Efficient Statistical Model Checking of Hardware Circuits With Multiple Failure Regions

Jayanand Asok Kumar, *Member, IEEE,* Seyed Nematollah Ahmadyan, *Student Member, IEEE,* and Shobha Vasudevan, *Member, IEEE*

***Abstract*—Statistical model checking (SMC) is a simulation-based approach for verifying the statistical properties of large, complex systems. If a large number of low-probability events (rare events) is required to be simulated, SMC is extremely time-consuming. In this paper, we present a methodology to accelerate the SMC of hardware circuits by generating rare events with a higher frequency. Unlike existing techniques, our methodology can be applied to circuits with multiple rare-event regions. We first sample the circuit uniformly to quickly generate a set of rare events. We employ variational Bayes, a variational inference technique used in machine learning, to infer the distribution of the rare events in the circuit. We then bias the statistical distribution toward these rare event regions. Finally, we employ the SMC using the biased distribution and adjust for the bias that we introduce. The use of variational Bayes enables our methodology to distinguish between multiple rare event regions in the circuit. We demonstrate the effectiveness of our biasing approach on two real-world hardware circuits. We consider the analog (i.e., continuous-time) behavior of these circuits. For an SRAM memory cell, which has a single failure region, we show that our approach provides around 31x speedup over regular SMC while verifying whether the failure rate is less than 10^{-4}. For a successive approximation ADC circuit, which has multiple failure regions, we demonstrate a speedup of 42x while verifying whether the failure rate is less than 10^{-5}.**

***Index Terms*—Importance sampling, multiple failure regions, statistical model checking (SMC), variational Bayesian inference.**

I. Introduction

STATISTICAL systems, such as cyber-physical systems (CPS), computer networks, and hardware circuits [36], require probabilistic verification for the properties pertaining to the metrics, such as reliability, safety, stability, and performance. Probabilistic model checking [20] employs numerical analyses to provide statistical guarantees for such systems. Although such analyses may become computationally intensive for very large systems. Statistical model checking (SMC) [21], [27], [39] is an inexact alternative strategy that provides guarantees by simulating several samples of the system while avoiding the exponential growth of the states. SMC is highly scalable and can be used to rigorously analyze massive, complex systems that cannot be verified exactly, using probabilistic model checking.

SMC samples the entire state space to find the statistical evidence for verifying the system. As a result, it is known to be inefficient and time consuming [24] in rare-event scenarios, i.e., scenarios that pertain to events which occur with very low probability ($<10^{-4}$). In such scenarios, a very large number of samples need to be generated to gather the statistical evidence required by the SMC engine. For example, while verifying whether the failure rate $<10^{-7}$, an average of 10^7 samples, needs to be generated before even a single failure is witnessed. In such cases, it is beneficial to reduce the search space by employing machine learning algorithms. In this paper, we present a technique to accelerate the SMC for hardware circuits in rare-event scenarios by identifying the rare-event regions and reducing the search space. We model check the statistical properties pertaining to the analog (i.e., continuous time) behavior of the hardware circuits.

SMC for rare-event scenarios can be accelerated by increasing the frequency with which failing samples (i.e., the rare events in this paper) are generated. This can be achieved by statistically biasing the distribution of the system toward the regions in which its failing samples are present. In probability estimation, this technique is referred to as importance sampling [11], [24], [30]. Recently, researchers [2], [7], [13] investigated increasing the efficiency of SMC using importance sampling by using analytical approaches. However, in such analytical approaches, the biased distribution is typically in a complex form that is hard to evaluate and generate samples from [24] and [30]. Therefore, although such approaches are theoretically sound for systems, in general, they are not practically viable.

We introduce a machine learning-based approach for accelerating the SMC of the hardware circuits in the rare-event scenarios. We employ variational Bayes [4], an iterative variational inference technique used in machine learning, to infer the failure regions of the hardware circuit. We train the algorithm using a set of failing samples that we generate as data. The variational Bayes technique then uses these failing samples to identify the failure intensive regions of the circuit. We accelerate SMC by statistically biasing the circuit to frequently sample the identified failing regions. We show that

Manuscript received August 18, 2013; revised November 10, 2013; accepted December 22, 2013. Date of current version May 15, 2014. This work was supported in part by the National Science Foundation under Grant CCF-0953767. This paper was recommended by Associate Editor W. Kunz.

The authors are with the Coordinated Science Laboratory, Electrical and Computer Engineering Department, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mails: jasokku2@illinois.edu; ahmadya2@illinois.edu; shobhav@illinois.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2014.2299957

our biasing approach is generic and practical for a large class of circuits.

In this paper, we intend our biasing approach to be used for verifying reliability properties of hardware circuits in the presence of variations. We consider reliability in the context of the analog (i.e., continuous behavior in space and time) circuit. We wish to verify that a hardware circuit M satisfies a reliability property Φ, where Φ deals with a rare-event scenario. In this paper, we consider failures as the rare events. We quickly generate several examples of the failures by uniformly sampling the space of the random variables in M. We then employ the variational Bayes to group these failing samples into distinct clusters which we interpret to be the failure regions of M. The variational Bayes technique, upon convergence after several iterations, also computes a mixture statistical distribution [4], where each component of the mixture is a Gaussian distribution centered on one of the inferred failure regions. We use this mixture distribution and modify the distribution of M to be biased toward the failure regions. We employ SMC by drawing samples according to the biased distribution. In order to preserve the accuracy of the model checking results, we adjust for the bias that we introduce in the distribution. Biasing, if applied correctly, reduces the variance estimate which in turn reduces the number of samples required for SMC. Therefore, in rare-event scenarios, our biasing approach provides significant performance benefits over regular SMC.

In this paper, we present an algorithmic approach for importance sampling in the context of SMC. The value of our methodology comes from the use of machine learning (i.e., variational Bayes) to learn about the rare failure intensive regions of the system. Without machine learning, information regarding the locations and distributions of these failure regions would not be readily available. Therefore, machine learning is a key in our strategy to statistically bias the hardware circuit toward the failure regions.

The use of variational Bayes enables our algorithm to distinguish between multiple failure regions in a hardware circuit. Furthermore, the variational Bayes does not require the user to specify the exact number of the failure regions in a circuit *a priori*. If the user overestimates the number of the failure regions, the contributions of the extra failure regions to the final mixture distribution are driven to zero upon convergence of the variational Bayes iterations. Therefore, the variational Bayes makes our biasing approach equally proficient for the circuits with a single failure region (eg., an SRAM cell), as well as for circuits with multiple failure regions (eg., a mixed signal SAR-ADC). In [16] and [19], the authors employ an importance sampling strategy for SRAM cells. However, their strategy has no mechanism that allows it to be extended to the circuits with multiple failure regions.

In [19], we presented a simple importance sampling strategy to accelerate the SMC for SRAM cells. In this paper, we significantly enhance this approach by making it applicable to the hardware circuits with multiple failure regions. We achieve this enhancement by incorporating the variational Bayes, a sophisticated machine learning technique, into the importance sampling strategy.

Our main contributions are as follows. We introduce a practically viable approach for accelerating the SMC in rare-event scenarios. We employ variational Bayes to infer the rare-event regions of the system. We present a novel use of variational Bayes as a practical strategy to bias a hardware circuit toward its rare-event regions.

We demonstrate the correctness and speedup of our biasing approach by applying it to verify the reliability property for both an SRAM cell [16] and for a successive approximation analog-to-digital converter (SAR-ADC) [1]. For the SRAM cell, we show that our approach provides up to 31x speedup over regular SMC. We observe that this is comparable to the speedup provided by the statistical blockade technique [29] under a similar setup. We also employ our approach on an SAR-ADC; a circuit has multiple failure regions. For the SAR-ADC, we demonstrate an average speedup of 52x across different failure rates. Statistical blockade has not been demonstrated to be applicable for circuits with multiple failure regions.

The rest of the paper is organized as follows. In Section II, we present a brief overview of related work. In Section III, we present preliminaries regarding the terminology and algorithms that we use in this paper. In Section IV, we present the broad intuition behind our biasing approach to accelerate the SMC. In Section V, we describe all the steps of our biasing approach in detail. In Section VI, we provide an analysis regarding the correctness, speedup, and complexity of our approach. In Section VII, we list the results of applying our biasing approach on two hardware circuits and then conclude our work in Section VIII.

II. Related Work

In this section, we briefly describe the related work in accelerated SMC.

SMC is used to verify whether the probability of occurrence of a relevant event (for example, a failure event) in a system exceeds a specified threshold or not. Although there are numerous varieties of SMC, the underlying algorithm in each of them can be broadly broken down into two steps: 1) sample generation and 2) statistical testing. In the sample generation step, the algorithm generates samples of the system in accordance with the statistical distribution of the system. In the statistical testing step, the algorithm determines whether each of the generated samples is a relevant event or not. The algorithm then aggregates the statistical information from these relevant events and uses it as an evidence to arrive at the verification result.

Sample generation is typically achieved by performing Monte Carlo simulations of the system. However, the method with which the statistical testing step is performed differs across the SMC techniques. Statistical testing methods can be broadly classified as 1) estimation testing methods and 2) hypothesis testing methods. When estimation testing is used [11], [40], the actual probability of occurrence of the relevant events is estimated. When hypothesis testing is used, the actual probability is not estimated. Instead, the statistical metrics other than the probability estimate are computed to arrive

at the verification result. The SMC techniques described in [5], [10], [21], [26], [37], and [39], use Wald's hypothesis testing [33]. In [6], [14], and [36], another flavor of hypothesis testing, called Bayesian hypothesis testing, is used. Hrault *et al.* [12] used another variant of hypothesis testing for the SMC of the software programs. A performance comparison of a selected set of the SMC techniques can be found in [18] and [38]. In this paper, we consider an SMC technique that uses estimation testing (Section III-C).

If the probability of occurrence of the relevant events is very small, a very large number of samples need to be generated before the statistical testing acquires sufficient evidence to determine the verification result. Consequently, all the SMC techniques are known to perform poorly in such rare-event scenarios. In such scenarios, importance sampling techniques [11] can be used to control sample generation and increase the rate at which the relevant events occur. As a result, the statistical testing step obtains the required statistical evidence at a much faster rate and therefore, quickly arrives at the verification result. All the SMC techniques, regardless of the statistical testing method that it employs, can benefit from importance sampling.

Importance sampling techniques [11] control sample generation by biasing the statistical distribution of the system toward the increased occurrence of relevant events. The main challenge in these techniques lies in ensuring the correctness of the verification result while improving the speed of the SMC. In [7], [13], and [25], cross-entropy minimization techniques are used to estimate the optimal biased statistical distribution for the system. However, these biased distributions are often expressed in complex forms which make them intractable for sample generation [30], [24]. Therefore, although such approaches are theoretically sound for large classes of systems, they are not practically viable. The importance sampling strategy that we present is based on the variational Bayes, a practical machine learning technique. Our strategy is applicable to a large class of the hardware circuits.

Several importance sampling strategies have been employed to accelerate the statistical estimation of failure rates in the SRAM cells [3], [8], [9], [15], [16], [17], [19], [23], [35]. These strategies bias the distribution of the SRAM cells by shifting it toward the failure region. Shahid [28] tailored a cross-entropy minimization technique for estimating the failure rate of the SRAM cells. In [29], [31], and [34], a technique called statistical blockade is employed to selectively generate samples only from the failure region of the SRAM cell. Statistical blockade achieves this by training a data mining classifier to identify the boundary between the failure region and the nonfailure region of the SRAM cell. Sun *et al.* [32] employed a Gibbs sampling-based approach to predict the failure rate of an SRAM cell. While all these techniques are extremely effective for SRAM cells, they implicitly rely on the fact that the SRAM cell has a single, continuous failure region. None of these techniques have been demonstrated to be applicable for the hardware circuits with multiple failure regions. We demonstrate that our importance sampling strategy is equally proficient for SRAM cells, as well as the circuits with multiple failure regions.

III. Preliminaries: Verifying Reliability of Hardware Circuit

We now present some preliminaries for modeling the variations in a hardware circuit. We then describe the reliability property that we wish to verify on the hardware circuit in the presence of such variations. In this section, we also provide a brief background for SMC and for the variational Bayes technique.

A. Modeling Variations in a Hardware Circuit

The variations arising from the manufacturing process may introduce randomness in the physical/electrical characteristics of hardware. In this paper, we consider two type of variations in hardware circuits: 1) static variations and 2) dynamic variations. The static variations are randomly sampled once at the beginning of the execution and from there on, remain constant. The static variations could result from modeling errors, or uncertainties and disturbances which exists in any realistic hardware circuit. For example, in most hardware circuits, the delay of the circuit under process variation is not allowed to exceed a predefined timing constraint. On the other hand, the dynamic variations change randomly during the execution of the circuit according to a random variable. For any dynamic variation, we assume there are only finite number of changes within a given bounded time interval. The dynamic variations could result from noises or unknown inputs. In the presence of such process variations, device parameters (for e.g., threshold voltages of transistors) in the hardware are typically modeled as Gaussian random variables [29]. In addition, the voltages/currents in the hardware are susceptible to random perturbations and are therefore, another source of randomness in the circuit.

A hardware circuit M can be viewed as a statistical entity with N random variables that model the process variations or the variations in voltage/current. Let V=$\{v_1, v_2, \ldots, v_N\}$ be the set of N random variables in M. Each variable $v_j \in V$ is real-valued and can be assigned a value in the range $[v_j^{min}\ v_j^{max}]$. In this paper, we consider V to be independent, truncated Gaussian variables [29]. There are other alternative distributions that would model the variations more accurately and are more appropriate, such as Beta distribution. However, modeling the variations as Gaussian random variables is a common practice widely used in industry [29]. Our algorithm is not restricted to just Gaussian random variables and users are free to choose any distribution they want to model the variations.

Definition 1: The sample space $\mathcal{S}$ of the hardware circuit M is the N-dimensional Euclidean space $[v_1^{min}\ v_1^{max}] \times [v_2^{min}\ v_2^{max}] \times \ldots \times [v_N^{min}\ v_N^{max}]$ spanned by the set of threshold voltages V.

Each point in the sample space $\mathcal{S}$ corresponds to a unique assignment of concrete, real values to variables V. We use the N-tuple $\{v_1, v_2, \ldots, v_N\}$ to denote a point in the sample space $\mathcal{S}$.

Let $g_j(v)$ denote the Gaussian probability density function (pdf) for variable v_j. In the case of process variations, the mean and variance of the distribution $g_j(v)$ can be obtained

from the specification of the devices in the process technology library [29].

Definition 2: The statistical distribution $D(V)$ of the hardware circuit M is given by the joint pdf of the threshold voltages V. Since the threshold voltages V are modeled to be statistically independent variables, the joint pdf of V can be computed as a product of the individual pdfs $g_j(v)$ of the variables v_j (j= 1 to N).

Let V^i=$\{v_1^i, v_2^i, \ldots, v_N^i\}$ be a sample that is drawn from the space $\mathcal{S}$ (Definition 1) of the hardware circuit. During Monte Carlo simulation of the hardware circuit according to the distribution $D(V)$, the probability density $D(V^i)$ of drawing the sample V^i is given by

$$D(V^i) = \prod_{j=1}^{N} g_j(v_j^i). \tag{1}$$

B. Reliability of Hardware Circuit

A hardware circuit is said to fail if its analog, (i.e., continuous-time) behavior does not satisfy a specified requirement under parametric variations. We wish to verify that the probability with which a hardware circuit fails is less than a threshold θ. We express this reliability requirement as a property

$$\Phi = P_{\leq\theta}[fail]. \tag{2}$$

In (2), fail implies the failure of the property. We use bounded LTL (BLTL) [36] to describe the failure properties. BLTL is a variant of well known linear temporal logic where the temporal properties are all equipped with a time bound, such as bounded until operator. Accordingly, Wang *et al.* [36] defined the standard bounded **F** and **G** operators and the semantics for BLTL over finite execution paths. Extending the BLTL to include unbounded properties is beyond the scope of this paper. If θ is very small, we consider that Φ deals with a rare-event scenario. For a reliable hardware circuit M, the failure rate $P[fail]$ is required to be very low. Therefore, the reliability property of M deals with a rare-event scenario.

Definition 3: A failing sample of a hardware circuit M is a sample where the analog behavior of M does not satisfy the specified requirement (i.e., the hardware circuit fails).

For each sample V^i drawn from the space $\mathcal{S}$, the delay of the hardware circuit can be measured by simulating the analog behavior of the circuit using the corresponding values v_j^i assigned to the variables $v_j \in V$ (j= 1 to N). The observed analog behavior can be compared against the specified requirement to check whether the sample is failing or not.

In a continuous sample space $\mathcal{S}$ (Definition 1), two samples are in the same neighborhood if the Euclidean distance between them is small. For circuits exhibiting continuous behavior across the state-space, the behavior of the circuit at one state is a good approximation for the behavior of the circuit in the region neighboring that state. Samples that are in the neighborhood of each other will exhibit similar behavior. Therefore, if a certain sample fails, it is reasonable to expect that the other samples in its neighborhood are also likely to fail. This is a reasonable assumption for the practical analog circuits. The above assumption does not hold true for the digital circuits (which exhibit discrete behavior and has a discrete state-space) or chaotic circuits (which are not typically practical in the real-world hardware implementations), which are beyond the scope of this paper.

Definition 4: The set of all failing samples in a neighborhood in $\mathcal{S}$ constitutes a failure region $\mathcal{S}_F^l$ ($\mathcal{S}_F^l \subseteq \mathcal{S}$) of the hardware circuit. In general, a hardware circuit can have K (dependent or independent) failure regions $\mathcal{S}_F^l$ (l = 1 to K).

If each of the failure regions of the hardware circuit can be bounded in the sample space $\mathcal{S}$, it is possible to analytically evaluate (as shown in Section VI-C) the performance improvement provided by the approach that we present.

Definition 5: $\mathcal{R}_F^l$ ($\mathcal{R}_F^l \subseteq \mathcal{S}$) is the smallest N-dimensional hyperrectangle in which the failure region $\mathcal{S}_F^l$ of the hardware circuit is completely contained. $\mathcal{R}_F^l$ can be viewed as a box that bounds the failure region $\mathcal{S}_F^l$.

C. Statistical Model Checking

We now briefly describe the SMC in the context of verifying the reliability of a hardware circuit. We wish to verify that a hardware circuit M verifies a reliability property Φ (2), denoted by $M \models \Phi$. We briefly describe the statistical model checking technique that we employ to verify $M \models \Phi$.

Let p_F denote the actual failure rate of M. If p_F is less than the threshold θ specified in Φ, then $M \models \Phi$. SMC obtains an estimate of the failure rate by performing the Monte Carlo simulations of M. $M \models \Phi$ is verified by comparing this estimated failure rate against θ.

Let V^i denote the i^{th} sample drawn according to the statistical distribution $D(V)$ (Definition 2) of M. We define $\mathbb{I}(V^i)$ to be an indicator function [22] that is equal to 1 if the sample V^i is failing (Definition 3) and 0 otherwise

$$\mathbb{I}(V^i) = \begin{cases} 1, & \text{if } V^i \text{ is a failing sample} \\ 0, & \text{if } V^i \text{ is not failing.} \end{cases} \tag{3}$$

After N_S samples have been generated, the expected (average) failure rate can be estimated as

$$\widehat{p_F} \quad = \quad \frac{1}{N_S} \sum_{i=1}^{N_S} \mathbb{I}(V^i). \tag{4}$$

However, in a different sampling run, another set of N_S samples could be used instead to estimate the failure rate. Therefore, the estimate is itself a random variable. For large N_S, the estimate is typically modeled as a Gaussian random variable (Fig. 1) with mean $\widehat{p_F}$. The variance $\sigma^2_{\widehat{p_F}}$ of the estimate is given by

$$\sigma^2_{\widehat{p_F}} \quad = \quad \frac{\sum_{i=1}^{N_S} [\mathbb{I}(V^i) - \widehat{p_F}]^2}{N_S(N_S - 1)}. \tag{5}$$

The Gaussian distribution represents how well $\widehat{p_F}$ estimates the actual failure rate p_F. p_F is more likely to be near the mean $\widehat{p_F}$ of the distribution and less likely to be in the tail regions.

SMC verifies $M \models \Phi$ by comparing $\widehat{p_F}$ against the threshold θ. In this paper, we consider a SMC technique that uses

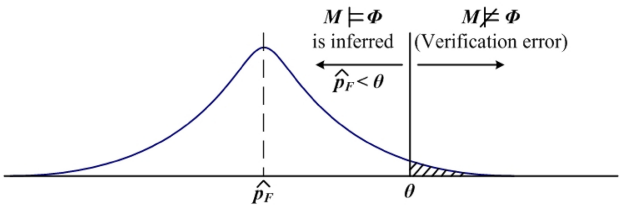

Fig. 1. Gaussian distribution of failure rate estimates, with mean $\widehat{p_F}$ and variance $\sigma^2_{\widehat{p_F}}$. The area of the shaded region is the probability of error in the verification result.

estimation testing. Since $\widehat{p_F}$ is only an estimate obtained using a limited set of simulations, the verification result may be inaccurate. SMC draws sufficient samples until the verification results are within the specified bounds of error α and β given as

$$P[(M \models \phi) \mid (M \not\models \phi) \text{ is claimed based on samples}] \leq \alpha$$
$$P[(M \not\models \phi) \mid (M \models \phi) \text{ is claimed based on samples}] \leq \beta \quad (6)$$

where α and β are bounds for the likelihood of error in the verification result, and therefore represent the verification accuracy. α is guaranteed to be an upper bound of the probability that SMC verifies the property to be FALSE based on the sample paths when in fact the property is TRUE in the system. Similarly, the β bounds the probability that a property is incorrectly verified to be TRUE. Tight error bounds (small α and β) can be provided by using a sufficiently large number of samples [39].

Fig. 1 depicts the scenario where $\widehat{p_F} < \theta$. In this scenario, the verification result is incorrect if the actual failure rate p_F is greater than the θ. Therefore, the probability of error is equal to the area of the shaded region in the figure. We require this probability to be less than the bound α (6). Similarly, if $\widehat{p_F} > \theta$, we require the probability of error to be less than β.

Verification errors arise when the actual failure rate p_F and the estimate $\widehat{p_F}$ lie on different sides of the threshold θ. As the number of samples N_S increases, the variance of the estimate (5) reduces and the Gaussian curve becomes narrower. As a result, the probability of occurrence of a verification error also reduces.

D. Variational Bayes

In this section, we briefly describe the variational Bayes technique [4]. Given a set of samples, variational Bayes can cluster those samples together and approximately model their statistical distribution as a Gaussian mixture distribution given by

$$\sum_{i=1}^{K} \pi_i \, \mathcal{N}(\mu_i, \Lambda_i^{-1}) \quad (7)$$

where K represents the number of Gaussian components in the mixture distribution with mean $\mu_\mathbf{i}$, variance $\Lambda_\mathbf{i}^{-1}$ ($\Lambda_\mathbf{i}$ is the precision) and $\pi_\mathbf{i}$ denotes the weight of each component in the mixture.

Let $\{\mathbf{x}_1, \ldots \mathbf{x}_n\}$ denote a set of samples from an N-dimensional sample space (Definition 1). Variational Bayes fits these samples to a mixture Gaussian distribution (7) by iteratively computing and updating the parameters $\mu_\mathbf{i}$, $\Lambda_\mathbf{i}^{-1}$, and $\pi_\mathbf{i}$ for each of the K components in the mixture.

Let z_{ij} indicates whether a corresponding sample $\mathbf{x}_i$ belongs to component j in the mixture where $1 \leq j \leq K$. Let $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ where each $\mathbf{z}_i$ corresponds to sample $\mathbf{x}_i$. The $\mathbf{z}_i$ is one-of-K vector with components z_{nk} for $k = 1 \ldots K$ where one of the elements is 1 and all other $K - 1$ elements are 0. The variational Bayes models the variable $\mathbf{Z}$ and the parameters mean μ, precision Λ, and mixture weight π as random variables (where mean follows a Gaussian distribution, the precision follows a Wishart distribution and weight mixture follows a Dirichlet distribution). We assume that the variational distribution can be factorized between the variable $\mathbf{Z}$ and the parameters mean μ, precision Λ, and mixture weight π. In order to compute these parameters, the algorithm iteratively alternates between two steps: 1) computing the responsibility of each cluster in explaining the samples and 2) using the responsibilities to update the distribution parameters. These iterations are repeated until the technique converges. The output of the algorithm is mean μ, the precision Λ, and the weight mixture π. Further details of this technique can be found in [4].

The variational Bayes computes the mixture weights as $\pi_i = \frac{1}{n} \sum_{j=1}^{n} r_{ji}$, where r_{ij} are responsibilities of each samples with respect to each component in the distribution [4]. The responsibilities and weight coefficients of components that provide inadequate explanation of the samples will converge to zero. Therefore, after convergence, components with negligible mixture weights are discarded. As a result, the technique does not require prior information that specifies the exact number of components in the mixture distribution. In [4], this feature is referred to as automatic relevance determination.

IV. Speeding Up SMC for Rare-Event Scenarios of Hardware Circuit

In this section, we define and establish the criteria for improving SMC in rare-event scenarios of a hardware circuit. We consider M to be the hardware circuit with the statistical distribution $D(V)$ (Definition 2). Let Φ (2) denote the reliability property of our interest.

We wish to verify whether $M \models \Phi$. The conventional SMC engine (Section III-C) infers whether $M \models \Phi$ by performing the Monte Carlo simulations of M according to the distribution $D(V)$ (Definition 2). In the rare-event scenarios, the failing samples (Definition 3) are generated with very low probability (1). Therefore, a very large number of samples need to be generated to gather sufficient statistical evidence regarding the failure rate. This makes SMC extremely time consuming for rare-event scenarios.

In this paper, we analyze a biasing approach to reduce the number of samples that need to be generated by the SMC in the rare-event scenarios for the hardware circuits. We achieve this by biasing the distribution of circuit M toward frequent occurrence of failures. In order to account for the increased failure rate, we will need to adjust for the bias while verifying $M \models \Phi$. Let $D'(V)$ be the joint distribution of the variables V

that is biased to generate the failing samples more frequently. We verify $M \models \Phi$ by drawing samples according to the distribution $D'(V)$ instead of $D(V)$. We show that the required statistical evidence can now be collected by generating fewer samples thereby improving the performance of the SMC in the rare-event scenarios. While verifying whether $M \models \Phi$, we exactly adjust for the statistical bias that we introduce in $D'(V)$. Therefore, verifying $M \models \Phi$ using the samples based on the distribution $D'(V)$ is now provably equivalent to regular SMC (Section III-C) that uses samples based on $D(V)$. This forms the basis of our biasing approach.

In order to construct the distribution $D'(V)$ that is biased toward failures, we wish to first broadly identify the failure regions (Definition 4), where the failing samples of the hardware circuit M are known to be present. In order to achieve this, we employ the variational Bayes (Section III-D). As the data for our inference technique, we first generate several examples of failures of M. We are not concerned with the statistical distribution of M at this stage. Therefore, we quickly generate the examples by uniformly sampling the space $\mathcal{S}$ (Definition 1) of M. We then employ the variational Bayes, which models the failing samples as a mixture Gaussian distribution $D'(V)$ consisting of K spatially components in the sample space. We interpret these K components of the mixture distribution to be the failure regions $\mathcal{S}_F^l$ ($l = 1$ to K) of M.

We consider the mixture distribution $D'(V)$ that is biased toward the failure regions $\mathcal{S}_F^l$ which we infer. Sampling based on the biased distribution $D'(V)$ generates several failing samples from the regions $\mathcal{S}_F^l$. We replace the statistical distribution $D(V)$ of M (Definition 2) with the distribution $D'(V)$. As a result, the Monte Carlo simulation of M is biased toward the failure regions $\mathcal{S}_F^l$ since the samples are drawn according to the distribution $D'(V)$.

We employ SMC using the samples drawn according to the biased distribution $D'(V)$. Sampling $D'(V)$ generates failures with the high frequency as compared to sampling the original distribution $D(V)$. We know the exact extent to which each sample is biased. We adjust for this bias analytically while estimating the failure rate to verify whether the $M \models \Phi$. Our approach can be viewed as performing importance sampling [11] on the hardware circuit M.

V. Our Biasing Approach for SMC of Hardware Circuit

We wish to check whether a hardware circuit M satisfies a reliability property Φ (2). We focus on the case where Φ deals with a rare-event scenario (Section III-B). We now describe all the steps in our approach (Fig. 2).

A. Identifying Failure Region of Hardware Circuit

We wish to broadly identify the K failure regions $\mathcal{S}_F^l$ of M with respect to Φ. We first generate a set of failing samples (Definition 3) of M. We then employ variational Bayes (Section III-D) that uses these samples as examples to approximately identify the mixture distribution of $\mathcal{S}_F^l$. We now describe these two steps in detail. We emphasize that we do not require a prior knowledge about K, the number of failure regions, as it will be determined by the variational Bayes algorithm.

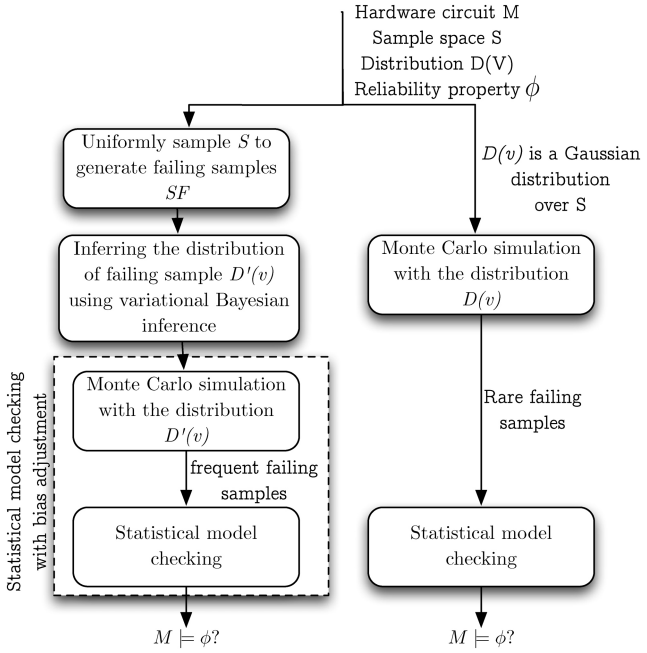

Fig. 2. Block diagram comparing the steps in our biasing approach (left branch) with those in regular SMC (right branch).

1) *Uniformly Sampling Space $\mathcal{S}$:* We can generate examples of the failures by performing the Monte Carlo simulations that sample the space $\mathcal{S}$ (Definition 1) based on the distribution $D(V)$. However, according to the statistical distribution $D(V)$ (Definition 2), all the failing samples of M occur with very low probability. Therefore, a large number of simulations would be required making example generation extremely time consuming.

At this stage of our algorithm, we are not concerned with the statistical distribution of M. With respect to determining the location of the failure region, all the failures of M are equally important regardless of the probabilities with which they occur. Therefore, we generate failures of M by uniformly sampling the space $\mathcal{S}$ (Definition 1). Such uniform sampling generates failing samples more frequently compared to sampling based on the original distribution $D(V)$. In the rare-event scenarios, the failures are concentrated in the tail region of this distribution. A uniform distribution has the highest variance and maximum entropy among all the possible continuous distributions defined on a bounded state space. Therefore, using a uniform distribution instead of the original distribution, is guaranteed to sample the tail region of the original distribution more frequently.

In order to uniformly sample the space $\mathcal{S}$, we sample the set of variables V (Section III-A) by treating them as a set of independent, uniformly distributed random variables. We uniformly sample $\mathcal{S}$ till the desired number of failing samples of the hardware circuit M are obtained.

2) *Identifying Mixture Distribution of Failing Samples:* To identify the distribution of the failing samples, we employ the variational Bayes (Section III-D) technique on the failing samples that we generate by uniformly sampling the state space $\mathcal{S}$. Variational Bayes fits the failing samples to a mixture Gaussian distribution (8).

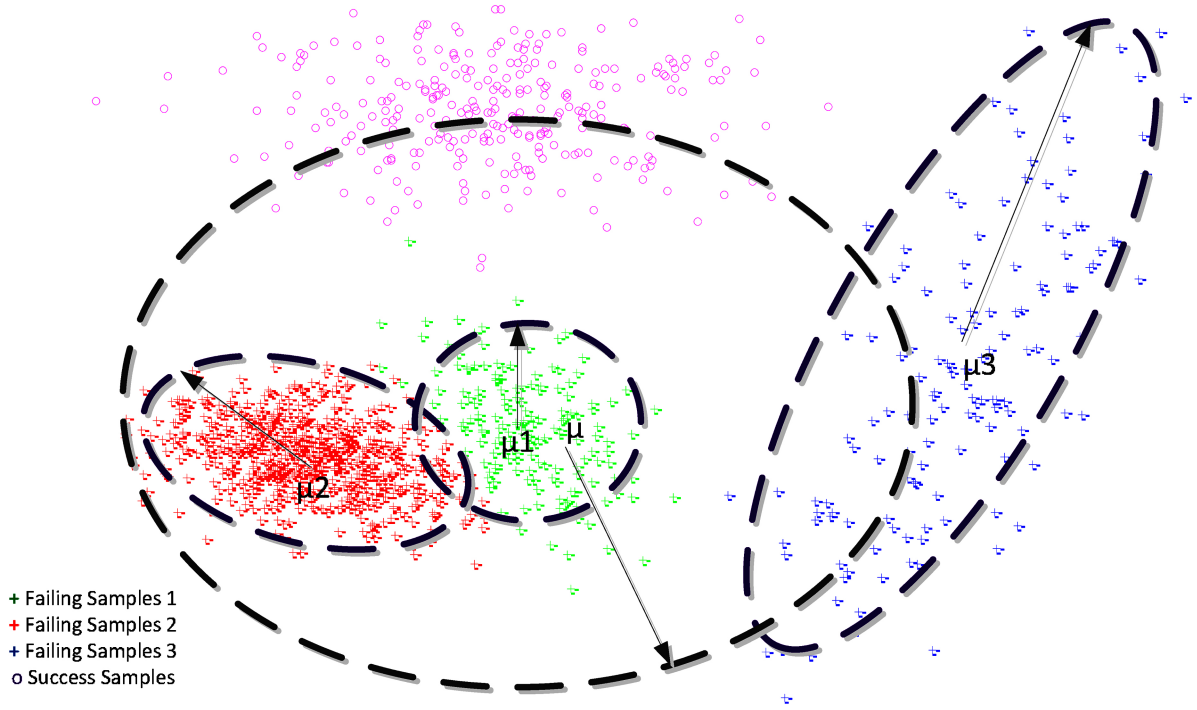

Fig. 3. Clustering in 2-D space for an example circuit with three failure regions. We identify the three distinct failure regions with means μ_1, μ_2, and μ_3. The dashed circles indicate the failing samples covered by each region. The single failure region with mean μ does not cover all the failing samples efficiently.

Variational Bayes requires a prior distribution of the samples to be specified, as an initial condition of the iterative process. In our sample generation phase, we assumed to have no knowledge about the prior distribution. Therefore, as in [4], we choose an independent Gaussian–Wishart distribution with mean $\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$, variance equal to 1 as a prior distribution. We set $\alpha_0 = 0$ (effective prior number of samples associated with each failure region) to guide the computed mixture distribution to be mostly dependent on the sample data rather than on the prior distribution that we provide. The variational Bayes, upon convergence, outputs the parameters of the mixture Gaussian distribution.

Some continuous-space hardware circuits, such as the SRAM cell (Section VII), have a single continuous failure region that is clearly demarcated from the region that contains successful samples. In some complex circuits, several distinct failure regions may be present. Furthermore, the distribution of failure regions might be dependent on each other. If we compute the failure distribution with only one component (K=1), then our algorithm infers a single failure region that may contain only a few failing samples (Fig. 3). As a result, biasing the circuit toward this failure region may not increase the failure rate. However, although inefficient, this does not affect the correctness of our approach.

In general, the number of failure regions in a circuit are not known *a priori*. Therefore, we adopt a conservative approach by assuming the number of failure regions K to a large value. The variational Bayes technique should be initialized with a number $K' \geq K$. The variational Bayes technique will quickly discard all the $K' - K$ redundant components and quickly converge on the actual K. In our case, we adopt a conservative approach and start with a very high value of K' (say 1000) which is highly likely to be greater than any value of K for a practical hardware circuit. We observe that the variational Bayes converges to the actual K within a few iterations. We then rely on the automatic relevance determination (Section III-D) feature of the variational Bayes, which drives the mixing weights of the extra components to zero. Therefore, the variational Bayes will eventually converge on the actual number of failure regions in the circuit.

Each component in the mixture distribution corresponds to a failure region in the circuit. The mixture distribution $D'(V)$ of the failing samples is given by

$$D'(V) = \sum_{i=1}^{K} \pi_i \, \mathcal{N}(\mu_i, \Lambda_i^{-1}) \tag{8}$$

where K is the number of failure clusters with nonnegligible contribution toward the mixture. This enables us to compute the centroids of multiple, distinct failure regions. Kanj *et al.* [16] used a simple averaging of all the generated failing samples to approximate the centroid of a single failure region in an SRAM cell (with single failure region). However, since they do not employ clustering, they have no mechanism to compute the centroids of multiple failure regions.

Definition 6: The centroid of the failure region $\mathcal{S}_F^l$ is the mean of the lth component in the mixture failure distribution (8).

The means μ_l computed by the variational Bayes approximates the centroid of each of the failure regions $\mathcal{S}_F^l$ (l = 1 to K). We also use the failing samples to approximately determine the boundaries of the hyperrectangle R_F (Definition 5) that bounds the failure region $\mathcal{S}_F$.

In Section V-B, we describe how the mixture distribution $D'(V)$ can be used to bias the distribution of the hardware circuit toward $\mathcal{S}_F^l$.

B. Biasing Distribution Toward Failure Regions

In order to increase the frequency of failing samples during the Monte Carlo simulation, the failure regions that we identify needs to be sampled more frequently. We achieve this by generating samples from the failure mixture distribution $D'(V)$ (8) instead of generating the samples from $D(V)$ (Definition 2). We refer to $D'(V)$ as the biased distribution of the hardware circuit M.

We modify M by substituting the distribution $D(V)$ with the biased distribution $D'(V)$. If the Monte Carlo simulation is performed using the distribution $D'(V)$, a larger number of failing samples are generated as compared to using the original distribution $D(V)$ (Fig. 2).

C. SMC With Bias Adjustment

We now perform the Monte Carlo simulations and sample the space $\mathcal{S}$ of the hardware circuit M based on its biased distribution $D'(V)$ (8). We use the generated samples to perform SMC and verify whether $M \models \Phi$. In order to maintain the accuracy of the SMC results, we adjust for the distribution bias in each sample.

Let $L(V^i)$ denote the extent to which a sample V^i is biased. We define $L(V^i)$ as

$$L(V^i) = \frac{D(V^i)}{D'(V^i)} \tag{9}$$

where $D(V^i)$ is the probability of occurrence of V^i according to the original distribution $D(V)$ (1). Similarly, $D'(V^i)$ is the probability of occurrence of V^i based on the biased distribution $D'(V)$ (8). In importance sampling, $L(V^i)$ is typically referred to as the inverse likelihood ratio [11].

After N'_S biased samples have been generated, the estimate $\widehat{p_F}'$ of the actual failure rate can be computed [11] as

$$\widehat{p_F}' = \frac{\sum_{i=1}^{N'_S} L(V^i)\mathbb{I}(V^i)}{N'_S} \tag{10}$$

where $\mathbb{I}(V^i)$ is the indicator function described in (3). The use of $L(V^i)$ adjusts the estimate for the statistical bias in each sample. The variance $\sigma'^2_{p_F}$ of the estimate, in the presence of biasing, is given by

$$\sigma'^2_{p_F} = \frac{\sum_{i=1}^{N'_S}[L(V^i)\mathbb{I}(V^i) - \widehat{p_F}']^2}{N'_S(N'_S - 1)}. \tag{11}$$

As described in Section III-C, the SMC engine draws samples based on the biased distribution $D'(V)$ until the error bounds specified by α and β (6) are satisfied. We find that the number of samples N'_S required to meet the error bounds using our approach is much smaller than the number of samples N_S required while using regular SMC (Section III-C).

VI. Analysis of Our Biasing Approach

We now briefly outline the correctness and speedup of our approach. We also derive analytical upper bounds for the variance of our failure rate estimates.

A. Correctness and Speedup

In order to verify whether a hardware circuit M satisfies a reliability property Φ, SMC estimates the probability of failure p_F and checks whether it is less than the specified probability threshold θ (Section III-C). In our approach, we estimate the probability of failure by drawing samples according to the biased distribution $D'(V)$ instead of the original distribution $D(V)$. If the estimation accuracy is preserved in the presence of biasing, the verification result would remain the same.

For probability estimation, the use of (10) has been proven to be correct [11] when biasing toward rare events. Therefore, the verification result that we obtain with our biasing approach is consistent with what we obtain with regular SMC.

A metric for speedup in SMC is the reduction in the number of samples required to provide the verification result. The SMC terminates when the estimation error is low enough to meet error bounds specified by α and β (Section III-C). If the error bounds can be met with fewer samples, SMC can be sped up.

From Fig. 1, we observe that the error in verification can be decreased by reducing the variance of the failure rate estimate (Section III-C). In other words, with variance reduction, a given error bound can be met with fewer samples. Statistical biasing, if applied correctly, is known to reduce estimation variance. As a result, biasing toward rare events can result in a speedup for SMC. As the failure rate p_F becomes smaller, the extent of such speedup typically becomes larger.

In Section VI-C, we derive analytical upper bounds for variance and show that our biasing approach guarantees a reduction in the variance over regular SMC. In Section VII-A, we also present empirical evidence for the variance reduction provided by our approach in comparison to the regular SMC.

B. Complexity

The additional samples (Section V-A) required for the variational Bayes is an overhead for our approach. As p_F decreases, the number of simulations N_E required to generate the examples of failing samples increases. However, since we use uniform sampling to generate these examples, the increase in N_E is not very severe in practice. Therefore, the overall speedup provided by our methodology still increases with decrease in p_F.

In theory, the complexity of the variational Bayes algorithm grows as $O(N_E \times V)$, N_E is the number of examples generated and V is a cost associated with updating the mixture distribution. The inference incurs a one-time overhead before performing the SMC on the biased system. We find that N_E can also be kept small.

C. Analytical Bounds for Variance of Estimates

In [11], the author presents analytical equations for computing the asymptotic variance of the failure rate estimates that are obtained with or without importance sampling. We now use these equations to derive upper bounds for the estimation variance.

In regular SMC (Section III-C), without importance sampling, the asymptotic variance AVar of the failure rate estimate

(4) is given as

$$\text{AVar} = \int_{\mathcal{S}} [\mathbb{I}(V)]^2 D(V) dV - {p_F}^2 \tag{12}$$

where the integration is performed over the sample space $\mathcal{S}$ of the hardware circuit (Definition 1). In other words, $\mathcal{S}$ is the domain of the N-D integration variable V.

Since $\mathbb{I}(V)$ is equal to 1 inside each failure region $\mathcal{S}_F^l$ and 0 elsewhere (3), the above equation for AVar can be written as

$$\text{AVar} = \sum_{l=1}^{K} \left(\int_{\mathcal{S}_F^l} D(V) dV - {p_F}^2 \right). \tag{13}$$

In the hardware circuits that we consider, each $\mathcal{S}_F^l$ is contained in the N-D hyperrectangle R_F^l (Definition 5). Therefore, in geometric terms, the volume of R_F^l is greater than or equal to the volume of $\mathcal{S}_F^l$. Since the integrand in (13) is strictly positive everywhere in $\mathcal{S}$, AVar can be bounded as

$$\begin{aligned} \text{AVar} &\leq \sum_{l=1}^{K} \left(\int_{R_F^l} D(V) dV - {p_F}^2 \right) \\ &\leq \sum_{l=1}^{K} \left(\int_{R_F^l} D(V) dV \right). \end{aligned} \tag{14}$$

In the presence of importance sampling, the asymptotic variance AVar' of the failure rate estimate is given as

$$\text{AVar'} = \int_{\mathcal{S}} [L(V)\mathbb{I}(V)]^2 D'(V) dV - {p_F}^2 \tag{15}$$

where $L(V)=\frac{D(V)}{D'(V)}$ as described in (9).

We employ the same line of reasoning described above and derive the upper bound of AVar' as

$$\begin{aligned} \text{AVar'} &= \sum_{l=1}^{K} \left(\int_{\mathcal{S}_F^l} L(V)^2 D'(V) dV - {p_F}^2 \right) \\ &\leq \sum_{l=1}^{K} \left(\int_{R_F^l} \frac{D(V)^2}{D'(V)} dV \right). \end{aligned} \tag{16}$$

The biased distribution $D'(V)$ depends on the centroid of the failure region (Definition 6) that we determine by using variational Bayes. However, since clustering uses data that is generated by uniformly sampling the space $\mathcal{S}$ (Section V-A), the centroid is a statistical estimate. Consequently, the upper bound AVar' defined in (16) is also a statistical metric. However, for a given estimate of the centroid, we can determine the corresponding value of AVar'.

In Section VII-A, we show that we can compute the upper bounds of AVar and AVar' by using empirical data from circuit simulations. However, the tightness of the upper bounds described in (14) and (16) depend on the tightness with which each R_F^l bounds the failure region $\mathcal{S}_F^l$. Moreover, we cannot determine the exact boundaries for R_F^l. Therefore, the upper bounds that we compute using empirical data are approximate. However, we find that these bounds are good indicators for the extent of variance reduction provided by our biasing approach over regular SMC.

If AVar' is less than AVar, our biasing approach can result in a variance reduction which in turn provides a speedup over regular SMC. In the worst-case scenario, a poorly estimated centroid (Definition 6) may cause the variance of our approach to be worse than that of regular SMC. However, we find that the number of failing samples required to obtain a good estimate of the centroid of the failure region, is small in practice.

The biased distribution can be constructed as a linear combination of $D(V)$ (1) and $D'(V)$ (8). The use of this mixture distribution [11], [16] guarantees that, even in the worst case, the variance estimate in the presence of biasing is bounded independently of the centroid estimate. While this approach guarantees an upper bound AVar' that is not statistical, it limits the maximum speedup that can be obtained using biasing. We shall not explore this mixture distribution approach further in this paper.

VII. Experimental Results

We now demonstrate our biasing approach on two hardware circuits: 1) an SRAM cell and 2) an SAR-ADC. We present the following detailed experimental analysis in this section.

1) We show correctness and effectiveness of our algorithm for an SRAM circuit with a single failure region. We provide empirical evidence for the speedup and variance reduction through our algorithm.
2) Our algorithm is most applicable for circuits with multiple failure regions. We demonstrate the effectiveness of our algorithm for an SAR-ADC circuit with multiple failure regions and report the speedup. We cannot provide a direct comparison with any other technique because, to the best of our knowledge, no other technique is capable of handling multiple failure regions.
3) We compare our algorithm with a state-of-the-art technique, statistical blockade [29] for SRAM circuit with a single failure region. We focus on a single failure region for the sake of comparison, even thought this is not the intended application of our algorithm. We demonstrate a speedup similar to statistical blockade, despite requiring much fewer samples for the training data.
4) We show that our algorithm is practical and converges quickly even for rare events with extreme deviation from the mean of the probability distribution (6σ and beyond). The number of required samples (for training and SMC) increases linearly even for failure regions with extreme deviation. We establish, through this experiment, a metric for comparison between our algorithm and any future work that can apply to circuits with multiple failure regions.

A. Case Study I: Reliability of SRAM Cell

We consider an SRAM cell circuit that comprises six transistors [29]. Due to process variations, the threshold voltages

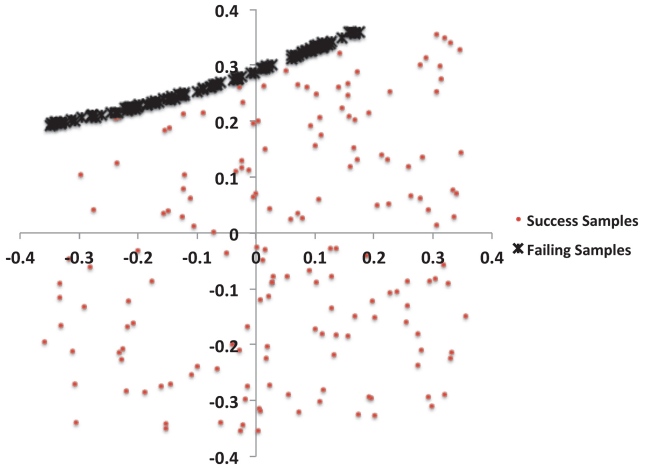

Fig. 4. 2-D projection of the failing samples that we generate by uniformly sampling (Section V-A) the 6-D sample space of the SRAM cell.

of these six transistors are typically modeled as independent, Gaussian random variables. Therefore, the SRAM cell can be viewed as a statistical entity with six random variables (Section III-A). The delay of the SRAM cell depends on the threshold voltages of the transistors. An SRAM cell is said to fail if its delay exceeds a predefined timing constraint. We wish to verify that the failure rate of the SRAM cell is less than a threshold θ (2).

We generate 30 examples of failures as data for identifying the failure region. Fig. 4 depicts the 2-D projection of the failing samples that we obtain. The SRAM cell has a single continuous failure region $\mathcal{S}_F$. Therefore, the variational Bayes identifies only one cluster.

In Table I, we provide empirical evidence for the variance reduction provided by our biasing approach (Column 5) over regular SMC. We choose three different timing constraints for the SRAM cell. For each timing constraint, SMC estimates the corresponding failure rate (Column 1) of the SRAM cell. For a fixed number of samples (=10^5), the variance in the estimate computed using our biasing approach (Column 4) is significantly less than that when biasing is not employed (Column 3). In Table I, we fix the number of samples to be 10^5. As a result, for a failure rate of around 10^{-5}, the estimate provided by regular SMC is highly unreliable since we observe only a single failure among 10^5 samples. On the other hand, when we use our biasing approach with 10^5 samples, we observe several failures and consequently obtain a more reliable estimate of the failure rate. The deviation of the estimates in Columns 1 and 3 of the last row demonstrates the effectiveness of our biasing approach over regular SMC.

We consider the 30 failing samples that we generate for identifying the failure region $\mathcal{S}_F$ and use them to estimate the boundaries of the hyperrectangle R_F (Definition 5). In Table II, we list the variance bounds that we compute using (14) and (16). We confirm that the analytical upper bound for variance in our biasing approach is less than that in regular SMC. Since we consider only a limited number of failing samples, the boundaries that we determine for R_F are approximate. Due to the coarseness of our approximation, the bounds that we compute are loose compared to the variance estimates in Table I. However, the reduction factors that we compute in both Tables I and II are comparable in magnitude.

TABLE I

DEMONSTRATING VARIANCE REDUCTION USING OUR BIASING APPROACH ON SRAM CELL. WE USE 10^5 SAMPLES FOR BOTH APPROACHES

Regular SMC		Our method		
Failure rate estimate $\widehat{p_F}$	Variance $\sigma^2_{p_F}$	Failure rate estimate $\widehat{p_F}'$	Variance $\sigma'^2_{p_F}$	Variance reduction $\frac{\sigma^2_{p_F}}{\sigma'^2_{p_F}}$
$2.20\text{x}10^{-3}$	$2.74\text{x}10^{-8}$	$2.16\text{x}10^{-3}$	$7.43\text{x}10^{-9}$	3.69x
$5.22\text{x}10^{-4}$	$6.64\text{x}10^{-9}$	$5.17\text{x}10^{-4}$	$9.6\text{x}10^{-10}$	6.92x
$8.76\text{x}10^{-5}$	$1.10\text{x}10^{-9}$	$4.53\text{x}10^{-5}$	$1.05\text{x}10^{-10}$	10.5x

TABLE II

DEMONSTRATING THAT OUR BIASING APPROACH PROVIDES A REDUCTION IN ANALYTICAL UPPER BOUND OF ESTIMATION VARIANCE

	Regular SMC	Our method	
Failure rate estimate $\widehat{p_F}$	AVar Upper bound	AVar' Upper bound	Upper bound reduction
$2.20\text{x}10^{-3}$	$1.93\text{x}10^{-3}$	$5.53\text{x}10^{-4}$	3.49x
$5.22\text{x}10^{-4}$	$3.18\text{x}10^{-4}$	$1.68\text{x}10^{-5}$	18.9x
$8.76\text{x}10^{-5}$	$1.79\text{x}10^{-5}$	$7.29\text{x}10^{-7}$	24.5x

In Table III, we provide evidence for the correctness and speedup provided by our approach. We executed our algorithm 10 times and reported the average number of samples required for the SMC. We also reported the average execution time of our tool (including our algorithm and the simulator). For the three different failure rates that we consider in Table I, we verify Φ (2) for low values of θ. The variance reduction provided by our approach results in a significant reduction in the number of samples required by the SMC engine to arrive at a result within the specified error bounds (Section III-C). Since we use uniform sampling to generate data for clustering, the number of samples required for identifying the failure region does not increase sharply with decrease in p_F. Therefore, our approach provides several orders of magnitude in overall speedup (Column 8). Moreover, our verification results (Column 7) are consistent with those obtained using regular SMC (Column 4).

B. Case Study II: Reliability of SAR-ADC

In this case study, our focus is on demonstrating that our biasing approach can be applied on circuits with multiple failure regions.

We consider an SAR-ADC circuit [1] which has two distinct failure regions. The SAR-ADC circuit (Fig. 5) converts an analog input v_{input} to a discrete, 10-bit digital representation (*Result*) via an implementation of a binary search algorithm. We implemented a 10 Megasamples/s, 10-bit mixed-signal SAR-ADC by modeling the digital controller circuitry in Verilog language and the analog circuitry in Verilog-A language. A detailed description of this circuit and its functionality can be found in [1].

TABLE III

DEMONSTRATING SPEEDUP OF OUR BIASING APPROACH ON SRAM CELL. WE SET ERROR BOUNDS α=β=0.01

		Regular SMC		Our method				
Failure rate estimate	Threshold θ	SMC samples	Result	Samples for identifying $\mathcal{S}_F$	SMC (biased) samples	Result	Speedup	time (hours)
$2.20\text{x}10^{-3}$	$2.5\text{x}10^{-3}$	44254	TRUE	4425	218.1	TRUE	9.53x	1.5 hrs
$5.22\text{x}10^{-4}$	$5\text{x}10^{-4}$	181720	FALSE	8152	419.1	FALSE	21.20x	2.8 hrs
$8.76\text{x}10^{-5}$	10^{-4}	322556	TRUE	9182	1586.5	TRUE	29.95x	3.3 hrs

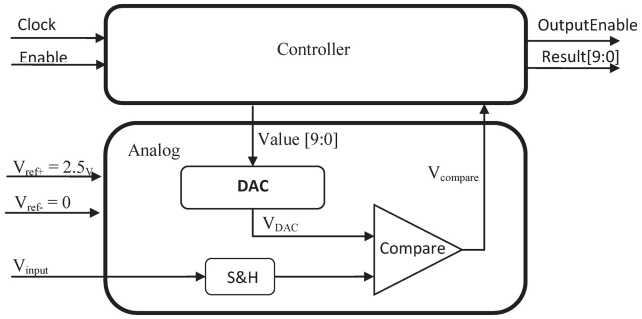

Fig. 5. Block diagram of the SAR-ADC circuit. The circuit consists of a digital controller, an analog DAC, a sample/hold circuit (S&H) and a comparator circuit (compare).

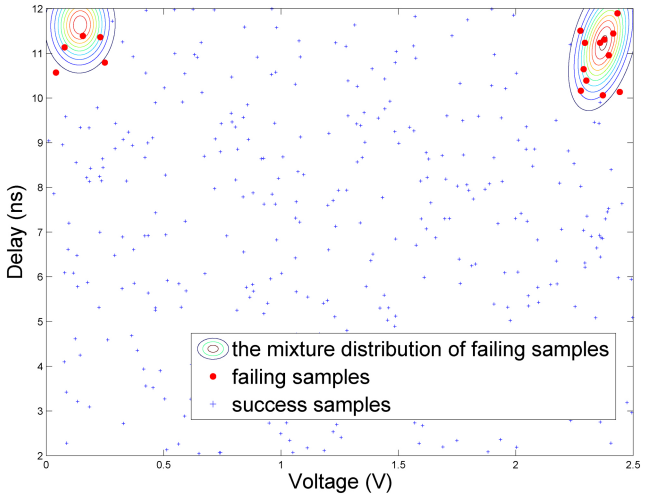

Fig. 6. Failing samples that we generate by uniformly sampling (Section V-A) the 2-D sample space of the SAR-ADC. Most failing samples are concentrated near two failing clusters. We used variational Bayes to cluster the failing samples together and then computed the mixture distribution of the failing samples.

We consider the following two random variables in this circuit: the input voltage v_{input} and the transient delay t_{delay} of the analog DAC component, where $0V \leq v_{\text{input}} \leq 2.5V$ and $2ns \leq t_{\text{delay}} \leq 12ns$. We model v_{input} as a Gaussian random variable with mean = 1.25 and variance = 1. We model t_{delay} as a Gaussian random variable with mean = 7 and variance = 1. Therefore, our SAR-ADC circuit can be viewed as a statistical entity with two random variables (Section III-A).

Let $Q = \dfrac{V_{ref+} - V_{ref-}}{2^B - 1}$ indicate the precision of the SAR-ADC, where B denotes the number of resolution bits in the SAR-ADC. The approximation error between the v_{input} and corresponding computed value *Result* is given by $|(Result \times Q) - v_{\text{input}}|$. The output of the SAR-ADC is said to be correct, if and only if this approximation error is smaller than the precision of the circuit. In other words, the SAR-ADC fails if $|(Result \times Q) - v_{\text{input}}| > Q$. Formally, we want to verify

$$\text{SAR-ADC} \models P_{\leq\theta}\mathbf{F}^{10^{ns}}(|\text{Result} \times Q - v_{\text{input}}| < Q). \quad (17)$$

Typically, the SAR-ADC circuit fails if there are bugs in the circuit that cause large approximation errors. In this case study, we consider a bug that is triggered depending on the values of v_{input} and t_{delay}. We observe 17 failing sample that are distributed in two failure regions in the 2-D sample space of the circuit, as shown in Fig. 6.

We wish to verify that the failure rate of the SAR-ADC, in the presence of the bug, is less than the threshold θ (2). We employ our biasing approach by setting K=2 and we identify the two distinct failure regions of the circuit (Fig. 6).

As in the case of the SRAM cell (Section VII-A), we show that our biasing approach provides a speedup over regular SMC while preserving correctness of the verification results (Table IV). We ran each experiments 100 times and recorded the average number of SMC samples. We also reported the average time required for the SMC on our setup (including our tool and the simulation time). For $\theta = 10^{-3}$, our biasing approach provides average speedup of 52x over regular SMC. We thus demonstrate, with empirical evidence, that our biasing approach is also effective for circuits with multiple failure regions.

C. Comparative Study of Our Algorithm for a Single Failure Region

To the best of our knowledge, there is no other technique that can analyze circuits with multiple failure regions. So we are unable to provide a direct comparison of our technique with other similar methods for circuits with multiple failure regions. Hence, we provide a comparison with a well known technique that analyzes a single failure region. The purpose of the direct comparison is only to provide a quantitative measure with a known algorithm. However, the real value of our algorithm is in analyzing circuits with multiple failure regions.

1) *Comparison With Statistical Blockade:* For direct comparison, we chose the statistical blockade [29] algorithm as an example of a state-of-the-art technique for accelerated Monte Carlo simulations. The original statistical blockade algorithm

TABLE IV
DEMONSTRATING SPEEDUP OF OUR BIASING APPROACH ON SAR-ADC CIRCUIT. WE SET ERROR BOUNDS α=β=0.05

	Regular SMC		Our biasing approach				
Threshold θ	SMC samples	Result	Samples for identifying $\mathcal{S}_F$	SMC (biased) samples	Result	Speedup	time (hours)
10^{-3}	31765	TRUE	543	41.31	TRUE	54.36x	1 hr
10^{-4}	35612	FALSE	543	40.67	FALSE	61.01x	1 hr
10^{-5}	24489	FALSE	543	41.77	FALSE	41.87x	1 hr

is used for fast yield estimation in SRAM circuits. In order to adapt this algorithm for property checking, we generated samples using statistical blockade in the sample generation loop of SMC. SMC iteratively generates samples and simulate those samples to find the statistical evidence for satisfaction of the given property. These samples can be generated from: 1) an unbiased distribution (the standard SMC); 2) a biased distribution (our algorithm): or 3) an unbiased distribution, then filtered through an SVM (the statistical blockade algorithm). Thereafter, our comparison method is as follows. Initially, we generated 30 examples of failure samples as training data for both our algorithm and the statistical blockade. The training data are shown in Fig. 4. The statistical blockade algorithms relies on SVM [29] algorithm to classify failure samples. We trained and optimized the SVM algorithm to classify failure samples. We then executed the SMC algorithm with the same parameters as our algorithm ($\alpha = \beta = 0.01$ and variable θ). We generated the samples using a normal distribution, but we only simulated the system for the samples that SVM classified as failure samples.

Fig. 7 shows the number of SMC samples required by the statistical blockade algorithm versus the number of samples required by our sample generation method. Given the limited number of training samples (30 failure samples), the accuracy of statistical blockade is relatively low. As a result, for circuits with single failure region and in cases where the number of training data is limited, we demonstrate our algorithm is as fast as statistical blockade. In comparison to our algorithm, for circuits with a single failure region, the statistical blockade requires a large number of failure samples to train the SVM for high accuracy, in order to achieve high speedup. In general, if we run statistical blockade in an ideal setup with a large set of training data (more than 1000 failure samples), we expect statistical blockade to outperform our algorithm for circuits with single failure region. We do not have the empirical results to support this hypotheses in this paper. Notably, the statistical blockade algorithm is unable to analyze circuits with multiple failure regions.

D. Extreme Value Analysis of Our Algorithm

We show applicability of our algorithm for failure regions with extreme deviation. For circuits with multiple failure regions, we employ our algorithm for rare events with extreme deviation from the mean. We report the number of SMC samples required for model checking with respect to the deviation from the mean of the distribution(σ). This quantity can be used as a metric for future techniques that address multiple failure regions as well.

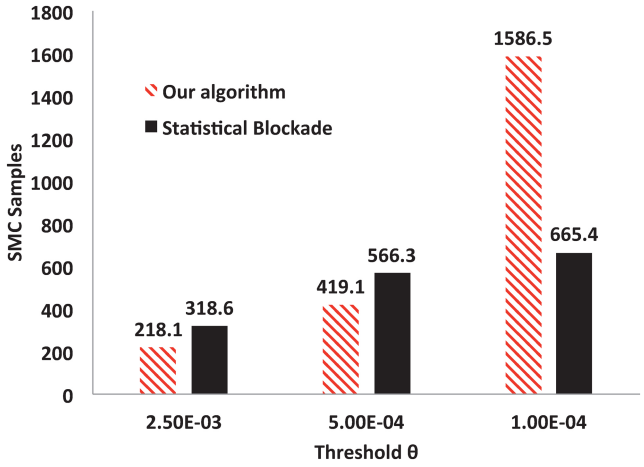

Fig. 7. Number of SMC samples used by our algorithm versus the statistical blockade algorithm for different threshold θs for an SRAM circuit with a single failure region.

The emphasis and applicability of our algorithm is on the circuits with multiple failure regions. However, to the best of our knowledge, other techniques are inapplicable to circuits with more than one failure region. We, therefore, demonstrate through an independent metric, the efficiency of our algorithm. We show that our algorithm is effective even for the rare events with increasing deviation from the mean of the distribution. We specify failure regions in terms of their deviation from the mean of the normal distribution.

We modified the SAR-ADC circuit to fail at different standard deviations from the mean of the voltage random variable. We modeled the voltage variable as a Gaussian distribution with mean = $1.25v$, standard deviation $\sigma = 0.125$, variance = $\sigma^2 = 0.125^2$, and set the delay at nominal value of 12^{ns} to simplify the analysis. We increased the number of bits in the SAR ADC circuit to increase the precision of the circuit. The SAR-ADC circuit will fail when the input voltage is $[0, 1.25 - \Delta V) \cup (1.25 + \Delta V, 2.5]$, where $\Delta V = n \times \sigma$, where n is the number of standard deviation from the mean and $n \in \{6, \ldots, 9\}$.

When the standard deviation increases, the probability of rare events decreases. As a result, our algorithm requires more training data to generate failure samples using uniform sampling and compute the distribution of failing samples to identify the failure regions. But after the training step is finished, the sample size of the SMC part remains low. Fig. 8 shows the result of the SAR-ADC circuit for different value of standard deviation. In each case, we identified two failure regions (at two tails of the distribution).

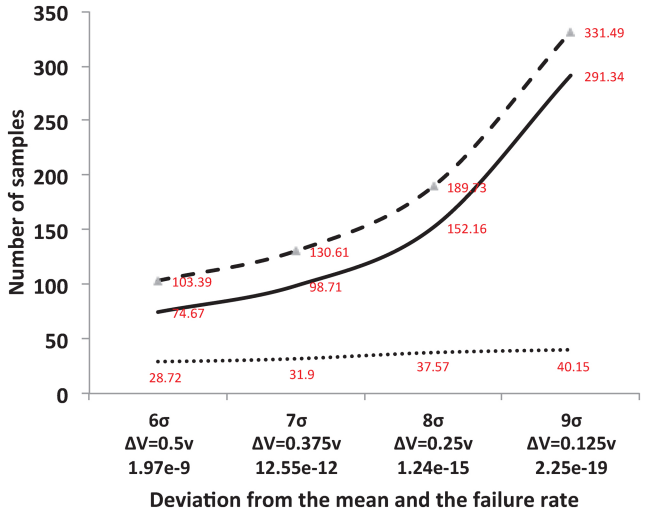

Fig. 8. Number of samples used by our algorithm for different failure regions with higher deviation from the mean of the distribution where $\theta = 10^{-4}$ for SAR-ADC circuit. As rare events become less probable and more extreme, the number of samples required by our algorithm increases almost linearly.

Fig. 8 shows the result of the SAR-ADC circuit for different values of standard deviation. As shown in Fig. 8, the number of samples required by our algorithm only increases linearly[1] for rare events with higher deviation in the far tail of the distribution. For example, when the probability of rare events decreases from 1.97×10^{-9} (corresponding to 6σ deviation) to 2.25×10^{-19} (corresponding to 9σ deviation) the number of samples only increases by 3.3 times at the most.

VIII. Conclusion

In conclusion, we have analyzed a practical, importance sampling-based approach for accelerating the SMC in rare-event scenarios for hardware circuit. We use the variational Bayes to identify the distribution of failure regions. We demonstrate, using two case studies, that our approach provides significant speedup while verifying the reliability of the hardware circuits.

References

[1] R. J. Baker, *CMOS Circuit Design, Layout, and Simulation*, 3rd ed. New York, NY, USA: Wiley, 2010.
[2] B. Barbot, S. Haddad, and C. Picaronny, "Coupling and importance sampling for statistical model checking," in *Proc. TACAS*, 2012, pp. 331–346.
[3] A. Bayrakci, A. Demir, and S. Tasiran, "Fast Monte Carlo estimation of timing yield with importance sampling and transistor-level circuit simulation," *IEEE Trans. Comput. Aided Design Integr. Circuits Syst.*, vol. 29, no. 9, pp. 1328–1341, Sep. 2010.
[4] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Germany: Springer, 2006.
[5] E. Clarke, A. Donzé, and A. Legay, "Statistical model checking of mixed-analog circuits with an application to a third order Δ: Σ modulator," in *Proc. HVC*, 2009, pp. 149–163.
[6] E. M. Clarke, J. R. Faeder, C. J. Langmead, L. A. Harris, S. K. Jha, and A. Legay, "Statistical model checking in bio lab: Applications to the automated analysis of T-cell receptor signaling pathway," in *Proc. CMSB*, 2008, pp. 231–250.
[7] E. M. Clarke and P. Zuliani, "Statistical model checking for cyber-physical systems," in *Proc. ATVA*, 2011, pp. 1–12.
[8] L. Dolecek, M. Qazi, D. Shah, and A. Chandrakasan, "Breaking the simulation barrier: SRAM evaluation through norm minimization," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2008, pp. 322–329.
[9] F. Gong, S. Basir-Kazeruni, L. Dolecek, and L. He, "A fast estimation of SRAM failure rate using probability collectives," in *Proc. ISPD*, 2012, pp. 41–48.
[10] R. Grosu and S. A. Smolka, "Monte Carlo model checking," in *Proc. TACAS*, 2005, pp. 271–286.
[11] T. C. Hesterberg, "Advances in importance sampling," Ph.D. dissertation, Dept. Statistics, Stanford Univ., Palo Alto, CA, 1988.
[12] T. Hrault, R. Lassaigne, F. Magniette, and S. Peyronnet, "Approximate probabilistic model checking," in *Verification, Model Checking, and Abstract Interpretation*, LNCS 2937, B. Steffen and G. Levi, Eds. Berlin/Heidelberg, Germany: Springer, 2004, pp. 73–84.
[13] C. Jegourel, A. Legay, and S. Sedwards, "Cross-entropy optimization of importance sampling parameters for statistical model checking," in *Proc. CAV*, 2012, pp. 327–342.
[14] S. K. Jha, E. M. Clarke, C. J. Langmead, A. Legay, A. Platzer, and P. Zuliani, "A Bayesian approach to model checking biological systems," in *Proc. CMSB*, 2009, pp. 218–234.
[15] R. Joshi, A. Pelella, A. Tuminaro, Y. Chan, and R. Kanj, "The dawn of predictive chip yield design: Along and beyond the memory lane," *IEEE Design Test Comput.*, vol. 27, no. 6, pp. 36–45, Nov./Dec. 2010.
[16] 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 *Proc. DAC*, 2006, pp. 69–72.
[17] K. Katayama, S. Hagiwara, H. Tsutsui, H. Ochi, and T. Sato, "Sequential importance sampling for low-probability and high-dimensional SRAM yield analysis," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2010, pp. 703–708.
[18] Y. Kim, M. Kim, and T.-H. Kim, "Statistical model checking for safety critical hybrid systems: An empirical evaluation," in *Hardware and Software: Verification and Testing*, LNCS 7857, A. Biere, A. Nahir, and T. Vos, Eds. Berlin/Heidelberg, Germany: Springer, 2013, pp. 162–177.
[19] J. A. Kumar. (2012). "Statistical guarantees of performance for RTL designs," Ph.D. dissertation, Dept. Elecl. Comput. Eng., Univ. Illinois Urbana-Champaign, Urbana, IL, USA [Online]. Available: http://hdl.handle.net/2142/30951
[20] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 2.0: A tool for probabilistic model checking," in *Proc. QEST*, 2004, pp. 322–323.
[21] A. Legay, B. Delahaye, and S. Bensalem, "Statistical model checking: An overview," in *Proc. RV*, 2010, pp. 122–135.
[22] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*, Cambridge, UK: Cambridge Univ. Press, 2005.
[23] M. Qazi, M. Tikekar, L. Dolecek, D. Shah, and A. Chandrakasan, "Loop flattening and spherical sampling: Highly efficient model reduction techniques for SRAM yield analysis," in *Proc. DATE*, 2010, pp. 801–806.
[24] D. P. Reijsbergen, P. T. de Boer, W. R. W. Scheinhardt, and B. R. H. M. Haverkort, "Rare event simulation for highly dependable systems with fast repairs," in *Proc. QEST*, 2010, pp. 251–260.
[25] A. Ridder, *Importance Sampling Simulations of Markovian Reliability Systems Using Cross-Entropy*, vol. 134. Norwell, MA, USA: Kluwer Academic, 2005, pp. 119–136.
[26] K. Sen, M. Viswanathan, and G. Agha, "On statistical model checking of stochastic systems," in *Proc. CAV*, 2005, pp. 266–280.
[27] K. Sen, M. Viswanathan, and G. Agha, "VESTA: A statistical model-checker and analyzer for probabilistic systems," in *Proc. QEST*, Sep. 2005, pp. 251–252.
[28] M. Shahid, "Cross entropy minimization for efficient estimation of SRAM failure rate," in *Proc. DATE*, 2012, pp. 230–235.
[29] A. Singhee and R. A. Rutenbar, "Statistical blockade: Very fast statistical simulation and modeling of rare circuit events and its application to memory design," *IEEE Trans. Comput. Aided Design*, vol. 28, no. 8, pp. 1176–1189, Aug. 2009.
[30] F. Southey, D. Schuurmans, and A. Ghodsi, "Regularized greedy importance sampling," in *Proc. NIPS*, 2002, pp. 753–760.
[31] H. Stratigopoulos, "Test metrics model for analog test development," *IEEE Trans. Comput. Aided Design Integr. Circuits Syst.*, vol. 31, no. 7, pp. 1116–1128, Jul. 2012.
[32] S. Sun, Y. Feng, C. Dong, and X. Li, "Efficient SRAM failure rate prediction via Gibbs sampling," *IEEE Trans. Comput. Aided Design Integr. Circuits Syst.*, vol. 31, no. 12, pp. 1831–1844, Dec. 2012.

[1] In Fig. 8 the x-axis is quasi-logarithmic. Although the figure looks exponential on logarithmic scale, the plot is in fact linear.

[33] A. Wald, "Sequential tests of statistical hypotheses," *Ann. Math. Stat.*, vol. 16, no. 2, pp. 117–186, 1945.
[34] J. Wang, A. Singhee, R. Rutenbar, and B. Calhoun, "Two fast methods for estimating the minimum standby supply voltage for large SRAMs," *IEEE Trans. Comput. Aided Design Integr. Circuits Syst.*, vol. 29, no. 12, pp. 1908–1920, Dec. 2010.
[35] J. Wang, S. Yaldiz, X. Li, and L. Pileggi, "SRAM parametric failure analysis," in *Proc. DAC*, 2009, pp. 496–501.
[36] Y.-C. Wang, A. Komuravelli, P. Zuliani, and E. M. Clarke, "Analog circuit verification by statistical model checking," in *Proc. 16th Asia and South Pacific Desn. Auto. Conf., (ASP-DAC)*, 2011, pp. 1–6.
[37] H. Younes, "Verification and planning for stochastic processes with asynchronous events," Ph.D. thesis, Dept. Compu. Sci., Carnegie Mellon Univ., Pittsburgh, Pennsylvania, 2005.
[38] H. Younes, M. Kwiatkowska, G. Norman, and D. Parker, "Numerical vs. statistical probabilistic model checking," *Softw. Tools Technol. Transfer*, vol. 8, no. 3, pp. 216–228, 2006.
[39] H. Younes and D. Musliner, "Probabilistic plan verification through acceptance sampling," in *Proc. AIPS Workshop Planning via Model Check.*, 2002, pp. 223–235.
[40] P. Zuliani, A. Platzer, and E. Clarke, "Bayesian statistical model checking with application to stateow/simulink verification," in *Proc. HSCC*, 2010, pp. 338–367.

Jayanand Asok Kumar (M'12) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology (IIT) Madras, Chennai, India, in 2005, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Illinois at Urbana-Champaign, Urbana, IL, USA, in 2009 and 2012, respectively.

He is currently with Qualcomm Atheros, San Jose, CA, USA. His current research interests include statistical analysis of hardware, probabilistic verification, and digital baseband design for communication systems.

Seyed Nematollah Ahmadyan (S'08) received the B.S. and M.S. degrees in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2009 and 2011, respectively. He is currently pursuing the Ph.D. degree with the University of Illinois at Urbana-Champaign, Urbana, IL, USA.

His current research interests include verification, validation, and testing of hardware systems and analog circuits.

Shobha Vasudevan (M'08) received the M.S. and Ph.D. degrees from the University of Texas, Austin, TX, USA.

She is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, USA. Her current research interests include hardware verification, validation, reliability, security, and software testing.

Dr. Vasudevan was a recipient of the 2010 National Science Foundation Career Award.