Fast Statistical Analysis of Rare Circuit Failure Events via Scaled-Sigma Sampling for High-Dimensional Variation Space

Shupeng Sun, *Student Member, IEEE*, Xin Li, *Senior Member, IEEE*, Hongzhou Liu, Kangsheng Luo, and Ben Gu

***Abstract*—Accurately estimating the rare failure rates for nanoscale circuit blocks (e.g., static random-access memory, D flip-flop, etc.) is a challenging task, especially when the variation space is high-dimensional. In this paper, we propose a novel scaled-sigma sampling (SSS) method to address this technical challenge. The key idea of SSS is to generate random samples from a distorted distribution for which the standard deviation (i.e., sigma) is scaled up. Next, the failure rate is accurately estimated from these scaled random samples by using an analytical model derived from the theorem of "soft maximum." Our proposed SSS method can simultaneously estimate the rare failure rates for multiple performances and/or specifications with only a single set of transistor-level simulations. To quantitatively assess the accuracy of SSS, we estimate the confidence interval of SSS based on bootstrap. Several circuit examples designed in nanoscale technologies demonstrate that the proposed SSS method achieves significantly better accuracy than the traditional importance sampling technique when the dimensionality of the variation space is more than a few hundred.**

***Index Terms*—Importance sampling, Monte Carlo (MC) analysis, parametric yield, process variation.**

I. Introduction

WITH aggressive technology scaling, process variation has become a growing concern for today's integrated circuits (ICs) [2]. As a complex IC may integrate numerous circuit components (e.g., millions of static random-access memory (SRAM) bit-cells in a high-performance microprocessor), a rare failure of each component may induce a not-so-rare failure for the entire system. Therefore, each component must be designed to be extremely robust to large-scale process variations. For instance, the failure rate of an SRAM bit-cell must be less than 10^{-8}–10^{-6} so that the full microprocessor system, containing millions of SRAM bit-cells, can achieve sufficiently high yield [3], [4]. For this reason, efficiently simulating the rare failure events for circuit components and accurately estimating their failure rates is an important task for the IC design community.

To address this issue, a large number of statistical algorithms and methodologies have been developed [5]–[15]. Most of these traditional methods focus on failure rate estimation for SRAM bit-cells that consist of few (e.g., 6–10) transistors. In these cases, only a small number of (e.g., 6–20) independent random variables are used to model process variations and, hence, the corresponding variation space is low-dimensional. However, several recent trends suggest us to revisit the aforementioned assumption of low-dimensional variation space.

1) *Dynamic SRAM Bit-Cell Stability Related to Peripherals:* It has been demonstrated that dynamic SRAM bit-cell stability depends not only on the bit-cell itself but also on its peripherals (e.g., other bit-cells connected to the same bit-line) [17]. Hence, a large number of transistors from multiple SRAM bit-cells and their peripherals must be considered to simulate the dynamic stability. As a result, many independent random variables must be used to model process variations, including device mismatches, for these transistors.
2) *Rare Failure Events for NonSRAM Circuits:* In addition to SRAM bit-cells, a complex IC system may contain a large number of other circuit components that must be designed with extremely low failure rates. Taking D flip-flop (DFF) as an example, it typically contains about 20 transistors [18] and the random mismatch of a single transistor is often modeled by 10–40 independent random variables at an advanced technology node. Hence, the total number of independent random variables can easily reach a few hundred for DFF analysis.

The combination of these recent trends renders a high-dimensional variation space that cannot be efficiently handled by most traditional techniques. It, in turn, poses an immediate need of developing a new CAD tool to accurately capture rare failure events in a high-dimensional variation space with low computational cost.

In this paper, we propose a novel scaled-sigma sampling (SSS) method to estimate the rare failure rate in a high-dimensional variation space. SSS is particularly developed to address the following two fundamental questions: 1) how to efficiently draw random samples from the rare failure region and 2) how to estimate the rare failure rate based on these random samples. Unlike the brute-force Monte Carlo (MC) analysis [22] that directly samples the variation space and therefore only few samples fall into the

Manuscript received July 24, 2014; revised November 12, 2014; accepted February 5, 2015. Date of publication February 20, 2015; date of current version June 16, 2015. This work was supported by the National Science Foundation under Contract CCF–1016890 and Contract CCF–1148778. This paper was presented at the International Conference on Computer-Aided Design in 2013 [1]. This paper was recommended by Associate Editor S. K. Lim.

S. Sun and X. Li are with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: shupengs@ece.cmu.edu).

H. Liu is with the Cadence Design Systems, Inc., Pittsburgh, PA 15238 USA.

K. Luo is with the Cadence Design Systems, Inc., Beijing 100080, China.

B. Gu is with the Cadence Design Systems, Inc., Austin, TX 78759 USA.

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.2015.2404895

0278-0070 © 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

failure region, SSS draws random samples from a distorted probability density function (PDF) for which the standard deviation (i.e., sigma) is scaled up. Conceptually, it is equivalent to increasing the magnitude of process variations. As a result, a large number of samples can now fall into the failure region.

Once the distorted random samples are generated, an analytical model derived from the theorem of soft maximum [23] is optimally fitted by applying maximum likelihood estimation (MLE). Next, the failure rate can be efficiently estimated from the fitted model. While most traditional techniques (e.g., importance sampling) become inefficient as the dimensionality increases, our proposed SSS approach does not suffer from such a dimensionality problem, as will be explained in the technical sections of this paper.

In addition, to make SSS of practical utility, several important implementation issues are further studied. First, a heuristic approach is developed to simultaneously estimate the failure rates for multiple performances and/or specifications by extracting the information from the same set of transistor-level simulations. Note that most traditional techniques (e.g., importance sampling) must repeatedly run different transistor-level simulations once the performance or its specification is changed, thereby resulting in extremely expensive simulation cost. From this point of view, SSS can efficiently facilitate circuit designers to explore the trade-offs between the specification setup and the failure rate for multiple performance metrics.

Second, to quantitatively assess the accuracy of SSS, we estimate the confidence interval of SSS based on bootstrap [20]. Our numerical experiments in Section VI demonstrate that the bootstrap method can accurately evaluate the estimation error for SSS, while the confidence interval estimated by the traditional minimum-norm importance sampling (MNIS) approach [11] is highly inaccurate and may misguide the circuit designer in practice.

The remainder of this paper is organized as follows. In Section II, we briefly review the background of rare failure event analysis. Next, we propose the SSS method for Gaussian random variables in Section III and then extend SSS to handle both Gaussian and uniform random variables in Section IV. Several implementation issues are discussed in detail in Section V. The efficacy of SSS is demonstrated by a number of circuit examples in Section VI. Finally, we conclude this paper in Section VII.

II. BACKGROUND

In a commercial process design kit, a set of independent random variables following Gaussian and/or uniform distributions are often defined. The device parameters (e.g., $V_{\mathrm{TH}}, T_{\mathrm{OX}}, W, L$, etc.) are modeled as functions of these independent random variables to capture process variations. The mapping from the independent random variables to the device parameters is often nonlinear. Hence, even though the independent random variables are Gaussian and/or uniform, the device parameters may follow other distributions (e.g., Poisson distribution, log-normal distribution, etc.), depending on the nonlinear mapping. Suppose that the M-dimensional vector

$$\mathbf{x} = [x_1 \; x_2 \; \cdots \; x_M]^T \tag{1}$$

contains all independent random variables with the joint PDF $f(\mathbf{x})$. The failure rate of a circuit can be mathematically represented as

$$P_f = \int_{\Omega} f(\mathbf{x}) \cdot d\mathbf{x} \tag{2}$$

where Ω denotes the failure region in the variation space, i.e., the subset of the variation space where the performance of interest does not meet the specification. Alternatively, the failure rate in (2) can be defined as

$$P_f = \int_{-\infty}^{+\infty} I(\mathbf{x}) \cdot f(\mathbf{x}) \cdot d\mathbf{x} \tag{3}$$

where $I(\mathbf{x})$ represents the indicator function

$$I(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in \Omega \\ 0 & \mathbf{x} \notin \Omega. \end{cases} \tag{4}$$

A. MC Analysis

The failure rate P_f can be estimated by a brute-force MC analysis. The key idea is to draw N random samples from $f(\mathbf{x})$, and then compute the mean of the indicator function $I(\mathbf{x})$ based on these samples

$$P_f^{\mathrm{MC}} = \frac{1}{N} \cdot \sum_{n=1}^{N} I\left[\mathbf{x}^{(n)}\right] \tag{5}$$

where $\mathbf{x}^{(n)}$ is the nth random sample. The variance of the estimator P_f^{MC} in (5) can be approximated as [21]

$$v_f^{\mathrm{MC}} = P_f^{\mathrm{MC}} \cdot \left(1 - P_f^{\mathrm{MC}}\right) / N. \tag{6}$$

When MC is applied to estimate P_f that is extremely small (e.g., 10^{-8}–10^{-6}), most random samples drawn from the PDF $f(\mathbf{x})$ do not fall into the failure region Ω. Therefore, a large number of (e.g., 10^7–10^9) samples are needed to accurately estimate P_f. Note that each MC sample is created by running an expensive transistor-level simulation. In other words, 10^7–10^9 simulations must be performed in order to collect 10^7–10^9 samples. It, in turn, implies that MC can be extremely expensive for our application of rare failure rate estimation.

B. Kernel Density Estimation

Without loss of generality, we consider a performance function $y(\mathbf{x})$, where y is the performance of interest. If y is continuous, we can estimate P_f defined in (2) by kernel density estimation (KDE). To clearly understand KDE, we first rewrite P_f in (2) as

$$P_f = \int_{D} \rho(y) \cdot dy \tag{7}$$

where $\rho(y)$ denotes the PDF of y and D denotes the failure region in the performance space, i.e., the subset of the performance space where y does not meet its specification. Unlike the failure region Ω that is often unknown in the variation space, D is explicitly specified by the circuit designer in advance. For instance, consider a simple circuit example where the delay of a sense amplifier is our performance of interest. Namely, the performance y is the delay from the input to the output of the sense amplifier. If y is larger than a predefined value $\mathrm{Delay}_{\mathrm{spec}}$, the sense amplifier is considered as "FAIL."

In this example, we can easily determine the failure region D as $(\text{Delay}_{\text{spec}}, +\infty)$.

To calculate P_f from (7), we need to know $\rho(y)$, which is, unfortunately, not known in advance. The objective of KDE is to estimate $\rho(y)$. To this end, we first draw N random samples $\{\mathbf{x}^{(n)}; n = 1, 2, \ldots, N\}$ from the PDF $f(\mathbf{x})$. After running transistor-level simulations, we obtain the corresponding performance values

$$y = \left[y^{(1)} \; y^{(2)} \; \cdots \; y^{(N)} \right]^T \tag{8}$$

where $y^{(n)}$ denotes the performance value associated with the nth random sample $\mathbf{x}^{(n)}$. Based on the idea of MLE, KDE approximates the PDF $\rho(y)$ as [19]

$$\rho^*(y) = \frac{1}{N} \cdot \sum_{n=1}^{N} \frac{1}{h} \cdot K\left[\frac{y - y^{(n)}}{h} \right] \tag{9}$$

where h is the kernel bandwidth that controls the smoothness of the estimator $\rho^*(y)$ and $K(\bullet)$ is the kernel function. In general, it is nontrivial to find the optimal h and $K(\bullet)$ without knowing the actual PDF of y, i.e., $\rho(y)$. However, we can often take advantage of our knowledge about the performance of interest to appropriately choose h and $K(\bullet)$ in practice. For example, if $\rho(y)$ approximately follows a Gaussian distribution, $K(\bullet)$ can be chosen as a Gaussian PDF with the optimal h as [19]:

$$h^{\text{OPT}} = (4/3)^{1/5} \cdot \sigma_y \cdot N^{-1/5} \tag{10}$$

where σ_y is the standard deviation of $\mathbf{y}$.

Once the PDF $\rho^*(y)$ in (9) is obtained, an estimator P_f^{KDE} can be built to estimate the failure rate P_f in (7)

$$P_f^{\text{KDE}} = \int_D \rho^*(y) \cdot dy. \tag{11}$$

The variance of the estimator P_f^{KDE} in (11) can be estimated by bootstrap [20]. The key idea of bootstrap is to construct a number of (i.e., N^{Boot}) datasets $\{\mathbf{y}_n; n = 1, 2, \ldots, N^{\text{Boot}}\}$ with the same size, where each dataset is obtained by randomly sampling the vector $\mathbf{y}$. For each dataset, (9) and (11) are used to calculate the corresponding failure rate $P_f^{\text{KDE}(n)}$. Once $\{P_f^{\text{KDE}(n)}; n = 1, 2, \ldots, N^{\text{Boot}}\}$ are available, the variance of $\{P_f^{\text{KDE}(n)}; n = 1, 2, \ldots, N^{\text{Boot}}\}$ is calculated and considered as the variance of the estimator P_f^{KDE} in (11).

Unlike the MC analysis in (5) that only relies on the information that whether the performance of interest meets the predefined specification (i.e., "PASS" or "FAIL"), KDE takes into account the continuous performance value when estimating the failure rate by (9) and (11). Intuitively speaking, KDE explores the extra information from the performance values and, therefore, can provide better accuracy than MC when the number of samples (i.e., N) is small.

Finally, we should make two important clarifications about the aforementioned KDE method. First, the performance y must be continuous. Otherwise, KDE may not be accurate because it assumes a smooth and continuous PDF for y. Second, to accurately estimate P_f by (11), we need to collect a number of random samples sitting in the failure region (i.e., D). As explained in Section II-A, generating a failure sample is extremely expensive for our application of rare failure rate estimation. Hence, directly applying KDE to rare failure rate estimation is not appropriate. Instead, KDE can be efficiently applied if and only if the failure rate is sufficiently large.

C. Importance Sampling

To reduce the computational cost, several statistical algorithms based on importance sampling have been proposed in [5], [8], [11], [13], and [15]. The key idea is to sample a distorted PDF $g(\mathbf{x})$, instead of the original PDF $f(\mathbf{x})$, so that most random samples fall into the failure region Ω. Here, $g(\mathbf{x})$ is positive for all $\mathbf{x}$ in Ω. In this case, the failure rate can be expressed as

$$P_f = \int_{-\infty}^{+\infty} \frac{I(\mathbf{x}) \cdot f(\mathbf{x})}{g(\mathbf{x})} \cdot g(\mathbf{x}) \cdot d\mathbf{x}. \tag{12}$$

If N random samples $\{\mathbf{x}^{(n)}; n = 1, 2, \ldots, N\}$ are drawn from $g(\mathbf{x})$, the failure rate in (12) can be approximated by

$$P_f^{\text{IS}} = \frac{1}{N} \cdot \sum_{n=1}^{N} f\left[\mathbf{x}^{(n)}\right] \cdot I\left[\mathbf{x}^{(n)}\right] \Big/ g\left[\mathbf{x}^{(n)}\right]. \tag{13}$$

When a finite number of samples are available, the results from (5) and (13) can be substantially different. If $g(\mathbf{x})$ is properly chosen for importance sampling, P_f^{IS} in (13) can be much more accurate than P_f^{MC} in (5). Theoretically, the optimal $g(\mathbf{x})$ leading to maximum estimation accuracy is defined as

$$g^{\text{OPT}}(\mathbf{x}) = f(\mathbf{x}) \cdot I(\mathbf{x}) / P_f. \tag{14}$$

Intuitively, if $g^{\text{OPT}}(\mathbf{x})$ is applied, P_f^{IS} in (13) becomes a constant with zero variance. Therefore, the failure rate can be accurately estimated by P_f^{IS} with very few samples.

Equation (14) implies that the optimal PDF $g^{\text{OPT}}(\mathbf{x})$ is nonzero if and only if the variable $\mathbf{x}$ sits in the failure region. We should directly sample the failure region to achieve maximum accuracy. Furthermore, $g^{\text{OPT}}(\mathbf{x})$ is proportional to the original PDF $f(\mathbf{x})$. In other words, the entire failure region should not be sampled uniformly. Instead, we should sample the high-probability failure region that is most likely to occur.

Applying importance sampling, however, is not trivial in practice. The optimal PDF $g^{\text{OPT}}(\mathbf{x})$ in (14) cannot be easily found, since the indicator function $I(\mathbf{x})$ is unknown. Most existing importance sampling methods attempt to approximate $g^{\text{OPT}}(\mathbf{x})$ by applying various heuristics. The key idea is to first search the high-probability failure region and then a distorted PDF $g(\mathbf{x})$ is constructed to directly draw random samples from such a high-probability failure region.

While the traditional importance sampling methods have been successfully applied to low-dimensional problems (e.g., 6–20 random variables), they remain ill-equipped to efficiently explore the high-dimensional variation space (e.g., 10^2–10^3 random variables) that is of great importance today. One major bottleneck lies in the high computational cost of the search algorithm, as it cannot easily find the high-probability failure region in a high-dimensional variation space. Such a computational cost issue is most pronounced, when the failure region of interest has a complicated (e.g., nonconvex or even discontinuous) shape. To the best of

our knowledge, there is no existing algorithm that can be generally applied to efficiently search a high-dimensional variation space. It, in turn, motivates us to develop a new approach in this paper to solve such high-dimensional problems.

III. SSS for Gaussian Distribution

In general, $\mathbf{x}$ may be modeled as a number of possible probability distributions such as Gaussian distribution, uniform distribution, etc. In this section, we describe SSS for Gaussian distribution. Namely, the random variable $\mathbf{x}$ follows a multivariate Gaussian distribution. In the next section, we will further extend SSS to "Gaussian-uniform" distribution where the random variables $\{x_m;\ m = 1, 2, \ldots, M\}$ are mutually independent, a subset of these random variables follow the standard Gaussian distributions and the other random variables are uniformly distributed. Such a multivariate Gaussian-uniform distribution has been used in many process design kits today.

Without loss of generality, we assume that the random variables $\{x_m;\ m = 1, 2, \ldots, M\}$ in the vector $\mathbf{x}$ are mutually independent and follow standard Gaussian distributions:

$$f(\mathbf{x}) = \prod_{m=1}^{M}\left[\frac{1}{\sqrt{2\pi}} \cdot \exp\left(-\frac{x_m^2}{2}\right)\right] = \frac{\exp\left(-\|\mathbf{x}\|_2^2/2\right)}{\left(\sqrt{2\pi}\right)^M} \tag{15}$$

where $\| \bullet \|_2$ denotes the L_2-norm of a vector. Any correlated random variables that are jointly Gaussian can be transformed to the independent random variables $\{x_m;\ m = 1, 2, \ldots, M\}$ by principal component analysis [22].

Unlike the traditional importance sampling methods that must explicitly identify the high-probability failure region, SSS takes a completely different strategy to address the following two fundamental questions: 1) how to efficiently draw random samples from the high-probability failure region and 2) how to estimate the failure rate based on these random samples. In what follows, we will derive the mathematical formulation of SSS for Gaussian distribution and highlight its novelties.

A. Statistical Sampling

For the application of rare failure rate estimation, $f(\mathbf{x})$ in (15) is often extremely small for a random sample $\mathbf{x}$ inside the failure region. It implies that the failure region is faraway from the origin $\mathbf{x} = \mathbf{0}$, as shown in Fig. 1(a). Since the failure rate is extremely small, MC cannot efficiently draw random samples from the failure region. Namely, many MC samples cannot reach the tail of $f(\mathbf{x})$.

In this paper, we apply a simple idea to address the aforementioned sampling issue. Given $f(\mathbf{x})$ in (15) for the M-dimensional random variable $\mathbf{x}$, we scale up the standard deviation of $\mathbf{x}$ by a scaling factor s ($s > 1$), yielding the following distribution:

$$g(\mathbf{x}) = \prod_{m=1}^{M}\left[\frac{1}{\sqrt{2\pi}s} \cdot \exp\left(-\frac{x_m^2}{2s^2}\right)\right] = \frac{\exp\left(-\|\mathbf{x}\|_2^2/2s^2\right)}{\left(\sqrt{2\pi}\right)^M s^M}. \tag{16}$$

Once the standard deviation of $\mathbf{x}$ is increased by a factor of s, we conceptually increase the magnitude of process variations.

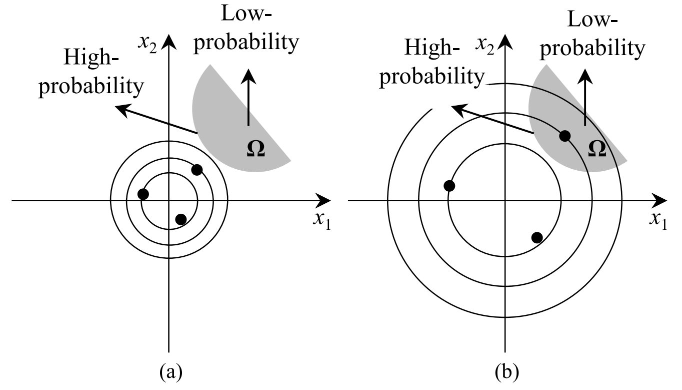

Fig. 1. Proposed SSS method for Gaussian distribution is illustrated by a 2-D example where the gray area Ω denotes the failure region and the circles represent the contour lines of the PDF. (a) Rare failure events occur at the tail of the original PDF $f(\mathbf{x})$ and the failure region is faraway from the origin $\mathbf{x} = \mathbf{0}$. (b) Scaled PDF $g(\mathbf{x})$ widely spreads over a large region and the scaled samples are likely to reach the faraway failure region.

Hence, the scaled PDF $g(\mathbf{x})$ widely spreads over a large region and the probability for a random sample to reach the faraway failure region increases, as shown in Fig. 1(b).

From an alternative viewpoint, the original random variables $\{x_m;\ m = 1, 2, \ldots, M\}$ follow the independent standard Gaussian distributions defined in (15). If we scale each of them (say, x_m) by a factor of s, the scaled random variables $\{s \cdot x_m;\ m = 1, 2, \ldots, M\}$ follow $g(\mathbf{x})$ in (16). Therefore, when sampling $g(\mathbf{x})$, we can first draw random samples from $f(\mathbf{x})$ and then scale each random sample by a factor of s. As a result, the scaled samples will move faraway from the origin $\mathbf{x} = \mathbf{0}$ and are likely to reach the failure region, as shown in Fig. 1(b).

On the other hand, it is important to note that the mean of $g(\mathbf{x})$ remains $\mathbf{0}$, which is identical to the mean of $f(\mathbf{x})$. Hence, for a given sampling location $\mathbf{x}$, the likelihood defined by $g(\mathbf{x})$ remains inversely proportional to the length of the vector $\mathbf{x}$ (i.e., $\|\mathbf{x}\|_2$). Namely, it is more (or less) likely to reach the sampling location $\mathbf{x}$, if the distance between the location $\mathbf{x}$ and the origin $\mathbf{0}$ is smaller (or larger). It, in turn, implies that the high-probability failure region associated with $f(\mathbf{x})$ remains the high-probability failure region after the PDF is scaled to $g(\mathbf{x})$, as shown in Fig. 1(a) and (b). Scaling the PDF from $f(\mathbf{x})$ to $g(\mathbf{x})$ does not change the location of the high-probability failure region. Instead, it only makes the failure region easy to sample.

Once the scaled random samples are drawn from $g(\mathbf{x})$ in (16), we need to further estimate P_f defined in (3). Since $g(\mathbf{x})$ and $f(\mathbf{x})$ are different, we cannot simply average the random samples generated by $g(\mathbf{x})$ to calculate P_f defined by $f(\mathbf{x})$. A major contribution of this paper is to derive an analytical model to accurately estimate the failure rate P_f from the scaled random samples, as will be discussed in detail in the next section.

B. Failure Rate Estimation

Given N random samples $\{\mathbf{x}^{(n)};\ n = 1, 2, \ldots, N\}$ drawn from $g(\mathbf{x})$ in (16), one straightforward way to estimate P_f is based upon the theory of importance sampling. Namely, since the random samples are generated by $g(\mathbf{x})$ that is different from $f(\mathbf{x})$, we can estimate the failure rate P_f by calculating

the average of $f(\mathbf{x}) \cdot I(\mathbf{x})/g(\mathbf{x})$, as shown by the estimator P_f^{IS} in (13).

Such a simple approach, however, does not result in an accurate failure rate, if the dimensionality of the variation space (i.e., M) is large. To understand the reason, let us calculate the variance of the importance sampling estimator P_f^{IS} in (13)

$$\mathrm{var}\left(P_f^{\mathrm{IS}}\right) = \mathrm{var}\left\{\frac{1}{N} \cdot \sum_{n=1}^{N} f\left[\mathbf{x}^{(n)}\right] \cdot I\left[\mathbf{x}^{(n)}\right] \Big/ g\left[\mathbf{x}^{(n)}\right]\right\} \quad (17)$$

where var(•) denotes the variance of a random variable. Since $\{\mathbf{x}^{(n)}; n = 1, 2, \ldots, N\}$ are independently drawn from $g(\mathbf{x})$ in (16), (17) can be rewritten as

$$\mathrm{var}\left(P_f^{\mathrm{IS}}\right) = \mathrm{var}\left[f(\mathbf{x}) \cdot I(\mathbf{x}) / g(\mathbf{x})\right] / N. \quad (18)$$

In (18), var[$f(\mathbf{x}).I(\mathbf{x})/g(\mathbf{x})$] can be expressed as [21]

$$\mathrm{var}\left[f(\mathbf{x}) \cdot I(\mathbf{x})/g(\mathbf{x})\right] = E\left\{\left[f(\mathbf{x}) \cdot I(\mathbf{x})/g(\mathbf{x})\right]^2\right\} - \left\{E\left[f(\mathbf{x}) \cdot I(\mathbf{x})/g(\mathbf{x})\right]\right\}^2 \quad (19)$$

where E(•) denotes the expected value of a random variable. We can rewrite (19) as

$$\begin{aligned}\mathrm{var}\left[f(\mathbf{x}) \cdot I(\mathbf{x})/g(\mathbf{x})\right] &= \int_{-\infty}^{+\infty} \frac{f^2(\mathbf{x})}{g^2(\mathbf{x})} \cdot I^2(\mathbf{x}) \cdot g(\mathbf{x}) \cdot d\mathbf{x} \\ &\quad - \left[\int_{-\infty}^{+\infty} \frac{f(\mathbf{x})}{g(\mathbf{x})} \cdot I(\mathbf{x}) \cdot g(\mathbf{x}) \cdot d\mathbf{x}\right]^2 \\ &= \int_{-\infty}^{+\infty} \frac{f^2(\mathbf{x})}{g(\mathbf{x})} \cdot I(\mathbf{x}) \cdot d\mathbf{x} \\ &\quad - \left[\int_{-\infty}^{+\infty} f(\mathbf{x}) \cdot I(\mathbf{x}) \cdot d\mathbf{x}\right]^2. \end{aligned} \quad (20)$$

Based on (3), (20) can be simplified as

$$\mathrm{var}\left[f(\mathbf{x}) \cdot I(\mathbf{x})/g(\mathbf{x})\right] = \int_{-\infty}^{+\infty} \frac{f^2(\mathbf{x})}{g(\mathbf{x})} \cdot I(\mathbf{x}) \cdot d\mathbf{x} - P_f^2. \quad (21)$$

Since $|I(\mathbf{x})|$ is less than or equal to 1, (21) is bounded by

$$\mathrm{var}\left[f(\mathbf{x}) \cdot I(\mathbf{x})/g(\mathbf{x})\right] \leq \int_{-\infty}^{+\infty} \frac{f^2(\mathbf{x})}{g(\mathbf{x})} \cdot d\mathbf{x} - P_f^2. \quad (22)$$

At the right-hand side of (22), P_f is the failure rate and, hence, the second term (i.e., $P_f{}^2$) is a constant. Based on (15) and (16), the first term at the right-hand side of (22) can be calculated as

$$\begin{aligned}\int_{-\infty}^{+\infty} \frac{f^2(\mathbf{x})}{g(\mathbf{x})} \cdot d\mathbf{x} &= \frac{s^M}{\left(\sqrt{2 \cdot \pi}\right)^M} \cdot \int_{-\infty}^{+\infty} \exp\left[-\frac{\left(2 \cdot s^2 - 1\right)}{2 \cdot s^2} \cdot \mathbf{x}^2\right] \cdot d\mathbf{x} \\ &= \left[s^4 \Big/ \left(2 \cdot s^2 - 1\right)\right]^{M/2}. \end{aligned} \quad (23)$$

Based on (18), (22), and (23), we have

$$\mathrm{var}\left(P_f^{\mathrm{IS}}\right) \leq \left\{\left[s^4 \Big/ \left(2 \cdot s^2 - 1\right)\right]^{M/2} - P_f^2\right\} \Big/ N. \quad (24)$$

Equation (24) shows that the upper bound of the variance of the estimator P_f^{IS} exponentially increases with the dimensionality M when s is greater than 1. It, in turn, implies that the variance of P_f^{IS} can be prohibitively large in a high-dimensional variation space. It is equivalent to saying that the estimator P_f^{IS} based on importance sampling may not be sufficiently accurate when the variation space is high-dimensional. It does not fit our need of high-dimensional failure rate estimation in this paper.

Instead of relying on the theory of importance sampling, our proposed SSS method attempts to estimate the failure rate P_f from a completely different avenue. We first take a look at the "scaled" failure rate corresponding to the scaled PDF $g(\mathbf{x})$

$$P_g = \int_{-\infty}^{+\infty} I(\mathbf{x}) \cdot g(\mathbf{x}) \cdot dx. \quad (25)$$

Our objective is to study the relation between the scaled failure rate P_g in (25) and the original failure rate P_f in (3). Toward this goal, we partition the M-dimensional variation space into a large number of identical hyper-rectangles with the same volume and the scaled failure rate P_g in (25) can be approximated as

$$P_g \approx \sum_k I\left[\mathbf{x}^{(k)}\right] \cdot g\left[\mathbf{x}^{(k)}\right] \cdot \Delta\mathbf{x} \quad (26)$$

where $\mathbf{x}^{(k)}$ represents the center of the kth hyper-rectangle and $\Delta\mathbf{x}$ denotes the volume of a hyper-rectangle. The approximation in (26) is accurate, if each hyper-rectangle is sufficiently small. Given (4), (26) can be rewritten as

$$P_g \approx \sum_{k \in \Omega} g\left[\mathbf{x}^{(k)}\right] \cdot \Delta\mathbf{x} \quad (27)$$

where $\{k;\ k \in \Omega\}$ represents the set of all hyper-rectangles that fall into the failure region. Substituting (16) into (27), we have

$$P_g \approx \frac{\Delta\mathbf{x}}{\left(\sqrt{2\pi}\right)^M s^M} \cdot \sum_{k \in \Omega} \exp\left[-\left\|\mathbf{x}^{(k)}\right\|_2^2 \Big/ 2s^2\right]. \quad (28)$$

Taking the logarithm on both sides of (28) yields

$$\log P_g \approx \log \frac{\Delta\mathbf{x}}{\left(\sqrt{2\pi}\right)^M} - M \cdot \log s + \underset{k \in \Omega}{\mathrm{lse}}\left[-\left\|\mathbf{x}^{(k)}\right\|_2^2 \Big/ 2s^2\right] \quad (29)$$

where

$$\underset{k \in \Omega}{\mathrm{lse}}\left[-\left\|\mathbf{x}^{(k)}\right\|_2^2 \Big/ 2s^2\right] = \log\left\{\sum_{k \in \Omega} \exp\left[-\left\|\mathbf{x}^{(k)}\right\|_2^2 \Big/ 2s^2\right]\right\} \quad (30)$$

stands for the log-sum-exp function. The function lse(•) in (30) is also known as the soft maximum from the mathematics [23]. It can be bounded by

$$\begin{aligned}\max_{k \in \Omega}\left[-\left\|\mathbf{x}^{(k)}\right\|_2^2 \Big/ 2s^2\right] + \log(T) &\geq \underset{k \in \Omega}{\mathrm{lse}}\left[-\left\|\mathbf{x}^{(k)}\right\|_2^2 \Big/ 2s^2\right] \\ &\geq \max_{k \in \Omega}\left[-\left\|\mathbf{x}^{(k)}\right\|_2^2 \Big/ 2s^2\right] \end{aligned} \quad (31)$$

where T denotes the total number of hyper-rectangles in Ω.

To clearly understand (31), let us consider two extreme cases. First, let us assume that all the hyper-rectangles $\{\mathbf{x}^{(k)};\ k \in \Omega\}$ have the same distance to the origin $\mathbf{0}$. In this case, the function lse(•) reaches its upper bound in (31). Second, we assume that only one hyper-rectangle in the set $\{\mathbf{x}^{(k)};\ k \in \Omega\}$ is close to the origin $\mathbf{0}$, and all other hyper-rectangles are

faraway from the origin $\mathbf{0}$. In this case, the function lse(•) reaches its lower bound in (31). For our application of rare failure event analysis, however, these two ideal cases rarely occur. Therefore, we cannot simply use the lower or upper bound in (31) to approximate the function lse(•) in (30).

In general, there exist a number of (say, T_0) dominant hyper-rectangles that are much closer to the origin $\mathbf{0}$ than other hyper-rectangles in the set $\{\mathbf{x}^{(k)}; k \in \Omega\}$. Without loss of generality, we assume that the first T_0 hyper-rectangles $\{\mathbf{x}^{(k)}; k = 1, 2, \ldots, T_0\}$ are dominant. Hence, we can approximate the function lse(•) in (30) as

$$\underset{k\in\Omega}{\text{lse}}\left[-\left\|\mathbf{x}^{(k)}\right\|_2^2\Big/2s^2\right] \approx \log\left\{\sum_{k=1}^{T_0}\exp\left[-\left\|\mathbf{x}^{(k)}\right\|_2^2\Big/2s^2\right]\right\}. \quad (32)$$

We further assume that these dominant hyper-rectangles $\{\mathbf{x}^{(k)}; k = 1, 2, \ldots, T_0\}$ have similar distances to the origin $\mathbf{0}$. Thus, (32) can be approximated by

$$\underset{k\in\Omega}{\text{lse}}\left[-\left\|\mathbf{x}^{(k)}\right\|_2^2\Big/2s^2\right] \approx \max_{k\in\Omega}\left[-\left\|\mathbf{x}^{(k)}\right\|_2^2\Big/2s^2\right] + \log(T_0). \quad (33)$$

Substituting (33) into (29) yields

$$\log P_g \approx \alpha + \beta \cdot \log s + \gamma/s^2 \quad (34)$$

where

$$\alpha = \log\left[\Delta\mathbf{x}\Big/\left(\sqrt{2\pi}\right)^M\right] + \log(T_0) \quad (35)$$

$$\beta = -M \quad (36)$$

$$\gamma = \max_{k\in\Omega}\left[-\left\|\mathbf{x}^{(k)}\right\|_2^2\Big/2\right]. \quad (37)$$

Equation (34) reveals the important relation between the scaled failure rate P_g and the scaling factor s. The approximation in (34) does not rely on any specific assumption of the failure region. It is valid, even if the failure region is nonconvex or discontinuous.

While (35)–(37) show the theoretical definition of the model coefficients α, β, and γ, finding their exact values is not trivial. For instance, the coefficient γ is determined by the hyper-rectangle that falls into the failure region Ω and is closest to the origin $\mathbf{x} = \mathbf{0}$. In practice, without knowing the failure region Ω, we cannot directly find out the value of γ. For this reason, we propose to determine the analytical model in (34) by linear regression. Namely, we first estimate the scaled failure rates $\{P_{g,q};\ q = 1, 2, \ldots, Q\}$ by setting the scaling factor s to a number of different values $\{s_q;\ q = 1, 2, \ldots, Q\}$. As long as the scaling factors $\{s_q;\ q = 1, 2, \ldots, Q\}$ are sufficiently large, the scaled failure rates $\{P_{g,q};\ q = 1, 2, \ldots, Q\}$ are large and can be accurately estimated with a small number of random samples. Next, the model coefficients α, β, and γ are fitted by linear regression for the model template in (34) based on the values of $\{(s_q, P_{g,q});\ q = 1, 2, \ldots, Q\}$. Once α, β, and γ are known, the original failure rate P_f in (3) can be predicted by extrapolation. Namely, we substitute $s = 1$ into the analytical model in (34)

$$\log P_f^{\text{SSS}} = \alpha + \gamma \quad (38)$$

where P_f^{SSS} denotes the value of P_f estimated by SSS. Applying the exponential function to both sides of (38), we have

$$P_f^{\text{SSS}} = \exp(\alpha + \gamma). \quad (39)$$

While the aforementioned discussions reveal the theoretical framework of the proposed SSS method, a number of implementation issues must be carefully studied to make SSS of practical utility. These implementation details will be further discussed in Section V.

IV. SSS for Gaussian-Uniform Distribution

In Section III, we assume that all random variables in $\mathbf{x}$ are mutually independent and follow the standard Gaussian distributions after applying principal component analysis. Such an assumption, however, may not always hold for today's nanoscale IC technologies. Namely, $\{x_m;\ m = 1, 2, \ldots, M\}$ may be modeled as other probability distributions (e.g., uniform distribution, etc.) in many practical applications.

In this section, we further extend SSS to Gaussian-uniform distribution. Without loss of generality, we rewrite $\mathbf{x}$ as

$$\mathbf{x} = \begin{bmatrix}\mathbf{x}_G \\ \mathbf{x}_U\end{bmatrix} \quad (40)$$

where the vector $\mathbf{x}_G = [x_{G,1}x_{G,2}, \ldots, x_{G,MG}]^T$ includes M_G random variables following the standard Gaussian distributions, and the vector $\mathbf{x}_U = [x_{U,1}x_{U,2}, \ldots, x_{U,MU}]^T$ includes M_U random variables following the uniform distributions, and $M = M_G + M_U$ is the total number of these random variables. Since all random variables in the vector $\mathbf{x}$ are mutually independent, we can express the joint PDF as

$$f(\mathbf{x}) = f_g(\mathbf{x}_G) \cdot f_u(\mathbf{x}_U) \quad (41)$$

$$f_g(\mathbf{x}_G) = \prod_{m=1}^{M_G}\left[\frac{1}{\sqrt{2\pi}} \cdot \exp\left(-\frac{x_{G,m}^2}{2}\right)\right] = \frac{\exp\left(-\|\mathbf{x}_G\|_2^2/2\right)}{\left(\sqrt{2\pi}\right)^{M_G}} \quad (42)$$

$$f_u(\mathbf{x}_U) = \prod_{m=1}^{M_U}\left(\frac{1}{u_m - l_m}\right) \cdot I\left(x_{U,m}\middle|l_m, u_m\right) \quad (43)$$

where l_m and u_m denote the lower and the upper bounds of the mth uniform random variable $x_{U,m}$ in the vector $\mathbf{x}_U$ respectively, and $I(x_{U,m}|l_m, u_m)$ represents the indicator function

$$I\left(x_{U,m}\middle|l_m, u_m\right) = \begin{cases}1 & l_m \le x_{U,m} \le u_m \\ 0 & \text{else}\end{cases} \quad (m = 1, 2, \ldots, M_U). \quad (44)$$

From (41) to (44), we can see that $f(\mathbf{x})$ is nonzero if and only if $\mathbf{x}$ belongs to the following set Ψ:

$$\Psi = \left\{\mathbf{x}\left|\begin{array}{c}x_{G,i} \in (-\infty, +\infty), i = 1, 2, \ldots, M_G \\ x_{U,j} \in \left[l_j, u_j\right], j = 1, 2, \ldots, M_U\end{array}\right.\right\}. \quad (45)$$

Hence, the random samples drawn from $f(\mathbf{x})$ must belong to Ψ.

Similar to Section III, we need to answer the following two fundamental questions when applying SSS to Gaussian-uniform distribution: 1) how to efficiently draw random samples from the high-probability failure region given the PDF $f(\mathbf{x})$ defined in (41) and 2) how to estimate the failure rate based on these random samples. The answers to these questions will be explained in the following sections.

A. Statistical Sampling

In this section, we focus on the failure region Ω that sits inside the set Ψ where Ψ is defined by (45). For any location $\mathbf{x}$ outside the set Ψ, the probability of reaching $\mathbf{x}$ is zero and, hence, it does not contribute to the failure rate of interest. Due to this reason, we consider the failure region Ω as a subset of Ψ. For any random sample $\mathbf{x} = [\mathbf{x}_G; \mathbf{x}_U]$ falling into the failure region Ω, the uniform PDF $f_u(\mathbf{x}_U)$ is constant while the Gaussian PDF $f_g(\mathbf{x}_G)$ can be extremely small. In other words, the failure event occurs at the tail of $f_g(\mathbf{x}_G)$. To efficiently draw random samples from the high-probability failure region, we apply the idea of SSS to the Gaussian random variable $\mathbf{x}_G$. Namely, we scale up the standard deviation of $f_g(\mathbf{x}_G)$ by a factor of $s(s > 1)$, while keeping $f_u(\mathbf{x}_U)$ unchanged. It, in turn, results in the following scaled PDF:

$$g(\mathbf{x}) = g_g(\mathbf{x}_G) \cdot f_u(\mathbf{x}_U) \tag{46}$$

$$g_g(\mathbf{x}_G) = \exp\left(-\|\mathbf{x}_G\|_2^2 / 2s^2\right) \Big/ \left[\left(\sqrt{2\pi}\right)^{M_G} \cdot s^{M_G}\right] \tag{47}$$

where $f_u(\mathbf{x}_U)$ is defined in (43). By sampling the scaled PDF $g(\mathbf{x})$ in (46), we can now reach the failure region Ω easily and a large number of random samples should sit inside Ω.

Since we assume that all random variables in $\mathbf{x}$ are mutually independent, we can draw random samples from the Gaussian and uniform distributions separately and then combine them together to form the random samples for the scaled PDF $g(\mathbf{x})$ in (46). In what follows, we will further discuss how to accurately estimate P_f in (3) based on these scaled random samples.

B. Failure Rate Estimation

To derive the analytical model for failure rate estimation of Gaussian-uniform distribution, we follow the same idea presented in Section III-B. Namely, we partition the M-dimensional variation space into a large number of small hyper-rectangles and the scaled failure rate P_g in (25) is approximated as (27). Substituting (46) into (27), we have

$$P_g \approx \sum_{k\in\Omega} g\left[\mathbf{x}^{(k)}\right] \cdot \Delta\mathbf{x} = \sum_{k\in\Omega} g_g\left[\mathbf{x}_G^{(k)}\right] \cdot f_u\left[\mathbf{x}_U^{(k)}\right] \cdot \Delta\mathbf{x} \tag{48}$$

where $\mathbf{x}^{(k)} = [\mathbf{x}_G^{(k)};\ \mathbf{x}_U^{(k)}]$ denotes the center of the kth hyper-rectangle. Substituting (43) and (47) into (48) yields

$$P_g \approx \sum_{k\in\Omega} \frac{\exp\left[-\left\|\mathbf{x}_G^{(k)}\right\|_2^2 / 2s^2\right] \cdot \Delta\mathbf{x} \cdot \prod_{m=1}^{M_U} I\left(x_{U,m}^{(k)} \middle| l_m, u_m\right)}{\left(\sqrt{2\pi}\right)^{M_G} \cdot s^{M_G} \cdot \prod_{m=1}^{M_U}(u_m - l_m)} \tag{49}$$

where $x_{U,m}^{(k)}$ represents the mth uniform random variable in $\mathbf{x}_U^{(k)} = [x_{U,1}^{(k)} x_{U,2}^{(k)}, \ldots, x_{U,MU}^{(k)}]^T$.

Since the failure region Ω is inside the set Ψ, the indicator functions $\{I(x_{U,m}^{(k)}|l_m, u_m);\ m = 1, 2, \ldots, M_U, k \in \Omega\}$ in (49) are all equal to 1. Therefore, (49) can be further simplified as

$$P_g \approx \frac{\Delta\mathbf{x}}{\left(\sqrt{2\pi}\right)^{M_G} \cdot s^{M_G} \cdot \prod_{m=1}^{M_U}(u_m - l_m)} \times \sum_{k\in\Omega} \exp\left[-\left\|\mathbf{x}_G^{(k)}\right\|_2^2 / 2s^2\right]. \tag{50}$$

By following the mathematical analysis described in Section III-B, we can approximate (50) as:

$$\log P_g \approx \alpha + \beta \cdot \log s + \gamma / s^2 \tag{51}$$

where

$$\alpha = \log\left\{\Delta\mathbf{x} \Big/ \left[\left(\sqrt{2\pi}\right)^{M_G} \cdot \prod_{m=1}^{M_U}(u_m - l_m)\right]\right\} + \log(T_0) \tag{52}$$

$$\beta = -M_G \tag{53}$$

$$\gamma = \max_{k\in\Omega}\left[-\left\|\mathbf{x}_G^{(k)}\right\|_2^2 \Big/ 2\right]. \tag{54}$$

In (52), T_0 denotes the number of dominant hyper-rectangles in the set $\{\mathbf{x}_G^{(k)};\ k \in \Omega\}$.

The analytical model in (51) for Gaussian-uniform distribution is identical to that in (34) for Gaussian distribution. Similar to the Gaussian distribution case, we first estimate the scaled failure rates $\{P_{g,q};\ q = 1, 2, \ldots, Q\}$ by setting the scaling factor s to a number of different values $\{s_q;\ q = 1, 2, \ldots, Q\}$. Next, the model coefficients α, β, and γ are fitted by linear regression based on $\{(s_q, P_{g,q});\ q = 1, 2, \ldots, Q\}$. Once α, β, and γ are known, the failure rate P_f in (3) can be predicted by using (39).

In summary, if both Gaussian and uniform distributions are used to model process variations, we only scale up the standard deviation of the Gaussian distribution without changing the uniform distribution. The failure rate P_f in (3) is then estimated by fitting an analytical model. The model fitting scheme is identical to what is described for Gaussian distribution in Section III.

Finally, it is important to mention that the SSS method presented in this section can only handle a special class of non-Gaussian distribution where all random variables in $\mathbf{x}$ are mutually independent, a subset of these random variables are Gaussian random variables and the other random variables are uniformly distributed. How to extend SSS to other non-Gaussian distributions remains an open question and will be further studied in our future research.

V. Implementation Details

To make the proposed SSS method of practical utility, a number of efficient algorithms are further studied in this section, including: 1) model fitting via MLE [22]; 2) confidence interval estimation via bootstrap [20]; and 3) simultaneous failure rate estimation for multiple performances and/or specifications. Since the aforementioned algorithms can be generally applied to both Gaussian and Gaussian-uniform distributions, we will not explicitly distinguish these two different cases when discussing the implementation details in this section.

A. Model Fitting via MLE

While the basic idea of SSS has been illustrated in Sections III and IV, we will develop a statistically optimal algorithm to implement it in this section. Our goal is to determine the MLE for the model coefficients α, β, and γ in (34) and (51). The MLE solution can be solved from an optimization problem and it is considered to be statistically optimal for a given set of random samples.

Without loss of generality, we assume that N_q scaled random samples $\{\mathbf{x}^{(n)};\ n = 1, 2, \ldots, N_q\}$ are collected for the scaling factor s_q. The scaled failure rate $P_{g,q}$ can be estimated by MC

$$P_{g,q}^{\text{MC}} = \frac{1}{N_q} \cdot \sum_{n=1}^{N_q} I\left[\mathbf{x}^{(n)}\right] \tag{55}$$

where $I(\mathbf{x})$ is the indicator function defined in (4). The variance of the estimator $P_{g,q}^{\text{MC}}$ in (55) can be approximated as [21]

$$v_{g,q}^{\text{MC}} = P_{g,q}^{\text{MC}} \cdot \left(1 - P_{g,q}^{\text{MC}}\right) \Big/ N_q. \tag{56}$$

If the number of samples N_q is sufficiently large, the estimator $P_{g,q}^{\text{MC}}$ in (55) follows a Gaussian distribution according to the central limit theorem [21]:

$$P_{g,q}^{\text{MC}} \sim \text{Gauss}\left(P_{g,q}, v_{g,q}^{\text{MC}}\right) \tag{57}$$

where $P_{g,q}$ denotes the actual failure rate corresponding to the scaling factor s_q.

If the performance of interest is continuous, we can apply KDE to estimate the scaled failure rate $P_{g,q}$, and then use bootstrap to evaluate the variance of the KDE estimator, as discussed in Section II-B. Here, we use $P_{g,q}^{\text{KDE}}$ to denote the estimated $P_{g,q}$ by KDE and $v_{g,q}^{\text{KDE}}$ to denote the variance of $P_{g,q}^{\text{KDE}}$. Similar to the estimator $P_{g,q}^{\text{MC}}$ in (57), $P_{g,q}^{\text{KDE}}$ also approximately follows a normal distribution when the number of samples N_q is sufficiently large:

$$P_{g,q}^{\text{KDE}} \sim \text{Gauss}\left(P_{g,q}, v_{g,q}^{\text{KDE}}\right). \tag{58}$$

In what follows, we will assume that the estimator $P_{g,q}^{\text{MC}}$ is applied to estimate the scaled failure rate in order to simplify our notation. If KDE is applied, we can simply replace $P_{g,q}^{\text{MC}}$ and $v_{g,q}^{\text{MC}}$ with $P_{g,q}^{\text{KDE}}$ and $v_{g,q}^{\text{KDE}}$ respectively, and the mathematical equations shown in the rest of this section should still hold.

Note that the model template in (34) and (51) are both expressed for $\log P_g$, instead of P_g. To further derive the probability distribution for $\log P_{g,q}^{\text{MC}}$, we adopt the first-order delta method from the statistics community [21]. Namely, we approximate the nonlinear function log(•) by the first-order Taylor expansion around the mean value $P_{g,q}$ of the random variable $P_{g,q}^{\text{MC}}$

$$\log P_{g,q}^{\text{MC}} \approx \log P_{g,q} + \frac{P_{g,q}^{\text{MC}} - P_{g,q}}{P_{g,q}} \approx \log P_{g,q} + \frac{P_{g,q}^{\text{MC}} - P_{g,q}}{P_{g,q}^{\text{MC}}}. \tag{59}$$

Based on the linear approximation in (59), $\log P_{g,q}^{\text{MC}}$ follows the Gaussian distribution:

$$\log P_{g,q}^{\text{MC}} \sim \text{Gauss}\left[\log P_{g,q}, v_{g,q}^{\text{MC}} \Big/ \left(P_{g,q}^{\text{MC}}\right)^2\right]. \tag{60}$$

Equation (60) is valid for all scaling factors $\{s_q;\ q = 1, 2, \ldots, Q\}$. In addition, since the scaled failure rates corresponding to different scaling factors are estimated by independent MC simulations, the estimated failure rates $\{P_{g,q}^{\text{MC}};\ q = 1, 2, \ldots, Q\}$ are mutually independent. Therefore, the Q-dimensional random variable

$$\log \mathbf{P}_g^{\text{MC}} = \left[\log P_{g,1}^{\text{MC}}\ \log P_{g,2}^{\text{MC}} \cdots\ \log P_{g,Q}^{\text{MC}}\right]^T \tag{61}$$

satisfies the following jointly Gaussian distribution:

$$\log \mathbf{P}_g^{\text{MC}} \sim \text{Gauss}\left(\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g\right) \tag{62}$$

where the mean vector $\boldsymbol{\mu}_g$ and the covariance matrix $\boldsymbol{\Sigma}_g$ are equal to

$$\boldsymbol{\mu}_g = \left[\log P_{g,1} \quad \log P_{g,2} \quad \cdots \quad \log P_{g,Q}\right]^T \tag{63}$$

$$\boldsymbol{\Sigma}_g = \text{diag}\left[\frac{v_{g,1}^{\text{MC}}}{\left(P_{g,1}^{\text{MC}}\right)^2}, \frac{v_{g,2}^{\text{MC}}}{\left(P_{g,2}^{\text{MC}}\right)^2}, \ldots, \frac{v_{g,Q}^{\text{MC}}}{\left(P_{g,Q}^{\text{MC}}\right)^2}\right] \tag{64}$$

where diag(•) denotes a diagonal matrix.

The diagonal elements of the covariance matrix $\boldsymbol{\Sigma}_g$ in (64) can be substantially different. In other words, the accuracy of $\{\log P_{g,q}^{\text{MC}};\ q = 1, 2, \ldots, Q\}$ associated with different scaling factors $\{s_q;\ q = 1, 2, \ldots, Q\}$ can be different, because the scaled failure rates $\{P_{g,q};\ q = 1, 2, \ldots, Q\}$ strongly depend on the scaling factors. In general, we can expect that if the scaling factor s_q is small, the scaled failure rate $P_{g,q}$ is small and, hence, it is difficult to accurately estimate $\log P_{g,q}$ from a small number of random samples. For this reason, instead of equally "trusting" the estimators $\{\log P_{g,q}^{\text{MC}};\ q = 1, 2, \ldots, Q\}$, we must carefully model the "confidence" for each estimator $\log P_{g,q}^{\text{MC}}$, as encoded by the covariance matrix $\boldsymbol{\Sigma}_g$ in (64). Such confidence information will be fully exploited by the MLE framework to fit a statistically optimal model.

Since the scaled failure rate $\{P_{g,q};\ q = 1, 2, \ldots, Q\}$ follows the analytical model in (34) and (51), the mean vector $\boldsymbol{\mu}_g$ in (63) can be rewritten as:

$$\boldsymbol{\mu}_g = \alpha + \beta \cdot \begin{bmatrix} \log s_1 \\ \log s_2 \\ \vdots \\ \log s_Q \end{bmatrix} + \gamma \cdot \begin{bmatrix} s_1^{-2} \\ s_2^{-2} \\ \vdots \\ s_Q^{-2} \end{bmatrix} = \mathbf{A} \cdot \boldsymbol{\Theta} \tag{65}$$

where

$$\mathbf{A} = \begin{bmatrix} 1 & \log s_1 & s_1^{-2} \\ 1 & \log s_2 & s_2^{-2} \\ \vdots & \vdots & \vdots \\ 1 & \log s_Q & s_Q^{-2} \end{bmatrix} \tag{66}$$

$$\boldsymbol{\Theta} = \left[\alpha \quad \beta \quad \gamma\right]^T. \tag{67}$$

Equation (65) implies that the mean value of the Q-dimensional random variable $\log \mathbf{P}_g^{\text{MC}}$ depends on the model coefficients α, β, and γ. Given $\{P_{g,q}^{\text{MC}};\ q = 1, 2, \ldots, Q\}$, the key idea of MLE is to find the optimal values of α, β, and γ so that the likelihood of observing $\{P_{g,q}^{\text{MC}};\ q = 1, 2, \ldots, Q\}$ is maximized.

Because the random variable $\log \mathbf{P}_g^{\text{MC}}$ follows the jointly Gaussian distribution in (62), the likelihood associated with the estimated failure rates $\{P_{g,q}^{\text{MC}}; q = 1, 2, \ldots, Q\}$ is proportional to:

$$L \sim \exp\left[-\frac{1}{2}\left(\log \mathbf{P}_g^{\text{MC}} - \boldsymbol{\mu}_g\right)^T \cdot \boldsymbol{\Sigma}_g^{-1} \cdot \left(\log \mathbf{P}_g^{\text{MC}} - \boldsymbol{\mu}_g\right)\right]. \tag{68}$$

Taking the logarithm for (68) yields

$$\log L \sim -\left(\log \mathbf{P}_g^{\text{MC}} - \boldsymbol{\mu}_g\right)^T \cdot \boldsymbol{\Sigma}_g^{-1} \cdot \left(\log \mathbf{P}_g^{\text{MC}} - \boldsymbol{\mu}_g\right). \tag{69}$$

Substituting (65) into (69), we have

$$\log L \sim -\left(\log \mathbf{P}_g^{\mathrm{MC}} - \mathbf{A}\cdot\mathbf{\Theta}\right)^T \cdot \mathbf{\Sigma}_g^{-1} \cdot \left(\log \mathbf{P}_g^{\mathrm{MC}} - \mathbf{A}\cdot\mathbf{\Theta}\right). \quad (70)$$

Note that the log-likelihood log$L$ in (70) depends on the model coefficients α, β, and γ, because the vector $\mathbf{\Theta}$ is composed of these coefficients as shown in (67). Therefore, the MLE solution of α, β, and γ can be determined by maximizing the log-likelihood function

$$\underset{\Theta}{\text{maximize}} -\left(\log \mathbf{P}_g^{\mathrm{MC}} - \mathbf{A}\cdot\mathbf{\Theta}\right)^T \cdot \mathbf{\Sigma}_g^{-1} \cdot \left(\log \mathbf{P}_g^{\mathrm{MC}} - \mathbf{A}\cdot\mathbf{\Theta}\right). \quad (71)$$

Since the covariance matrix $\mathbf{\Sigma}_g$ is positive definite, the optimization in (71) is convex. In addition, since the log-likelihood log$L$ is simply a quadratic function of $\mathbf{\Theta}$, the unconstrained optimization in (71) can be directly solved by inspecting the first-order optimality condition [23]

$$\frac{\partial}{\partial \mathbf{\Theta}}\left[-\left(\log \mathbf{P}_g^{\mathrm{MC}} - \mathbf{A}\cdot\mathbf{\Theta}\right)^T \cdot \mathbf{\Sigma}_g^{-1} \cdot \left(\log \mathbf{P}_g^{\mathrm{MC}} - \mathbf{A}\cdot\mathbf{\Theta}\right)\right] = 2\cdot\mathbf{A}^T\cdot\mathbf{\Sigma}_g^{-1}\cdot\left(\log \mathbf{P}_g^{\mathrm{MC}} - \mathbf{A}\cdot\mathbf{\Theta}\right) = \mathbf{0}. \quad (72)$$

Based on the linear equation in (72), the optimal value of $\mathbf{\Theta}$ can be determined by

$$\mathbf{\Theta} = \left(\mathbf{A}^T\cdot\mathbf{\Sigma}_g^{-1}\cdot\mathbf{A}\right)^{-1}\cdot\mathbf{A}^T\cdot\mathbf{\Sigma}_g^{-1}\cdot\log \mathbf{P}_g^{\mathrm{MC}}. \quad (73)$$

Studying (73) reveals an important fact that the estimators $\{\log P_{g,q}^{\mathrm{MC}};\ q = 1, 2, \ldots, Q\}$ are weighted by the inverse of the covariance matrix $\mathbf{\Sigma}_g$. Namely, if the variance of the estimator $\log P_{g,q}^{\mathrm{MC}}$ is large, $\log P_{g,q}^{\mathrm{MC}}$ becomes noncritical when determining the optimal values of α, β, and γ. In other words, the MLE framework has optimally weighted the importance of $\{\log P_{g,q}^{\mathrm{MC}};\ q = 1, 2, \ldots, Q\}$ based on the confidence level of these estimators. Once α, β, and γ are solved by MLE, the original failure rate P_f can be estimated by (39).

B. Confidence Interval Estimation via Bootstrap

While the MLE algorithm presented in the previous section is able to optimally estimate the model coefficients α, β, and γ, and then predict the failure rate P_f, it remains an open question how we can quantitatively assess the accuracy of our SSS method. Since SSS is based upon MC simulation, a natural way for accuracy assessment is to calculate the confidence interval of the estimator P_f^{SSS}. However, unlike the traditional estimator where a statistical metric is estimated by the average of multiple random samples and, hence, the confidence interval can be derived as a closed-form expression, our proposed estimator P_f^{SSS} is calculated by linear regression with nonlinear exponential/logarithmic transformation, as shown in Section V-A. Accurately estimating the confidence interval of P_f^{SSS} is not a trivial task.

In this paper, we apply bootstrap [20] to address the aforementioned challenge. The key idea of bootstrap is to regenerate a large number of random samples based on a statistical model without running additional transistor-level simulations. These random samples are then used to repeatedly calculate the value of P_f^{SSS} in (39) for multiple times. Based on these repeated runs, the statistics (hence, the confidence interval) of the estimator P_f^{SSS} can be accurately estimated.

In particular, we start from the estimated failure rates $\{P_{g,q}^{\mathrm{MC}};\ q = 1, 2, \ldots, Q\}$. Each estimator $P_{g,q}^{\mathrm{MC}}$ follows the Gaussian distribution in (57). The actual mean $P_{g,q}$ in (57) is unknown; however, we can approximate its value by the estimated failure rate $P_{g,q}^{\mathrm{MC}}$. Once we know the statistical distribution of $P_{g,q}^{\mathrm{MC}}$, we can resample its distribution and generate N_{RS} sampled values $\{P_{g,q}^{\mathrm{MC}(n)};\ n = 1, 2, \ldots, N_{\mathrm{RS}}\}$. This resampling idea is applied to all scaling factors $\{s_q;\ q = 1, 2, \ldots, Q\}$, thereby resulting in a large data set $\{P_{g,q}^{\mathrm{MC}(n)};\ q = 1, 2, \ldots, Q, n = 1, 2, \ldots, N_{\mathrm{RS}}\}$. Next, we repeatedly run SSS for N_{RS} times and get N_{RS} different failure rates $\{P_f^{\mathrm{SSS}(n)};\ n = 1, 2, \ldots, N_{\mathrm{RS}}\}$. The confidence interval of P_f^{SSS} can then be estimated from the statistics of these failure rate values.

C. Multiple Performances and/or Specifications

In many practical applications, circuit designers are interested in both the overall failure rate considering all performance metrics and the individual failure rates corresponding to individual performance metrics. These performance metrics may be evaluated by the same transistor-level simulation. For instance, both the read access time and the dynamic read margin of the SRAM bit-cell can be assessed from a single transient simulation [16].

To estimate the failure rates for multiple performances and/or specifications, most traditional techniques require different sets of transistor-level simulations. Taking importance sampling as an example, it must draw different random samples and, hence, run different transistor-level simulations based on different distorted PDFs, because the high-probability failure regions associated with different performances and/or specifications can be completely different. It, in turn, results in extremely expensive simulation cost.

Our proposed SSS method, however, does not suffer from such a limitation. Note that we only need to run the transistor-level simulations with a set of predefined scaling factors once. When estimating the overall failure rate, we first construct an equivalent metric (EM). If all performance metrics meet their design specifications, EM is considered as "PASS." Otherwise, EM is considered as "FAIL." It is easy to see that the failure rate of EM is equivalent to the overall failure rate of the circuit. We then calculate the scaled failure rates for EM based on transistor-level simulations. Once the scaled failure rates are known, we can further estimate the failure rate of EM by our proposed flow. Namely, we first fit the model template in (34) and (51) by MLE and then apply extrapolation to estimate the original failure rate. When analyzing each individual performance metric, we directly calculate its scaled failure rates based on the simulation results. The individual failure rate is then estimated by our proposed flow. Note that even though we analyze each performance metric individually, these performance metrics are not necessarily independent.

There are two important clarifications that should be made. First, as previously mentioned, the transistor-level simulation cost often dominates the overall computational cost for failure rate analysis. Hence, by sharing the transistor-level simulations for multiple performances and/or specifications, we can substantially reduce the total number of simulation runs and, hence, the computational cost. It is an extremely attractive feature offered by the proposed SSS method.

Algorithm 1 SSS

1. Start from a set of preselected scaling factors $\{s_q;\ q = 1, 2, \ldots, Q\}$, and H performances and/or specifications that we want to analyze.
2. For each scaling factor s_q where $q \in \{1, 2, \ldots, Q\}$, sample the scaled PDF $g(\mathbf{x})$ in (16) for Gaussian distribution or (46) for Gaussian-uniform distribution by setting $s = s_q$, and generate N_q scaled random samples by running transistor-level simulations.
3. For each $h \in \{1, 2, \ldots, H\}$
4. Calculate the scaled failure rates $\{P_{g,q}^{MC};\ q = 1, 2, \ldots, Q\}$ by (55) for the hth performance and/or specification based on the simulation results collected in Step 2.
5. For each estimator $P_{g,q}^{MC}$ where $q \in \{1, 2, \ldots, Q\}$, calculate its variance $v_{g,q}^{MC}$ by (56).
6. Form the Q-dimensional vector $\log\mathbf{P}_g^{MC}$ by taking the logarithm for the estimated failure rates $\{P_{g,q}^{MC};\ q = 1, 2, \ldots, Q\}$, as shown in (61).
7. Form the diagonal matrix $\mathbf{\Sigma}_g$ in (64) and the matrix $\mathbf{A}$ in (66).
8. Calculate the MLE solution $\mathbf{\Theta}$ based on (73), where the vector $\mathbf{\Theta}$ is composed of the model coefficients α, β, and γ as shown in (67).
9. Approximate the failure rate P_f for the hth performance and/or specification by the estimator P_f^{SSS} in (39).
10. For each estimator $P_{g,q}^{MC}$ where $q \in \{1, 2, \ldots, Q\}$, resample the Gaussian distribution in (57) for which the actual mean $P_{g,q}$ is approximated by its estimated value $P_{g,q}^{MC}$, and generate N_{RS} resampled values $\{P_{g,q}^{MC(n)};\ n = 1, 2, \ldots, N_{RS}\}$.
11. For each data set $\{P_{g,q}^{MC(n)};\ q = 1, 2, \ldots, Q\}$ where $n \in \{1, 2, \ldots, N_{RS}\}$, repeat Steps 5–9 to calculate the failure rate $P_f^{SSS(n)}$.
12. Based on the data set $\{P_f^{SSS(n)};\ n = 1, 2, \ldots, N_{RS}\}$, estimate the confidence interval of the estimator P_f^{SSS}.
13. End For

Second, our discussion in this section assumes that multiple performances can be assessed from the same transistor-level simulation. In practice, this assumption may not hold for all performance metrics. However, as long as the same transistor-level simulation can be shared by a subset of the performances of interest, our proposed SSS method is able to reduce the computational cost over other traditional approaches.

D. Summary

Algorithm 1 summarizes the simplified flow of the proposed SSS algorithm. It assumes that a set of preselected scaling factors $\{s_q;\ q = 1, 2, \ldots, Q\}$ are already given. In practice, appropriately choosing these scaling factors is a critical task due to several reasons. First, if these scaling factors are too large, the estimator P_f^{SSS} based on extrapolation in (39) would not be accurate, since the extrapolation point $s = 1$ is faraway from the selected scaling factors. Second, if the scaling factors are too small, the scaled failure rates $\{P_{g,q};\ q = 1, 2, \ldots, Q\}$ are extremely small and they cannot be accurately estimated from a small number of scaled random samples. Third, the failure rates for different performances and/or specifications can be quite different. To estimate them both accurately and efficiently, we should choose small scaling factors for the performance metrics with large failure rates, but large scaling factors for the performance metrics with small failure rates. Hence, finding an appropriate set of scaling factors for all performances and/or specifications can be extremely challenging.

In this paper, a number of evenly distributed scaling factors covering a relatively large range are empirically selected and provided as the input of Algorithm 1. For the performance metrics with large failure rates, the scaled failure rates corresponding to a number of small scaling factors can be used to fit the model template in (34) and (51). On the other hand, the scaled failure rates corresponding to a number of large scaling factors can be used for the performance metrics with small failure rates. As such, a broad range of performances and/or specifications can be accurately analyzed by our proposed SSS method. In our future research, we will study efficient methodologies to further optimize these scaling factors and improve the efficacy of SSS.

Finally, two additional clarifications should be made for Algorithm 1. First, the confidence intervals estimated for different performances and/or specifications are different. Once the simulation results are obtained from Step 2, Steps 4–12 are performed for each individual performance and/or specification, resulting in a unique confidence interval. Second, while most traditional methods cannot efficiently estimate the rare failure rate in a high-dimensional variation space, the proposed SSS algorithm does not suffer from such a dimensionality problem. None of the steps in Algorithm 1 is sensitive to the dimensionality of the variation space. As will be demonstrated in Section VI, SSS achieves superior accuracy over the traditional importance sampling method when the dimensionality of the variation space exceeds a few hundred.

VI. Numerical Examples

In this section, three circuit examples are used to demonstrate the efficacy of the proposed SSS method. For testing and comparison purposes, three different approaches are implemented: 1) brute-force MC analysis; 2) MNIS [11]; and 3) proposed SSS method. MC is used to generate the "golden" failure rates so that the accuracy of MNIS and SSS can be quantitatively evaluated. The implementation of MNIS consists of two stages, as described in [11]. In the first stage, 2000 transistor-level simulations are used to search the variation space and find the failure event that is most likely to occur. Next, importance sampling is applied with a shifted Gaussian distribution to estimate the rare failure rate. On the other hand, when implementing the proposed SSS method, six different scaling factors are empirically chosen to estimate the failure rate and 200 resampled data points are generated to estimate the confidence interval (i.e., $Q = 6$ and $N_{\mathrm{RS}} = 200$) in Algorithm 1. All numerical experiments are run on a 3.16 GHz computer with 12 GB memory.

A. DFF Delay With Gaussian Distribution

Fig. 2 shows the simplified circuit schematic for a DFF designed in a commercial 32 nm CMOS process. The DFF

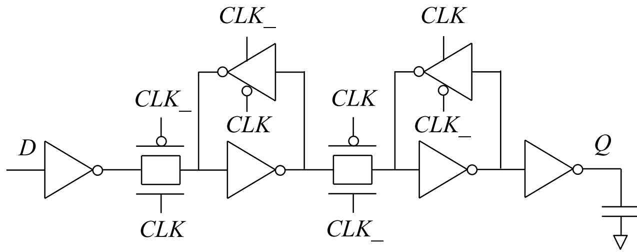

Fig. 2. Simplified schematic is shown for a DFF designed in a 32 nm CMOS process where the delay from the clock signal CLK to the output Q is considered as the performance of interest.

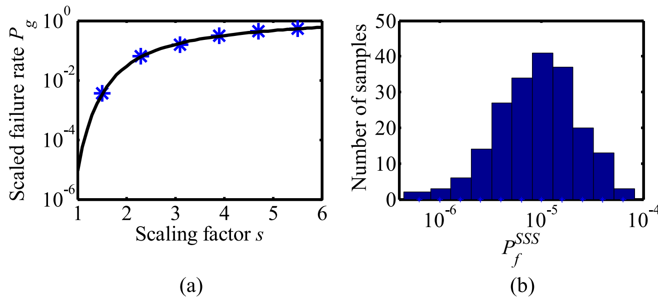

Fig. 3. (a) Scaled failure rate P_g is plotted as a function of the scaling factor s where the blue stars represent six empirically selected scaling factors and the estimated failure rates corresponding to these scaling factors. (b) Histogram is generated by 200 resampled data points to estimate the confidence interval of the estimator P_f^{SSS}.

consists of 20 transistors and the random mismatch of each transistor is modeled by 14 independent Gaussian random variables in the process design kit. Thus, there are 280 independent Gaussian random variables in total. In this example, we consider the delay from the clock signal CLK to the output Q as the performance of interest. In particular, there are two different delay metrics: 1) the delay from CLK to Q when the input data D is "ZERO" (Delay_0) and 2) the delay from CLK to Q when the input data D is "ONE" (Delay_1). Both Delay_0 and Delay_1 must be less than their predefined specifications (Spec_0 and Spec_1) so that the DFF circuit is considered as "PASS." Otherwise, the DFF circuit is considered as "FAIL." If we define the metric $D_{\text{CLK}\to Q}$ as

$$D_{\text{CLK}\to Q} = \max\left(\text{Delay}_0 - \text{Spec}_0, \text{Delay}_1 - \text{Spec}_1\right) \quad (74)$$

the aforementioned performance requirement can be equivalently defined by using the metric $D_{\text{CLK}\to Q}$. Namely, the DFF circuit is considered as "PASS" if $D_{\text{CLK}\to Q}$ is negative and "FAIL" otherwise.

We first apply MC with 10^7 random samples and the estimated failure rate is 10^{-5}. It is considered as the golden result to compare the accuracy for other statistical methods in our experiment. Next, we apply the proposed SSS method (i.e., Algorithm 1) to estimate the failure rate. Fig. 3(a) shows six empirically selected scaling factors $\{s_q;\ q = 1, 2, \ldots, 6\}$ and their corresponding scaled failure rates $\{P_{g,q}^{\text{MC}};\ q = 1, 2, \ldots, 6\}$. In total, 10^4 transistor-level simulations are used to generate these six data points $\{(s_q, P_{g,q}^{\text{MC}});\ q = 1, 2, \ldots, 6\}$. The black curve in Fig. 3(a) shows the analytical model in (34) that is optimally fitted by MLE. The DFF failure rate is then predicted by the estimator P_f^{SSS} in (39) based on the extrapolation at $s = 1$. Fig. 3(b) further shows the histogram generated by resampling, as described in Algorithm 1. The histogram is calculated from 200 resampled data points, and is used to estimate the confidence interval of the estimator P_f^{SSS}. In our experiment, we notice that the computational cost of SSS is completely dominated by the transistor-level simulations required to generate the random samples. The computational time of post-processing the sampling data in Algorithm 1 takes less than 0.2 s and, hence, is negligible.

TABLE I
FAILURE RATES AND 95% CONFIDENCE INTERVALS $[P_f^{\text{Low}}, P_f^{\text{Up}}]$ ESTIMATED BY MNIS AND SSS (GOLDEN FAILURE RATE = 10^{-5})

# of Sims	MNIS [11]			SSS (Proposed)		
	P_f^{Low}	P_f^{MNIS}	P_f^{Up}	P_f^{Low}	P_f^{SSS}	P_f^{Up}
5×10^3	0	3.3×10^{-6}	9.6×10^{-6}	1.8×10^{-6}	2.0×10^{-5}	1.7×10^{-4}
6×10^3	0	2.2×10^{-7}	5.8×10^{-7}	2.7×10^{-6}	2.5×10^{-5}	1.6×10^{-4}
7×10^3	0	1.7×10^{-7}	4.6×10^{-7}	2.1×10^{-6}	1.6×10^{-5}	1.1×10^{-4}
8×10^3	0	1.8×10^{-7}	4.3×10^{-7}	1.6×10^{-6}	1.1×10^{-5}	7.0×10^{-5}
9×10^3	0	1.6×10^{-7}	3.7×10^{-7}	1.5×10^{-6}	9.0×10^{-6}	5.4×10^{-5}
1×10^4	0	1.5×10^{-7}	3.4×10^{-7}	1.5×10^{-6}	8.9×10^{-6}	4.9×10^{-5}

Table I compares the failure rates and the 95% confidence intervals estimated by MNIS [11] and SSS based on different numbers of transistor-level simulations. Table I reveals two important observations. First, the failure rate estimated by MNIS is substantially different from the golden result (i.e., 10^{-5}). We believe that MNIS fails to identify the critical failure region that is most likely to occur, since the variation space is high-dimensional (i.e., consisting of 280 independent random variables) in this example. Therefore, the importance sampling implemented at the second stage of MNIS fails to estimate the failure rate accurately. On the other hand, the proposed SSS method successfully estimates the failure rate even if the number of transistor-level simulations is as small as 5×10^3.

Second, but more importantly, the 95% confidence interval estimated by MNIS is not accurate either. As shown in Table I, MNIS does not result in a confidence interval $[P_f^{\text{Low}}, P_f^{\text{Up}}]$ that overlaps with the golden failure rate (i.e., 10^{-5}). In other words, the confidence interval estimated by MNIS based on importance sampling is highly biased. It is one of the major limitations of MNIS and, in general, the importance sampling technique. Namely, as the confidence interval is inaccurate, it provides a wrong assessment of the accuracy and may completely misguide the user in practical applications. On the other hand, the proposed SSS method is unbiased in estimating both the failure rate and the confidence interval.

In order to further validate the confidence interval estimated by SSS, we repeatedly run Algorithm 1 for 200 times. During each run, the failure rate and the corresponding 95% confidence interval $[P_f^{\text{Low}}, P_f^{\text{Up}}]$ are estimated from 10^4 transistor-level simulations, resulting in 200 different values for both P_f^{Low} and P_f^{Up}. Fig. 4(a) and (b) shows the histograms of these 200 values for P_f^{Low} and P_f^{Up}, respectively. For only 11 cases out of 200 runs, the 95% confidence interval $[P_f^{\text{Low}}, P_f^{\text{Up}}]$ does not overlap with the golden failure rate (i.e., 10^{-5}), as shown in Fig. 4. In other words, the probability

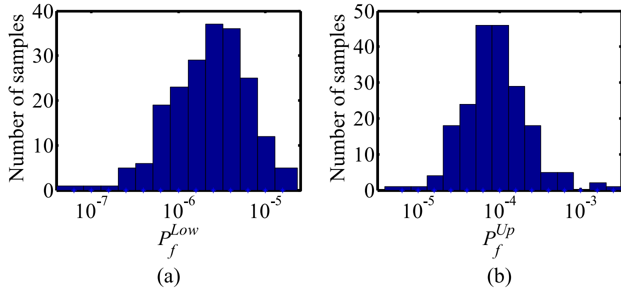

Fig. 4. Histograms of the lower and upper bounds of the 95% confidence interval $[P_f^{\text{Low}}, P_f^{\text{Up}}]$ are estimated from 200 repeated runs. (a) Lower bound P_f^{Low}. (b) Upper bound P_f^{Up}.

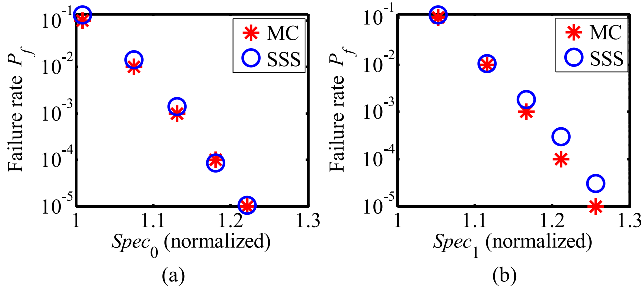

Fig. 5. Estimated failure rate P_f is plotted as a function of the performance specification for (a) Delay_0 and (b) Delay_1. The red stars represent the estimated failure rates by MC with 10^7 random samples, and they are considered as the golden results. The blue circles denote the estimated failure rates by SSS with 10^4 random samples.

for the golden failure rate to fall out of the estimated confidence interval is $11/200 \approx 5\%$. It, in turn, demonstrates that our confidence interval estimation based on bootstrap resampling (i.e., Algorithm 1) is highly accurate and it is practically more attractive than the traditional MNIS method based on importance sampling.

In addition to the failure rate associated with $D_{\text{CLK}\to Q}$ that combines two different performance metrics Delay_0 and Delay_1 as shown in (74), circuit designers are often interested in knowing the failure rates corresponding to Delay_0 and Delay_1, respectively. Furthermore, estimating the failure rates of Delay_0 and Delay_1 as functions of the specifications Spec_0 and Spec_1 is of great importance, since they provide the additional design insights that are valuable for circuit optimization. To validate the proposed SSS algorithm for multiple performances and/or specifications, we set Spec_0 and Spec_1 to several different values and estimate the corresponding failure rates. Unlike the traditional importance sampling techniques (e.g., MNIS) that require different sets of random samples to estimate the failure rates for Delay_0 and Delay_1 with different performance specifications, SSS only needs to generate a single set of random samples by running transistor-level simulations, as discussed in Section V-C.

Fig. 5(a) and (b) shows the estimated failure rate P_f as a function of the performance specification for Delay_0 and Delay_1, respectively. The red stars denote the estimated failure rates by MC with 10^7 random samples, and they are considered as the golden results. The blue circles denote the estimated failure rates by our proposed SSS method with

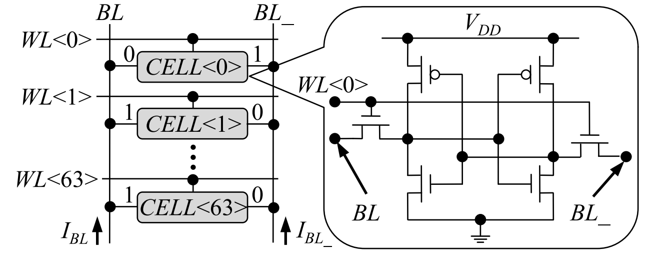

Fig. 6. Simplified schematic is shown for an SRAM column designed in a 45 nm CMOS process where the read current is defined as the difference between two bit-line currents: $I_{\text{READ}} = I_{\text{BL}} - I_{\text{BL_}}$.

10^4 random samples. Fig. 5 shows two important observations. First, the failure rate increases when the performance specification becomes tight, which is consistent with our understanding of the circuit. Second, the failure rates estimated by SSS accurately approximate the golden results. Armed with Fig. 5, we can easily determine the feasible performance specification for a given failure rate of our design. It, in turn, demonstrates an important advantage of the proposed SSS method over other traditional approaches.

B. SRAM Read Current With Gaussian Distribution

Fig. 6 shows the simplified schematic of an SRAM column designed in a 45 nm CMOS process. It consists of 64 bit-cells that are connected to two bit-lines: BL and BL_. When reading the first bit-cell CELL<0>, we first precharge both bit-lines to the supply voltage V_{DD}. Next, the word-line WL<0> is turned on and the bit-cell CELL<0> is activated. All other word-lines are turned off so that the corresponding bit-cells are de-activated. The current through CELL<0> then discharges the bit-lines and creates a voltage difference between BL and BL_.

To mimic the worst-case scenario for read operation, we store ZERO in CELL<0> and ONE in all other bit-cells. As such, the read current of CELL<0> discharges BL, while the leakage current of all other bit-cells discharges BL_, thereby slowing down the read operation. In this example, the read current $I_{\text{READ}} = I_{\text{BL}} - I_{\text{BL_}}$ is our performance of interest. It directly impacts the read delay and, therefore, is an important performance metric. If I_{READ} is greater than a predefined specification Spec, the SRAM circuit is considered as "PASS." Otherwise, it is considered as "FAIL."

To consider process variations in this experiment, we model the local V_{TH} mismatch of each transistor as an independent Gaussian random variable. Since one SRAM column consists of 64 bit-cells and each bit-cell is composed of six transistors, there are 384 transistors and, hence, 384 independent Gaussian random variables in total. It, in turn, renders a high-dimensional variation space for this SRAM example.

We first run MC with 10^8 random samples and the estimated failure rate is 1.1×10^{-6}. Table II compares the failure rates and the 95% confidence intervals estimated by MNIS [11] and SSS. Similar to the DFF example in the previous section, MNIS cannot predict the failure rate or the confidence interval accurately. On the other hand, the proposed SSS method estimates both of them accurately, even if the number of simulations is as small as 5×10^3. From this point of view, the

TABLE II
FAILURE RATES AND 95% CONFIDENCE INTERVALS $[P_f^{Low}, P_f^{Up}]$ ESTIMATED BY MNIS AND SSS (GOLDEN FAILURE RATE = 1.1×10^{-6})

# of Sims	MNIS [11]			SSS (Proposed)		
	P_f^{Low}	P_f^{MNIS}	P_f^{Up}	P_f^{Low}	P_f^{SSS}	P_f^{Up}
5×10^3	0	3.5×10^{-11}	7.6×10^{-11}	7.7×10^{-9}	1.5×10^{-6}	1.4×10^{-4}
6×10^3	0	1.8×10^{-10}	4.0×10^{-10}	6.2×10^{-8}	2.3×10^{-6}	7.6×10^{-5}
7×10^3	0	3.3×10^{-11}	6.6×10^{-11}	4.5×10^{-8}	2.2×10^{-6}	1.1×10^{-4}
8×10^3	0	5.9×10^{-9}	1.5×10^{-8}	2.6×10^{-8}	1.3×10^{-6}	2.2×10^{-5}
9×10^3	0	1.5×10^{-10}	3.6×10^{-10}	3.7×10^{-8}	1.3×10^{-6}	5.3×10^{-5}
1×10^4	2.7×10^{-11}	2.2×10^{-10}	4.2×10^{-10}	2.5×10^{-8}	9.6×10^{-7}	2.1×10^{-5}

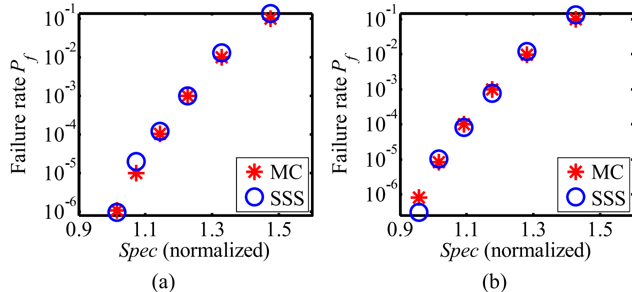

Fig. 7. Estimated failure rate P_f is plotted as a function of the performance specification for I_{READ}, where the local V_{TH} mismatch of each transistor follows (a) Gaussian distribution. (b) Gaussian-uniform distribution. The red stars represent the estimated failure rates by MC with (a) 10^8 and (b) 1.3×10^8 random samples, and they are considered as the golden results. The blue circles denote the estimated failure rates by SSS with 10^4 random samples.

SRAM example again demonstrates that our proposed SSS method is superior over the traditional MNIS approach in a high-dimensional variation space.

Finally, we apply SSS to study the relation between the failure rate P_f and its corresponding performance specification Spec. Fig. 7(a) shows the estimated failure rate P_f as a function of Spec. The red stars denote the estimated failure rates by MC with 10^8 random samples, and they are considered as the golden results. The blue circles denote the estimated failure rates by SSS with 10^4 random samples. Fig. 7(a) shows that our proposed SSS method accurately estimates the failure rates for different performance specifications from the same set of random samples. It implies that SSS is able to provide more useful guidance for memory designers and, therefore, is more attractive than other traditional approaches.

C. SRAM Read Current With Gaussian-uniform Distribution

In Sections VI-A and VI-B, the process variations are modeled as Gaussian distributions. In this section, we further demonstrate the efficacy of our proposed SSS method for Gaussian-uniform distribution.

To this end, we consider the SRAM column shown in Fig. 6 as our example. In Section VI-B, the local V_{TH} mismatch of each transistor is simply modeled by an independent Gaussian random variable. In this section, however, we use a different process design kit where the local V_{TH} mismatch of each transistor is modeled by one Gaussian random variable and one uniform random variable that are mutually independent. In total, we have 768 independent random variables (i.e., 384 Gaussian random variables and 384 uniform random variables). Similar to Section VI-B, the read current $I_{READ} = I_{BL} - I_{BL_}$ is considered as our performance of interest. If I_{READ} is greater than a predefined specification Spec, the SRAM circuit is considered as "PASS." Otherwise, it is considered as "FAIL."

TABLE III
FAILURE RATES AND 95% CONFIDENCE INTERVALS $[P_f^{Low}, P_f^{Up}]$ ESTIMATED BY SSS (GOLDEN FAILURE RATE = 7.9×10^{-7})

# of Sims	SSS (Proposed)		
	P_f^{Low}	P_f^{SSS}	P_f^{Up}
5×10^3	5.2×10^{-9}	9.1×10^{-7}	1.0×10^{-4}
6×10^3	2.6×10^{-8}	1.3×10^{-6}	5.6×10^{-5}
7×10^3	1.7×10^{-8}	1.0×10^{-6}	6.0×10^{-5}
8×10^3	6.0×10^{-9}	4.3×10^{-7}	8.8×10^{-6}
9×10^3	9.6×10^{-9}	4.1×10^{-7}	2.0×10^{-5}
1×10^4	6.5×10^{-9}	3.1×10^{-7}	8.2×10^{-6}

We first run MC with 1.3×10^8 random samples and the estimated failure rate is 7.9×10^{-7}, which is considered as the golden result. Next, we run SSS, and Table III lists the failure rates and the 95% confidence intervals estimated by SSS with different numbers of transistor-level simulations. Since MNIS [11] can handle Gaussian distribution only, no comparison between MNIS and SSS is made in this section. Table III shows that SSS estimates both the failure rate and the 95% confidence interval accurately, even if the number of simulations is as small as 5×10^3. It again demonstrates the efficacy of our proposed SSS approach in a high-dimensional variation space.

Finally, we apply SSS to study the relation between the failure rate P_f and its corresponding performance specification Spec. Fig. 7(b) shows the estimated failure rate P_f as a function of Spec. The red stars denote the estimated failure rates by MC with 1.3×10^8 random samples, and they are considered as the golden results. The blue circles denote the estimated failure rates by SSS with 10^4 random samples. As shown in Fig. 7(b), the failure rates estimated by SSS accurately approximate the golden results, implying that SSS can efficiently handle Gaussian-uniform distribution with high accuracy.

VII. CONCLUSION

In this paper, a novel statistical analysis method, referred to as SSS, is developed to accurately estimate the rare failure rates for nanoscale ICs in a high-dimensional space. The proposed SSS approach is based upon an analytical model derived from the theorem of soft maximum. It is statistically formulated as a regression modeling problem and optimally solved by MLE. With a single set of transistor-level simulations, the proposed SSS method can simultaneously estimate the rare failure rates for multiple performances and/or specifications. Such a unique feature can provide the users with valuable design insights, but it is not offered by most traditional algorithms. To quantitatively assess the accuracy of SSS, the confidence interval is estimated by bootstrap for our proposed SSS estimator. Several numerical experiments demonstrate that our proposed SSS approach achieves superior accuracy over the traditional importance sampling technique when the dimensionality of the variation space is more than a few hundred.

Finally, we would like to mention that our proposed SSS approach should be primarily applied to rare failure rate estimation in a high-dimensional variation space with a large number of (e.g., a few hundred) random variables. When the variation space is low-dimensional and there are only a few (e.g., 6–20) random variables, the traditional approaches [5]–[15] may be more accurate and, hence, preferred to SSS. Due to the page limit, we do not include a detailed comparison between SSS and the traditional methods in a low-dimensional variation space in this paper.

REFERENCES

[1] S. Sun, X. Li, H. Liu, K. Luo, and B. Gu, "Fast statistical analysis of rare circuit failure events via scaled-sigma sampling for high-dimensional variation space," in *Proc. Int. Conf. Comput.-Aided Design*, San Jose, CA, USA, 2013, pp. 478–485.

[2] B. Calhoun *et al.*, "Digital circuit design challenges and opportunities in the era of nanoscale CMOS," *Proc. IEEE*, vol. 96, no. 2, pp. 343–365, Feb. 2008.

[3] A. Bhavnagarwala, X. Tang, and J. Meindl, "The impact of intrinsic device fluctuations on CMOS SRAM cell stability," *IEEE J. Solid-State Circuits*, vol. 36, no. 4, pp. 658–665, Apr. 2001.

[4] R. Heald and P. Wang, "Variability in sub-100 nm SRAM designs," in *Proc. Int. Conf. Comput.-Aided Design*, San Jose, CA, USA, 2004, pp. 347–352.

[5] 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. Design Autom. Conf.*, San Francisco, CA, USA, 2006, pp. 69–72.

[6] C. Gu and J. Roychowdhury, "An efficient, fully nonlinear, variability aware non-Monte-Carlo yield estimation procedure with applications to SRAM cells and ring oscillators," in *Proc. Asia South Pac. Design Autom. Conf.*, Seoul, Korea, 2008, pp. 754–761.

[7] M. Abu-Rahma *et al.*, "A methodology for statistical estimation of read access yield in SRAMs," in *Proc. Design Autom. Conf.*, Anaheim, CA, USA, 2008, pp. 205–210.

[8] L. Dolecek, M. Qazi, D. Shah, and A. Chandrakasan, "Breaking the simulation barrier: SRAM evaluation through norm minimization," in *Proc. Int. Conf. Comput.-Aided Design*, San Jose, CA, USA, 2008, pp. 322–329.

[9] J. Wang, S. Yaldiz, X. Li, and L. Pileggi, "SRAM parametric failure analysis," in *Proc. Design Autom. Conf.*, San Francisco, CA, USA, 2009, pp. 496–501.

[10] A. Singhee and R. 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.

[11] 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. Design Autom. Test Europe Conf. Exhibit.*, Dresden, Germany, 2010, pp. 801–806.

[12] R. Fonseca *et al.*, "A statistical simulation method for reliability analysis of SRAM core-cells," in *Proc. Design Autom. Conf.*, Anaheim, CA, USA, 2010, pp. 853–856.

[13] 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. Int. Conf. Comput.-Aided Design*, San Jose, CA, USA, 2010, pp. 703–708.

[14] R. Kanj, R. Joshi, Z. Li, J. Hayes, and S. Nassif, "Yield estimation via multi-cones," in *Proc. Design Autom. Conf.*, San Francisco, CA, USA, 2012, pp. 1107–1112.

[15] S. Sun, Y. Feng, C. Dong, and X. Li, "Efficient SRAM failure rate prediction via Gibbs sampling," *IEEE Trans. Comput.-Aided Design*, vol. 31, no. 12, pp. 1831–1844, Dec. 2012.

[16] W. Dehaene, S. Cosemans, A. Vignon, F. Catthoor, and P. Geens, "Embedded SRAM design in deep deep submicron technologies," in *Proc. 23rd Eur. Solid State Circuits Conf. (ESSCIRC)*, Munich, Germany, 2007, pp. 384–391.

[17] R. Joshi *et al.*, "Design of sub-90 nm low-power and variation tolerant PD/SOI SRAM cell based on dynamic stability metrics," *IEEE J. Solid-State Circuits*, vol. 44, no. 3, pp. 965–976, Mar. 2009.

[18] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*. Boston, MA, USA: Addison-Wesley, 2010.

[19] B. Silverman, *Density Estimation for Statistics and Data Analysis*. London, U.K.: Chapman and Hall, 1986.

[20] B. Efron and R. Tibshirnani, *An Introduction to the Bootstrap*. New York, NY, USA: Chapman and Hall, 1993.

[21] A. Papoulis and S. Pillai, *Probability, Random Variables and Stochastic Process*. Boston, MA, USA: McGraw-Hill, 2002.

[22] C. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2007.

[23] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2009.

Shupeng Sun (S'11) received the B.S. degree in automation from Tsinghua University, Beijing, China, in 2010. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA.

His current research interests include memory yield verification and performance modeling.

Xin Li (S'01–M'06–SM'10) received the B.S. and M.S. degrees from Fudan University, Shanghai, China, in 1998 and 2001, respectively, both in electronics engineering, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2005.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Carnegie Mellon University. His current research interests include computer-aided design, neural signal processing, and power system analysis and design.

Dr. Li was the recipient of the IEEE Donald O. Pederson Best Paper Award in 2013, the National Science Foundation CAREER Award in 2012, the Design Automation Conference Best Paper Award in 2010, and two International Conference on Computer-Aided Design Best Paper Awards in 2004 and 2011.

Hongzhou Liu received the B.S. degree from Peking University, Beijing, China, the M.S. degree from Tsinghua University, Beijing, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 1997, 1999, and 2004, respectively.

He is currently with Cadence Design Systems, Inc., Pittsburgh, PA, USA, involved in the areas of analog design automation and statistical variation aware design methodology. He holds ten U.S. patents related to analog and statistical design automation.

Mr. Liu was the recipient of the Best Paper Award from Design Automation Conference in 2002.

Kangsheng Luo received the B.S. degree from Xidian University, Xi'an, China, and the Ph.D. degree from Tsinghua University, Beijing, China, in 2004 and 2011, respectively, both in electrical engineering.

He is currently an Engineering Manager with Cadence Design Systems, Inc., Beijing. His current research interests include worst case corners, high-yield estimation, and variation aware design.

Ben Gu received the B.S. degree from Shanghai Jiaotong University, Shanghai, China, and the M.S. degree from Pennsylvania State University, University Park, PA, USA, in 1996 and 2004, respectively, both in electrical engineering.

From 2004 to 2010, he was with Freescale Semiconductors, Austin, TX, USA, developing an in-house SPICE circuit simulator, Mica. From 2010 to 2012, he was with Magma Design Automation, where he was the Architect of the Finesim SPICE circuit simulator. He has been with Cadence Design Systems, Inc., Austin, since 2012, where he is currently an Engineering Director responsible for research and development of the Voltus IC power grid simulator.