PLL examples. Moreover, the CSD was able to validate the NORMAL PLL example, whose divider and PFD were carefully sized to suitably follow a high-frequency clock. Note that the described algorithm detected a global convergence failure very early. In this PLL example, the worst-case lock-time of the PLL was as long as 8 $\mu s$ but the CSD algorithm detected the failures as early as 12 $ns$ and 200 $ns$ for the SLOWDIV and the SLOWPFD examples, respectively ($T_{sim}$ in Table 3.4).

When the algorithm concluded global convergence, the number of remaining simulation trajectories after the proposed merging operation can be used as a measure to check the progress of the CSD algorithm during the analysis. As the circuit simulations proceed, each sample trajectory will gradually converge to the circuit's attractor and the trajectories in the state space will thus become closer. In other words, the trajectories will merge with each other as time goes on and will finally merge into one if the circuit has only one attractor in the state space. Thus, by counting the number of remaining simulation trajectories, we can examine the progress of the CSD – fewer remaining trajectories indicate greater progress of the CSD. For example, Fig. 3.9 (a) shows the number of remaining trajectories for each time step of the CSD in the NORMAL PLL, and one can check the progress of the CSD at each time step according to the ratio of the number of remaining trajectories to the initial number of trajectories.

Table 3.4. PLL analysis results

| | NORMAL | | | SLOWDIV | | | SLOWPFD | | |
|---|---|---|---|---|---|---|---|---|---|
| # of initial conditions ($N_{init}$) | 128 | 256 | 512 | 128 | 256 | 512 | 128 | 256 | 512 |
| Simulation time ($T_{sim}; ns$) | 283 | 250 | 230 | 12 | 39.6 | 360 | 200 | 22.4 | 4.9 |
| # of clusters reported($N_{clt}$) | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 |
| Execution time ($T_{sec}; min$) | 68 | 97.6 | 184 | 8.4 | 31.1 | 49.3 | 45.8 | 24.8 | 20.8 |

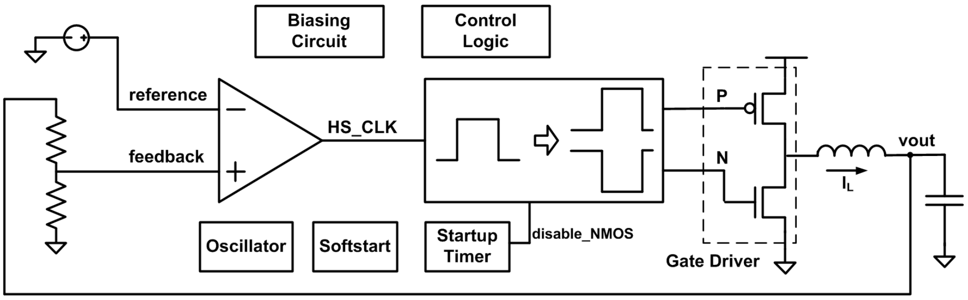


Figure 3.16. Step-down DC-DC converter.

### Step-down DC-DC converter

The last example is a step-down DC-DC converter circuit implemented in a LDMOS process, with 3,295 circuit nodes. It is a high-frequency synchronous buck converter which takes an input voltage ranging from 2.3V to 4.8V and maintains a constant 1.8V voltage output via a 6-MHz, pulse-width modulation control scheme (Fig. 3.16).

Although the example circuit was composed of thousands of circuit nodes, the proposed algorithm was able to reduce the sample dimension to as low as 1,500 - the example DC-DC converter contains many digital logic components for configuration and testability, whose internal circuit nodes quickly settle to certain values (i.e., non-X) - and finally concluded global convergence (Table 3.5). Although the cost of simulating the sample trajectories remained high for this large circuit, the CSD algorithm was able to finish the global convergence analysis within a practical time - the execution time ($T_{exec}$ in Table 3.5) was approximately half a day for 512 initial sample trajectories. In comparison to the previous PLL, the computational time increased by only 3.6-4.9x despite the fact that the circuit size increased by about 9x. This occurred because the proposed CSD was able to reduce the simulation

Table 3.5. DC-DC converter analysis results

| # of initial conditions ($N_{init}$) | 128 | 256 | 512 |
|---|---|---|---|
| Simulation time ($T_{sim}; \mu s$) | 11 | 11 | 13.8 |
| # of clusters reported($N_{clt}$) | 1 | 1 | 1 |
| Execution time ($T_{sec}; min$) | 4.1 | 6.9 | 14.9 |

overhead by merging close trajectories during the analysis (as shown in Fig. 3.9 (b)). Furthermore, the CSD was able to deliver the specified confidence level with a certain number of initial conditions and the failure probability bound even for this large a circuit, as the CSD makes a conclusion from the samples and because their resulting relationship in (2.5) does not rely on the circuit size.

In summary, this section has explained a cluster-split detection algorithm to address a global convergence failure problem that is becoming increasingly prevalent in mixed-signal systems. The approach is based on a data cluster analysis with random pilot simulations, and the proposed method is readily applicable to practical circuits and is not constrained by the circuit topology or type of steady state. By applying KNN-clustering to samples of trajectories, it detects a global convergence failure as soon as a split is found among the trajectory samples in the state space. Although the required number of initial samples (therefore, the required validation efforts) for an overly high confidence and an extremely low tolerance could be also prohibitively high, the proposed cluster-split detection was able to detect global convergence failures and to conclude global convergence with at least 99.9% confidence level and 1.34% tolerance for various circuits whose numbers of circuit nodes range from 6 to 3,295, including ring oscillators, a DCO, PLLs and a step-down DC-DC converter. To further enhance the algorithm to confirm that a given circuit is free of ppm-order GCFs with a high confidence, an advanced sampling technique such as importance sampling is necessary.[5]

[5] Section 3.3 will give such examples.

## 3.3 Efficient Covering and Sampling of Parameter Space

In this section, methods to better take samples from a circuit's parameter space (*i.e.*, initial conditions in case of global convergence analysis) are investigated. First, a way to cover the space by volumes by finding transient regions and pruning them out is investigated upon the observation that the effective dimensionality of a circuit's state space quickly decreases with time progress [15]. Second, a way to quickly find problematic parameters that cause failures is discussed.

The point is that we don't need to test all the exponentially growing number of samples for increasing dimensionality of the parameter space in global convergence analysis due to the following two reasons. The first is that the effective dimensionality of a typical circuit's state space quickly reduces to a far less number than the number of circuit nodes as time evolves. The second is that most of circuit responses have a smoothness property, indicating that previous samples' results nearby a new sample location can provide relevant information.

Hence, this section first shows that much of the region in a circuit's state space are transient and investigates a method that attempts to cover the circuit's state space by finding transient regions, finally enabling to take samples from only non-transient regions. Second, the smoothness property of typical response of a circuit is further used to efficiently find a parameter that leads the circuit to fail.

### 3.3.1 Attempt to cover the parameter space – finding transient regions in circuit's state space

To verify global convergence of a circuit, one may want to efficiently cover the given circuit's state space with a certain number of samples. Then, we need to uniformly test every possible initial state in a circuit's state space. Thus, one could expect that the required number of samples for a given resolution would inevitably grow exponentially (e.g., to have at least 8 samples per each axis in the 10-dimensional space, $8^{10}$ samples are necessary).

However, each volume in the parameter space could be verified with a practical number of samples. This is because first, the effective dimensionality of the parameter space is usually lower than its raw dimensionality (as will be shown in the

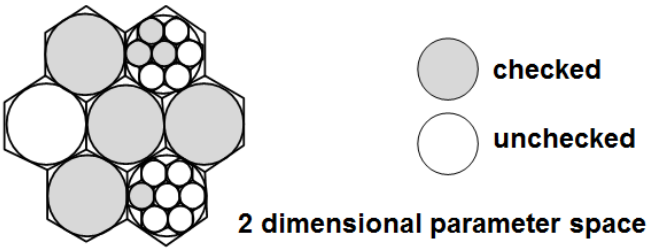


Figure 3.17. Attempt to cover the parameter space by volumes.

following subsection); and second, a sample can provide a lot information about the close region around it. Consequently, we first attempt to check each region in the parameter space one by one as Fig. 3.17, using the transience property of a circuit's state space. In the following sub-section, global convergence property is checked by verifying each state space region's transience and pruning them out from the parameter space.

### Effective dimensionality of circuit's state distribution

The effective dimensionality of the circuit's state space can be much lower than the apparent dimensionality of the state space [32]. The high dimensional uniform state distribution at time zero will soon transform to a lower dimension space as time $t$ goes according to the circuit's dynamics (Fig. 3.4) [30]. The spanned space that is formed after time $t$ is usually more confined to a specific region and it will eventually converge to a limit set region in a circuit. In other words, the effective dimensionality will eventually converge to the minimum number of variables needed to describe the steady-state dynamics of the given circuit [30]. For instance, the effective dimensionality of an oscillator, which has a periodic steady-state, will finally converge to one.

Thus, as the simulation time advances, the effective dimensionality of the state distribution is expected to decrease and hence the effective resolution of samples will increase. The amount of reduction in dimensionality after a small time $t$ would be significant because a circuit's state tends to converge to a steady-state at an exponential rate [30]. Put differently, the state points in a significant portion of space will converge to the existing steady-states almost exponentially and accordingly the

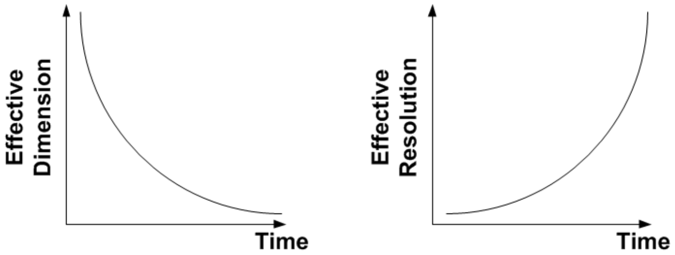


Figure 3.18. Decreasing effective dimensionality of state distribution versus time and increasing effective resolution of finite number of samples versus time.

effective dimensionality will decrease very quickly (Fig. 3.18). If we simply define the resolution ($Q$) as $2^{QD} = N$, where $N$ is the number of samples and $D$ is the dimensionality, the decreasing effective dimensionality will result in the increase in the effective resolution of the fixed number of samples (Fig. 3.18).

$$D_{info} = \lim_{e \to 0} \frac{E[\log p_e]}{\log(1/e)} \approx \frac{H}{\log(bins)} \tag{3.8}$$

where it is assumed that the state space is divided by hyper-cubes whose side length is $e$: $p_e$ represents the probability of a box to be visited at a certain time; $H$ is the entropy of the estimated histograms; and $bins$ is the number of bins in the histograms. However, estimating information dimension could become impractical for high dimensional data set as the number of boxes to consider exponentially increases.

Accordingly, this work proposes to instead efficiently compute the upper bound of $D_{info}$ by projecting the high dimensional data (*i.e.*, final circuit states at $t$ in our case) to low dimensional spaces and inspecting them. For example, $H(X, Y, Z)$ is always lower than $H(X) + H(Y) + H(Z)$ so instead trying to estimate high dimensional entropy with a large number of samples, we correctly estimate the low dimensional entropy with a small number of samples and acquire the upper bound of $H(X, Y, Z)$.

Furthermore, this upper bound can get tightened by grouping highly-correlated variables. The highly-correlated variables are guessed with connectivity information in a circuit netlist because the correlation between circuit nodes would decrease as the distance between two nodes in the topology increases. In other words, two directly-connected circuit nodes are likely to have significant mutual information between them. Thus, we group circuit variables when they are fully connected (*i.e.*, clique) in the netlist.

Therefore, we can find a tight upper bound on the effective dimensionality of a circuit's state space by grouping the fully-connected circuit variables in a circuit netlist, estimating their joint entropy and combining them to finally calculate the upper-bound of the full joint entropy. Specifically, we first construct a circuit graph by mapping circuit nodes to graph nodes and circuit connections to edges. Then, deterministic circuit nodes are found by estimating the uncertainty of each node by entropy. The circuit's final state distribution at $t$ is estimated by histograms and we remove the 'non-X' circuit nodes whose entropy values are zero from the circuit graph while maintaining connectivity (this is referred to as X graph). After constructing the X graph like Fig. 3.19 (a), a clique graph is built (Fig. 3.19 (b)). Then, each joint entropy of the grouped variables, variables in the same clique, is empirically estimated from the data. However, when two cliques share common nodes, we use the conditional entropy of one clique instead of the joint entropy to prevent overestimating the total joint entropy. For instance, for $ABCD$ and $DE$ cliques in Fig. 3.19 (b), we start from $ABCD$ and count their contribution to the total joint entropy as $H(A, B, C, D)$. Next, for $DE$ clique, we count its contribution to the total joint by $H(E|D) = H(D, E) - H(D)$ as variable $D$ is already considered when we visited the $ABCD$ clique. In this way, we traverse all the nodes in the clique graph and the upper bound of the total joint entropy $H(A, B, C, D, E, F, G, H)$ is finally computed as $H(A, B, C, D) + H(E|D) + H(F|C) + H(G, H|F)$. This value can be shown to always upper-bound the full joint entropy using the chain rule of entropy [36]. For example, by chain rule, $H(A, B, C, D, E, F, G, H) = H(A, B, C, D) + H(E|A, B, C, D) + H(F|A, B, C, D, E) + H(G, H|A, B, C, D, E, F)$ and this is smaller than $H(A, B, C, D) + H(E|D) + H(F|C) + H(G, H|F)$ since conditioning always decreases entropy. Once the joint entropy value is estimated, the upper limit of the effective dimensionality is easily found by dividing the computed entropy upper bound value by logarithm of the number of histogram bins used as in (3.8).

Since the algorithm requires estimating only joint distribution of members of clique, complexity is bounded by the maximum clique size in the X graph and

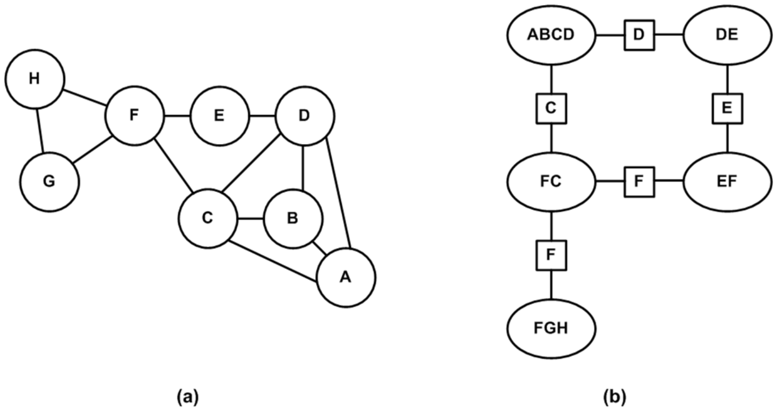


Figure 3.19. (a) Example X graph and (b) clique graph.

therefore the upper bound of the full joint entropy can be estimated with a practical number of samples. This is because the maximum clique size would not be exceedingly high for many practical circuits: *e.g.*, the maximum clique sizes were 4-6 in the bandgap reference circuit in [37], the DCO in [34] and the PLL in [34]. For instance, when we generated 8192 samples according to the $P(A, B, C, D, E, F, G, H)$, the full joint entropy value is hardly to be estimated with the given number of samples because the dimensionality is high, i.e. eight, giving only $13/8 = 1.625$ bit resolution. However, if 8192 samples are used to estimate $H(A, B, C, D)$, $H(E|D)$, $H(F|C)$ and $H(G, H|F)$, then its maximum dimensionality is bound to four and the resolution that we can get is 3.25, 6.5, 6.5 and 4.3 bit, respectively.

The proposed method was applied to estimate the effective dimensionality of various circuits state distribution – a bandgap reference circuit (BGR) in [37], a supply-regulated digitally-controlled oscillator (DCO) and a phase-locked loop (PLL) in [34]. The results shown in Fig. 3.20 compare the effective dimensionality of the state distributions measured after various time durations and estimated with multiple methods: the proposed upper bound entropy estimation without grouping and with grouping according to clique and the principal component analysis (PCA) [11]. In detail, 2048 random initial conditions were generated and their distributions after specified time duration were binned to 8-bin histograms to estimate the entropy. For PCA, 0.99 is used as the threshold to select the principal axes.

As expected, the effective dimensionality rapidly decreased as the simulation

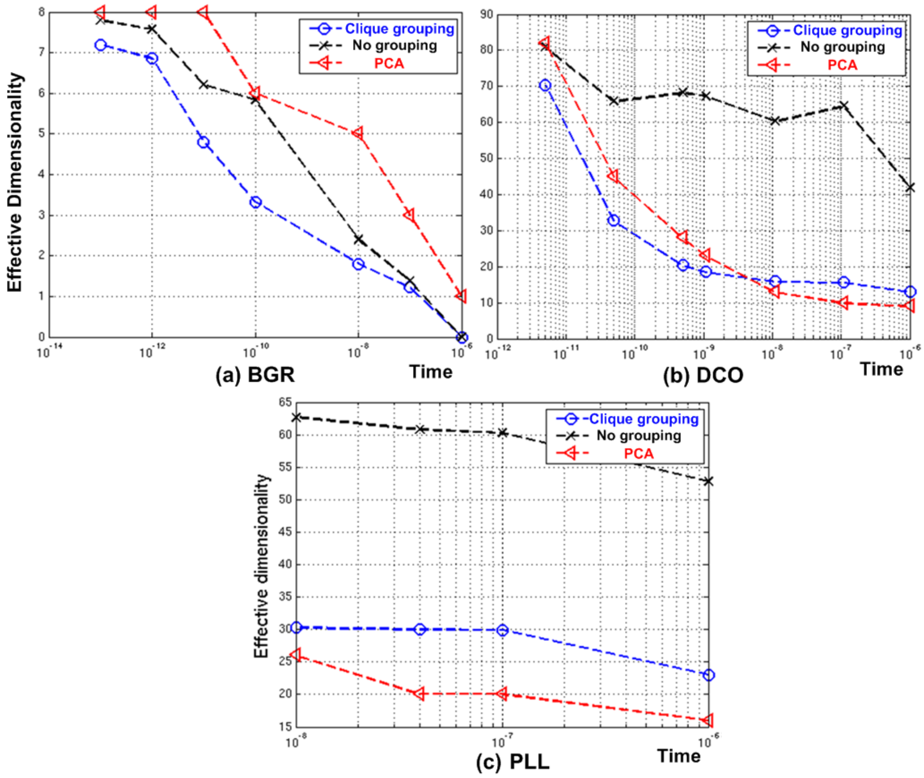


Figure 3.20. Effective dimensionality of the state distribution at various time instants, measured by the proposed entropy based method and PCA, of (a) BGR, (b) DCO and (c) PLL.

time increased. For example, the effective dimensionality of the circuits after 100ps, 500ps and 10ns were estimated to be as low as 3.33, 20.43, and 30.3, respectively, even though dimensionality of raw data was as high as 8, 112 and 355 for the BGR, the DCO and the PLL, respectively. Moreover, the proposed entropy based dimensionality estimation method was shown to closely estimate effective dimensionality when compared to the popular PCA method. Even though the proposed method gave the upper bound value of the effective dimensionality, its value was close to the value estimated by PCA as shown in Fig. 3.20 – it even gave the better estimation for the BGR example and for the DCO example, especially at the early phase.

### Finding transient regions

Recognizing that much of the region in a circuit's state space are transient, in this sub-section, we try to cover the parameter space by finding and pruning transient regions. In other words, the parameter space of global convergence analysis is the initial state space and we can use the transience property, which is explained in the previous subsection, for effectively covering the parameter space.

The idea is first to quantize the space uniformly and next check each region whether it is transient or not. First, quantization can be efficiently performed by sphere packing (or K-means algorithm [11]). In another word, a circuit's state space is divided uniformly into a finite number of regions which will be inspected separately. The second step is to check dynamics of each sphere by taking samples and simulating them. The number of required samples to simulate is usually finite because of the inherent smoothness property in AMS circuits. For instance, a circuit's next state function $f(x,t)$, where $x$ is current state and $t$ is time, typically forms a continuous and smooth function. Therefore, with only a finite number of samples, we can accurately estimate a circuit's smooth response via regression and verify the region. That is, after a certain number of sampling and simulations, the circuit's next state can be well predicted by regression and thus we can use its prediction instead of expensive simulation, saving simulation cost. In this study, we use Gaussian process regression.

Among many candidate regression techniques, Gaussian process (GP) regression is chosen for the following advantages [11, 38]. First, it is non-parametric regression technique. Second, it can well deal with bias-variance problem by taking Bayesian view and computing marginal likelihood of the model that helps model selection without requirements of additional test samples. Finally, it is known that the GP is capable to estimate many nonlinear function well [38]. Actually, it can be shown that the GP is equivalent to the radial basis regression with an infinite number of basis functions when its covariance function is chosen to be the squared exponential.[6]

To be precise, we propose to find transient regions as following (Fig. 3.21). First, a circuit's state space is uniformly divided. Second, we take a random sample from each region and simulate the sample until it leaves the region. This step is repeated until the region is sufficiently checked satisfying the stopping criteria. The stopping criteria of the circuit simulation is when the Gaussian process regression with the collected samples' results gives an accurate prediction, bounding the prediction error under the given threshold (*e.g.*, 5%). Then, we build the transition matrix that tells the probability of transition from one region to another region from the collected sample results and the GP regression. The total number of samples ($M$) to draw from each region is determined by the given confidence level ($1 - \delta$) and interval ($\epsilon$) for transition probability estimation. The relationship among them can be mathematically expressed as (3.9) using the Hoeffding bound [13].

$$M \geq \frac{\ln{(2/\delta)}}{2\epsilon^2} \tag{3.9}$$

Upon the estimated transition matrix, transient regions can be filtered out by the depth-first search on its transition diagram [39]. That is, a graph node is created for each quantized state region and when a sample in one region can move to another region, two regions are connected from the source to the destination (Fig. 3.21 (b)). Then, strongly connected components in the graph are found by the depth-first search and the graph is condensed. Afterward, any node in the condensed graph that has no descendant is recurrent node and all the states (*i.e.*, the quantized state regions) in the condensed node are recurrent. On the other hand, all the other state regions not in the recurrent node in the condensed graph are transient.[7]

[6] Gaussian process regression will be further explained in Section 3.3.2

[7] Moreover, if there is more than one supernode with no descendants, we can detect that there is more than one steady-state and thus conclude that the system is with the possibility of global

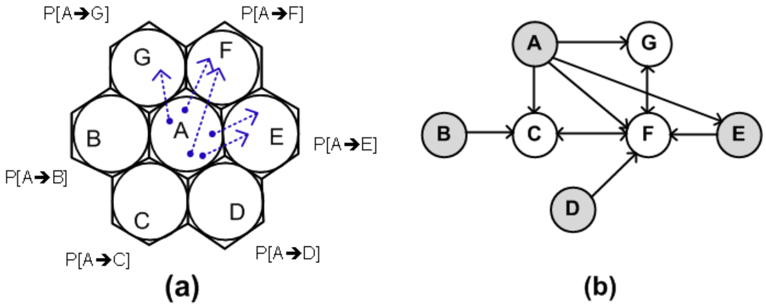


Figure 3.21. (a) Dividing the parameter region and checking each region's transitions, (b) building the transition diagram and finding transient regions.

Furthermore, this test gives a way to check the uniqueness of stationary distribution of the transition matrix. If the condensed graph has only one supernode with no leaving edges, then there is a unique stationary distribution [39]. Thus, after the graph analysis confirms that there is only one supernode without any descendant in the condensed graph, we can compute the unique stationary distribution using the estimated transition matrix [40]. Then, all the states in the stationary distribution with near zero probability (*e.g.*, $\leq 10^{-6}$) can additionally be cut out as transient.

In addition, the proposed method can be recursively applied to further find transient regions. In other word, after pruning the found transient regions, the remaining non-transient regions can be more finely quantized and transience can be checked again. This recursion would provide a better efficiency because it can focus on the crowded region (*i.e.*, where it is highly likely to be visited by a trajectory that starts from a random initial condition) after the iteration, leading to better identification of transient regions.

This procedure was first tested on the 3-stage ring oscillator in Fig. 3.22 (a). In this experiment, the circuit was implemented using 65 nm CMOS process with 1.1V supply voltage and the proposed method found that more than a half of the circuit's state space was transient. Specifically, the circuit's 3-dimensional state space was first uniformly divided by 87, 155, 218, 390 and 502 regions respectively. After the proposed analysis, we could find 11, 37, 59, 192 and 287 transient regions, meaning that it was able to reduce the parameter space for global convergence analysis

convergence failure.

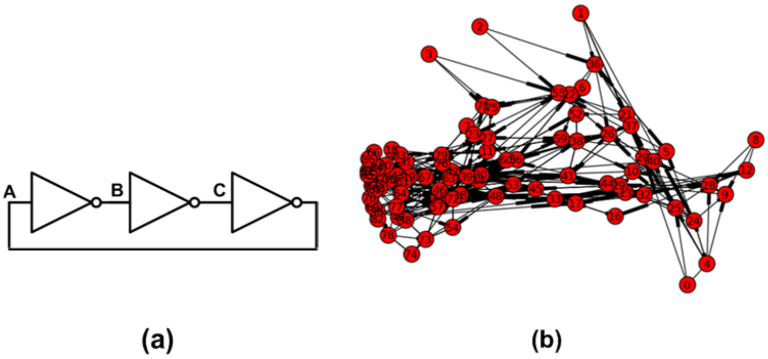


Figure 3.22. (a) Three-stage ring oscillator and (b) resulting transition diagram after the experiment when the space was divided into 87 regions.

by 13%, 24%, 27%, 49% and 57%. Fig. 3.23 shows the portion of detected transient regions with different number of quantization. The red indicates the portion of detected transient regions when only graph analysis is used and the blue shows the portion when we additionally find transient regions after computing stationary distribution. As expected, we could detect more transient regions when the space quantization was finer and the computation of stationary distribution aided us to further find transient regions. The used confidence level and interval in this experiment were 99% and 0.05, respectively, and the required number of samples for estimating transition probabilities with that accuracy was 1,060 according to (3.9). The sampling and simulation for estimating each region's transition probability were repeated until the taken samples could predict the circuit's next state with a sufficient accuracy as Fig. 3.24.

Using the GP regression significantly accelerated the step for estimating transition probability. In the given setting, for example, we should draw 1,060 samples for each divided region but simulating all these samples for 100 regions would take 59 hours since SPICE took 2 seconds to simulate each sample in the 3-stage ring oscillator example. However, using the GP regression, we can simulate only a few hundreds samples and use the GP predictions afterward. This, for example, can reduce the run-time to 6.7 hours assuming we only simulate 100 samples for each region and use the GP prediction values for the rest 960 samples.

The proposed method could be applied recursively as explained and provide

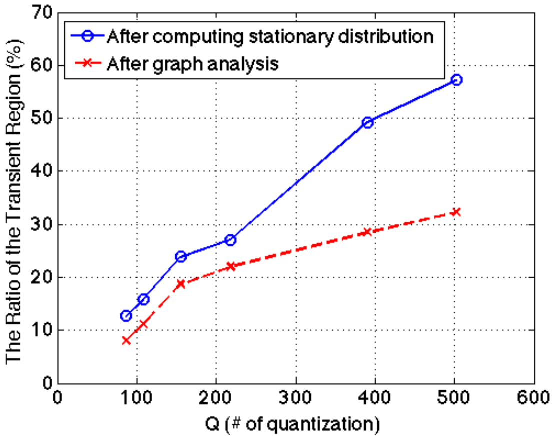


Figure 3.23. Detected transient region's volume ratio versus the number of divided regions in the 3-stage ring oscillator example.

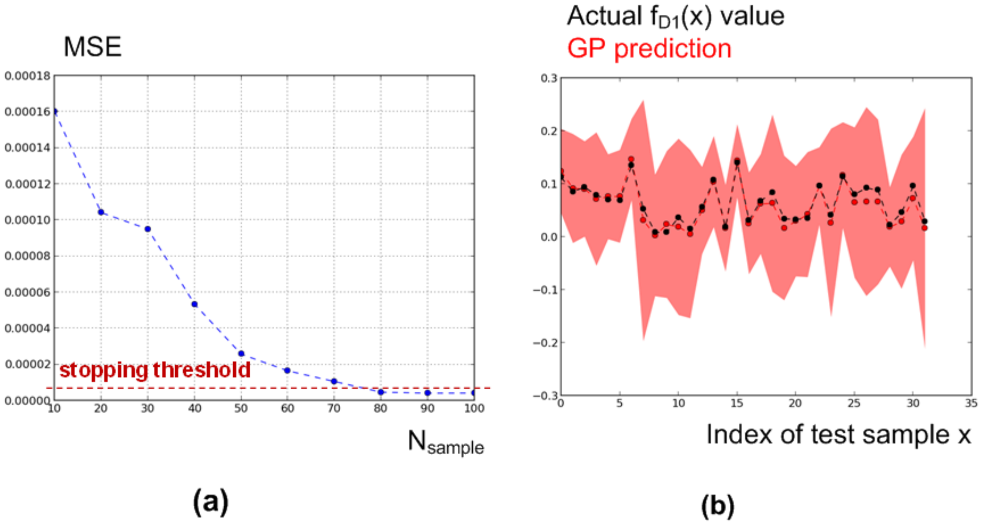


Figure 3.24. (a) Mean squared error of the GP's prediction for the samples in the 73th sphere; (b) the actual value of one element of the next state vector function and the predicted value by the GP regression. The red points are sample predictions, the red regions are 95% confidence interval of the GP predictions, and the black points are the actual values.

better efficiency. For instance, when we first had partitioned the state space of the 3-stage ring oscillator into 390, the method had found 49% transient regions. Next, the remaining region was quantized finer, 192 regions (among 390) had been found to be non-transient after the first iteration so its space was more precisely partitioned into 292 quanta for the next iteration. Then, the method was able to detect 15% more transient regions. In total, it found 64 % transient regions using 485 quanta, giving better efficiency when we used just one iteration of the algorithm using 502 quanta.

As another example, we used the coupled ring oscillator implemented 0.13um using 1.2V supply voltage (Fig. 3.3). The tested oscillator's size ratio of the primary inverter to the coupling inverter ($W1 : W2$) was set to 24:12 to have a single oscillation mode and the proposed procedure was applied. It was turned out that 12, 35 and 122 out of 96, 242 and 485 divided regions were transient. In another word, we could reduce the initial condition space to explore by 25% for this oscillator when the parameter space was divided by 485.

The method was tested on the dual-delay-path differential ring oscillator in Fig. 3.11 as well. The size ratio of the devices in the oscillator was set to have one oscillation mode and the space was partitioned into 200. The result was that the method reported 20% region as transient.

Even though the proposed method reduces the time complexity by using the GP regression instead of expensive circuit simulation, it is still with a certain computation overhead and it would be best applicable to the circuits whose size is under ten such as an oscillator. One reason is that the time complexity of the method's regression increases in proportional to the circuit size because it has to estimate the next state function of the circuit, which grows with the circuit size. Moreover, the efficiency of the uniform quantization could degrade very quickly as the circuit size increases due to the curse of dimensionality [11, 41].

### 3.3.2 Rare-event failure simulation using Gaussian process

Although the Monte Carlo method has many advantages, it has a limitation that a prohibitively large number of samples are required to detect a rare-event failure [42]. For example, to take at least one sample of a rare-event failure which occurs with the probability $\rho$, we should try at least $\frac{1}{\rho}$ trials on average, indicating that we should pay for more than million simulations to detect a ppm-order failure.

To avoid this, an analytical model and method could be used to mathematically guarantee the non-existence of any failure in the system [19, 43]. However, its practicality tends to be limited by the required assumptions and simplifications in the model and method [44]. The scope of circuits and systems that can satisfy the required conditions of the analytical model and method (such as linearity) tends to be very narrow.

Yet, there is another technique called *importance sampling* that can reduce the number of required samples to generate a rare-event failure and enhance the Monte Carlo method [42]. The basic idea of the importance sampling is to bias the sampling distribution so that we can generate more "important" samples which are in the failure parameter region [45]. Prior works [44, 46–50] have shown that importance sampling technique can be effective for failure rate prediction of the static random access memory (SRAM) and several other types of circuits.

However, the choice of the bias (*i.e.*, the proposal distribution) is very critical in the importance sampling. If the proposal distribution is sub-optimal, the importance sampling can rather degrade the performance of the Monte Carlo [51]. However, finding the effective proposal distribution for an arbitrary non-linear circuit is very difficult because (1) the original parameter distribution can be non-exponential family distribution (*e.g.*, not a Gaussian), which is difficult to analytically handle, and (2) a failure region is never known in prior and can be very irregular (*e.g.*, it can consist of multiple narrow regions).

In this section, a non-parametric importance sampling method that can be applied to a broad class of circuits with small efforts is discussed. The idea is to select a smooth guidance metric in the circuit that can be estimated by Gaussian process regression. By choosing the proper guidance metric that can help us to locate where

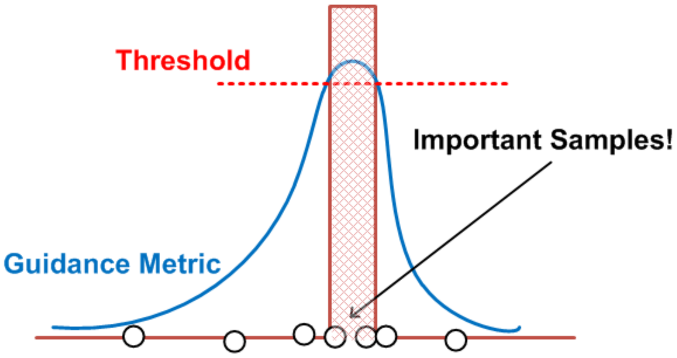


Figure 3.25. Importance sampling by using a smooth guidance metric and a regression.

the failure is, we can tell which samples are more "important" so that rare failure event samples can be quickly generated.

**Finding a failure event using smooth guidance metric and regression**

We propose to choose a *smooth* guidance metric that is closely related to the failure region so that we can use it to effectively find an important region in the parameter space (Fig. 3.25). The very first reason of using a *smooth guidance metric* is to use a regression technique to estimate a given guidance metric (*i.e.*, response surface modeling) so that we can get a direction for the failure region [52]. This smooth metric is chosen to be proportional (or inversely proportional) to the distance to the failure region in the parameter space so that it can guide us toward the failure region.

In case of global convergence failure analysis, this guidance metric could be either a settling time or a signal-to-noise ratio (SNR). The setting time is defined to be the time for a system to converge to its desired steady-state. For an oscillator example, it can be defined to be the first time when the average time difference of the output's consecutive rising edge crossings becomes lower than a certain threshold [52]. On the other hand, the SNR is defined to be the ratio of the power of the desired oscillating frequency to the rest in the spectrum at a certain time [19]. Then, the failure region in the parameter space can be defined to be the region with an infinite settling time or the region with a very low SNR. Because the settling time should deal with *infinite* value when there is a failure and the measurement of the settling time cannot guarantee a quick simulation ending, the later SNR is chosen as the smooth guidance metric in this work.

When a function or a metric is smooth enough, the function's value at a new location can be predicted well using a regression. In other words, *smoothness* property allows us to use non-parametric regression methods such as Gaussian process (GP) regression to estimate it even though the guidance metric function is also unknown.

Thus, for a given set of samples, we can predict the smooth guidance metric values using regression and the predicted values can be used as samples' indicators of importance before running simulation [53]. The Monte Carlo circuit simulation is composed of two steps: (1) generating a circuit instance using a given process design kit (PDK); (2) simulating the instance by SPICE. The high cost occurs primarily at the step 2 when we have to numerically simulate a given circuit instance. However, we can reduce the number of simulations required to generate a sample causing the failure by prioritizing the samples according to their likelihoods to fail. Specifically, after taking a large number of samples according to the correct distribution at the step 1, samples can be ranked according to their regression values before simulation. Then, we can decide the order to actually simulate them according to the rank. This would increase the probability to early generate a sample leading to a rare-event failure as long as the prediction is moderately correct and the sample bunch is large enough to contain a few failure samples.

### Gaussian process regression

Gaussian process (GP) regression is a Bayesian non-parametric regression analysis [38]. This approach treats the unknown function $f(x)$ as a random process and sets a prior for the random process as a Gaussian process, expressed in (3.10), with a certain mean function $m(x)$ and covariance function $k(x, x')$. By the covariance function (*i.e.*, covariance kernel) and the mean function, a family of candidate functions to consider is first set and they are further narrowed down by actual observations of outputs $\mathbf{y}$ at different inputs $\mathbf{x}$ (*i.e.*, $y_1 = f(x_1), y_2 = f(x_2), \ldots, y_N = f(x_N)$). Moreover, it is usually taken that the observations are with an additive Gaussian noise, forming the normal distribution $N(f(x), \sigma_{noise}^2 I)$. With this formulation, the posterior distribution of the process after the observation also becomes a Gaussian process. As a result, a predictive distribution of $y^*$ at a new input $x^*$ (*i.e.*, $P(y^*) = P(f(x^*))$) forms a Gaussian distribution which is given by (3.11), where $K$ is a matrix with the values $K(i,j) = k(x_i, x_j)$. In this work, the mean function is chosen as a constant zero function and the squared exponential kernel is used for the covariance function.

$$f(x) = GP(m(x), k(x, x')) \tag{3.10}$$

$$\begin{aligned} &f(x^*)|x^*, \mathbf{x}, \mathbf{y} = \\ &N(k(x^*, \mathbf{x})^T [K(\mathbf{x}, \mathbf{x}) + \sigma_{noise}^2 I]^{-1} \mathbf{y}, \\ &k(x^*, x^*) + \sigma_{noise}^2 - k(x^*, \mathbf{x})^T [K(\mathbf{x}, \mathbf{x}) + \sigma_{noise}^2 I]^{-1} k(x^*, \mathbf{x})) \end{aligned} \tag{3.11}$$

**Improving the effectiveness and soundness of the Monte Carlo by regression model and sample ordering**

In [54,55], the weakness of the classical Monte Carlo and importance sampling is pointed out. First, the importance sampling depends on the choice of the proposal distribution which is irrelevant information and this violates the *Likelihood Principle* in statistics. Second, the classical Monte Carlo ignores the values of the samples even though the newly drawn sample is tried previously, which means that it throws away relevant information and wastes resources.

However, we can overcome the first issue by a bunch sampling and ordering as previously suggested (Fig. 3.26). A large number of samples (*e.g.*, a million) are first generated according to the original parameter distribution and their simulation results are first predicted by the GP. These samples are sorted by their predicted guidance values that indicate the likelihoods for them to be in the failure region. Afterwards, the samples are simulated in the sorted order. It should be noted that the whole samples will be tested instead of taking only partial portion of "important" samples. In other words, the proposed method do not filter out samples but only determine samples' simulation order so that the worst performance not to be degraded by the sub-optimal distribution. This would guarantee that the method

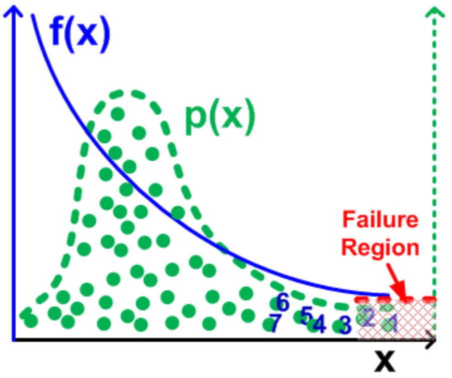


Figure 3.26. Ranking samples according to the GP predictions.

at least gives the performance of the standard Monte Carlo method for every bunch of samples, preventing degradation that may come from a sub-optimal proposal distribution in importance sampling [53].

Next, the second problem is relieved by using the GP regression model. The proposed method saves the simulation cost by preventing unnecessary simulations of samples. If a new sample is near to the previously tested sample, the new sample will give very similar value for the guidance metric and the value can be accurately predicted by the GP without simulation. That is, when the predicted uncertainty (*i.e.*, variance) at a new sample location is very low and the predicted value is certainly away from the failure region, it is unnecessary to run that sample. Therefore, we can block the unnecessary simulation when a sample's guidance metric can be accurately predicted, finally saving simulation cost.

### Experimental results

The proposed method was tested on two coupled ring oscillator examples. The first example was a single-ended coupled ring oscillator in [26]. The example oscillator may have an unwanted steady-state other than the desired differential oscillation mode depending on the ratio of the primary inverter ($W1$) and the coupling inverter ($W2$) – the oscillator may have the start-up failure problem. In the first experiment, this size ratio was changed to make the oscillators with different start-up failure probability, from 0.03% to 4%. Fig. 3.27 shows that the proposed method can achieve tens gains for the example oscillator. That is, the required number of samples to first detect the failure was remarkably reduced by 15-84.

The second test case was the dual-delay-path differential ring oscillator in [25]. This oscillator can also be set to have a start-up failure with a different failure probability by changing the size ratio of the primary device ($W1$), the auxiliary device ($W2$), and the latch device ($W3$). The gain from the proposed method is shown in Fig. 3.28. That is, the required number of samples to first detect the failure was significantly reduced by 5-173.

The complexity issue may arise when the proposed method is extended to a large circuit. This is because (1) the number of inputs of the scalar function to estimate (*e.g.*, the SNR function) increases in proportional to the circuit size, and (2) a large circuit would require a significant number of samples to be simulated for the correct prediction. However, this method could be still applicable to a practical circuit as long as the scalar function (*i.e.*, the SNR) could be estimated by the GP.[8] In contrast to the method proposed in the previous section, it requires to estimate one scalar function rather than a vector function so its degradation for a large circuit would be much smaller. Moreover, its performance is still at least similar to the basic Monte Carlo method for every bunch.

[8] Even though the data is high-dimensional, they are highly likely to lie on locally low-dimensional manifolds that makes the GP method be still effective [11, 54]

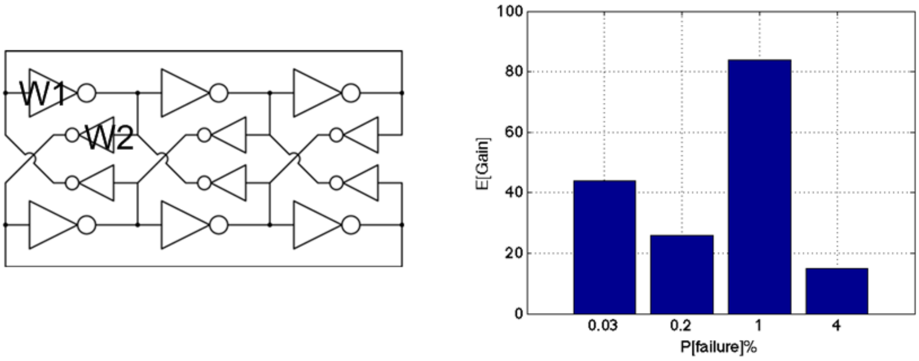


Figure 3.27. Experimental results (right) of the proposed method to the coupled ring oscillator (left).

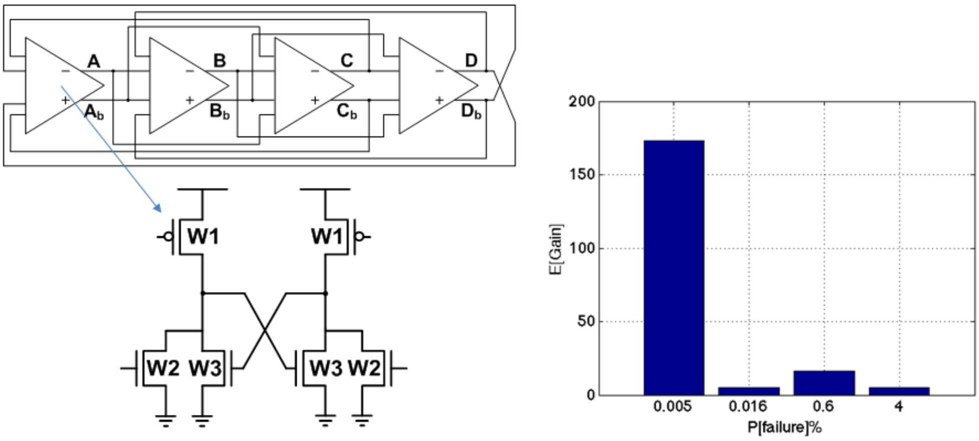


Figure 3.28. Experimental results (right) of the proposed method to the coupled ring oscillator (left).

## 3.4 Preventing Global Convergence Failure via Indeterminate State $X$ Elimination

The previous sections describe the methods that detect the existence of global convergence failure (GCF). In addition to these ways to detect the GCF, this section gives a practical procedure to prevent the GCF.[9] The proposed flow eliminates the uncertainty of a system's starting state and hence prevent GCFs in AMS systems. The idea is to introduce the notion of an indeterminate value that is analogous to the "don't know" value $X$ used in digital logic simulators. Basically, the procedure finds all the nodes in a circuit whose voltages do not settle to any deterministic value within a prescribed amount of time and suggests the locations as to where proper reset circuits must be added. Examples with a digitally controlled oscillator (DCO) and a phase-locked loop (PLL) demonstrate that the proposed design flow is effective in preventing GCFs.

### 3.4.1 Preventing start-up failure by eliminating all indeterminate states

The widespread awareness of GCFs has driven active research efforts in this area, with most work focused on detecting the existence of GCFs [21–23]. In contrast to these existing works which addressed ways to detect the existence of GCFs, this section presents a practical design flow that can identify circuit nodes that are poorly initialized and suggest where reset circuits should be added in order to reduce the discovered ambiguity. Interestingly, the proposed method is analogous to the way digital systems have addressed GCFs since the 1970s, with the introduction of the unknown, indeterminate state value of 'X' [56, 57]. Basically, this work extends the same philosophy to mixed-signal systems by defining 'X' for systems with continuous signals and by providing a dependency analysis to track the originators of the X's.

**Preventing GCFs in digital systems based on the indeterminate state $X$**

It is noteworthy that digital systems were also once plagued by similar GCFs before the four-valued logic simulation was introduced and adopted in the main-

[9] In other words, this section gives a debugging method for global convergence failure problems.

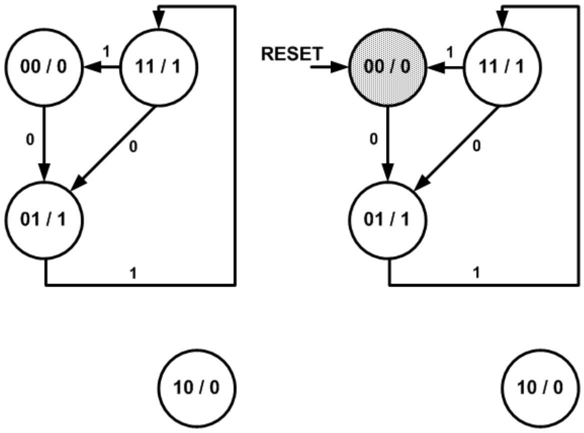
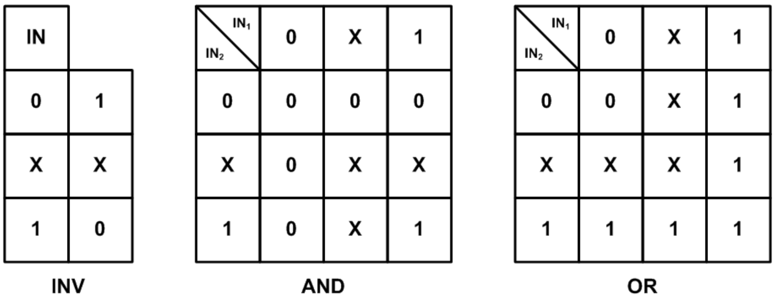


Figure 3.29. Finite state machine without reset (left) and with reset (right).

stream [56–58]. In fact, digital systems are highly nonlinear systems and are hence bound to undergo various start-up failures such as GCFs. A simple finite-state machine (FSM) example in Fig. 3.29 illustrates a possible GCF in a digital system. The FSM in Fig. 3.29 is composed of three valid states (00, 01, and 11) and one unused state (10) when the state is encoded in a two-bit format. Suppose that this unused state is isolated; i.e., when the FSM starts from the 10 state, it has no transitional path to reach the other valid states. Such an FSM will exhibit intermittent functional failures. In other words, the FSM will work correctly when it starts from the states other than 10 and fail otherwise. The failure of the FSM operation depends on the random conditions of the internal circuit nodes upon power-on.

Common remedies for these GCFs in digital systems involve either creating the paths from the unused states to the valid ones or providing a reset mechanism that can properly initialize the state of the FSM. Still, a method that guarantees no GCFs for large complex digital systems without exhaustively adding resets to all of the state elements in the system (i.e., flip-flops) is necessary.

Instead of proving the nonexistence of GCFs directly, modern digital design practices use a much simpler yet effective way of avoiding GCFs [56, 57]. This is the very reason why four-valued logic was introduced, where each logic signal can take a value of ‘X’ (don’t know or don’t care) or ‘Z’ (high impedance or floating) in addition to the normal Boolean values of 0 and 1. Among them, ‘X’ indicates that the signal



| IN | |
|---|---|
| 0 | 1 |
| X | X |
| 1 | 0 |

INV

| $\mathrm{IN}_2$ \ $\mathrm{IN}_1$ | 0 | X | 1 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| X | 0 | X | X |
| 1 | 0 | X | 1 |

AND

| $\mathrm{IN}_2$ \ $\mathrm{IN}_1$ | 0 | X | 1 |
|---|---|---|---|
| 0 | 0 | X | 1 |
| X | X | X | 1 |
| 1 | 1 | 1 | 1 |

OR

Figure 3.30. Extended Boolean algebra including the indeterminate value X.

currently has an ambiguous value, which can be either 0 or 1 (either because it was not initialized or because there is some contention between two drivers). In other words, the signal has an unknown, indeterminate value. Using this notion of ‘X’, logic simulators assume that each signal in the system starts as an X at time zero and remains so until it is initialized or driven by a strong driver (called a ‘dominator’). Also, the logical operators are extended to handle this new value ‘X’ in order to model the propagation of ambiguity in a conservative fashion, as listed in Fig. 3.30. For instance, $1 \cdot X = X, 0 \cdot X = 0, 1 + X = 1$ and $0 + X = X$.

The manner in which this notion of ‘X’ prevents GCFs is very interesting. If there is some degree of ambiguity in the system’s initial states, some of its signals will take the value of ‘X’. For instance, Fig. 3.31 depicts the simulated waveforms of the described FSM example when one of its two-bit state signals ($STATE1$) is not properly initialized. As these X’s do not provide any information about the system’s behavior (only signifying“don’t know”), designers therefore try to eliminate these X’s by adding proper reset mechanisms to the system. Once all of these X’s are eliminated by adding an adequate amount of resets, designers can finally observe the actual behavior of the system. Note that this practice leads to the realization of a system whose initial states will be fully determined with known and chosen values when the reset signal is asserted and therefore free of GCFs. The only thing that remains to be verified is whether the system reaches the correct operational state when it starts from this chosen, initial state. For this FSM example, GCFs can be prevented by identifying any ambiguity in $STATE1$ after a four-valued logic simulation and adding a proper reset mechanism to the sequential component that

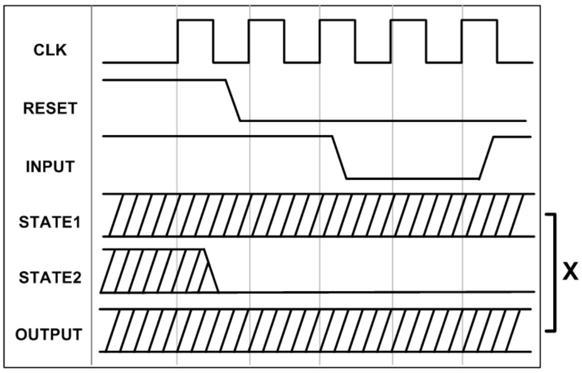


Figure 3.31. Example of the four-valued simulation in a digital system.

encodes $STATE1$ (Fig. 3.29 (right)).

### Extending an indeterminate *X* to analog systems

The goal of this section is to adopt a similar design practice to prevent GCFs in analog/mixed-signal systems by extending this notion of 'X' to analog circuits that have continuous-valued states. Once we can express and evaluate the uncertainty of a given node's condition, GCFs can be avoided by adding enough resets until all the uncertainties are eliminated. In other words, the problem of guaranteeing the global convergence of a system becomes simpler if the system always starts from a fixed, known state and always reaches the correct steady state after starting from such a state.

The only difference is that analog circuits need to be considered as a dynamical system with continuous state variables rather than a FSM with Boolean variables [19]. For example, Fig. 3.32 illustrates an example of a global convergence failure in a second-order dynamic system that has two DC equilibrium states, marked as EQ1 and EQ2. This type of plot is known as a phase portrait [59], which visualizes the progression of the system's state as trajectories in the state space. Specifically, the two axes of the plot correspond to the two state variables of the system (*e.g.*, $x_1$ and $x_2$); hence, each point in the space corresponds to one possible state of the system. The arrow on the plot indicates the direction of the state's progression, *i.e.*, $dx_1/dt$ and $dx_2/dt$. With this set of arrows, it is easy to determine how the system's state

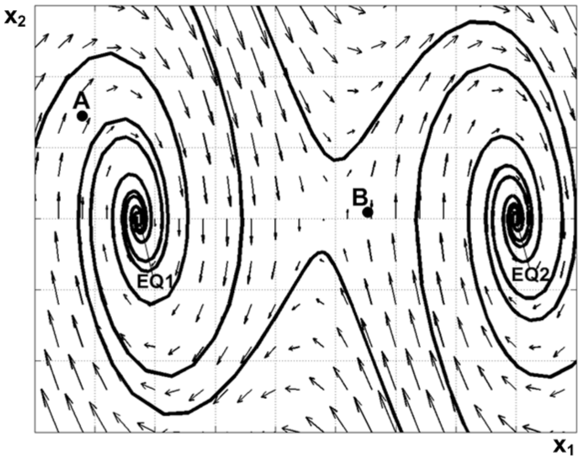


Figure 3.32. State space with two equilibrium points.

will change over time when it is initialized to a certain point. For example, initialized at point $A$, the system will converge to $EQ1$. On the other hand, initialized at point $B$, it will converge to $EQ2$. If $EQ1$ is the desired equilibrium, the existence of an alternate equilibrium is problematic. The design practice we are aiming to adopt in this paper is seen as restricting the initial state of the system (*e.g.*, to point $A$) by guiding the designers to add enough resets so that the system is always guaranteed to converge to the desired operating point (*e.g.*, $EQ1$).

The notion of an indeterminate or ambiguous state ('X') can be extended to analog circuits by assessing the distribution of the possible values that a signal can take at a certain time point. For instance, suppose that we are examining a system's state at time $t_0$ when its state at time 0 is randomly initialized. Fig. 3.33 shows the time trajectories of two node voltages ($A$ and $B$) of a circuit, where multiple trajectories on a given plot are obtained by initializing the system state differently at time 0. If a state variable, in this case a node voltage, at time $t_0$ has a single-point distribution regardless of the initial condition at time 0, then the voltage has a determinate value and contains no uncertainty (as in $A$ in Fig. 3.33). On the other hand, if the voltage takes a distribution with a non-zero width, the voltage value is not fully determined (indeterminate) and contains uncertainty depending on the initial condition (as in $B$).

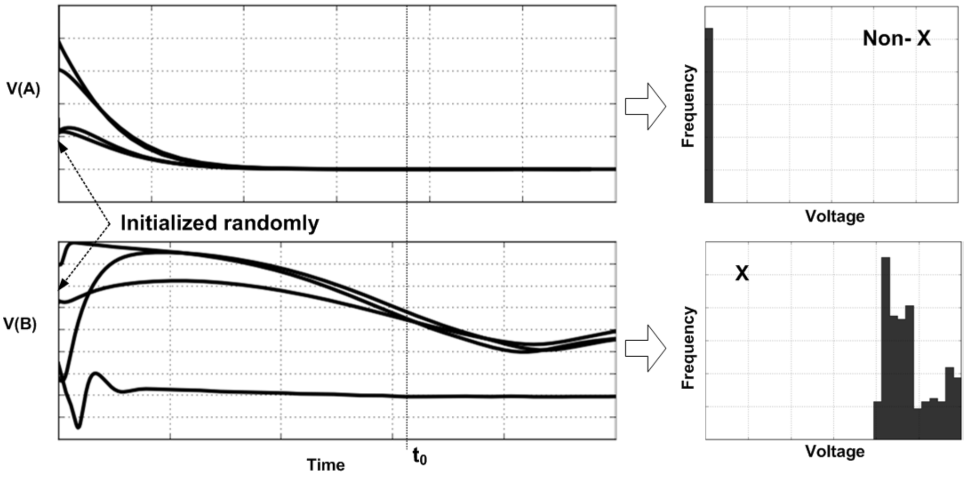


Figure 3.33. Indeterminate state X in mixed-signal systems.

The uncertainty of a signal can be quantified based on the established information theory using entropy [36]. For a discrete-valued signal $A$, its entropy can be expressed in terms of its probability distribution as (3.12), where $a$ is a possible value of the signal and $p(a)$ denotes its probability. If the signal has no uncertainty and is thus fully determined, its entropy would take a value of zero. Otherwise, it would be a positive value.

$$H(A) = \Sigma_{a \in A} p(a) \log \frac{1}{p(a)} \tag{3.12}$$

While the entropy can also be defined for continuous-valued signals (called differential entropy), this work evaluates the entropy of a given node voltage's distribution by approximating the distribution as a histogram with finite-interval M-bins and computing (3.12), as illustrated in Fig. 3.34. This lends itself well to our procedure, in which the distributions are empirically estimated from a finite-sample Monte-Carlo simulation that randomizes the initial conditions of the system. While the entropy in (3.12) and differential entropy may not have the same absolute values [36], differential entropy can be estimated by a histogram of the observations [60]. Furthermore, uncertainty measurement by quantization and (3.12) has an advantage in that it has an explicit lower (*i.e.*, 0) and upper bound (*i.e.*, $\log M$) in contrast to differential entropy, which has no bounds. Therefore, the entropy expression in

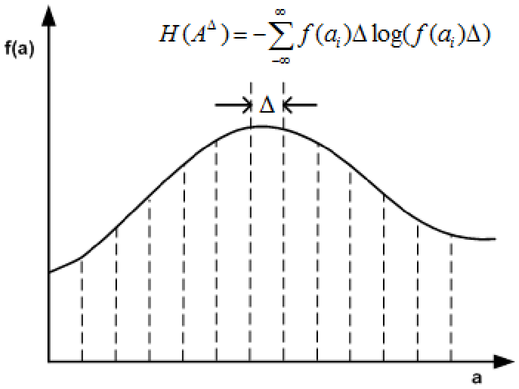

f(a)
$H(A^{\Delta}) = -\sum_{-\infty}^{\infty} f(a_i)\Delta \log(f(a_i)\Delta)$
Δ
a

Figure 3.34. Quantization of a continuous random variable.

(3.12) serves our purpose of assessing the uncertainty of a distribution well, even for a continuous-valued state variable in the system.

The procedure of determining whether a signal is an ‘X’ can be summarized as follows. First, the statistical distribution of a signal across a set of randomly initialized transient-mode simulations is formulated as a histogram with a finite number of bins (in this work, 20 bins). The entropy value is then calculated according to (3.12). If all of the signal samples lie within the same bin, then the entropy would be 0, indicating that the signal is fully determined, in other words, it is not an X. Otherwise, a positive entropy value will denote the relative uncertainty of this signal compared to the other indeterminate signals in the system – the X’s. Note that the interval width of the histogram bin sets the tolerance, or minimum spread, in the distribution for a signal to be considered as an X (Algorithm (4)).

**Algorithm 4** Procedure that identifies X nodes.

```
Estimate the nodal voltage distributions (i.e., $p(A(t_0))$) of every node in the circuit at $t_0$ using $M$-bin histogram.
Compute entropy $H(A)$ according to (3.12)
return All nodes with non-zero entropy ($H(A) > 0$)
```

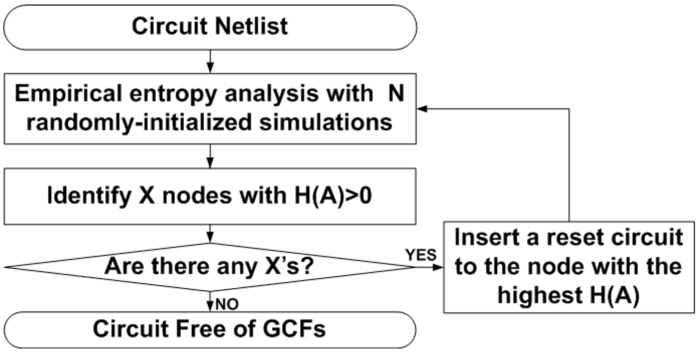

Circuit Netlist
Empirical entropy analysis with N randomly-initialized simulations
Identify X nodes with H(A)>0
Are there any X's?
YES
Insert a reset circuit to the node with the highest H(A)
NO
Circuit Free of GCFs

Figure 3.35. X elimination procedure.

### 3.4.2 Procedure of eliminating indeterminate states with the extended *X* for AMS systems

Now with this extended definition of X for analog signals, a similar X elimination procedure can be devised for preventing GCFs in analog/mixed-signal systems (Fig. 3.35). First, all of the indeterminate nodes in the circuit (X's) are identified based on the empirical entropy analysis, as described in the previous section. Second, reset circuits are added incrementally to the nodes with the largest entropy values, until all the X's are eliminated. For instance, a conservative procedure to reduce the number of added resets would be to add one reset circuit at a time to the node with the maximum entropy. Every time a new reset circuit is added, a new entropy analysis is performed to find the remaining X's in the system, and the process repeats until all the X's are eliminated. In the present work, the process of adding a reset circuit still requires a designer's intervention, as it may require domain-specific knowledge, especially for analog circuits. However, it typically boils down to choosing a proper reset value for the signal, which is usually not a difficult task for designers who are familiar with the circuit's operation. Also, as will be shown later, the number of additional resets required is typically small for a medium-sized circuit with hundreds of circuit nodes (e.g., only six resets for a phase-locked loop with 383 nodes). Hence, the burden on the designer is low.

This procedure prevents GCFs in a mixed-signal system. The proposed method adds the proper number of resets to the X nodes in the system until none remains. This is equivalent to restricting the system's initial states to a certain region, which is a small hypercube in the state space whose edge length is equal to the interval width of the histogram. Then, the system becomes free of GCFs once it is verified that the system reaches the correct steady state starting from these known states. With the assumption that the limited initial state region is small enough to be within the same basin of attraction, a representative initial condition can be taken and verified to converge to the correct steady state via a circuit simulation. For the example in Fig. 3.32, if the initial states of the system are confined to a small region around point $A$ after inserting proper resets, the system will always converge to the desired operating point, $EQ1$.

The proposed approach avoids GCFs not by verifying that all initial conditions converge to the desired steady state but by limiting the initial condition of the system. This is advantageous because establishing global stability in a nonlinear system in general is a very difficult task which has not been solved for nearly a century [59]. The approach allows a designer to avoid putting a great amount of effort in modifying his/her system so that it has only one basin of attraction while still being guided to avoid convergence to an unwanted steady state such that it restricts the initial state of the system to a small region in the basin of attraction of the valid steady state.

The number of Monte-Carlo samples ($N$) and that of the histogram bins ($M$) in the empirical entropy analysis need not be impractically large. For the experimental examples presented in Section 3.4.4, values as low as $N = 32$ and $M = 8$ were found sufficient to identify most of the X's in the system. As a rule of thumb, no more than 100 Monte-Carlo samples are required to find the uncertainty in the system, as the probability of 100 randomly initialized systems giving the same responses despite the underlying uncertainty is very low [11,61]. Moreover,the number is not expected to grow significantly with the circuit size or with the number of nodes. Even if the circuit size grows, each node in the circuit will still interact with a small number of neighboring nodes when the circuit simulation time is short enough,[10] and the overall circuit can be regarded as a collection of small sub-circuits whose uncertainties can be individually analyzed [62,63]. Fig. 3.36 plots the number of samples ($N$) versus

[10] this is the property that enables the efficient, sparse matrix representation of the circuit in SPICE-like simulators [62].

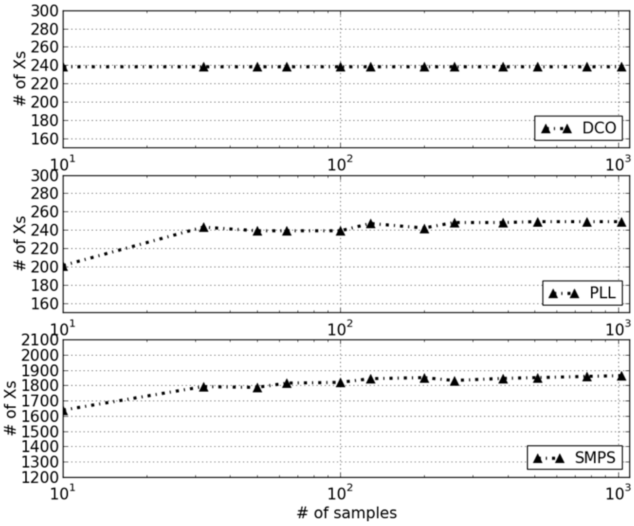


Figure 3.36. Number of samples (N) versus the number of detected X's in a digitally controlled oscillator, a phase-locked loop circuit and a switching-mode power supply.

the number of detected X's ($M$ is set to 20) for a large switching-mode power supply circuit (SMPS) with 3597 circuit nodes in addition to the digitally controlled oscillator and the phase-locked loop circuits described in Section 3.4.4, with 307 and 383 circuit nodes, respectively. This empirically demonstrates that $N = 256$ and $M = 20$ were enough to reveal most of the X's in the circuits. Moreover, in the digitally controlled oscillator (DCO) and the PLL examples with GCFs, the problematic circuit nodes (*i.e.*,*vbp* for the DCO and *vctrl* for the PLL) are listed as one of the most independent and uncertain X nodes even when $N$ is as low as 32.

Although the required number of samples to estimate the X's in the system does not need to be impractically large, even for a circuit with thousands of nodes (Fig. 3.36), the proposed X elimination method may add unnecessarily large number of resets if the method is applied to a large circuit without designers' intervention. Since the uncertainty of one node may propagate to other nodes, the number of X's in the system tends to increase rapidly as the circuit size grows. As a result, the procedure is more likely to fail in distinguishing an important X node from less important resultant X nodes, even with the proposed dependency analysis in Section 3.4.3. Empirically, the proposed method is able to eliminate X's with tens of resets for the medium-sized circuits (*i.e.*, circuits with hundreds of nodes, such as DCO and PLL circuits), but it tends to add too many resets for large circuits (*i.e.*, circuits with thousands of nodes, such as SMPS circuits). Thus, the process of adding a reset circuit still requires a designer's intervention to divide the system properly and apply the proposed procedure with the designer's knowledge to eliminate X's from the system with the lowest possible number of resets.

The duration of each transient-mode simulation, $t_0$, deserves some discussion. Basically, there are no strong constraints on this duration $t_0$ as long as the resets are added to the system until the X's evaluated based on the duration are all eliminated. However, the choice $t_0$ of may have some practical implications. For instance, a long duration $t_0$ of is desired to provide enough time for all the non-X nodes in the system to settle within a single interval bin of the histogram, as otherwise some non-X nodes may be detected as X's and unnecessary resets may thus be added. On the other hand, a short duration is desired to keep the simulation time short. One complication is that the uncertainty of one node may propagate to other nodes as the time progresses, making a long duration less desirable for the purpose of minimizing the number of resets to be added. However, the additional entropy analysis presented in the next section can classify these as dependent X's, for which the designers need not add resets. In practical settings, we conducted our entropy-analysis with the minimum duration first and increased it incrementally only when the number of resets required was considered to be excessive. In most cases, a few multiples of the circuit's dominant time-constant or longest clock period was found to be a reasonable choice.

### 3.4.3 Reducing reset circuits in the $X$ elimination procedure

The initial X elimination procedure exhibited a problem in that it reported too many nodes as X's and hence required too many reset circuits. Among them, some were floating but isolated nodes that never influenced the circuit's operation (*e.g.*, the internal nodes between the two stacked nMOS devices of a two-input NAND gate when both of its inputs are fixed at 0). Others were nodes driven by the uncertainty of still other nodes. We refer to the former nodes as *isolated* $X$'s, while the latter as *dependent* $X$'s. These two types of nodes do not cause GCFs; the isolated X nodes bear harmless uncertainty and the dependent X nodes are not the originators of uncertainty. Therefore, adding reset circuits to these nodes would be wasteful.

This section describes a more detailed entropy analysis that can classify the identified X's either as isolated X's, dependent X's, or as truly independent X's only which require reset circuits. It is noteworthy that the necessity of identifying the true originators of the state's uncertainty is higher in analog/ mixed-signal systems than in digital systems. In digital systems, the states can be stored only in state storage elements such as flip-flops and latches, and adding unnecessary resets to these elements may not degrade the performance as much as it would in most cases. On the other hand, in analog circuits, each and every node (as well as the branch currents of some circuit elements such as inductors) of the circuit can constitute a state variable; adding too many unnecessary resets to these nodes is likely to impact the circuit's performance and area significantly.

### Independent *X*

First, we would like to identify the X nodes that must be reset to the known initial values with the highest priority. The early version of the X-elimination procedure chose the node with the highest entropy for this purpose, but the problem with this approach was that the entropy metric alone cannot distinguish between an X node that truly originates the uncertainty and an X node that simply propagates the uncertainty. Without a means of distinguishing these two types of X's, originators and propagators, unnecessary reset circuits may be added to the circuit, increasing the number of iterations required to eliminate all of the X's in the system.

The originators of uncertainty can be identified by evaluating a self-conditional entropy value of each X node, $H(A(t_0)|A(0))$, which denotes the uncertainty of node A's voltage at time $t_0$, given its known voltage value at time 0. In some sense, this conditional entropy estimates the entropy of the node voltage if a reset circuit were added to fix its initial value. If this conditional entropy yields a low value, e.g., 0, it implies that a reset circuit would be effective in eliminating the uncertainty at that particular node. On the other hand, if the conditional entropy $H(A(t_0)|A(0))$ has a high value without much reduction from the original entropy value $H(A(t_0))$, it implies that a reset circuit would not be so effective.

The self-conditional entropy is estimated from the joint probability distribution of the node voltage values at time $t_0$ and time 0 [36]. Similar to when the entropy was estimated in (3.12), the joint probability distribution can be approximated as a two-dimensional histogram as in (3.13).

$$\begin{aligned} H(A(t_0)|A(0)) &= H(A(t_0), A(0)) - H(A(0)) \\ &= \Sigma_{a_{t_0} a_0 \in A(t_0), A(0)} p(a_{t_0}, a_0) \log \frac{1}{p(a_{t_0}, a_0)} - H(A(0)) \end{aligned} \tag{3.13}$$

Based on this new metric, the improved X elimination procedure adds resets to the nodes with low self-conditional entropy values first.

**Algorithm 5** Procedure that identifies independent X's.

```
1: Estimate joint distribution $p(a_{t_0}, a_0)$ of every remaining X at $t_0$ via M-bin histogram.
2: Compute self-conditional entropy $H(A(t_0)|A(0))$ according to (3.13)
3: Sort X nodes according to the self-conditional entropy (A lower value indicates higher independency)
4: return the sorted X nodes (the X node with the lowest self-conditional entropy is the most important X node)
```

### Isolated *X*

Some indeterminate X nodes are independent yet harmless; they do not cause GCFs and therefore do not require resets. Such an example is a floating node that is completely isolated from the other parts of the circuit. These floating nodes can be found in mixed-signal systems with digital configurability, in which the configuration logic may contain logic gates whose inputs stay fixed for most of the time and where the system may have inactive flip-flops depending on the configuration. One example is a static CMOS NAND gate with both the inputs fixed at 0, as illustrated in Fig. 3.37 (a). In this case, the shared node between the two nMOS devices is never connected to the other nodes of the circuit and is therefore isolated. Another example is an inactive flip-flop of which the clock input is fixed at 0 ($ck$ = 0 and $ckb$ = 1), as shown in Fig. 3.37 (b). This circuit also contains isolated nodes that are left floating. While these floating nodes will be reported as independent X's and adding resets to them will indeed reduce the uncertainty, those resets are in fact unnecessary because the uncertainty of these isolated nodes never propagates to the other parts of the circuit. This section describes two methods of identifying these isolated X's: the first method [33] requires a hierarchical description of the circuit while the second one does not [34].

Assuming that the circuit is represented as a set of sub-circuit elements, an X node can be identified as an isolated X when it meets the following two criteria. First, all of the input/output ports of the sub-circuit containing this node must be determinate nodes (i.e., non-X nodes). Second, the uncertainty of the X node must not propagate to any of the determinate nodes within the sub-circuit. The second condition is evaluated by examining any increase in the entropy values of the nodes within the sub-circuit for a designated period of time. If any determinate node exists whose entropy value has increased, the internal X's are deemed to be propagating. These two criteria can identify the floating nodes in Fig. 3.37 as isolated X's [33].

When a hierarchical description is not available from the designer, the isolated X's can be identified based on a connectivity graph of the circuit. Suppose that a graph is constructed by creating a node in the graph for each circuit node and making a connection between two nodes if a circuit element exists that connects the two. Using this graph, isolated X's can be identified by performing the following procedure. First, all of the non-X nodes are removed from the graph. Second, a set of connected X nodes that does not have a path connected to the observable ports of the circuit is found. Finally, this set of X nodes is considered to be *isolated* if the entropy values of the surrounding non-X nodes do not increase over a designated period of time. The procedure is illustrated in Fig. 3.38 and Algorithm (6).

**Algorithm 6** Procedure that identifies isolated X's.

```
Construct a circuit graph from connectivity information in the circuit netlist
Eliminate all non-X nodes from the circuit graph
Find connected components in the circuit graph by the graph analysis (e.g., the
  depth-first search)
return  X nodes in the connected components with no observable outputs
```

The described graph-based procedure can be thought of as establishing conditional independence between different parts of the circuit. Because a circuit node

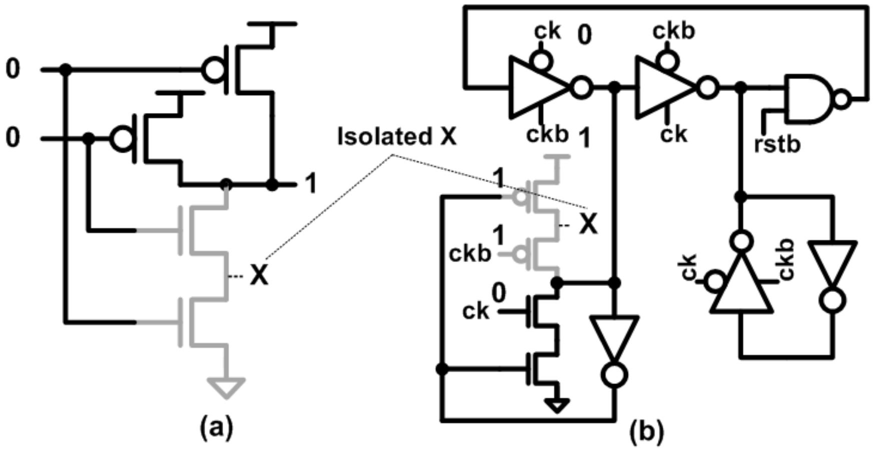


Figure 3.37. Isolated X in (a) a NAND gate with fixed zero inputs and in (b) a flip-flop with inactive clocks.

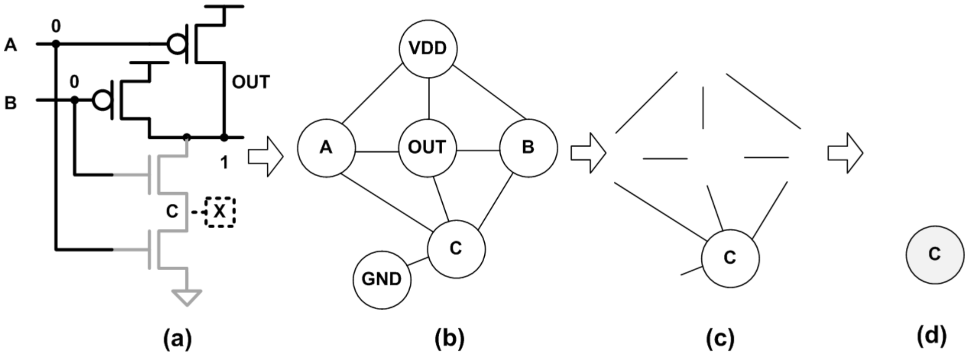


Figure 3.38. Finding isolated X in (a) a NAND gate by (b) constructing a circuit graph, (c) eliminating all non-X nodes and (d) finding a connected component which has no path to the observable outputs.

only interacts with the nodes to which it has a direct connection, when a group of X nodes is fully surrounded by non-X nodes on the graph, the group is isolated from other parts of the circuit. In other words, the groups of X's are conditionally independent from the other parts of the circuit given that the surrounding nodes have fixed determinate values.

The designated time period to determine whether or not an X is propagating to the neighboring nodes must be long enough to prevent the misidentification of non-isolated X's as isolated X's. Also, for the same reason, a tighter tolerance is applied when determining whether the entropy of a determinate node is strictly decreasing.

### Dependent *X*

Conditional entropy can also be used to infer a functional relationship between two neighboring circuit nodes [36]. For example, one node may be an input to a CMOS static inverter while the other node is the output. In this case, the output node is driven and completely determined by the input node. Such dependency can be found by evaluating the conditional entropy $H(A|B)$ or $H(B|A)$, where $A$ and $B$ are the two nodes under consideration. If $H(A|B)$ is zero, this implies that the node $A$'s condition is fully determined by the node $B$'s condition, *i.e.*, $A$ is completely dependent on $B$. In this case, it is meaningless to add a reset circuit to initialize $A$; initializing $B$ would be sufficient. For example, in Fig. 3.39, one can infer the dependencies of $B$, $C$, and $OUT$ on $A$ by finding that the conditional entropies $H(B|C)$, $H(C|B)$ and $H(OUT|C)$ are zero.

This dependency analysis complements the analysis that finds the independent X's in the earlier subsection, especially when the node voltage's distribution changes slowly over time. For example, in the second circuit shown in Fig. 3.39, the nodal voltage $E$ is determined by nodal voltages $D$ and $F$ (*i.e.*, $E = (R_1F + R_2D)/(R_1 + R_2)$). Thus, the uncertainties on $D$, $E$ and $F$ can be eliminated by adding reset circuits to only $D$ and $F$. However, if $D$ and $F$ are floating nodes whose entropy values remain constant, the independent metric using conditional entropy will fail to distinguish $E$ from $D$ and $F$, as the self-conditional entropy $H(E(t_0)|E(0))$ would have a value similar to that of $H(D(t_0)|D(0))$ and $H(F(t_0)|F(0))$; *i.e.*, the distribution of $E$ remains the same when $D$ and $F$ do not

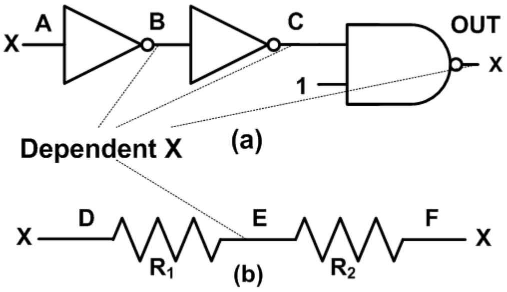


Figure 3.39. Finding dependent relationships.

change. On the other hand, with conditional entropy $H(E|D, F)$, node $E$ can be identified as a dependent X node that does not require a reset circuit.

In general, dependent X's can be found by examining the conditional entropy of each circuit node ($A$) given its neighboring nodes in the connectivity graph (the $N_i$ values), *i.e.*, $H(A|N_i), H(A|N_i, N_{i+1}), \ldots, H(A|N_i, N_{i+1}, \ldots, N_{i+k})$. Here, $k$ is the maximum number of neighboring nodes considered during the dependency analysis. The conditional entropy can be computed via the following (3.14). If the computed conditional entropy $H(A|N_1, \ldots, N_k)$ is zero, this implies that $A$ is fully dependent on its neighbors $N_i$ values and that $A$ is therefore a dependent X. In our work, we limited $k$ to 2 to compromise for the exponentially increasing computational costs with $k$.

$$H(A|N_1, \ldots, N_k) = H(A, N_1, \ldots, N_k) - H(N_1, \ldots, N_k) \tag{3.14}$$

### Improved *X* elimination procedure

The improved X elimination procedure augments the previously described procedure with analyses that find the independent X's, isolated X's, and dependent X's. Basically, the isolated and dependent X's are excluded from the list of candidate nodes that require resets, and the reset priority is evaluated based on the self-conditional entropy $H(A(t_0)|A(0))$ rather than only on $H(A)$. This way, the number of reset circuits needed to eliminate all the X's in the system and hence the number of X-elimination iterations can be reduced significantly.

Fig. 3.40 describes the final X elimination algorithm. First, an empirical entropy

**Algorithm 7** Procedure that identifies dependent X's.

```
Construct a circuit graph from connectivity information in the netlist
Eliminate all non-X nodes from the circuit graph
for each node A in the graph do
    Construct A's neighboring node set, $N = N_0, N_1, N_2, \dots$
    Enumerate 1 to k combinations fro the set $N$: $E = \{(N_0), (N_1), (N_2), \dots, (N_0, N_1), (N_0, N_2), (N_1, N_2), \dots\}$
    for each enumeration in E do
        Estimate the joint distribution $p(A(t_0)), N_0(t_0), \dots)$ of nodes in E using an M-bin histogram
        Compute the conditional entropy $H(A(t_0)|(N_0(t_0), \dots))$ according to (3.14)
        if $H(A(t_0)|(N_0(t_0), \dots)) = 0$ then
            add A to dependent X set $Set_{DepX}$
        end if
    end for
end for
return $Set_{DepX}$
```

analysis with N-sample Monte-Carlo simulations is performed to identify the indeterminate X nodes and classify them as isolated, dependent, or truly independent X's using the aforementioned analyses. Then, a new reset circuit (*e.g.*, a switch that either pulls up to $VDD$ or pulls down to $GND$) is inserted into one of the truly independent X nodes which has the lowest self-conditional entropy and thus the highest independence. This process is repeated until all of the truly independent X's are eliminated from the system.

Since the process of inserting a reset circuit still involves intervention by the designer, the designer may choose a better location at which to add the reset based on his or her domain-specific knowledge, and this may lead to a smaller number of resets required. This is especially necessary when the system is large and contains many digital circuits. Otherwise, it is likely that many unnecessary resets are added even with the proposed dependency analysis. For instance, the use of four-valued logic simulation is recommended to eliminate the uncertainties in the digital parts of the system first rather than relying on our procedure entirely. Then, our procedure can be applied to the rest of the system to remove the remaining uncertainties. Nonetheless, the described X-elimination procedure effectively guides the designer

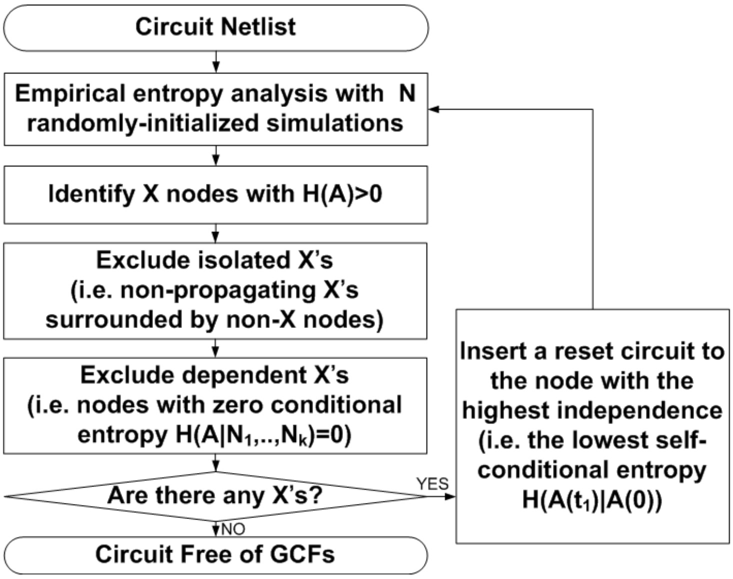

Circuit Netlist
Empirical entropy analysis with N randomly-initialized simulations
Identify X nodes with H(A)>0
Exclude isolated X's (i.e. non-propagating X's surrounded by non-X nodes)
Exclude dependent X's (i.e. nodes with zero conditional entropy H(A|N1,..,Nk)=0)
Are there any X's?
YES
NO
Circuit Free of GCFs
Insert a reset circuit to the node with the highest independence (i.e. the lowest self-conditional entropy H(A(t1)|A(0))

Figure 3.40. Flowchart of the improved X elimination procedure.

as to whether uncertainty still exists and the system requires additional reset circuits.

### 3.4.4 Experimental results

The described X elimination procedure is demonstrated on two real circuit examples: a digitally controlled oscillator with supply regulation and a phase-locked loop.

**Digitally controlled oscillator**

Fig. 3.41 shows the circuit schematic of a DCO. It is implemented in a 45 nm CMOS process and its frequency ranges from 0.6 to 1.9 GHz with a 5-bit resolution. Basically, the DCO consists of three components: a ring oscillator, a supply regulator, and a digitally controlled resistor (DCR). First, the digital inputs control the resistance of the DCR, which is a serially connected chain of resistors that are selectively shorted out with switches. Second, the supply regulator operates on a principle similar to that of a constant-$g_m$ [64], which adjusts the supply voltage of the ring oscillator (*vreg*) to a value that makes the oscillator's output resistance equal to the DCR's resistance. Finally, the ring oscillator generates a clock signal

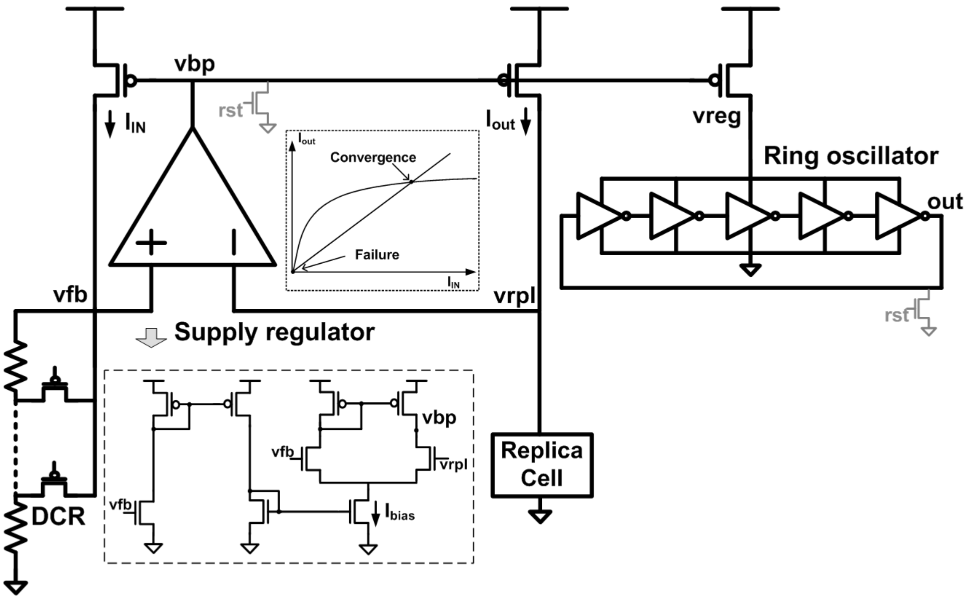


Figure 3.41. DCO experiencing a global convergence failure.

whose frequency is adjustable via the digital inputs, as the oscillation period is proportional to the product of the driving resistance and the load capacitance of the oscillator's stages.

The self-biased feedback loop employed within the supply regulator has a well-known start-up failure. The bias current of the amplifier stage ($I_{bias}$) determines the voltage that drives the gates of the pMOS current sources (*vbp*), while this *vbp* in turn sets the bias current level ($I_{bias}$) via a current-mirror network. The start-up failure stems from the fact that there are two possible DC steady-state solutions in this circuit: one with a positive $I_{bias}$ value (desired) and the other with $I_{bias}$ at 0 and *vbp* at the full supply voltage. If the DCO converges to the latter equilibrium, the oscillator will be stuck at a lower frequency than the intended value, resulting in a global convergence failure.

The entropy analysis is carried out with a set of 256 transient simulations ($N = 256$), each of which is run for $12.1ns$ with a different initial condition. When the final ($t_0 = 12.1ns$) values of each node are binned into a 20-bin ($M$=20) histogram, 238 out of the total 307 nodes are identified as indeterminate X's. Among them, 91 were classified as dependent X's (see Fig. 3.42). Fig. 3.43 lists the X nodes with the

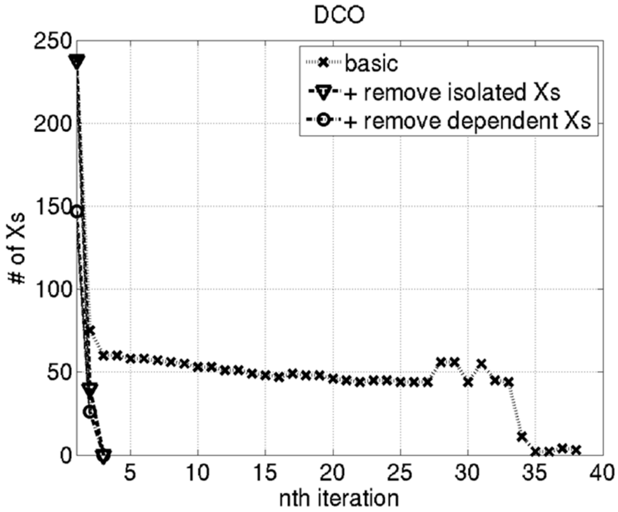
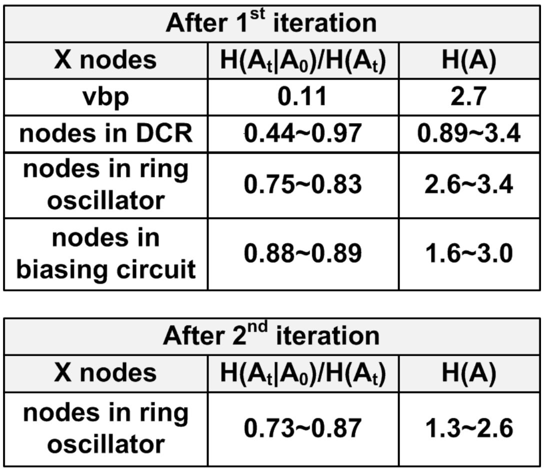


Figure 3.42. Number of X nodes during the X elimination procedures.

highest levels of independence and uncertainty after the first and second iterations. As shown in the table, the proposed procedure correctly identifies the problematic node *vbp* as an uncertain and independent X node requiring a reset. The rest were the nodes which were affected by the DCO biasing circuit and the initial phase of the ring oscillator clock.

Once the iterations to eliminate all of the X's are completed by adding pull-up or pull-down reset switches to the identified X's (i.e., charging/discharging switches that can be implemented as pMOS/nMOS elements that connect its source to VDD/GND, respectively, and its gate to a reset input), all of the important indeterminate nodes in the DCO example are resolved to known values within 12.1 *ns* after the last reset assertion. The revised procedure indeed reduces the number of resets added, as demonstrated by this example in which the improved procedure added only two resets after three iteration cycles to eliminate all of the X's, whereas the basic procedure added 37 resets after 38 iteration cycles (Fig. 3.42). The revised procedure adds the resets to the *vbp* node and one node within the ring oscillator in order to initialize its phase (Fig. 3.43). Once the reset circuits are added with the X elimination procedure, the problematic initial conditions that lead the system to an incorrect equilibrium can be avoided by ensuring that the DCO starts from the



| After 1st iteration | | |
|---|---|---|
| X nodes | $H(A_t\|A_0)/H(A_t)$ | H(A) |
| vbp | 0.11 | 2.7 |
| nodes in DCR | 0.44~0.97 | 0.89~3.4 |
| nodes in ring oscillator | 0.75~0.83 | 2.6~3.4 |
| nodes in biasing circuit | 0.88~0.89 | 1.6~3.0 |

| After 2nd iteration | | |
|---|---|---|
| X nodes | $H(A_t\|A_0)/H(A_t)$ | H(A) |
| nodes in ring oscillator | 0.73~0.87 | 1.3~2.6 |

Figure 3.43. Lists of X nodes in the DCO after the first and second enhanced X elimination iterations.

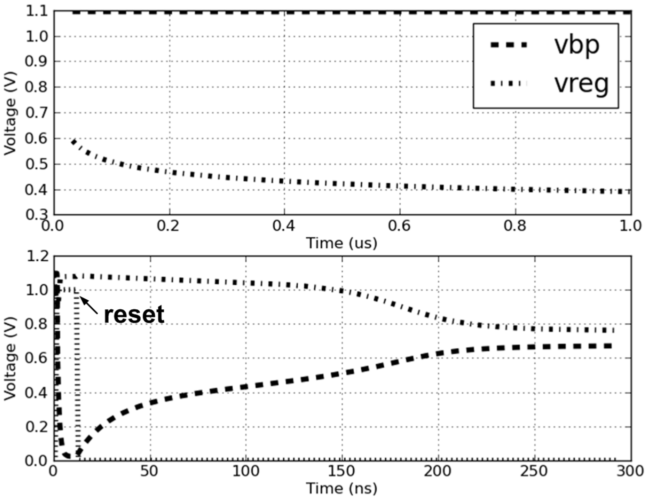


Figure 3.44. Trajectory starting from the problematic initial condition in the original DCO (up) and the trajectory after inserting reset circuits according to the X elimination procedure (bottom).

proven initial state (Fig. 3.44).

With regard to the circuit complexity and the execution time of the improved X elimination procedure, the DCO circuit contains 307 nodes, as described by 431 circuit equations. The total execution time of the final algorithm (Fig. 3.40) on a machine with an Intel Xeon X3440 processor and 8 GB memory was 2.46 hours. The proposed algorithm took only 15% of the total execution time, while 85% was consumed by circuit simulations.

### Phase-locked loop

The proposed X elimination procedure is also applied to a PLL example with the circuits similar to the example described in [35]. The example PLL is implemented in a 130 nm CMOS process, taking an 833 MHz reference clock and generating 833 MHz - 2.44 GHz output clocks with a programmable multiplication ratio. The PLL possesses a GCF (Fig. 3.45). The GCF scenario is as follows. Suppose that the voltage-controlled oscillator (VCO) starts at a high frequency which is beyond the operating range of the divider. In this case, the divider may occasionally swallow the input clock pulses, resulting in an effectively higher division ratio and lower frequency of the clock at its output. Despite the fact that the VCO frequency is too high, the PFD determines that the feedback clock frequency is too low and drives the control voltage towards an even higher voltage. This way, the PLL is stuck at a dead-lock condition and fails to acquire the correct lock [26,33].

A set of 256 simulations ($N = 256$), each of which is $15.05ns$ ($t_0 = 15.05ns$) long, is run with individually different initial conditions, and 20-bin ($M = 20$) histograms are collected for each circuit-node voltage distribution. The distributions are examined, and the results indicate that 184 out of 383 nodes are indeterminate X nodes. Among them, 55 isolated X's and 40 dependent X's were found, leaving only 89 nodes as candidate nodes that may require reset circuits (Fig. 3.46).

The table in Fig. 3.47 lists the nodes with high independence levels (*i.e.*, those with low self-conditional entropy values) after the first and second iterations. Not surprisingly, the most independent X nodes were the control voltage of the PLL, the internal states of the oscillator and divider, and finally the slowly moving bias nodes (Fig. 3.45). Many of the X nodes were related to the loop filter, such as

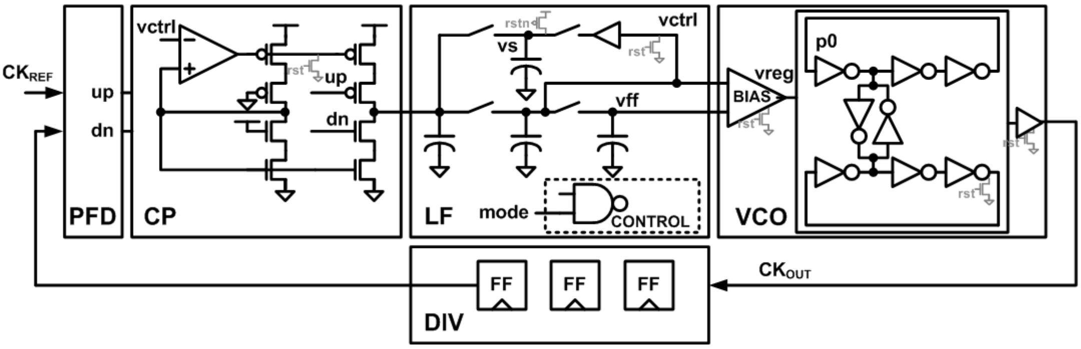


Figure 3.45. PLL experiencing a global convergence failure.

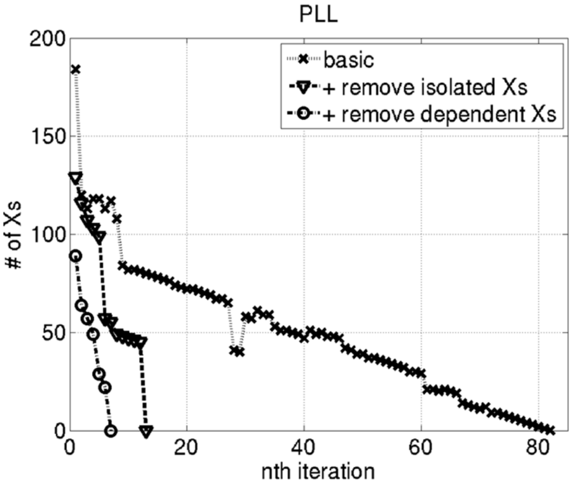
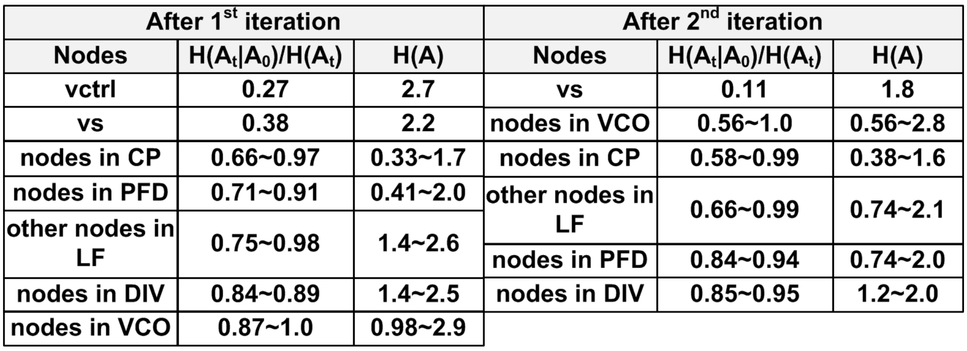


Figure 3.46. Number of X nodes during the X elimination procedures.



| After 1st iteration | | | After 2nd iteration | | |
|---|---|---|---|---|---|
| Nodes | $H(A_t\|A_0)/H(A_t)$ | H(A) | Nodes | $H(A_t\|A_0)/H(A_t)$ | H(A) |
| vctrl | 0.27 | 2.7 | vs | 0.11 | 1.8 |
| vs | 0.38 | 2.2 | nodes in VCO | 0.56~1.0 | 0.56~2.8 |
| nodes in CP | 0.66~0.97 | 0.33~1.7 | nodes in CP | 0.58~0.99 | 0.38~1.6 |
| nodes in PFD | 0.71~0.91 | 0.41~2.0 | other nodes in LF | 0.66~0.99 | 0.74~2.1 |
| other nodes in LF | 0.75~0.98 | 1.4~2.6 | nodes in PFD | 0.84~0.94 | 0.74~2.0 |
| nodes in DIV | 0.84~0.89 | 1.4~2.5 | nodes in DIV | 0.85~0.95 | 1.2~2.0 |
| nodes in VCO | 0.87~1.0 | 0.98~2.9 | | | |

Figure 3.47. Lists of X nodes in the PLL after the first and second X elimination iterations.

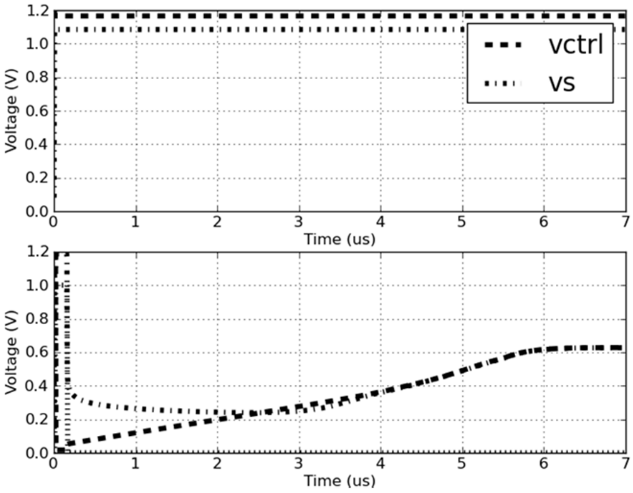


Figure 3.48. Trajectory starting from the problematic initial condition in the original PLL (up) and the trajectory after inserting reset circuits according to the X elimination procedure (bottom).

$vctrl$, $vs$ and $vff$ , as well as the internal nodes of the VCO and the charge-pump's bias generators. Some others were related to the initial phase of the VCO clocks. In addition, a large number of isolated X nodes were reported inside the inactive parts of the programmable divider and the CP/LF control logic. Once the process of adding resets (switches that either pull up to VDD or pull down to GND) is completed, all of the nodes of the circuit settle to their determinate values upon a reset.

Again, it is noteworthy that the basic X elimination procedure (Fig. 3.35) required 81 iterations to remove all the X nodes after adding 80 reset circuits, whereas the enhanced procedure (Fig. 3.40) required only six reset circuits after seven iterations (Fig. 3.46). If the isolated X's and dependent X's were not excluded, the X elimination procedure would have wasted a majority of the reset circuits on the initialization of the isolated and dependent X nodes. Once all of the necessary reset circuits are added, the PLL becomes free of GCFs (Fig. 3.48).

The proposed X elimination procedure has low computational cost. In this PLL example, the entropy and dependence metric of each node were measured through 256 Monte-Carlo simulations, each of which was 15.05-$ns$ long. This duration is far shorter than the PLL's worst-case lock time of $8\mu s$, meaning that a brute-force approach involving the running of 256 simulations 8-$\mu s$ long each would have been 532 times slower.

Regarding the circuit complexity and the execution time of the improved X elimination procedure in the PLL example, the PLL contains 383 nodes, and it is described by 1830 circuit equations. The total execution time of the final algorithm (Fig. 3.40) on a machine with an Intel Xeon X3440 processor and 8 GB memory was 1.85 hours. The proposed algorithm took only 9% of the total execution time, while 91% was spent by circuit simulations.

In conclusion, this section has extended the notion of the indeterminate state $X$ in digital systems to analog/mixed-signal systems based on entropy and described a procedure that effectively prevents global convergence failures by adding resets and eliminating all the X's. This procedure guarantees that the system would always start from a known state upon a reset and is free of GCFs if it can reach the correct equilibrium state from that state. With various entropy measures that can determine the independent X's, isolated X's, and dependent X's, the number of resets required can be greatly reduced. This essentially describes a design practice that can effectively prevent GCFs without having to detect their existence. Its main advantage is that it is a highly pragmatic method that can be easily integrated into existing circuit simulators.

# Chapter 4
# Bug Localization using Probabilistic Graphical Models in Post-Silicon Validation

Upon the observation of a circuit failure, problematic circuit blocks and parameters need to be localized before they can be fixed or by-passed, which has traditionally been highly manual and problem-specific. In this chapter, we present a bug localization methodology that can automatically identify and rank potential root-causes in a probabilistic manner. We model linear and nonlinear sub-circuits using the corresponding probabilistic graphical models, and formulate the bug localization problem as a statistical inference problem given partially observed data. We infer the posterior distribution of underlying circuit parameters, which provides a statistical measure of whether the bug lies in each sub-circuit.

## 4.1 Problem Formulation

Debug is known to be a bottleneck during circuit design and validation. It is not uncommon to take days to months to find a root-cause of the problem [5, 6, 65], especially for analog/mixed-signal circuits. In this chapter, we focus on the problem of *bug localization*, *i.e.*, identifying the sub-circuit/parameter that leads to the failure observed during simulation or measurement. For example, in a high-speed I/O link, the eye diagram at the receiver is a key link characteristic and might be observed to be "closed" during measurement. Upon the failure observation, we would like to identify whether the problem is caused by the transmitter, the channel, or the receiver, and subsequently, which parameter of the sub-circuit causes the eye closure.

While there have been various techniques developed for debugging digital circuits (such as formal methods [66], ATPG [67], trace mining [68], *etc.*), analog/mixed-signal debug is largely empirically based and ad hoc. The localization problem is often tackled manually by expert designers, based on a series of controlled experiments relying on extensive knowledge of how the circuit works. Furthermore, during post-silicon validation, the limited observability and controllability of circuit (*e.g.*, inability to probe internal voltages and currents) makes the bug localization problem more difficult. As a result, the circuit debugging is usually the most time-consuming process during post-silicon validation and presents a critical bottleneck for product qualification.

In this chapter, we present a bug localization methodology that automatically identifies and ranks potential failure root-causes for analog and mixed-signal systems, relying on limited observation of circuit responses. The key idea and novelty is to model the circuit using probabilistic graphical models and to employ statistical inference to estimate the posterior distribution of the (unknown) circuit parameters based on partially observed circuit responses. The use of probabilistic graphical models is beneficial in several aspects:

- Both linear and nonlinear circuits can be modeled by specifying the corresponding conditional probabilities (Section 4.2).
- Circuit and environmental uncertainties can be naturally modeled in the probabilistic graphical model (Section 4.2).
- Partially observed data can be easily incorporated in the statistical inference framework (Section 4.3.1).
- Circuit controllability can be modeled by expanding the probabilistic graphical model (Section 4.3.3).
- Potential bugs can be localized and ranked in a probabilistic manner based on the estimated posterior distributions (Section 4.3).

In particular, we treat hidden circuit parameters as random variables, and apply a Markov Chain Monte Carlo algorithm (Gibbs sampling) [11] to estimate the posterior distribution of hidden circuit parameters as well as internal signals that are not observable. The resulting posterior distributions are then used to probabilistically localize the bug. We illustrate the effectiveness of our method on a high-speed

I/O link. We demonstrate that our method can successfully identify the sub-circuit that causes the eye closure at the receiver side of the link, given only the input and output waveforms of limited length.

The rest of the chapter is organized as follows. Section 4.2 introduces the modeling of analog/mixed-signal circuits using graphical models. Section 4.3 describes the bug localization methodology which is composed of statistical inference and probabilistic ranking. Section 4.3.3 addresses a few critical issues in order to successfully apply the proposed methodology. Section 4.4 presents experimental results on a high-speed I/O link circuit.

## 4.2 Modeling of AMS Circuits using Probabilistic Graphical Models

Deterministic models are commonly used to model analog/mixed-signal circuits. At the transistor level, Modified Nodal Analysis (MNA) equations are used in SPICE(-like) simulators [69]. At the system level, behavioral models of various types [70] (*e.g.*, response surface models [71], phase macromodels [72, 73] ) and implementations (*e.g.*, SystemVerilog, VerilogAMS, Simulink [4]) are often used in practice. Based on the deterministic model, randomness is then handled by imposing probability distributions on circuit parameters and input stimuli.

While deterministic modeling is extremely useful for circuit simulation, it is probably not the best choice for the problem of bug localization, at least for two reasons. First, the values of the unobserved signals and parameters are non-deterministic. In fact, due to system noises as well as the limited observations of input/output waveforms, they should admit a range of values, each associated with a "confidence measure". Second, in the post-silicon setting, the deterministic models can only predict rough circuit responses due to the inaccuracy of device models and the ignorance of various noise sources.

Instead of deterministic models, we propose to use probabilistic graphical models to model analog/mixed-signal circuits, and they naturally address the two technical challenges raised above.[1] In this section, we focus on explaining the key concepts of

[1] We refer the readers to three excellent books [11, 13, 14] on probabilistic graphical models.

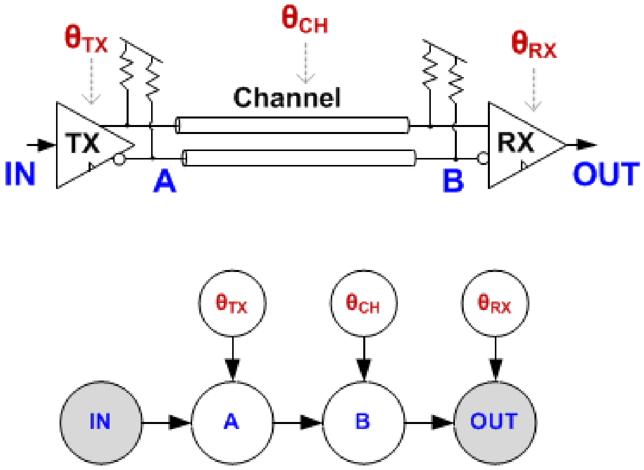


Figure 4.1. Probabilistic graphical model of a I/O link system.

probabilistic graphical models as well as how to apply them to model analog/mixed-signal circuits.

### 4.2.1 Probabilistic graphical models

The concepts of probabilistic graphical modeling can be explained through a concrete example. Fig. 4.1 shows the diagram of a typical high-speed I/O link as well as its graphical model. The link is composed of a transmitter (TX), a channel (CH) and a receiver (RX). We denote the output of the TX to be $A$ and the output of the channel to be $B$. Therefore, the signal flows as $IN \rightarrow A \rightarrow B \rightarrow OUT$. We further denote the parameters for TX, CH and RX to be $\theta_{TX}$, $\theta_{CH}$ and $\theta_{RX}$, respectively.

A probabilistic graphical model consists of two components – a graph denoting independence structure of random variables (*i.e.*, conditional independency) and a joint probability distribution in factorized form. In Fig. 4.1, the link is modeled by a directed acyclic graph (DAG) where each node represent the signals or parameters, and the edges denote the dependence structure (*e.g.*, signal $B$ depends on the channel input $A$ and channel parameters $\theta_{CH}$). The graphical model essentially characterizes circuit behaviors using the joint probability distribution of all signals and parameters.

Yet, multi-dimensional joint probability distributions can be extremely expensive to characterize. However, the structure defined by the graph allows us to factorize the joint distribution of all the random variables in the form of $\prod_x p(x|\text{parents}(x))$. In particular, for the example above, we can factorize the joint distribution as

$$\begin{aligned} & p(IN, A, B, OUT, \theta_{TX}, \theta_{CH}, \theta_{RX}) \\ =& p(IN)p(\theta_{TX})p(A|IN, \theta_{TX}) \\ & p(\theta_{CH})p(B|A, \theta_{CH})p(\theta_{RX})p(OUT|B, \theta_{RX}). \end{aligned} \tag{4.1}$$

To fully specify the joint distribution, we just need to characterize the conditional probability distributions $p(A|IN, \theta_{TX})$, $p(B|A, \theta_{CH})$ and $p(OUT|B, \theta_{RX})$. In the following, we present two conditional probability distribution (CPD) models that are suitable for modeling linear and nonlinear circuits, respectively.

### Gaussian Bayesian network (GBN)

In a Gaussian Bayesian Network (GBN), the CPD of a node is a Gaussian distribution whose mean is a linear combination of its parent nodes and whose variance is fixed. Therefore, it can be used to model linear time-invariant circuits such as a linear equalizer or a channel, together with a term that models noises at the output.

In particular, we can model a linear time-invariant circuit by its transfer function (in Laplace domain or $z$-domain), and equivalently by a set of differential equations or difference equations. Take the difference equation as an example, the output is a linear combination of past values of the input and output signals. Take the difference equation as a representation of a circuit (with a proper time step for time discretization), the output is a linear combination of past values of the input and output signals. Therefore, the corresponding CPD can be set as a Gaussian distribution whose mean is determined by the underlying difference equation and whose variance models noises in the circuit.

For example, an active continuous-time linear equalizer (CTLE) with 1 zero and 2 poles (as shown in Fig. 4.2(a)) can be modeled by a difference equation

$$y[n] = b_0x[n] + b_1x[n-1] + a_1y[n-1] + a_2y[n-2], \tag{4.2}$$

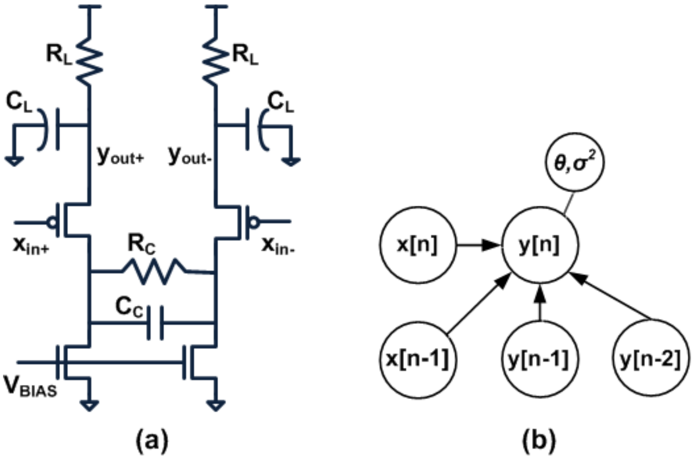


Figure 4.2. (a) Continuous-time linear equalizer and (b) its GBN model.

where $x$ is the input, $y$ is the output, $n$ is time index of the sampled signals, and $\boldsymbol{a} = [a_1, a_2]$ and $\boldsymbol{b} = [b_0, b_1]$ are the coefficients of the difference equation (and hence the transfer function).

Therefore, the GBN for the CTLE can be built as shown in Fig. 4.2(b), where the CPD for $y[n]$ is a Gaussian distribution with mean $b_0x[n] + b_1x[n-1] + a_1y[n-1]+a_2y[n-2]$, and variance $\sigma^2$ which models the variance of the noise. For notational convenience, we denote $\theta = [a_1, a_2, b_0, b_1]$ to be all the unknown parameters of the circuit.

### Table Bayesian network (TBN)

While GBN is suitable for modeling linear circuits, it is not easy to extend it to model nonlinear circuits. We resort to Table Bayesian Network (TBN) model for modeling nonlinear circuits.

In a TBN, both the parent and child nodes take discrete values, and the CPD of a node is simply a look-up table (LUT) which stores the conditional probabilities for all combinations of values of parent nodes – we call this LUT the $\gamma$-table which stores all the parameters for a TBN, just like $\theta$ and $\sigma$ parameters in a GBN model.

In our application, however, the signal and parameter nodes take continuous values. To model a circuit using TBN, we discretize each signal $x$ into a number of intervals $[l_i, h_i), i = 1, \cdots, K$, and denote the corresponding discrete values to be $i$.

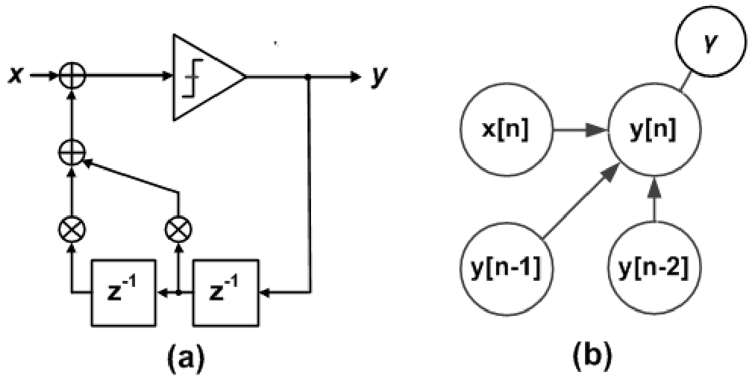


Figure 4.3. (a) Decision feedback equalizer and (b) its TBN model.

When $x$ takes values in $[l_i, h_i)$, we assume that it is uniformly distribution in $[l_i, h_i)$. Therefore, we can fill in the $\gamma$-table by computing the probability of output being in an interval given that input (including its past values) is in another interval. Since the TBN essentially stores the input/output relationship in a LUT, we can model arbitrary nonlinearity in the circuit in a probabilistic way.

For example, a decision-feedback loop equalizer (DFE) with 2 taps can be modeled by TBN, as shown in Fig. 4.3. In this case, $y[n]$ depends on $x[n]$, $y[n-1]$ and $y[n-2]$. By discretizing all the signals, we can compute the $\gamma$-table by randomly sampling $x[n]$, $y[n-1]$ and $y[n-2]$, and fill in the probabilities for $y[n]$. With the $\gamma$-table, the DFE's output $y[n]$ can be probabilistically modeled by a multinomial distribution. For instance, if the DFE's input signal $x$ is at the quantized level 7 and its output signal $y$ was at quantization level 0 and 1 for the two past sampling times, all the table entries with $(x[n], y[n-1], y[n-2])$=$(7, 0, 1)$ are loaded from the LUT, and normalized to form a multinomial distribution for $y[n]$.

Since the TBN requires to store output distributions in the $\gamma$-table for every possible input combinations, the space complexity of the model may grow explosively. Yet, the $\gamma$-table in TBN models tends to be sparse and space complexity grows gracefully. This is because (1) the conditioning variables in the TBN model are limited to the local neighbors in its graphical model and (2) a target sub-circuit for TBN modeling may be nearly deterministic and a distribution for a given input in the $\gamma$-table tends to have a non-zero element for only a few entries in the table.

### 4.2.2 Generating probabilistic graphical models for AMS circuits

With GBN and TBN models, we can model analog/mixed-signal circuits using probabilistic graphical models. That is, linear sub-circuits can be modeled by GBNs and non-linear sub-circuits by TBNs. For example, a I/O link circuit may be composed of a TX feed-forward equalization with 3 taps (FFE), a channel (CH), a continuous-time linear equalizer (CTLE) and a decision feedback equalizer (DFE). Since the FFE, CH and CTLE can be reasonably assumed to be linear circuits, we can model them by GBNs. DFE is a nonlinear block and can be modeled by TBN. The FFE has 3 taps, and hence, its GBN model takes the current and the past three samples of the input signal as the parent nodes of the output node. The channel can also be modeled by pole/zero model and converted to a GBN model. Specifically, if an S-parameter model is given, we can approximate the response by a transfer function with a few poles and zeros [74], which can then be converted to a GBN model. The CTLE in the example is an active linear equalizer usually characterized by 1 zero and 2 poles and its GBN model can be constructed accordingly. The DFE contains a (highly) nonlinear quantizer and thus it is modeled by a TBN model.

The resulting graphical model for the I/O link example is shown in Fig. 4.4 – it is a mixture of GBN and TBN models. Each node represent either a signal or a parameter. The signal nodes represent the signal value at a particular time sample (*e.g.*, $A_2$ is the value of signal $A$ at time 2). The parameter nodes represent the unknown parameters of the circuit (*e.g.*, $\theta_{CH}$ is the parameter for the CH), and are shared by all signal nodes of the same sub-circuit.

From the above discussion, we remark that we can address the two challenges raised in the beginning of this section. First, based on partial observation, we may compute the posterior distribution of the circuit parameters (as well as signals), which can be viewed as a range of values and their corresponding confidence measure. Second, it is straightforward to model circuit uncertainties and model inaccuracies by modifying conditional probability distributions. For example, to model an additive noise at the output of the TX, we can add variance term in $p(A|IN, \theta_{TX})$.

It is important to note, however, that size of the probabilistic graphical model (*i.e.*, the number of nodes) can grow quickly. In particular, if the Table Bayesian

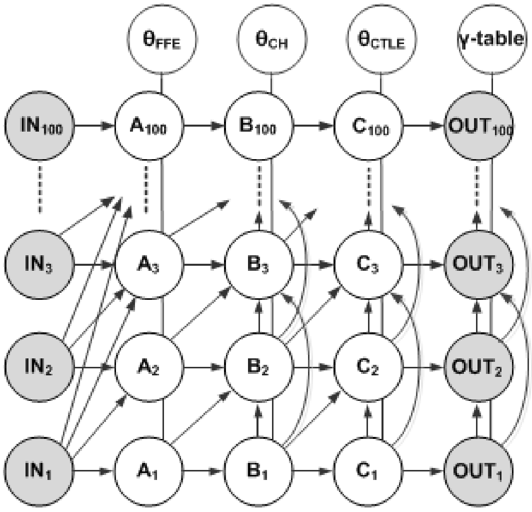


Figure 4.4. Bayesian network model of a four-stage I/O link system with the 100-length input/output waveform evidences. Shaded nodes indicate presence of evidences on the nodes.

Network (TBN, Section 4.2.1) is used, then the size grows exponentially with respect to the maximum number of random variables in all conditional probability distributions. However, due to the locality of circuits (*i.e.*, each node is determined given its neighbors, and the number of neighboring nodes is small), the size of the graphical model grows not quickly with respect to the circuit size.[2]

## 4.3 Probabilistic Bug Localization using Probabilistic Graphical Models

With the probabilistic graphical models in hand, our methodology localizes circuit bugs in two steps:

1. Using the observed waveforms as evidence, estimate the posterior distributions of unknown circuit parameters in the graphical model using statistical inference.
2. Based on the posterior distributions of circuit parameters, localize and rank potential bugs in the circuit.

[2] When the connectivity in a circuit is not complex, it is almost linear to the circuit size.

### 4.3.1 Posterior estimation using statistical inference

In the first step, we estimate the posterior distribution of each circuit parameter (*i.e.*, $\theta$ parameters of GBN models, and $\gamma$-table parameters of TBN models), based on the observed waveforms. In another word, we would like to compute the marginal conditional distribution of the parameters conditioned on the observed random variables.

This is known as the *statistical inference* problem in statistics and machine learning literatures [13]. There have been many existing techniques developed for this problem. In this paper, we use a Markov Chain Monte Carlo method called Gibbs sampling, which is a sampling-based approximate inference algorithm.[3]

Gibbs sampling is typically used when the conditional distribution of each random variable is known and is easy to sample from – this is exactly the case for the GBN and TBN models we established in Section 4.2. Assuming there are $N$ random variables $x_1, \cdots, x_N$ whose joint distribution is $p(x_1, \cdots, x_N)$. Gibbs sampling starts with an initial value for all the variables $x_1, \cdots, x_n$, and then iteratively samples each variable $x_i$ from the conditional distribution $p(x_i|x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_N)$. It can be shown [13] that under certain mild conditions, the distribution of the samples for $x_i$ converges to the marginal distribution of $x_i$. If the random variables $x_{ob}$ are observed, we can simply set them to be the observed value. In that case, the distribution of Gibbs samples of $x_i$ converges to the conditional marginal distribution $p(x_i|x_{ob})$.

For example, consider a simple graphical model shown in Fig. 4.5 where signal $B$ is observed. We would like to estimate $p(A, C, D|B_{ob})$ using Gibbs sampling. Gibbs sampling begins with an initial guess $(A_0, C_0, D_0|B_{ob})$, and it iteratively samples variables $A, C, D$ from their conditional distributions, *i.e.*, sampling from $p(A_i|B_{ob}, C_{i-1}, D_{i-1})$, $p(C_i|B_{ob}, A_i, D_{i-1})$ and $p(D_i|B_{ob}, A_i, C_i)$ iteratively. We repeat this process $N$ times and discard the first few ($K$, known as the burn-in time) samples. If $N$ is large enough, $(A_{K+1}, C_{K+1}, D_{K+1})$,...,$(A_N, C_N, D_N)$ are the

[3] There is, however, no reason one cannot use other inference algorithms. Indeed, we have also tried exact inference algorithms such as junction tree algorithm. The major drawback for exact inference algorithms is that their space and time complexity are too large for our bug localization problem.

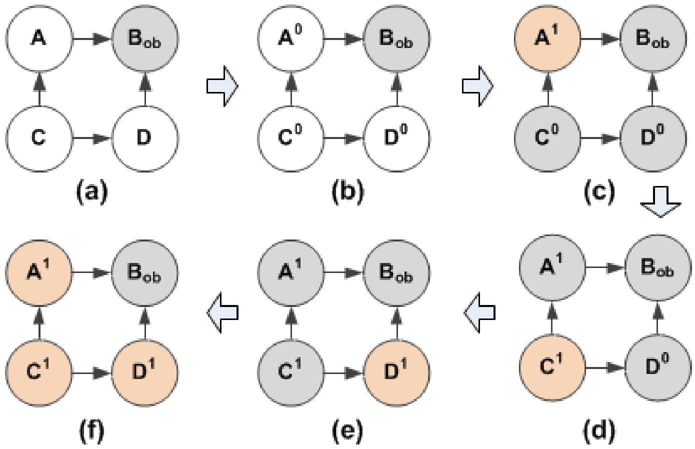


Figure 4.5. (a) Estimating $p(A, C, D|B_{ob})$ by Gibbs sampling: (b) initial guess $(A_0, B_{ob}, C_0, D_0)$; (c) sampling A according to $p(A|B_{ob}, C_0, D_0)$; (d) sampling C according to $p(C|A_1, B_{ob}, D_0)$; (e) sampling D according to $p(D|A_1, B_{ob}, C_1)$; (e) a Gibbs sample $(A_1, B_{ob}, C_1, D_1)$.

samples from the desired posterior distribution $p(A, C, D|B_{ob})$ [11, 13, 14].

A critical choice for the Gibbs sampling in our graphical model is how to choose the order of random variables to be sampled – the order affects the convergence of the algorithm. In our implementation, signal nodes are traversed first before the parameter nodes, and empirically we observe that it converges reasonably fast.

Specifically, we first guess the unknown circuit parameters according to design specifications, and then propagate the signal values from the input to the output (*e.g.*, in the order of $A$,$B$ and $C$ in Fig. 4.4). Second, Gibbs samples of signal nodes in the Bayesian network are taken. For example, in Fig. 4.4, Gibbs sampling starts from the left bottom node $A_1$. For each visit to a node, the node's Markov blanket (*i.e.*, all the parents of the node, the children of the node and all the parents of the children) are used to compute the node's conditional probability $p(A_1|IN_1, B_1, B_2, A_2)$ and a Gibbs sample is taken from the computed distribution. When the node is a GBN model, the conditional probability $p(A_1|IN_1, B_1, B_2, A_2)$ is also Gaussian [11] which can be easily sampled. In the case of a TBN node, the conditional probability of $C_1$ (i.e., $P(C_1|B_1, B_2, B_3, C_2, C_3, OUT_1)$) is computed by accessing all the associated $\gamma$-table in its Markov blanket, loading all the rows with the previous Gibbs sample values (e.g. $B_1, B_2, B_3, C_2, C_3, OUT_1$) from the tables, and multiplying them. From the computed conditional probability, a new sample value is taken and these steps are repeated for every signal node in the Bayesian network.

After sampling the signal nodes, parameter nodes are visited and parameter Gibbs samples are taken. Gibbs samples of $\theta$ parameters in GBN can be taken from Gaussian distribution because both the CPD and the prior distribution of the parameter are Gaussian [11]. It is also possible to take Gibbs samples of $\gamma$-tables in TBN because a sample of a Dirichlet distribution can be taken from a Gamma distribution [13].

### 4.3.2 Probabilistic bug localization and ranking

From the Gibbs samples, we can estimate the posterior distribution of each parameter. The posterior distribution represents our belief about what values the parameter should take given all the observations we made about the circuit. Hence, if the reference value of the parameter (*i.e.*, the true value specified by designers) is not "covered" by the posterior distribution, we claim that the parameter is potentially a bug root-cause, and therefore the associated sub-circuit is a possible bug root-cause.

To be more precise, we assume a *spec-range* is given for each parameter (usually in the form of a lower bound and an upper bound), and we compute the probability of a parameter being within the spec-range from the estimated posterior distribution. If this probability is too small, we report that the parameter and the associated circuit is a potential bug root-cause.[4]

For example, Fig. 4.6-(3) shows the posterior distributions of three parameters from TX, CH and RX respectively. It is observed that, $\theta_{TX}$ and $\theta_{RX}$ are "covered" by their posterior distributions, while $\theta_{CH}$ is in the tail of its posterior distribution. Hence, we report the channel to be the root-cause of the output failure.

It is highly possible that several parameters are simultaneously likely to be the actual bug root-cause – bug ranking is necessary. In the proposed methodology, we rank these root-causes according to the probability of the parameters being within their spec-range.

[4] The Gibbs samples can also be used to estimate multivariate posterior distributions of parameters and signals.

The complexity of the algorithm depends on the constructed graphical model. The first is the size of the graphical model and the second is the number of samples we take for inference. The size of the graphical model will grow as the number of hidden signals, parameters (i.e, horizontal direction in Fig. 4.4) and the length of evidence waveform (i.e, vertical direction in Fig. 4.4) increase. Hence, the computational cost is roughly linear with respect to the size of the graph and the number of samples.[5]

To summarize, the proposed bug localization and ranking flow consists of 5 steps, as is illustrated in Fig. 4.6:

1. Generate a probabilistic graphical model (Bayesian Network) for a given circuit using Gaussian Bayesian network (GBN) for linear circuits and Table Bayesian network (TBN) for non-linear circuits.
2. Obtain observations of the circuit from measurement or simulation (*e.g.*, input and output waveforms of a circuit).
3. Obtain Gibbs samples of unknown parameters (*i.e.*, $\theta$s and $\gamma$-tables) of each sub-circuit, conditioned on observed evidence.
4. Estimate the conditional marginal posterior probability of each parameter to satisfy a given spec from the Gibbs samples.
5. Rank the potential problematic parameters according to the posterior probabilities of different parameters.

### 4.3.3 Implementation details

#### Adding controllability

When there is limited observability, two phenomena are sometimes observed in our experiments. First, the posterior distribution of parameters tends to be very wide – this may greatly degrade the accuracy of the proposed method. Second, the posterior distribution may exhibit multiple modes. There may be several reasons for this phenomenon. One major reason is that from the given evidence about in-

[5] It should be noted again that the complexity also depends on how sophisticated the sub-circuit is. For example, in an extreme case when the circuit is very complex requiring to consider at least 10 sample history of its input and modeled by TBN, the number of bins in the table increases very quickly and its complexity becomes very high.

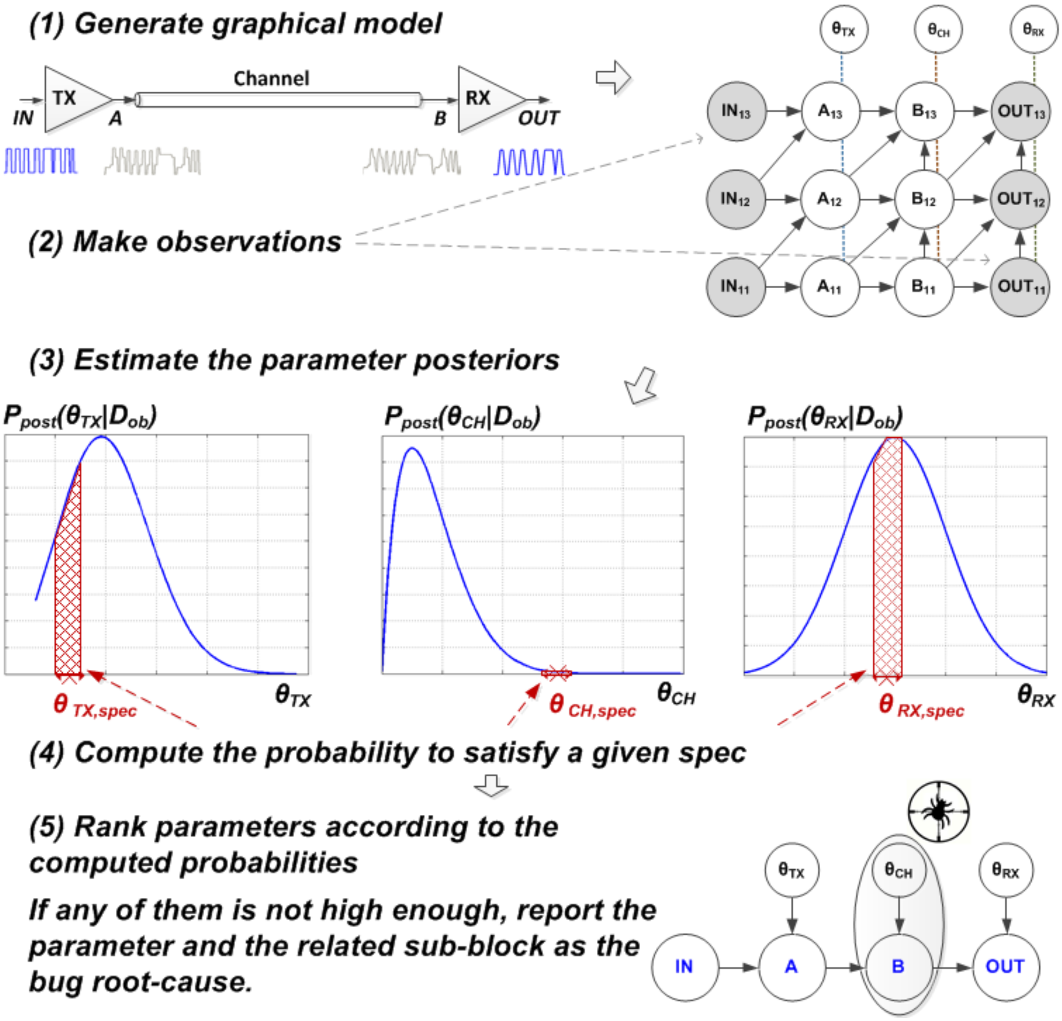

(1) Generate graphical model
TX
Channel
RX
IN
A
B
OUT
$\theta_{TX}$
$\theta_{CH}$
$\theta_{RX}$
$IN_{13}$
$A_{13}$
$B_{13}$
$OUT_{13}$
$IN_{12}$
$A_{12}$
$B_{12}$
$OUT_{12}$
$IN_{11}$
$A_{11}$
$B_{11}$
$OUT_{11}$
(2) Make observations
(3) Estimate the parameter posteriors
$P_{post}(\theta_{TX}|D_{ob})$
$P_{post}(\theta_{CH}|D_{ob})$
$P_{post}(\theta_{RX}|D_{ob})$
$\theta_{TX,spec}$
$\theta_{TX}$
$\theta_{CH,spec}$
$\theta_{CH}$
$\theta_{RX,spec}$
$\theta_{RX}$
(4) Compute the probability to satisfy a given spec
(5) Rank parameters according to the computed probabilities
If any of them is not high enough, report the parameter and the related sub-block as the bug root-cause.
$\theta_{TX}$
$\theta_{CH}$
$\theta_{RX}$
IN
A
B
OUT

Figure 4.6. Proposed bug localization method using probabilistic graphical models.

put/output waveforms, the actual bug root-cause is not distinguishable from another possible root-cause.

For example, for the link circuit example in Fig. 4.4, if only input and output of the link are observed, pole and zero parameters of different linear sub-components can be freely exchanged while still producing the same output for the given input. Therefore, it is impossible to localize the bug unless extra information is provided.

Fortunately, in practice, we often have several controllable knobs in the circuit that may be tuned in post-silicon stage (*e.g.*, the resistors in the active CTLE in Fig. 4.2), and we may control environmental variables such as temperature and voltage. Moreover, we may change the experiment setup by swapping dies and boards.

All these situations lead to a graphical model almost identical to the original one, but with one or two sub-circuits substituted. It is naturally handled by our methodology by expanding the graphical model. For example, Fig. 4.7 shows a graphical model with two CTLEs (with different resistance settings) connected to the same channel – note that $OUT_{1\cdot}$ and $OUT_{2\cdot}$ depend on the same channel parameters but different CTLE parameters.

With extra controllability, we greatly reduce the variance of the posterior distributions and increase the parameter estimation accuracy. Additionally, we avoid the potential confusion between different circuit blocks. This will be further illustrated in Section 4.4 on a high-speed I/O example.

### Handling linear circuits in series

Another serious problem for localizing bugs exists for linear circuits in serious – it is hard, if not impossible, to estimate the gain of different stages based on only input/output waveforms. In fact, there are infinite number of valid gain assignments for the sub-circuits as long as their net multiplication value remains same.

To overcome this problem, we force the sub-circuit gain to be unity, and add a *virtual gain* stage at the end of a chain of linear sub-circuits. The virtual gain is simply the product of the gains of all sub-circuits. This effectively constrains the $\theta$ parameters in GBN to be correlated to only pole/zero locations. The valid parameter space of $\theta$ is further restricted, again reducing the variance of the estimated posterior distributions. This is illustrated in Fig. 4.7 where signal $D$ is inserted to the original

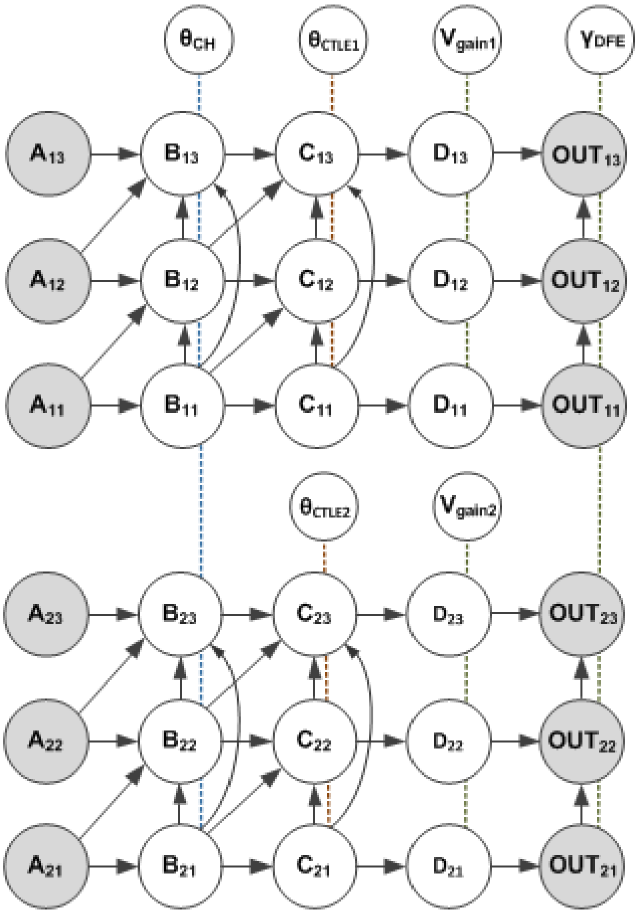


Figure 4.7. Adding controllability ($\theta_{CTLE1}$ and $\theta_{CTLE2}$) and virtual gain stages ($V_{gain1}$ and $V_{gain2}$) in the probabilistic graphical model.

graphical model and the sub-circuit from $C$ to $D$ is simply a gain stage.

### Connection to circuit parameters

In bug localization, we may want to pin-point the particular circuit parameter (*e.g.*, such as a resistance, a transistor) that causes the failure. To do that using the proposed methodology, we can further expand the graphical model by adding nodes for these circuit parameters, and adding edges between these parameters and $\theta$ or $\gamma$ parameters. For instance, for the CTLE in Fig. 4.2, additional graph nodes for capacitors and resistors in the circuit can be created and $\theta$ parameter can be set to be child of them. We can then apply Gibbs sampling on the expanded graphical model and use the same ranking criteria as described in Section 4.3.[6]

### Difference from conventional methods

While one can view the proposed method as a system identification method, it is different from conventional linear system identification methods [75] in several ways. The major difference is that our method computes a posterior distribution of each underlying parameter, rather than a single "best" value in traditional system identification methods. This brings several benefits. First, the "best" estimate in traditional methods can be misleading because there might be multiple local optima that are close to the global optimum, while a probability distribution can capture all possible parameter values. Second, the variance of the posterior distribution gives a rough measure of the estimation accuracy and confidence. For example, assume that the posterior of a parameter is a uniform distribution over $[a, b]$, $b \gg a$, *i.e.*, any value in $[a, b]$ might be a valid assignment for this parameter, and it also tells that we cannot estimate this parameter accurately from the given evidence, and need to gather more evidence for better estimation. However, the "best" estimate from Maximum Likelihood Estimation might be the mean $(a + b)/2$, which could be

[6] So far, we have focused on the case where the graphical model is directed and the circuit is feed-forward. However, it is possible to extend the model to undirected graphs and circuits with feedback. For example, PLLs and $\Sigma - \Delta$ ADCs are intrinsically a feedback system and the underlying graphical model will be a directed cyclic graph. For another example, the signal at the output of one block depends on the load of the subsequent block – the dependency may be modeled by undirected graphs as the signal flow is less clear.

misleading for bug identification purpose.

Besides the above major difference, our method gives a framework to handle various practical constraints. For example, it naturally handles limited controllability and observability in the debug process – *e.g.*, we may control only the bit pattern at the input signal, but not its magnitude/jitter, and we may observe quantized waveform or statistics of only a few output/internal signals. This fits well with our Bayesian inference approach where any unknown signal and parameter is treated as a random variable, and any observed quantity is treated as evidence. Our method also deals with arbitrary nonlinearity by modeling the circuit using TBN, while traditional methods usually assume a parameterized nonlinear model.

## 4.4 Experimental Results

The proposed probabilistic graphical model and the bug localization algorithm have been tested and verified for a realistic 5Gbps I/O link example. The example includes the 3-tap pre-emphasizer, the FR408 PCB backplane which is modeled in S-parameter and approximated by a pole/zero model using 1 zero and 2 poles, the active continuous-time linear equalizer (CTLE) with 1 zero and 2 poles, and lastly the decision feedback equalizer (DFE) with 2 taps. The first three sub-blocks are linear systems and thus are modeled as GBN. The last DFE contains nonlinear quantization operation and thus is modeled as TBN.

To obtain evidence for statistical inference, we simulate the circuit using a PRBS pattern, and observe only the input waveforms of the channel and the output waveforms of the CTLE. We also consider the controllability of the resistance $R_C$ in the CTLE that contributes to the zero and DC gain of the CTLE.

We apply our method and estimate the posterior distributions of the circuit parameters. For example, Fig. 4.8 shows the distribution of parameter $b_1$ for the CTLE. The black line represent the actual parameter value, and the blue distribution is the posterior distribution estimated using Gibbs sampling – it is observed that the posterior distribution covers the actual value.

In addition, from Fig. 4.8, the posterior distribution without controllability (Fig. 4.8(a)) exhibits multiple peaks and is much more dispersed than that with

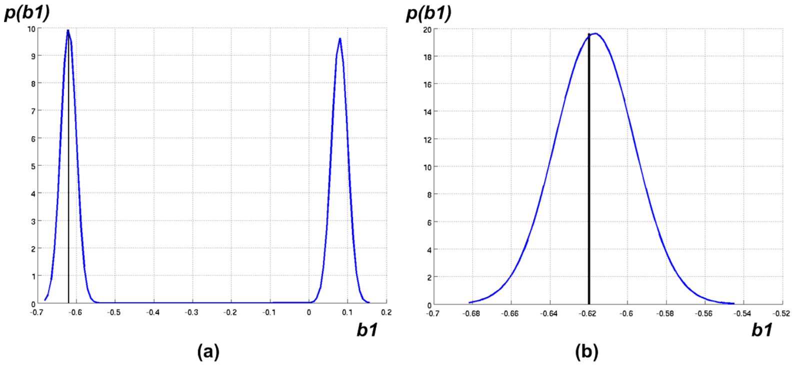


Figure 4.8. Estimated posterior distribution of one of the CTLE parameters. The black spike is the actual value of the parameter. (a) The posterior distribution when no controllability is applied in the experiments. (b) The posterior distribution when we consider controllability of the CTLE.

controllability (Fig. 4.8(b)), as is expected from the discussion in Section 4.3.3. It is more obvious to see this from the pole-zero plot, as shown in Fig. 4.9. Since the controllability is applied on the CTLE resistance $R_C$ that contributes to the zero, the estimation of the zero is much more accurate than that without controllability. More importantly, the distribution of the zero has a single mode instead of two modes when there is no controllability. From Fig. 4.9, we also observe that the estimated variance of the poles is reduced, even though the controllability is with respect to the zero. In our experiments, we observe roughly 45%-50% variance reduction for the poles.

Next, we further restrict observability by taking the output waveform of the DFE instead of the CTLE. In other words, two pairs of input/output waveforms with different settings for the resistance $R_C$ in the CTLE are given as available evidences. This is a much harder problem since the quantization of DFE removes much information of the circuit response. To demonstrate the capability of our method on the problem of bug localization, we use a different lossy channel (instead of a normal one), and apply our algorithm to estimate the posterior distribution of the channel parameters. Fig. 4.10 shows the estimated posterior distributions of the $\theta$ parameters (by 10,000 Gibbs samples) as well as the expected channel parameter values. We observe that the expected channel parameter value is not “covered” by

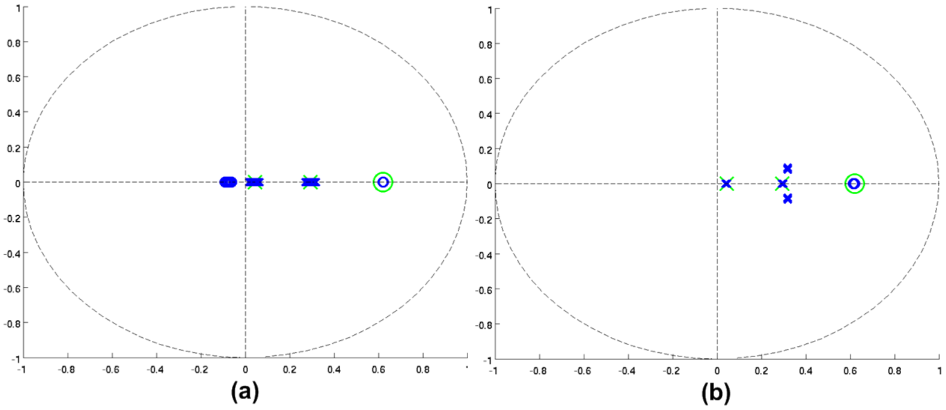


Figure 4.9. Pole-zero plot. Green crosses and circles represent the actual values of poles and zeros, respectively. The blue crosses and circles represent the Gibbs samples of the poles and zeros, respectively. (a) When used a single pair of input/output waveform evidences without controllability and (b) when used two pairs of input/output waveform evidences with controllability.

the posterior distribution, and therefore the algorithm concludes that the bug is caused by the channel variation. Table 4.1 shows the posterior probability of the pole/zero parameters in the system to be within the given spec-range (i.e., ±0.1 of the desired parameter value) and the buggy lossy channel is indeed identified with the low posterior probabilities. In the following figures (Fig. 4.11, Fig. 4.13, Fig. 4.12 and Fig. 4.14), the posterior distributions of the estimated parameters in the experiment are given in detail.

In summary, we have presented a method that uses probabilistic graphical models to perform bug localization for AMS systems. The underlying algorithm uses a novel probabilistic graphical modeling of the circuit, and applies statistical inference to estimate the posterior distributions of parameters for bug localization purpose. The proposed method has been successfully tested and verified for a realistic I/O link circuit example.

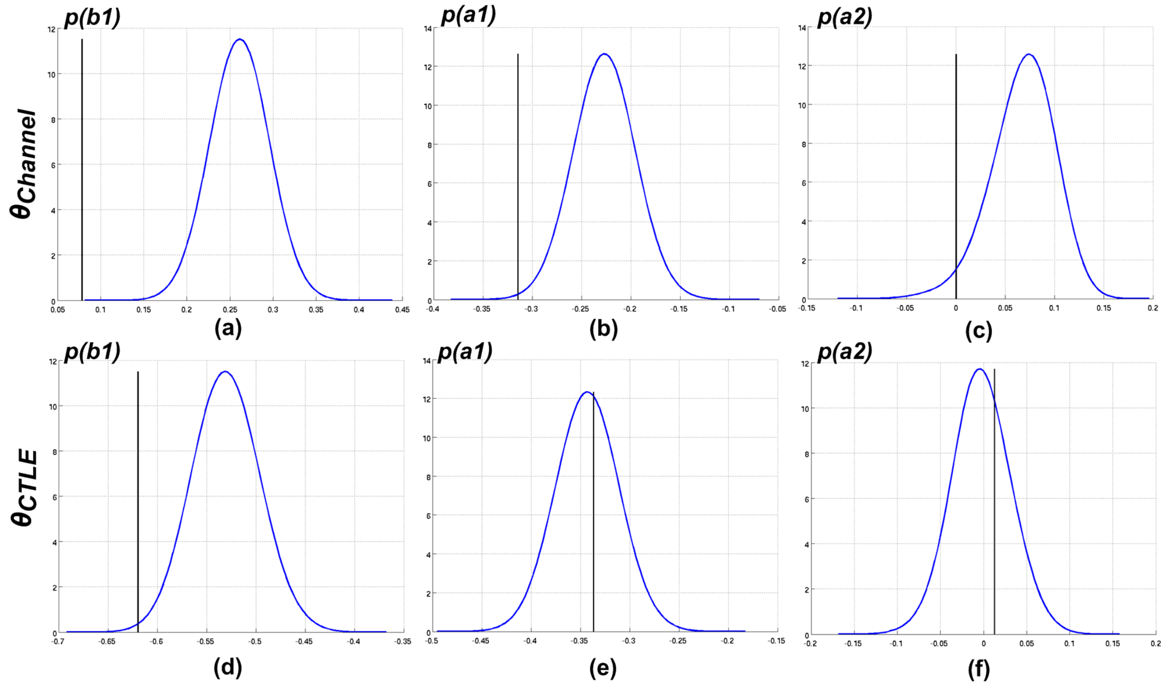


Figure 4.10. Posterior distributions of the channel parameters (a,b and c) and the CTLE parameters (d,e and f). The black spike is the expected value of the parameter.

Table 4.1. Posterior probabilities of the pole/zero parameters of the channel and the CTLE to be within the given spec-range.

| $P(\theta\ in\ \theta_{spec})$ | $Re(z_1)$ | $Im(z_1)$ | $Re(p_1)$ | $Im(p_1)$ | $Re(p_2)$ | $Im(p_2)$ |
|---|---|---|---|---|---|---|
| Channel | 0.17% | 100% | 1.7% | 5.4% | 15% | 5.4% |
| CTLE | 65% | 100% | 58% | 90% | 63% | 90% |

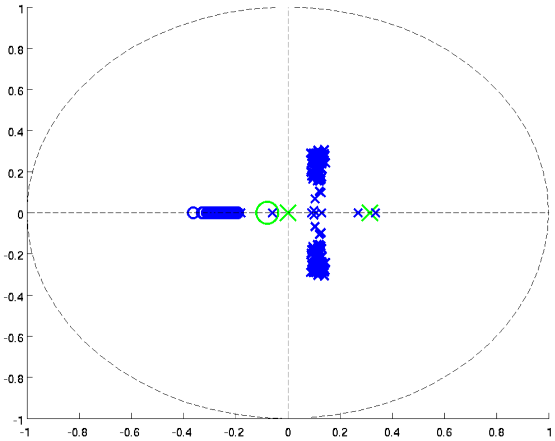


Figure 4.11. Posterior distributions of the channel's pole/zero parameters when the input of the channel and the output of the DFE are observed. The channel is modeled as 1 zero and 2 poles and samples of their posterior distributions of zero (o) and pole (x) locations are plotted in pole/zero map (blue). The green crosses (x) and a circle (o) are the desired pole/zero locations. From this figure, it can be seen that the posterior distribution does not cover the desired pole/zero location so that we can detect that the channel is buggy in this example.

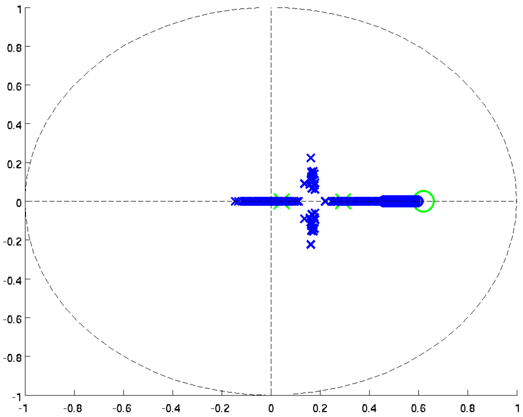


Figure 4.12. Posterior distributions of the CTLE's pole/zero parameters when the input of the channel and the output of the DFE are observed. Each sub-component (i.e., channel and CTLE) has 1 zero and 2 poles and samples of their posterior distributions of zero (o) and pole (x) locations are plotted in pole/zero map (blue). The green crosses (x) and a circle (o) are the desired pole/zero locations.

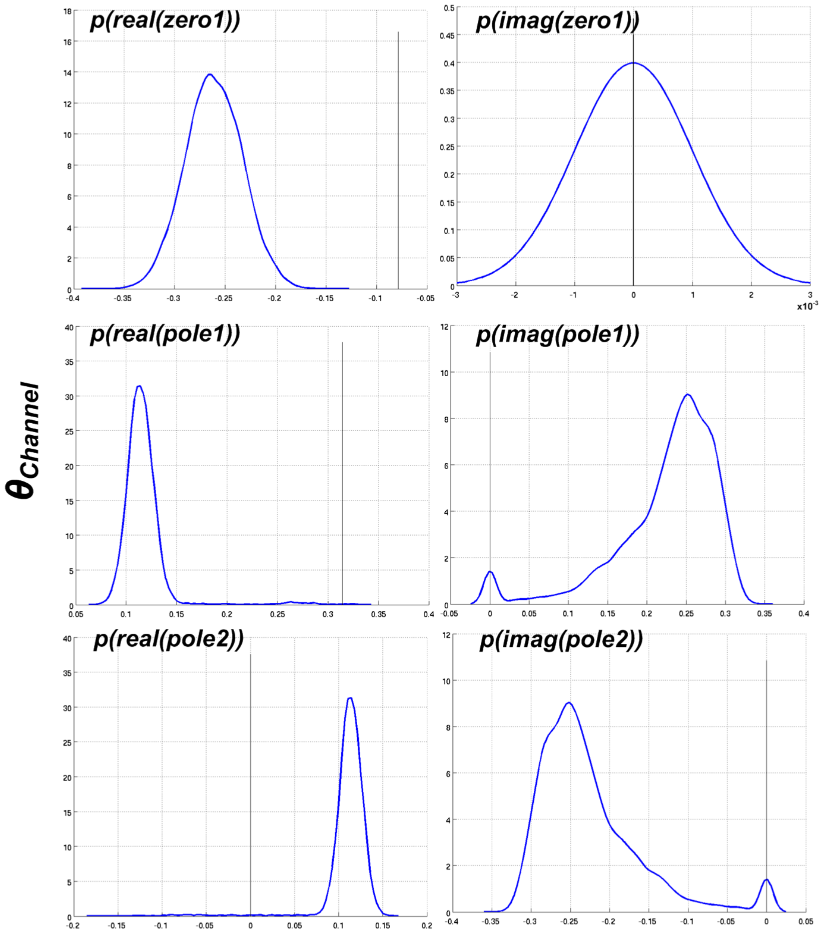


Figure 4.13. Posterior distributions of the channel's pole/zero parameters when the input of the channel and the output of the DFE are observed. Each sub-component (i.e., channel and CTLE) has 1 zero and 2 poles and their posterior distributions of real part and imaginary part are displayed. The black spike indicates the desired parameter value.

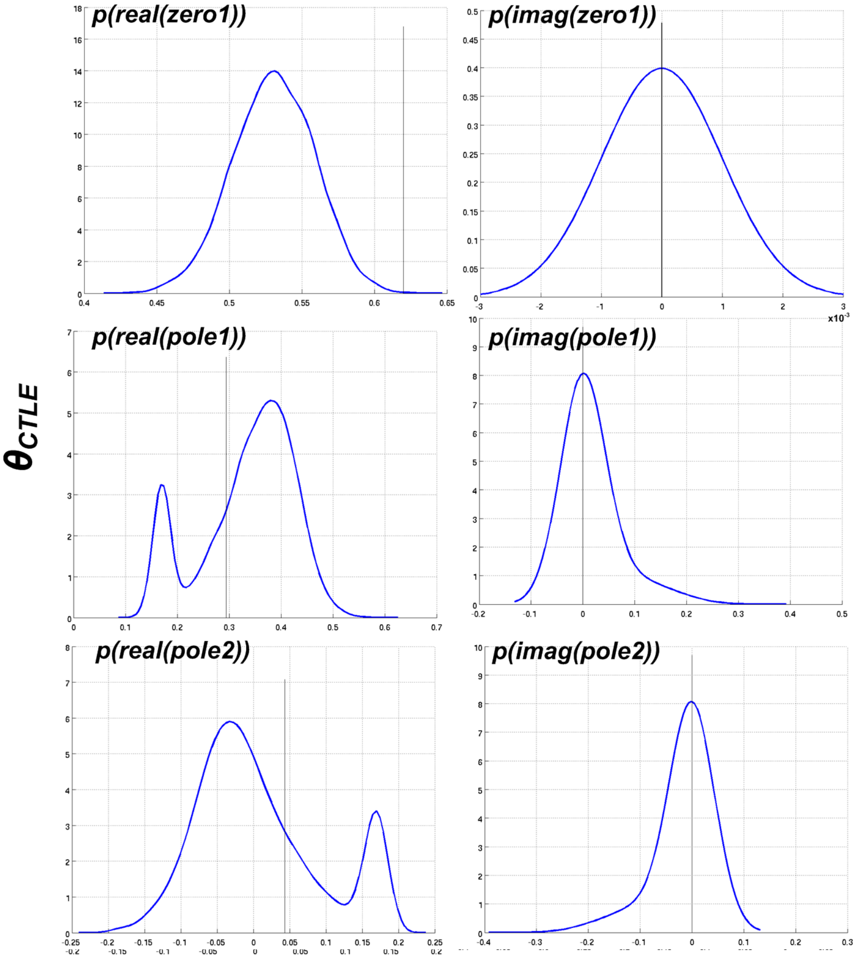


Figure 4.14. Posterior distributions of the channel's pole/zero parameters when the input of the CTLE and the output of the DFE are observed. The CTLE has 1 zero and 2 poles and their posterior distributions of real part and imaginary part are displayed. The black spike indicates the desired parameter value.

## 4.5 Possible Extensions of Graphical Models – Equivalence Checking

Today's high-performance and complex VLSI systems are partly enabled by the widespread use of formal equivalence checking techniques. For example, a high performance digital system such as a microprocessor is designed in a top-down fashion-from a high-level model such as transaction level model (TLM), through the register transfer level (RTL), and down to the transistor level. Equivalence checking is a key step in this design flow which confirms that the design description in one abstraction level is equivalent to that in [76].

However, the notion of equivalence checking is poorly defined for analog and mixed-signal (AMS) systems while it is well established for digital systems [32, 77]. Hence, the prior works and uses of equivalence checking have been primarily for digital circuits, where the prevailing approach is to simplify the problem of comparing two general machines to that of checking two combinational logics [76]. In contrast, the main hurdle to establishing the equivalence between two AMS systems is the fact that the systems have continuous states and nonlinear behaviors. For linear systems, one possible way is to compare their transfer functions. For nonlinear systems, a transformation may exist that enables the similar comparison, but finding such a transformation is difficult for general AMS circuits [77].

It is noteworthy, however, that there exists an effective verification method that is widely used in the current AMS design flow: namely, layout versus schematic (LVS). In LVS, the circuit netlists described in the schematic and in the layout are compared to check their equivalence in connectivity [76, 78]. Basically, LVS checks whether the graphs represented by the two netlists are isomorphic.

This section briefly explores an approach that extends LVS and checks the functional equivalence as well as topological equivalence using a probabilistic graphical model. The main idea is to describe a circuit as a Markov network (MN) that represents not only the topological configuration but also its functional behavior.

### Markov network representation of AMS systems

Typically, an analog circuit is represented by a transistor-level netlist, i.e., a set of connected devices such as resistors, capacitors, and transistors. In other words, the circuit's behavior is described by an ordinary differential equation (ODE), e.g. constructed based on the modified nodal analysis algorithm [76]. A solution to the ODE can be found by solving an approximate, discrete-time algebraic equation, e.g. via Newton-Rhapson method.

The connectivity information in this circuit netlist reveals the conditional independence properties among the nodes and it implies that the derived connectivity graph of the circuit can be considered as a Markov network (MN). In this MN, each unknown nodal voltage corresponds to a random variable in the MN and the random variables in the MN satisfy all the conditional independences implied by the graph. For instance, if two nodes, $A$ and $B$, have no direct connection in-between and the values of the intermediate nodes between $A$ and $B$ that block any path from $A$ and $B$ is known, the circuit nodal equations can be partitioned into two sets and solved independently. More precisely speaking, a Markov network can be constructed by creating a graph node for each circuit node and forming an edge between every two nodes that are connected to the same primitive instance. Each vertex in the graph is a two dimensional random vector, consisting of the current state and previous state. This is because a circuit can be described by a discrete-time algebraic equation $X[n] = X[n-1] + F(X[n-1], U[n])$ , where $X[n-1]$, $X[n]$ and $U[n]$ represents the previous state, current state and current input, respectively, given that the time difference between (n-1)-th and n-th step is small enough. If only the DC characteristics of circuits are in question, each vertex in the MN becomes a single-valued random variable which satisfies the nonlinear algebraic DC nodal equations.

The Markov network can capture the circuit's characteristics via its probabilistic relationships among the nodes. The idea is to express the relationships of the circuit nodes using the joint distributions instead of the circuit equations. For example, the ideal CMOS NAND gate in Fig. 4.15 (a) is composed of $A, B, C$ and $OUT$ nodes and the circuit's characteristic could be expressed as a set of distributions. That is, if both $A$ and $B$ are conditioned to be between $VDD$ and $VDD/2$ (i.e., logic

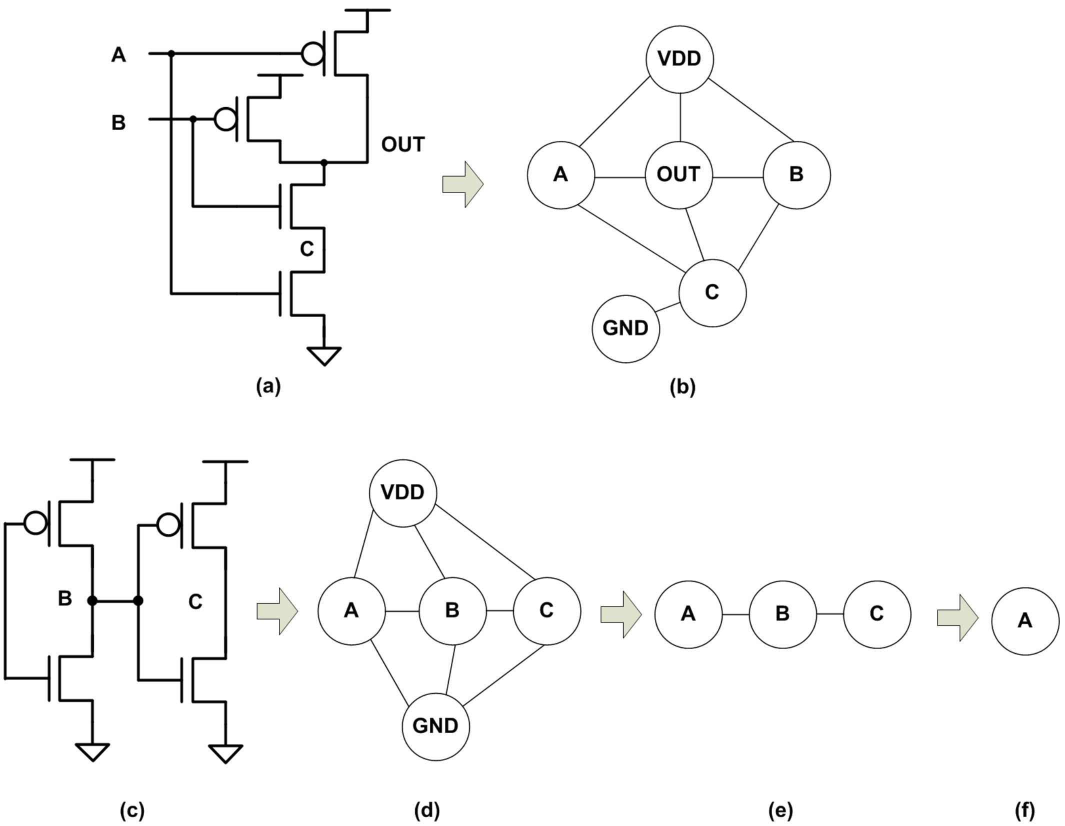


Figure 4.15. (a) CMOS NAND gate and (b) representing the NAND gate by a Markov Network. (c) Inverter chain, (d) MN representation, (e) reducing the MN by eliminating certain nodes, and (f) reducing the MN by eliminating dependent nodes.

1), $OUT$ will always be $GND$ (logic 0) with the probability of 1.0. If either $A$ or $B$ is between $VDD/2$ and $GND$ (i.e., logic 0), $OUT$ will always be $VDD$ (logic 1) with the probability of 1.0, where $VDD$ denotes the supply voltage and $GND$ denotes the ground. Thus, the NAND gate's characteristic can be expressed as the conditional probability $P(OUT|A, B)$ and with the given distributions of input $A$ and $B$, the joint distribution of the nodes, $P(A, B, OUT)$, can fully represent the characteristic of the NAND gate.

One big advantage of this Markov network representation is that the joint distribution of the MN can be factorized into a set of potential functions, which correspond to the maximal cliques in the MN [11]. This is advantageous since the required memory to store a D-dimensional distribution and process it grows exponentially with D. In contrast to a naive approach, the complexity of comparing two MNs can be significantly reduced to the complexity that is proportional to the size of the maximum clique in the MN. Moreover, the variations in process, voltage, and temperature conditions can be reflected to this Markov network model when the nodal relationships are expressed using probabilistic distributions.

### Markov network based equivalence checking

The Markov networks representing two circuits can be reduced in size, in order to save the computational efforts in comparing the two. It turns out not every node in a circuit needs to be included in Markov network. For instance, the variables with fixed values are irrelevant in determining the equivalence and can be omitted. Also, the variables whose values are completely dependent on other variables can be omitted as well.

These fixed-value nodes and dependent nodes can be found by the X analysis previously described in [34]. That is, the uncertainty of each node's distribution is measured by the entropy, after running a number of short transient simulations with randomly generated initial conditions. If the node has zero entropy, it implies that there is no uncertainty in its value and can be removed from the MN (Fig. 4.15 (e)). On the other hand, a node that is completely dependent on the other adjacent nodes can be found by measuring a conditional entropy. The conditional entropy can assess any dependent relationships between the adjacent nodes since a zero conditional entropy $H(B|A) = 0$ would indicate that the value of $B$ is completely determined by $A$. Thus, any node that has a zero conditional entropy with any of its adjacent nodes in the graph could be removed from the MN while maintaining the connectivity with the neighbors by adding new edges (Fig. 4.15 (f)).

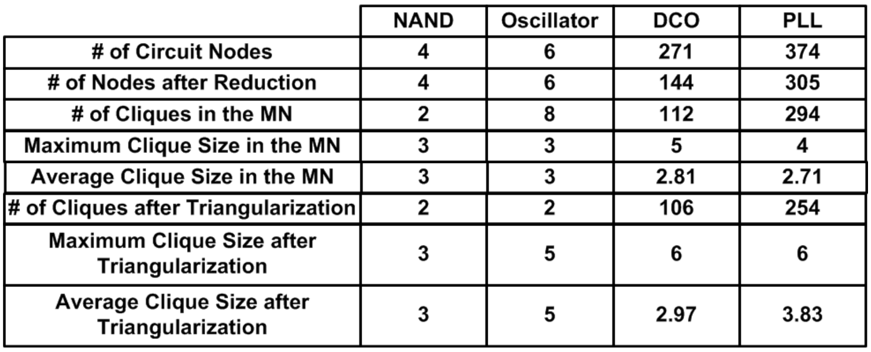
After reducing the size of the MNs, the equivalence between the two MNs can be first checked based on the graph structure of the reduced MN. If they are not isomorphic, the two circuits are reported to be different.

If the two MNs are isomorphic, the difference between the two MNs is then measured by comparing the corresponding potentials of their joint distributions. However, learning potentials of the MN that factorize the distribution and converting them into canonical potential forms can be a difficult task in general [14]. However, if the graph structure of the MN is decomposable (i.e., chordal), the potentials of the MN can be estimated as marginal distributions of cliques and their intersections. Thus, the MN structure is first transformed to a decomposable graph by triangularization and the marginal distributions of the cliques are examined. The marginal distributions are estimated by the histograms collected from a set of very short circuit transient simulations with randomized initial conditions as in the X analysis. Afterwards, the distance between two decomposed potentials (i.e., marginals) of the MNs is measured by the Jensen-Shannon divergence (JSD) [36]. If there is any pair of marginals whose JSD distance is larger than the user-specified threshold, it is concluded that the two circuits are not equivalent. The threshold should be between 0 and 1 as Jensen-Shannon divergence is limited within that range and the threshold is considered as the maximum deviation allowed. Although factorization reduces the dimensionality of the distributions to deal with, the dimensionality of the decomposed potentials doubles since the number of variables to consider for each MN node is two (i.e., the previous state and current state). Thus, in this work, only the marginal distributions of the current states are collected and compared as approximation.

#### Experimental results

The proposed method significantly reduces the dimensionality of the problem by first reducing the MN size using the X analysis and decomposing them according to the cliques in the graph after triangularization. Fig. 4.16 shows the results of the proposed dimensionality reduction after the X analysis and clique decomposition for a CMOS NAND gate (Fig. 4.15), a coupled ring oscillator a digitally controlled oscillator (DCO) and a phase locked loop (PLL) in Section 3.2.3. For DCO and PLL examples, the maximum dimensionality of distributions to compare is reduced from 271 to 6 and 374 to 6 respectively.

Next, the proposed equivalence checking is applied to the coupled ring oscillator and CMOS NAND gate (Fig. 4.15) examples, assuming that they are migrated from a 130$nm$ to a 90$nm$ technology maintaining the same transistor size ratios and varying only the device scaling factor. For both examples, 500 initial conditions are



| | NAND | Oscillator | DCO | PLL |
|---|---|---|---|---|
| # of Circuit Nodes | 4 | 6 | 271 | 374 |
| # of Nodes after Reduction | 4 | 6 | 144 | 305 |
| # of Cliques in the MN | 2 | 8 | 112 | 294 |
| Maximum Clique Size in the MN | 3 | 3 | 5 | 4 |
| Average Clique Size in the MN | 3 | 3 | 2.81 | 2.71 |
| # of Cliques after Triangularization | 2 | 2 | 106 | 254 |
| Maximum Clique Size after Triangularization | 3 | 5 | 6 | 6 |
| Average Clique Size after Triangularization | 3 | 5 | 2.97 | 3.83 |

Figure 4.16. Dimensionality reduction in a NAND gate, a coupled ring oscillator, a digitally controlled oscillator and a phase locked loop using the X analysis and clique decomposition.

generated pseudo-randomly, $11ps$ time step is used for the short transient simulations and 9-bin histogram is used to estimate the nodal distributions after the simulations. For the coupled ring oscillator, the proposed algorithm predicts that the oscillator in the $90nm$ process becomes the closest to that in $130nm$ when the scaling factor is set to $49nm$ (Fig. 4.17 (a) top). This prediction is correct because their difference in period (Fig. 4.17 (a) middle) and the integration of differences between two waveforms after transient simulations with the same initial condition (Fig. 4.17 (a) bottom) were minimum at the same scaling factor. For the CMOS NAND gate example, the proposed algorithm predicts that the NAND gate in $90nm$ process becomes most similar to that in $130nm$ when scaling factor is set to $48nm$ (Fig. 4.17 (b) top). This prediction is correct because their difference in delay from input to output (Fig. 4.17 (b) middle) and the integration of differences between two waveforms after transient simulations with the same initial condition and inputs (Fig. 4.17 bottom) were minimum at the same scaling factor.

### Summary

This section has explored a way of establishing the equivalence between two analog/mixed-signal circuits based on their Markov network (MN) representations. The proposed method first constructs an MN from the circuit netlist and reduces the MN using the entropy measures as the guide. Then, the behavioral characteristics

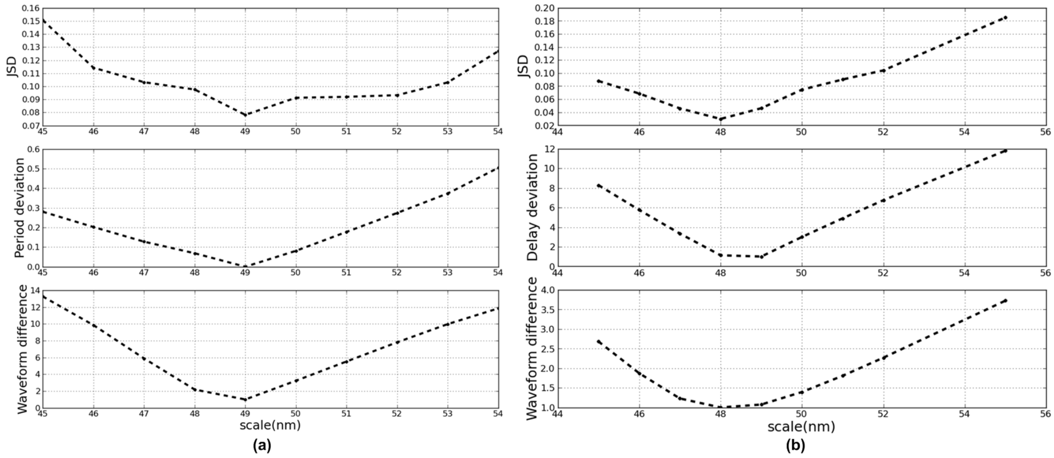


Figure 4.17. (a) Average JSD distance (top), period difference (middle; normalized), and deviation of transient waveforms with the same initial condition (bottom; normalized) of ring oscillator. (b) Average JSD distance (top), input to output delay difference (middle; normalized), and deviation of transient waveforms with the same initial condition (bottom; normalized) of CMOS NAND gate.

of the circuit are represented by the distributional relationships among the cliques in the MNs. The proposed method compares the graph structures of the two MNs and quantifies the difference between the two by measuring the Jessen-Shannon divergence of the cliques' marginals of the MNs. The experimental results show that the method can significantly reduce the size of the problem by decomposing the joint distribution into a product of marginal distributions. Moreover, it is shown that the method correctly measures the difference of two circuits for the coupled ring oscillator and NAND gate examples.

However, for the proposed method to be useful, it is necessary to derive a direct link that connects the measured JSD to actual circuit characteristics and parameters so that designers can assure that their design is okay in terms of the real circuit characteristics, specifications and parameters. Yet, this method could be improved, in future, to expand the MN model to include actual circuit parameters as another parameter nodes in its graphical model.

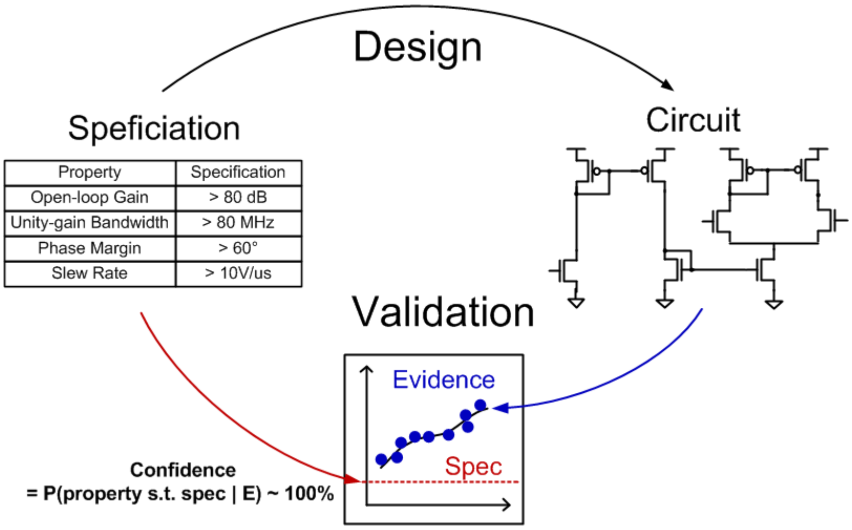
# Chapter 5
# Conclusion

This dissertation has investigated and developed many efficient validation methods for AMS systems, particularly pre-silicon global convergence property checking and post-silicon bug localization. The challenge is to effectively explore a large parameter space of a given AMS circuit, which is continuous and has an infinite number of parameters, so that we can efficiently detect failures and prevent them.

To resolve this challenge and validate AMS systems properly, this dissertation has formulated validation as an inference problem taking Bayesian perspective (*i.e.*, probabilistic validation approach as shown in Fig. 5.1). The validation is stated as the process that (1) efficiently collects sufficient evidence and (2) assures the circuit to satisfy the specification with a high degree of confidence, which can be computed by inference.

Having the probabilistic formulation of AMS system validation, this work has attempted to solve two specific validation problems in both pre-silicon and post-silicon validation problems. First, the pre-silicon validation methodologies for global convergence property checking have been investigated, mainly Monte Carlo methods. The verification has been accelerated by fast sample batch verification via cluster analysis and fast failure event generation using Gaussian process regression. Second, probabilistic graphical models and statistical inference have been used for the post-silicon bug localization.

It should be noted that the proposed Bayesian viewpoint in validation and resulting confidence metric can provide a *coverage* in AMS system validation. The higher the confidence after the validation is, the more bug-free and the safer it is to use the design. Moreover, confidence can be used to compare many existing validation methods because we can quantify the effectiveness of the validation by confidence (Fig. 5.2). That is, the faster the confidence for the invested validation efforts increases, the better the validation method is. Since balancing trade-off



| Property | Specification |
|---|---|
| Open-loop Gain | > 80 dB |
| Unity-gain Bandwidth | > 80 MHz |
| Phase Margin | > 60° |
| Slew Rate | > 10V/us |

Figure 5.1. Validation as inference.

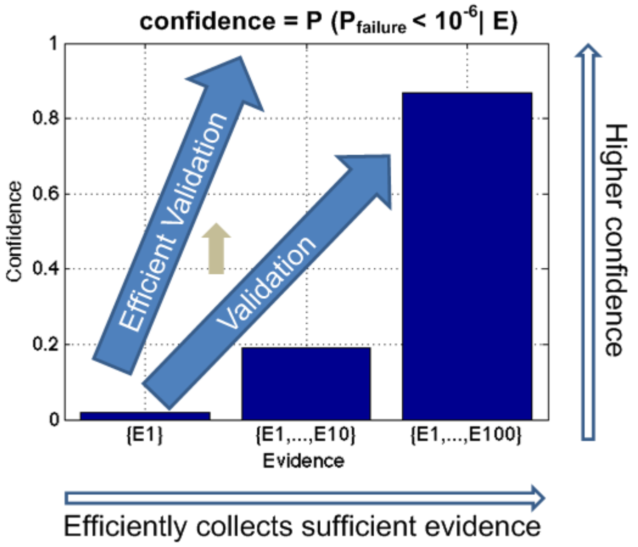


Figure 5.2. Collecting evidence efficiently using confidence as guidance.

among validation efforts, confidence and tolerance is possible, we could eventually evaluate different methods and efficiently plan validation.

In conclusion, this dissertation has proposed two kinds of validation methods for AMS systems that check global convergence property in pre-silicon verification and automatically localize the root-causes of the observed failure in post-silicon validation, taking probabilistic approaches. Validation is formulated as the procedure of efficient evidence collection and inference, and confidence is proposed as an effective *coverage* metric in AMS system validation. The probabilistic approach may demand significant validation efforts especially when it is necessary to confirm that a system is free of rare-event failures (such as *ppm*-order failures). Nevertheless, the proposed probabilistic formulation still allows us to explicitly quantify the required validation efforts (*e.g.*, the number of validation sample trials) and plan a balanced validation. To further conquer and draw even more precise conclusions (*i.e.*, the validation that can confirm a system to be free of *ppm*-order failure), a different technique would be necessary, such as the importance sampling used in Section 3.3.2. This dissertation has also proposed to use the probabilistic graphical model as one effective abstraction for AMS systems. The graphical model has been shown to be effective for the post-silicon bug localization and it would possibly be extended to many other validation applications such as the pre-silicon equivalence checking.

# Bibliography

[1] W. Lam, *Hardware Design Verification: Simulation and Formal Method-Based Approaches*, ser. Prentice Hall Modern Semiconductor Design Series. Prentice Hall Professional Technical Reference, 2005.

[2] J. Bergeron, *Writing Testbenches: Functional Verification of HDL Models*. Springer US, 2003.

[3] D. Perry and H. Foster, *Applied Formal Verification : For Digital Circuit Design*, ser. McGraw-Hill electronic engineering series. McGraw-Hill Professional Publishing, 2005.

[4] H. Chang and K. Kundert, "Verification of complex analog and RF IC designs," *Proceedings of the IEEE*, vol. 95, no. 3, pp. 622–639, 2007.

[5] J. Keshava, N. Hakim, and C. Prudvi, "Post-silicon validation challenges: how EDA and academia can help," in *Proceedings of the 47th Design Automation Conference*. ACM, 2010, pp. 3–7.

[6] S. Mitra, S. A. Seshia, and N. Nicolici, "Post-silicon validation opportunities, challenges and recent advances," in *Design Automation Conference (DAC), 2010 47th ACM/IEEE*. IEEE, 2010, pp. 12–17.

[7] C. Gu, "Challenges in post-silicon validation of high-speed I/O links," in *Proceedings of the International Conference on Computer-Aided Design*. ACM, 2012, pp. 547–550.

[8] A. Evans, A. Silburt, G. Vrckovnik, T. Brown, M. Dufresne, G. Hall, T. Ho, and Y. Liu, "Functional verification of large ASICs," in *Design Automation Conference, 1998. Proceedings*, June 1998, pp. 650–655.

[9] Y. Chen, S. Safarpour, J. Marques-Silva, and A. Veneris, "Automated design debugging with maximum satisfiability," *Computer-Aided Design of Integrated*

*Circuits and Systems, IEEE Transactions on*, vol. 29, no. 11, pp. 1804–1817, 2010.

[10] O. Maler, "Algorithmic verification of continuous and hybrid systems."

[11] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning.* springer New York, 2006, vol. 1.

[12] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*, ser. Springer Series in Statistics. Springer, 2009.

[13] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques.* The MIT Press, 2009.

[14] D. Barber, *Bayesian reasoning and machine learning.* Cambridge University Press, 2012.

[15] S. Youn and J. Kim, "Establishing global convergence in mixed-signal systems," *Proceedings of the SRC Techcon Conference*, 2013.

[16] S. Youn and J. Kim, "Markov network based equivalence checking in mixed-signal systems," *Frontiers in Analog CAD*, 2013.

[17] S. Youn, C. Gu, and J. Kim, "Probabilistic bug localization via statistical inference based on partially observed data," in *Proceedings of the 51th Design Automation Conference.* ACM, 2014.

[18] A. Freno and E. Trentin, *Hybrid Random Fields: A Scalable Approach to Structure and Parameter Learning in Probabilistic Graphical Models*, ser. Intelligent Systems Reference Library. Springer, 2011.

[19] W.-K. Chen, *Feedback, nonlinear, and distributed circuits.* CRC Press, 2010.

[20] K. Yamamura, N. Tamura, and K. Suda, "An efficient algorithm for finding all DC solutions of nonlinear circuits using lp narrowing," in *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on.* IEEE, 2009, pp. 2081–2084.

[21] M. R. Greenstreet and S. Yang, "Verifying start-up conditions for a ring oscillator," in *Proceedings of the 18th ACM Great Lakes symposium on VLSI.* ACM, 2008, pp. 201–206.

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

[23] G. Russo, M. di Bernardo, and J.-J. Slotine, "A graphical approach to prove contraction of nonlinear circuits and systems," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 58, no. 2, pp. 336–348, 2011.

[24] A. Stacey and R. Stonier, "Analytic estimates for the boundary of the region of asymptotic attraction," *Dynamics and Control*, vol. 8, no. 2, pp. 177–189, 1998.

[25] Z.-Z. Chen and T.-C. Lee, "The design and analysis of dual-delay-path ring oscillators," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 58, no. 3, pp. 470–478, 2011.

[26] S. Youn, J. Kim, and M. Horowitz, "Global convergence analysis of mixed-signal systems," in *Proceedings of the 48th Design Automation Conference.* ACM, 2011, pp. 498–503.

[27] K. Kundert, J. White, and A. Sangiovanni-Vincentelli, *Steady-State Methods for Simulating Analog and Microwave Circuits*, ser. The Kluwer international series in engineering and computer science : VLSI, computer architecture and digital signal processing. Springer, 1990.

[28] M. P. Kennedy, "Three steps to chaos. I. evolution," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 40, no. 10, pp. 640–656, 1993.

[29] F. Najm, *Circuit Simulation.* Wiley, 2010.

[30] T. Parker and L. Chua, *Practical Numerical Algorithms for Chaotic Systems.* Springer London, Limited, 2011.

[31] W. Lohmiller and J.-J. E. Slotine, “On contraction analysis for non-linear systems,” *Automatica*, vol. 34, no. 6, pp. 683–696, 1998.

[32] J. Kim, M. Jeeradit, B. Lim, and M. A. Horowitz, “Leveraging designer’s intent: A path toward simpler analog CAD tools,” in *Custom Integrated Circuits Conference, 2009. CICC’09. IEEE.* IEEE, 2009, pp. 613–620.

[33] S. Youn, J. Kim, and M. A. Horowitz, “Preventing global convergence failures in mixed-signal systems by eliminating indeterminate states,” *Frontiers in Analog Circuit Synthesis and Verification*, 2011.

[34] S. Youn and J. Kim, “Preventing global convergence failure in mixed-signal systems via indeterminate state (X) elimination,” *IEEE Transactions on Circuits and SystemsI: Regular Papers*, vol. 60, no. 10, p. 2561, 2013.

[35] J. Kim, “Adaptive-bandwidth phase-locked loop with continuous background frequency calibration,” *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 56, no. 3, pp. 205–209, 2009.

[36] T. Cover and J. Thomas, *Elements of Information Theory.* Wiley, 2012.

[37] H. Banba, H. Shiga, A. Umezawa, T. Miyaba, T. Tanzawa, S. Atsumi, and K. Sakui, “A cmos bandgap reference circuit with sub-1-v operation,” *Solid-State Circuits, IEEE Journal of*, vol. 34, no. 5, pp. 670–674, 1999.

[38] C. E. Rasmussen, “Gaussian processes for machine learning,” 2006.

[39] J. P. Jarvis and D. R. Shier, “Graph-theoretic analysis of finite markov chains.” [Online]. Available: http://www.ces.clemson.edu/~shierd/Shier/markov.pdf

[40] J. R. Norris, *Markov chains.* Cambridge university press, 1998, no. 2008.

[41] S. Dasgupta and Y. Freund, “Random projection trees for vector quantization.” *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3229–3242, 2009.

[42] G. Rubino and B. Tuffin, *Rare Event Simulation using Monte Carlo Methods.* Wiley, 2009.

[43] C. Chi-Tsong, “Linear system theory and design,” 1999.

[44] C. Dong and X. Li, “Efficient SRAM failure rate prediction via gibbs sampling,” in *Proceedings of the 48th Design Automation Conference.* ACM, 2011, pp. 200–205.

[45] J.-C. Chen, D. Lu, J. S. Sadowsky, and K. Yao, “On importance sampling in digital communications. I. fundamentals,” *Selected Areas in Communications, IEEE Journal on*, vol. 11, no. 3, pp. 289–299, 1993.

[46] R. Kanj, R. Joshi, and S. Nassif, “Mixture importance sampling and its application to the analysis of SRAM designs in the presence of rare failure events,” in *Proceedings of the 43rd annual Design Automation Conference.* ACM, 2006, pp. 69–72.

[47] A. Singhee and R. A. Rutenbar, “Statistical blockade: a novel method for very fast monte carlo simulation of rare circuit events, and its application,” in *Design, Automation, and Test in Europe.* Springer, 2008, pp. 235–251.

[48] A. Singhee, J. Wang, B. H. Calhoun, and R. A. Rutenbar, “Recursive statistical blockade: an enhanced technique for rare event simulation with application to SRAM circuit design,” in *VLSI Design, 2008. VLSID 2008. 21st International Conference on.* IEEE, 2008, pp. 131–136.

[49] L. Dolecek, M. Qazi, D. Shah, and A. Chandrakasan, “Breaking the simulation barrier: SRAM evaluation through norm minimization,” in *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design.* IEEE Press, 2008, pp. 322–329.

[50] J. Wang, S. Yaldiz, X. Li, and L. T. Pileggi, “SRAM parametric failure analysis,” in *Proceedings of the 46th Annual Design Automation Conference.* ACM, 2009, pp. 496–501.

[51] P. J. Smith, M. Shafi, and H. Gao, "Quick simulation: A review of importance sampling techniques in communications systems," *Selected Areas in Communications, IEEE Journal on*, vol. 15, no. 4, pp. 597–613, 1997.

[52] T. Kim, D.-G. Song, S. Youn, J. Park, H. Park, and J. Kim, "Verifying start-up failures in coupled ring oscillators in presence of variability using predictive global optimization," in *Proceedings of the International Conference on Computer-Aided Design.* IEEE Press, 2013, pp. 486–493.

[53] T. McConaghy and J. Hogan, *Variation-Aware Design of Custom Integrated Circuits: A Hands-on Field Guide.* Springer, 2013.

[54] C. E. Rasmussen and Z. Ghahramani, "Bayesian Monte Carlo," *Advances in neural information processing systems*, pp. 505–512, 2003.

[55] A. O'Hagan, "Monte carlo is fundamentally unsound," *The Statistician*, pp. 247–249, 1987.

[56] E. B. Eichelberger, "Hazard detection in combinational and sequential switching circuits," *IBM Journal of Research and Development*, vol. 9, no. 2, pp. 90–99, 1965.

[57] E. B. Eichelberger and T. W. Williams, "A logic design structure for lsi testability," in *Proceedings of the 14th design automation conference.* IEEE Press, 1977, pp. 462–468.

[58] P. Wilcox, "Digital logic simulation at the gate and functional level," in *Proceedings of the 16th Design Automation Conference.* IEEE Press, 1979, pp. 242–248.

[59] J. Slotine and W. Li, *Applied Nonlinear Control*, ser. Prentice-Hall International Editions. Prentice-Hall, Incorporated, Englewood Cliffs, N.J., 1991.

[60] J. Beirlant, E. J. Dudewicz, L. Györfi, and E. C. Van der Meulen, "Nonparametric entropy estimation: An overview," *International Journal of Mathematical and Statistical Sciences*, vol. 6, pp. 17–40, 1997.

[61] K. Singh and M. Xie, "Bootstrap: a statistical method," *Unpublished Working Paper. Rutgers University*, 2008. [Online]. Available: http://www.stat.rutgers.edu/home/mxie/RCPapers/bootstrap.pdf

[62] A. R. Newton and A. L. Sangiovanni-Vincentelli, "Relaxation-based electrical simulation," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 3, no. 4, pp. 308–331, 1984.

[63] W. John, W. Rissiek, and K. Paap, "Circuit partitioning for waveform relaxation," in *Design Automation. EDAC., Proceedings of the European Conference on.* IEEE, 1991, pp. 149–153.

[64] T. Lee, *The Design of CMOS Radio-Frequency Integrated Circuits.* Cambridge University Press, 2004.

[65] L. S. Milor, "A tutorial introduction to research on analog and mixed-signal circuit testing," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 45, no. 10, pp. 1389–1407, 1998.

[66] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model checking.* MIT press, 1999.

[67] L. Zhang, I. Ghosh, and M. Hsiao, "Efficient sequential ATPG for functional RTL circuits," in *ITC*, vol. 3. Citeseer, 2003, pp. 290–298.

[68] W. Li, A. Forin, and S. A. Seshia, "Scalable specification mining for verification and diagnosis," in *Design Automation Conference (DAC), 2010 47th ACM/IEEE.* IEEE, 2010, pp. 755–760.

[69] A. L. Sangiovanni-Vincentelli, "Circuit simulation," in *Computer Design Aids for VLSI Circuits.* Springer, 1981, pp. 19–112.

[70] G. G. Gielen and R. A. Rutenbar, "Computer-aided design of analog and mixed-signal integrated circuits," *Proceedings of the IEEE*, vol. 88, no. 12, pp. 1825–1854, 2000.

[71] G. E. Box and N. R. Draper, *Empirical Model Building and Response Surfaces.* John Wiley & Sons, 1987.

[72] A. Demir, A. Mehrotra, and J. Roychowdhury, “Phase noise in oscillators: a unifying theory and numerical methods for characterization,” *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 47, no. 5, pp. 655–674, 2000.

[73] J. Kim, B. S. Leibowitz, and M. Jeeradit, “Impulse sensitivity function analysis of periodic circuits,” in *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on.* IEEE, 2008, pp. 386–391.

[74] B. Gustavsen and A. Semlyen, “Rational approximation of frequency domain responses by vector fitting,” *Power Delivery, IEEE Transactions on*, vol. 14, no. 3, pp. 1052–1061, 1999.

[75] L. Ljung, *System identification.* Wiley Online Library, 1999.

[76] L. Scheffer, L. Lavagno, and G. E. Martin, *EDA for IC implementation, circuit design, and process technology.* CRC Press, 2006.

[77] M. H. Zaki, S. Tahar, and G. Bois, “Formal verification of analog and mixed signal designs: A survey,” *Microelectronics Journal*, vol. 39, no. 12, pp. 1395–1404, 2008.

[78] H. S. Baird and Y. E. Cho, “An artwork design verification system,” in *Proceedings of the 12th Design Automation Conference.* IEEE Press, 1975, pp. 414–420.

# 초 록

반도체 시스템의 복잡도와 공정 후 불확실성의 증가는 설계 및 검증에 큰 어려움을 초래하고 있다. 따라서 본 논문에서는 반도체 설계 단계 그리고 칩 제작 후에 아날로그 혼성신호 회로를 효과적으로 검증할 수 있는 방법론에 대한 연구가 이루어졌다. 본 연구는 기본적으로 회로 검증 문제를 베이지안 추론 (Bayesian inference) 문제로 기술하고 확률적인 방법으로 다룬다. 예를 들면, 회로의 실패 확률을 표본 추출 후 베이지안 방법으로 계산하고 이를 통해 검증성공에 대한 신뢰도를 구할 수 있다. 이러한 방식으로 본 논문은 먼저 설계 단계에서의 회로 검증을 연구하였다. 즉, 시스템의 global convergence 특성을 확인하기 위한 두 가지 효율적인 몬테카를로 (Monte Carlo) 검증 방법론, 즉 클러스터링 기법 (Cluster analysis) 을 사용하여 효과적으로 다수의 샘플을 검증하는 방법과 가우시안 프로세스 (Gaussian process) 를 사용하여 효율적으로 샘플링 (Sampling) 을 하는 방법이 제안되었다. 이에 더하여 회로의 global convergence failure를 방지하기 위하여 불확정 상태 $X$ 가 아날로그 혼성신호 회로에 대하여 확장되어 사용되었다. 반도체 칩 제작 후 검증에 대한 연구도 확률 그래프 모델 (Probabilistic graphical model) 을 아날로그 혼성신호 시스템에 대한 모델로 제안하여 이루어졌다. 제안된 확률 그래프 모델과 추론을 통하여 시스템 안의 각 변수가 표준 규격 (Specification) 을 만족시킬 확률을 구하고 이를 통하여 가능한 버그 (Bug) 를 한정하고 비교할 수 있다. 제안된 모델과 방법은 측정과 제어가 제한된 상태에서도 사용될 수 있기 때문에 칩 제작 후 검증에 특히 유효하다.

키워드: 반도체 설계 검증, 반도체 칩 제작 후 검증, 확률적 검증, 확률 그래프 모델, 아날로그 혼성신호 시스템
학번: 2010-30991