Fast Statistical Analysis of Rare Circuit Failure Events via Subset Simulation in High-Dimensional Variation Space

Shupeng Sun and Xin Li
Electrical & Computer Engineering Department, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213 USA
{shupengs, xinli}@ece.cmu.edu

ABSTRACT

In this paper, we propose a novel subset simulation (SUS) technique to efficiently estimate the rare failure rate for nanoscale circuit blocks (e.g., SRAM, DFF, etc.) in high-dimensional variation space. The key idea of SUS is to express the rare failure probability of a given circuit as the product of several large conditional probabilities by introducing a number of intermediate failure events. These conditional probabilities can be efficiently estimated with a set of Markov chain Monte Carlo samples generated by a modified Metropolis algorithm, and then used to calculate the rare failure rate of the circuit. To quantitatively assess the accuracy of SUS, a statistical methodology is further proposed to accurately estimate the confidence interval of SUS based on the theory of Markov chain Monte Carlo simulation. Our experimental results of two nanoscale circuit examples demonstrate that SUS achieves significantly enhanced accuracy over other traditional techniques when the dimensionality of the variation space is more than a few hundred.

1. INTRODUCTION

As deep sub-micron technology advances, the ever increasing process variation has become a growing concern for today's integrated circuits (ICs) [1]. A complex IC, containing numerous circuit components (e.g., millions of SRAM bit-cells integrated in an advanced microprocessor), is required to meet the design specification not only at the nominal process corner, but also under large-scale process variations. To achieve sufficiently high yield, each component must be designed to be extremely robust. For instance, the failure rate of an SRAM bit-cell must be smaller than 10^{-8}~10^{-6} for a typical SRAM design [2]-[3]. For this reason, efficiently analyzing the rare failure events for individual circuit components is an important task for the IC design community.

To address this issue, a large number of methods have been proposed in the literature [4]-[14]. Most of these traditional methods focus on failure rate estimation for SRAM bit-cells, and only a small number of (e.g., 10~50) independent random variables are used to model process variations. Hence, the corresponding variation space is low-dimensional. It has been demonstrated in [15] that estimating the rare failure probability in a high-dimensional space (e.g., hundreds of independent random variables to model the device-level variations for a DFF) becomes increasingly important. Unfortunately, such a high-dimensional problem cannot be efficiently handled by most traditional methods [4]-[14]. To address this challenge, the scaled-sigma sampling (SSS) method has been proposed in [15]. Though no dimensionality curse is observed for SSS, the estimated failure rate may not be highly accurate since SSS is based on extrapolation. In a word, how to efficiently and accurately estimate the rare failure probability in a high-dimensional variation space remains an open question.

In this paper, a novel subset simulation (SUS) technique is proposed to address this technical challenge. The key idea of SUS, borrowed from the statistics community [16]-[18], is to express the rare failure probability as the product of several large conditional probabilities by introducing a number of intermediate failure events. As such, the original problem of rare failure probability estimation is cast to an equivalent problem of estimating a sequence of conditional probabilities via multiple phases. Since these conditional probabilities are relatively large, they are substantially easier to estimate than the original rare failure rate.

When implementing the proposed SUS method, it is difficult, if not impossible, to directly draw random samples from the conditional probability density functions (PDFs) and estimate the conditional probabilities, since these conditional PDFs are unknown in advance. To address this issue, a modified Metropolis (MM) algorithm is adopted from [16] to generate random samples by constructing a number of Markov chains. The conditional probabilities of interest are then estimated from these random samples. Unlike most traditional techniques [4]-[14] that suffer from the dimensionality issue, SUS can be efficiently applied to high-dimensional problems, as will be explained in the technical sections of this paper.

In addition, a statistical methodology is further developed to accurately estimate the confidence interval of SUS based on the theory of Markov chain Monte Carlo (MCMC) simulation [19]. As will be demonstrated by the numerical experiments in Section 5, SUS can achieve substantially better accuracy than the traditional methods when hundreds of independent random variables are used to model process variations.

The remainder of this paper is organized as follows. We briefly review the background of rare failure probability estimation in Section 2, and then describe the proposed SUS approach in Section 3. Several implementation issues, including (i) how to define the intermediate failure events, and (ii) how to quantitatively assess the accuracy of the proposed SUS estimator, are discussed in Section 4. Two circuit examples are presented to demonstrate the efficacy of SUS in Section 5. Finally, we conclude in Section 6.

2. BACKGROUND

Suppose that the vector

$$\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_M] \tag{1}$$

is an M-dimensional random variable modeling the device-level process variations and its joint PDF is $p(\mathbf{x})$. Without loss of generality, we further assume that the random variables $\{x_m;\ m = 1, 2, \cdots, M\}$ in the vector $\mathbf{x}$ are mutually independent [4]-[15]

$$p(\mathbf{x}) = \prod_{m=1}^{M} p_m(x_m), \tag{2}$$

where $p_m(x_m)$ is the 1-D PDF for x_m.

Consider a circuit-level performance of interest (PoI) and it

can be expressed as a function of the device-level variations: $y(\mathbf{x})$. The failure rate P_F can be expressed as

$$P_F = \Pr\left[y(\mathbf{x}) \in F\right], \tag{3}$$

where Pr(•) denotes the probability that an event occurs, and F stands for the failure region in the performance space (i.e., the subset of the performance space where the PoI does not meet the specification).

In theory, the failure rate P_F can be estimated by brute-force Monte Carlo (MC) analysis. The key idea of MC is to draw N random samples $\{\mathbf{x}^{(n)}; n = 1, 2, \cdots, N\}$ from $p(\mathbf{x})$, and perform transistor-level simulations to evaluate their PoI values $\{y(\mathbf{x}^{(n)}); n = 1, 2, \cdots, N\}$. The failure rate P_F is then estimated by

$$P_F^{MC} = \frac{1}{N} \cdot \sum_{n=1}^{N} I_F\left[y\left(\mathbf{x}^{(n)}\right)\right], \tag{4}$$

where $I_F[y(\mathbf{x})]$ represents the indicator function

$$I_F\left[y(\mathbf{x})\right] = \begin{cases} 1 & y(\mathbf{x}) \in F \\ 0 & y(\mathbf{x}) \notin F \end{cases}. \tag{5}$$

When MC is applied, around $1/P_F$ random samples are required on average to obtain a single random sample whose PoI falls into the failure region F. Evaluating the PoI for each random sample requires an expensive transistor-level simulation and, hence, a large number of (e.g., 10^7~10^9) simulations are needed by MC when the failure rate P_F is extremely small (e.g., 10^{-8}~10^{-6}). From this point of view, MC cannot be efficiently applied to the problem of rare failure rate estimation in practice.

3. SUBSET SIMULATION

Instead of directly estimating the rare failure probability, SUS expresses the rare failure probability as the product of several large conditional probabilities by introducing several intermediate failure events. Without loss of generality, we assume that the performance function $y(\mathbf{x})$ is continuous. Define K intermediate failure events $\{F_k; k = 1, 2, \cdots, K\}$ as the subsets of the performance space

$$F = F_K \subset F_{K-1} \subset \cdots \subset F_2 \subset F_1. \tag{6}$$

Based on (6), we can express P_F in (3) as

$$P_F = \Pr\left[y(\mathbf{x}) \in F\right] = \Pr\left[y(\mathbf{x}) \in F_K\right] = \Pr\left[y(\mathbf{x}) \in F_K, y(\mathbf{x}) \in F_{K-1}\right]. \tag{7}$$

In (7), if $y(\mathbf{x})$ belongs to F_K, it will undoubtedly belong to F_{K-1} because F_K is a subset of F_{K-1} as shown in (6). Eq. (7) can be re-written as [20]

$$P_F = \Pr\left[y(\mathbf{x}) \in F_K \middle| y(\mathbf{x}) \in F_{K-1}\right] \cdot \Pr\left[y(\mathbf{x}) \in F_{K-1}\right]. \tag{8}$$

Here, the conditional probability $\Pr[y(\mathbf{x}) \in F_K \mid y(\mathbf{x}) \in F_{K-1}]$ represents the probability of $y(\mathbf{x}) \in F_K$ given $y(\mathbf{x}) \in F_{K-1}$. Similarly, we can express $\Pr[y(\mathbf{x}) \in F_{K-1}]$ as

$$\Pr\left[y(\mathbf{x}) \in F_{K-1}\right] = \Pr\left[y(\mathbf{x}) \in F_{K-1} \middle| y(\mathbf{x}) \in F_{K-2}\right] \cdot \Pr\left[y(\mathbf{x}) \in F_{K-2}\right]. \tag{9}$$

From (6), (8) and (9), we can easily derive

$$P_F = \Pr\left[y(\mathbf{x}) \in F_1\right] \cdot \prod_{k=2}^{K} \Pr\left[y(\mathbf{x}) \in F_k \middle| y(\mathbf{x}) \in F_{k-1}\right] = \prod_{k=1}^{K} P_k, \tag{10}$$

where

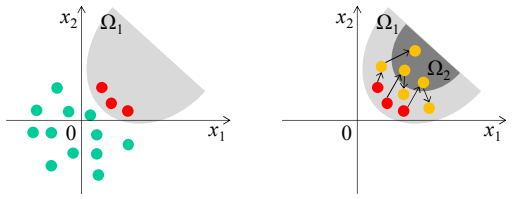
$$P_1 = \Pr\left[y(\mathbf{x}) \in F_1\right] \tag{11}$$

$$P_k = \Pr\left[y(\mathbf{x}) \in F_k \middle| y(\mathbf{x}) \in F_{k-1}\right] \quad (k = 2, 3, \cdots, K). \tag{12}$$

If the failure events $\{F_k; k = 1, 2, \cdots, K\}$ are properly chosen, all the probabilities $\{P_k; k = 1, 2, \cdots, K\}$ are large and can be efficiently estimated. Once $\{P_k; k = 1, 2, \cdots, K\}$ are known, the rare failure probability P_F can be easily calculated by (10). In what follows, we will first use a simple 2-D example to intuitively illustrate the basic flow of SUS in Section 3.1, and then further generalize SUS to high dimension in Section 3.2.

3.1 A Simple 2-D Example

Figure 1 shows a simple 2-D example with two random variables $\mathbf{x} = [x_1\ x_2]$ to model the device-level process variations. In Figure 1, the failure regions Ω_1 and Ω_2 denote the subsets of the variation space where the PoI $y(\mathbf{x})$ belongs to F_1 and F_2 respectively, i.e., $\Omega_1 = \{\mathbf{x} \mid y(\mathbf{x}) \in F_1\}$ and $\Omega_2 = \{\mathbf{x} \mid y(\mathbf{x}) \in F_2\}$. Note that Ω_1 and Ω_2 are depicted for illustration purposes in this example. In practice, we do not need to explicitly know Ω_1 and Ω_2. Instead, we can run a transistor-level simulation to determine whether a sample $\mathbf{x}$ belongs to Ω_1 and/or Ω_2.



(a) (b)

Figure 1. A two-dimensional example is used to illustrate the procedure of probability estimation via multiple phases by using SUS: (a) generating MC samples and estimating P_1 in the 1st phase, and (b) generating MCMC samples and estimating P_2 in the 2nd phase.

Our objective is to estimate the probabilities $\{P_k; k = 1, 2, \cdots, K\}$ via multiple phases. Starting from the 1st phase, we simply draw L_1 independent random samples $\{\mathbf{x}^{(1,l)}; l = 1, 2, \cdots, L_1\}$ from the PDF $p(\mathbf{x})$ to estimate P_1. Here, the superscript "1" of the symbol $\mathbf{x}^{(1,l)}$ refers to the 1st phase. Among these L_1 samples, we identify a subset of samples $\{\mathbf{x}_F^{(1,t)}; t = 1, 2, \cdots, T_1\}$ that fall into Ω_1, where T_1 denotes the total number of the samples in this subset. As shown in Figure 1 (a), the red points represent the samples that belong to Ω_1 and the green points represent the samples that are out of Ω_1. In this case, P_1 can be estimated as

$$P_1^{SUS} = \frac{1}{L_1} \cdot \sum_{l=1}^{L_1} I_{F_1}\left[y\left(\mathbf{x}^{(1,l)}\right)\right] = \frac{T_1}{L_1}, \tag{13}$$

where P_1^{SUS} denotes the estimated value of P_1, and $I_{F1}[y(\mathbf{x})]$ represents the indicator function

$$I_{F_1}\left[y(\mathbf{x})\right] = \begin{cases} 1 & y(\mathbf{x}) \in F_1 \\ 0 & y(\mathbf{x}) \notin F_1 \end{cases}. \tag{14}$$

If P_1 is large, it can be accurately estimated by the aforementioned MC method with a small number of random samples (e.g., L_1 is around 10^2~10^3).

Next, in the 2nd phase, we need to estimate the conditional probability $P_2 = \Pr[y(\mathbf{x}) \in F_2 \mid y(\mathbf{x}) \in F_1]$ which can be re-written as $\Pr(\mathbf{x} \in \Omega_2 \mid \mathbf{x} \in \Omega_1)$. Towards this goal, one simple idea is to directly draw random samples from the conditional PDF $p(\mathbf{x} \mid \mathbf{x} \in \Omega_1)$ and then compute the mean of the indicator function $I_{F2}[y(\mathbf{x})]$

$$I_{F_2}\left[y(\mathbf{x})\right] = \begin{cases} 1 & y(\mathbf{x}) \in F_2 \\ 0 & y(\mathbf{x}) \notin F_2 \end{cases}. \tag{15}$$

This approach, however, is practically infeasible since $p(\mathbf{x} \mid \mathbf{x} \in \Omega_1)$ is unknown in advance. To address this challenge, we apply a modified Metropolis (MM) algorithm [16] to generate a set of random samples that follow the conditional PDF $p(\mathbf{x} \mid \mathbf{x} \in \Omega_1)$.

MM is a Markov chain Monte Carlo (MCMC) technique [19].

Starting from each of the samples $\{\mathbf{x}_F^{(1,t)}; t = 1, 2, \cdots, T_1\}$ that fall into Ω_1 in the 1st phase, MM generates a sequence of samples that form a Markov chain. In other words, there are T_1 independently generated Markov chains in total and $\mathbf{x}_F^{(1,t)}$ is the 1st sample of the t-th Markov chain. To clearly explain the MM algorithm, we define the symbol $\mathbf{x}^{(2,t,1)} = \mathbf{x}_F^{(1,t)}$, where $t \in \{1, 2, \cdots, T_1\}$. The superscripts "2" and "1" of $\mathbf{x}^{(2,t,1)}$ refer to the 2nd phase and the 1st sample of the Markov chain respectively.

For our 2-D example, we start from $\mathbf{x}^{(2,1,1)} = [x_1^{(2,1,1)}\ x_2^{(2,1,1)}]$ to form the 1st Markov chain. To generate the 2nd sample $\mathbf{x}^{(2,1,2)}$ from $\mathbf{x}^{(2,1,1)}$, we first randomly sample a new value x_1^{NEW} from a 1-D transition PDF $q_1[x_1^{NEW} \mid x_1^{(2,1,1)}]$ that must satisfy the following condition [16]

$$q_1\left[x_1^{NEW}\middle|x_1^{(2,1,1)}\right] = q_1\left[x_1^{(2,1,1)}\middle|x_1^{NEW}\right]. \quad (16)$$

There are many possible ways to define $q_1[x_1^{NEW} \mid x_1^{(2,1,1)}]$ in (16) [16]. For example, a 1-D Normal PDF can be used

$$q_1\left[x_1^{NEW}\middle|x_1^{(2,1,1)}\right] = \frac{1}{\sqrt{2\pi}\cdot\sigma_1}\cdot\exp\left\{-\left[x_1^{NEW} - x_1^{(2,1,1)}\right]^2 \Big/ 2\cdot\sigma_1^2\right\}, \quad (17)$$

where $x_1^{(2,1,1)}$ and σ_1 are the mean and standard deviation of the distribution respectively. Here, σ_1 is a parameter that should be empirically chosen by following the heuristics in Section 3.2.

Next, we compute the ratio

$$r = p_1\left(x_1^{NEW}\right) \Big/ p_1\left(x_1^{(2,1,1)}\right), \quad (18)$$

where $p_1(x_1)$ is the original PDF of the random variable x_1 shown in (2). A random sample u is then drawn from a 1-D uniform distribution with the following PDF

$$f(u) = \begin{cases} 1 & 0 \le u \le 1 \\ 0 & \text{Otherwise} \end{cases}, \quad (19)$$

and the value of $x_1^{(2,1,2)}$ is set as

$$x_1^{(2,1,2)} = \begin{cases} x_1^{NEW} & u \le \min(1, r) \\ x_1^{(2,1,1)} & u > \min(1, r) \end{cases}. \quad (20)$$

A similar procedure is applied to generate $x_2^{(2,1,2)}$. Once $x_1^{(2,1,2)}$ and $x_2^{(2,1,2)}$ are determined, we form a candidate $\mathbf{x}^{NEW} = [x_1^{(2,1,2)}\ x_2^{(2,1,2)}]$ and use it to create the sample $\mathbf{x}^{(2,1,2)}$

$$\mathbf{x}^{(2,1,2)} = \begin{cases} \mathbf{x}^{NEW} & \mathbf{x}^{NEW} \in \Omega_1 \\ \mathbf{x}^{(2,1,1)} & \mathbf{x}^{NEW} \notin \Omega_1 \end{cases}. \quad (21)$$

By repeating the aforementioned steps, we can create other samples to complete the Markov chain $\{\mathbf{x}^{(2,1,l)}; l = 1, 2, \cdots, L_2\}$, where L_2 denotes the length of the Markov chain in the 2nd phase. In addition, all other Markov chains can be similarly formed. Since there are T_1 Markov chains and each Markov chain contains L_2 samples, the total number of the MCMC samples is $T_1 \cdot L_2$ for the 2nd phase. Figure 1 (b) shows the sampling results for our 2-D example. In Figure 1 (b), the red points represent the initial samples $\{\mathbf{x}^{(2,t,1)}; t = 1, 2, \cdots, T_1\}$ of the Markov chains and they are obtained from the 1st phase. The yellow points represent the MCMC samples created via the MM algorithm in the 2nd phase.

It has been proved in [16] that if the initial sample $\mathbf{x}^{(2,t,1)}$ follows the distribution $p(\mathbf{x} \mid \mathbf{x} \in \Omega_1)$, all the samples $\{\mathbf{x}^{(2,t,l)}; l = 1, 2, \cdots, L_2\}$ in the Markov chain follow $p(\mathbf{x} \mid \mathbf{x} \in \Omega_1)$. In our 2-D example, since the initial samples $\{\mathbf{x}^{(2,t,1)}; t = 1, 2, \cdots, T_1\}$ are randomly drawn from $p(\mathbf{x})$ and belong to Ω_1, they follow the distribution $p(\mathbf{x} \mid \mathbf{x} \in \Omega_1)$. Hence, all the MCMC samples $\{\mathbf{x}^{(2,t,l)}; t = 1, 2, \cdots, T_1, l = 1, 2, \cdots, L_2\}$ in Figure 1 (b) follow $p(\mathbf{x} \mid \mathbf{x} \in \Omega_1)$. In other words, we have successfully generated a number of random samples that follow our desired distribution for the 2nd phase.

Among all the MCMC samples $\{\mathbf{x}^{(2,t,l)}; t = 1, 2, \cdots, T_1, l = 1, 2, \cdots, L_2\}$, we further identify a subset of samples $\{\mathbf{x}_F^{(2,t)}; t = 1, 2, \cdots, T_2\}$ that fall into Ω_2, where T_2 denotes the total number of the samples in this subset. The conditional probability P_2 can be estimated as

$$P_2^{SUS} = \frac{1}{T_1 \cdot L_2} \cdot \sum_{t=1}^{T_1}\sum_{l=1}^{L_2} I_{F_2}\left[y\left(\mathbf{x}^{(2,t,l)}\right)\right] = \frac{T_2}{T_1 \cdot L_2}, \quad (22)$$

where P_2^{SUS} denotes the estimated value of P_2, and $I_{F2}[y(\mathbf{x})]$ is the indicator function defined in (15).

By following the aforementioned idea, we can estimate all the probabilities $\{P_k; k = 1, 2, \cdots, K\}$. Namely, for the k-th phase where $k > 2$, we need to estimate the conditional probability $P_k = \Pr[y(\mathbf{x}) \in F_k \mid y(\mathbf{x}) \in F_{k-1}]$ by generating MCMC samples via the MM algorithm. Once the values of $\{P_k; k = 1, 2, \cdots, K\}$ are estimated, the rare failure rate P_F is calculated by

$$P_F^{SUS} = \prod_{k=1}^{K} P_k^{SUS}, \quad (23)$$

where P_F^{SUS} represents the estimated value of P_F by using SUS.

3.2 High-dimensional Case

If we have more than two random variables, estimating the probabilities $\{P_k; k = 1, 2, \cdots, K\}$ can be pursued in a similar way. Algorithm 1 summarizes the major steps of the proposed SUS method for high dimension. It consists of K phases. During the 1st phase, we randomly sample the PDF $p(\mathbf{x})$ to estimate P_1. Next, we apply MM (i.e., Step 7~15 in Algorithm 1) to estimate the conditional probability P_k during the k-th phase, where $k \in \{2, 3, \cdots, K\}$. When estimating P_k, we construct T_{k-1} Markov chains. Each Markov chain contains L_k MCMC samples $\{\mathbf{x}^{(k,t,l)}; l = 1, 2, \cdots, L_k\}$ that are created by the MM algorithm. Hence, there are $T_{k-1} \cdot L_k$ samples in total, and the probability P_k is estimated by these samples. Finally, the rare failure rate P_F is estimated from $\{P_k; k = 1, 2, \cdots, K\}$ by using (23).

Algorithm 1: Subset Simulation (SUS)

1. Start from a set of pre-defined failure events $\{F_k; k = 1, 2, \cdots, K\}$.
2. Generate L_1 random samples $\{\mathbf{x}^{(1,l)}; l = 1, 2, \cdots, L_1\}$ from $p(\mathbf{x})$.
3. From the random samples $\{\mathbf{x}^{(1,l)}; l = 1, 2, \cdots, L_1\}$, identify the samples for which $y[\mathbf{x}^{(1,l)}] \in F_1$. Label these samples as $\{\mathbf{x}_F^{(1,t)}; t = 1, 2, \cdots, T_1\}$, where T_1 represents the total number of samples satisfying the condition. Calculate P_1^{SUS} by (13).
4. Initialize $k = 2$.
5. Set $\mathbf{x}^{(k,t,1)} = \mathbf{x}_F^{(k-1,t)}$, where $t = 1, 2, \cdots, T_{k-1}$.
6. For $t = 1, 2, \cdots, T_{k-1}$
7. For $l = 2, 3, \cdots, L_k$
8. For $m = 1, 2, \cdots, M$
9. Generate a random value x_m^{NEW} from the 1-D transition PDF $q_m[x_m^{NEW} \mid x_m^{(k,t,l-1)}]$. For instance, the 1-D transition PDF can be a Normal distribution [16]

$$q_m\left[x_m^{NEW}\middle|x_m^{(k,t,l-1)}\right] = \frac{1}{\sqrt{2\pi}\cdot\sigma_m}\cdot\exp\left\{-\left[x_m^{NEW} - x_m^{(k,t,l-1)}\right]^2 \Big/ 2\cdot\sigma_m^2\right\}. \quad (24)$$

10. Calculate the ratio

$$r = p_m\left(x_m^{NEW}\right) \Big/ p_m\left(x_m^{(k,t,l-1)}\right). \quad (25)$$

11. Draw a random value u from the uniform distribution in (19) and set the value of $x_m^{(k,t,l)}$ as

$$x_m^{(k,t,l)} = \begin{cases} x_m^{NEW} & u \le \min(1, r) \\ x_m^{(k,t,l-1)} & u > \min(1, r) \end{cases}. \quad (26)$$

12. End For
13. Form a candidate $\mathbf{x}^{NEW} = [x_1^{(k,\,t,\,l)}\ x_2^{(k,\,t,\,l)} \cdots x_M^{(k,\,t,\,l)}]$, and run a transistor-level simulation to evaluate $y(\mathbf{x}^{NEW})$.
14. Set the value of $\mathbf{x}^{(k,\,t,\,l)}$ as

$$\mathbf{x}^{(k,t,l)} = \begin{cases} \mathbf{x}^{NEW} & y(\mathbf{x}^{NEW}) \in F_{k-1} \\ \mathbf{x}^{(k,t,l-1)} & y(\mathbf{x}^{NEW}) \notin F_{k-1} \end{cases}. \quad (27)$$

15. End For
16. End For
17. From the $T_{k-1} \cdot L_k$ MCMC samples $\{\mathbf{x}^{(k,\,t,\,l)};\ t = 1, 2, \cdots, T_{k-1},\ l = 1, 2, \cdots, L_k\}$, identify the samples for which $y[\mathbf{x}^{(k,\,t,\,l)}] \in F_k$. Label these samples as $\{\mathbf{x}_F^{(k,\,t)};\ t = 1, 2, \cdots, T_k\}$, where T_k represents the total number of samples satisfying the condition.
18. Calculate P_k^{SUS} by

$$P_k^{SUS} = \frac{T_k}{T_{k-1} \cdot L_k}. \quad (28)$$

19. If $k < K$, set $k = k + 1$ and go to Step 5. Otherwise, go to Step 20.
20. Calculate P_F^{SUS} by (23).

There are several important clarifications that should be made for Algorithm 1. First, sampling the transition PDF $q_m[x_m^{NEW} \mid x_m^{(k,\,t,\,l-1)}]$ in (24) at Step 9 or the uniform PDF $f(u)$ in (19) at Step 11 involves no transistor-level simulation and, hence, its computational cost is almost negligible. The computational cost of Algorithm 1 is dominated by the transistor-level simulation to evaluate $y(\mathbf{x}^{NEW})$ at Step 13.

Second, MM successively samples a set of 1-D transition PDFs $\{q_m[x_m^{NEW} \mid x_m^{(k,\,t,\,l-1)}];\ m = 1, 2, \cdots, M\}$, instead of any high-dimensional joint PDF, to generate a new MCMC sample. For this reason, MM does not suffer from any dimensionality issue and can efficiently handle high-dimensional problems in practice. More detailed discussions about the efficiency of MM for high dimension can be found in [16].

Finally, the 1-D transition PDFs $\{q_m[x_m^{NEW} \mid x_m^{(k,\,t,\,l-1)}];\ m = 1, 2, \cdots, M\}$ play an important role in sampling the failure region Ω_{k-1} at the k-th phase, where Ω_{k-1} denotes the subset of the variation space $\{\mathbf{x} \mid y(\mathbf{x}) \in F_{k-1}\}$. For illustration purposes, we consider the 1-D transition PDF in (24) as an example. For this Normal distribution, if its standard deviation σ_m is too large, it is likely that the new sample x_m^{NEW} is far away from the previous sample $x_m^{(k,\,t,\,l-1)}$. In other words, we attempt to "jump" over a long distance via the Markov chain. However, the new sample $\mathbf{x}^{NEW}$ may eventually fall out of the failure region Ω_{k-1} (i.e., $y(\mathbf{x}^{NEW}) \notin F_{k-1}$) and get rejected, as shown in (27).

On the other hand, if σ_m is too small, it is likely that the new sample x_m^{NEW} is extremely close to the previous sample $x_m^{(k,\,t,\,l-1)}$. In this case, it may require many MCMC samples to fully explore the failure region Ω_{k-1}. The aforementioned discussions imply an important fact that the value of σ_m must be appropriately chosen in order to make Algorithm 1 efficient. As a heuristic approach [16], we simply set σ_m equal to the standard deviation of the original PDF $p_m(x_m)$ shown in (2). Intuitively, if the standard deviation of $p_m(x_m)$ is large, the random variable x_m can vary over a large range. In this case, we want to set σ_m to a relatively large value so that the resulting Markov chain can quickly explore a large region of the variation space.

4. IMPLEMENTATION DETAILS

In this section, we further discuss several important implementation issues for SUS, including (i) subset selection, and (ii) confidence interval estimation.

4.1 Subset Selection

In Algorithm 1, we assume that the failure events $\{F_k;\ k = 1, 2, \cdots, K\}$ are pre-defined. In practice, however, we have to carefully define $\{F_k;\ k = 1, 2, \cdots, K\}$ so that our proposed SUS method is computationally efficient. Otherwise, if $\{F_k;\ k = 1, 2, \cdots, K\}$ are not appropriately chosen, $\{P_k;\ k = 1, 2, \cdots, K\}$ can be extremely small and, hence, cannot be efficiently estimated by a small number of samples.

To address this practical issue, we propose to adaptively choose the failure events $\{F_k;\ k = 1, 2, \cdots, K\}$ to satisfy the following conditions

$$\begin{matrix} P_k^{SUS} = P_{OBJ} \quad (k = 1, 2, \cdots, K-1) \\ P_K^{SUS} \ge P_{OBJ} \end{matrix}, \quad (29)$$

where P_{OBJ} ($0 < P_{OBJ} < 1$) is a user-specified parameter that defines the "desired" value for P_k at the k-th phase. To intuitively illustrate our proposed strategy for subset selection, we consider a sense amplifier example where the PoI $y(\mathbf{x})$ is the delay from the input to its output. If the delay $y(\mathbf{x})$ is greater than or equal to a pre-defined specification y_{SPEC}, we consider the sense amplifier as "FAIL". In this example, the failure events $\{F_k;\ k \in 1, 2, \cdots, K\}$ can be defined by setting different values for the delay specification. The delay specification should be tight (i.e., small) for F_1 and loose (i.e., large) for F_K. Our objective is to adaptively determine a set of monotonically increasing values $\{y_1 < y_2 < \cdots < y_{K-1} < y_K = y_{SPEC}\}$ and define the failure event F_k as $F_k = \{y \mid y(\mathbf{x}) \ge y_k\}$, where $k \in \{1, 2, \cdots, K\}$, so that the conditions in (29) are satisfied.

Based upon this idea, during the 1st phase of SUS for the sense amplifier example, we need to take the random samples $\{y[\mathbf{x}^{(1,\,l)}];\ l = 1, 2, \cdots, L_1\}$ created in Step 2 of Algorithm 1, and determine the value of y_1 so that the condition $y[\mathbf{x}^{(1,\,l)}] \ge y_1$ holds for $T_1 = L_1 \cdot P_1^{SUS} = L_1 \cdot P_{OBJ}$ samples. To this end, we sort $\{y[\mathbf{x}^{(1,\,l)}];\ l = 1, 2, \cdots, L_1\}$ and then set y_1 to the T_1-th largest value of y of the sorted samples. Similarly, during the 2nd phase, we need to take the random samples $\{\mathbf{x}^{(2,\,t,\,l)};\ t = 1, 2, \cdots, T_1,\ l = 1, 2, \cdots, L_2\}$ created in Step 6~16 of Algorithm 1, and determine the value of y_2 so that the condition $y[\mathbf{x}^{(2,\,t,\,l)}] \ge y_2$ holds for $T_2 = T_1 \cdot L_2 \cdot P_2^{SUS} = T_1 \cdot L_2 \cdot P_{OBJ}$ samples. We sort $\{\mathbf{x}^{(2,\,t,\,l)};\ t = 1, 2, \cdots, T_1,\ l = 1, 2, \cdots, L_2\}$ and then set y_2 to the T_2-th largest value of y of the sorted samples. The aforementioned procedure is repeatedly applied to further determine the values of $\{y_3, y_4, \cdots\}$ until we reach the K-th phase where setting $y_K = y_{SPEC}$ results in the probability P_K^{SUS} that is greater than P_{OBJ}.

Two important clarifications should be made for our proposed subset selection. First, combining (10), (23) and (29) implies

$$P_F \approx P_F^{SUS} = P_{OBJ}^{K-1} \cdot P_K^{SUS} \ge P_{OBJ}^K. \quad (30)$$

From (30), we observe that the total number of phases (i.e., K) depends on the failure rate P_F and the user-specified parameter P_{OBJ}. The value of K is approximately equal to the minimum integer that is greater than $\log(P_F) / \log(P_{OBJ})$. Since we do not know P_F in advance, we cannot pre-determine K. Instead, the value of K must be adaptively set when running the SUS algorithm.

Second, but more importantly, the user-specified parameter P_{OBJ} substantially impacts the efficiency of SUS. If P_{OBJ} is too

small, the probabilities $\{P_k;\ k = 1, 2, \cdots, K\}$ are small. Hence, estimating these probabilities requires a large number of samples and, therefore, is not computationally efficient. On the other hand, if P_{OBJ} is too large, estimating $\{P_k;\ k = 1, 2, \cdots, K\}$ becomes trivial. However, based on (30), a large number of phases are needed to estimate the failure rate P_F (i.e., K is large), thereby resulting in high computational cost. For these reasons, it is crucial to appropriately choose P_{OBJ} to make SUS efficient for practical circuit analysis problems.

In this paper, an empirical value is chosen for P_{OBJ} (say, $P_{OBJ} = 0.1$). In this case, even if the failure rate P_F is extremely small (e.g., 10^{-8}~10^{-6}), SUS only needs a small number of (e.g., K = 6~8) phases to complete. Furthermore, it only requires a few hundred samples during each phase to accurately estimate the probability P_k that is close to 0.1, where $k \in \{1, 2, \cdots, K\}$. It, in turn, results in an efficient implementation of SUS for rare failure rate estimation, as will be demonstrated by our experimental results in Section 5.

4.2 Confidence Interval Estimation

To quantitatively assess the accuracy of the proposed SUS estimator P_F^{SUS} shown in (23), we must estimate its confidence interval (CI). To this end, we need to know the distribution of P_F^{SUS}. Since P_F^{SUS} is equal to the multiplication of $\{P_k^{SUS};\ k = 1, 2, \cdots, K\}$, we should carefully study the statistical property of $\{P_k^{SUS};\ k = 1, 2, \cdots, K\}$ in order to derive the distribution for P_F^{SUS}.

As described in Algorithm 1, the estimators $\{P_k^{SUS};\ k = 1, 2, \cdots, K\}$ are calculated from the random samples either drawn from $p(\mathbf{x})$ in Step 2 or created by MM in Step 6~16 of Algorithm 1. To be specific, P_1^{SUS} is calculated by using (13) with L_1 independent and identically distributed (i.i.d.) samples drawn from $p(\mathbf{x})$ in Step 2 of Algorithm 1. According to the central limit theorem (CLT) [20], P_1^{SUS} approximately follows a Normal distribution

$$P_1^{SUS} \sim N(P_1, v_1), \tag{31}$$

where the mean value P_1 is defined in (11) and the variance value v_1 can be approximated as [20]

$$v_1 \approx \frac{1}{L_1} \cdot P_1^{SUS} \cdot \left(1 - P_1^{SUS}\right). \tag{32}$$

On the other hand, P_k^{SUS}, where $k \in \{2, 3, \cdots, K\}$, is calculated by using (28) with the MCMC samples $\{\mathbf{x}^{(k,\,t,\,l)};\ t = 1, 2, \cdots, T_{k-1},\ l = 1, 2, \cdots, L_k\}$ created by MM in Step 6~16 of Algorithm 1. It has been proved in [16] that all these MCMC samples $\{\mathbf{x}^{(k,\,t,\,l)};\ t = 1, 2, \cdots, T_{k-1},\ l = 1, 2, \cdots, L_k\}$ follow the conditional PDF $p(\mathbf{x} \mid \mathbf{x} \in \Omega_{k-1})$. Eq. (28) can be re-written as

$$P_k^{SUS} = \frac{1}{T_{k-1} \cdot L_k} \cdot \sum_{t=1}^{T_{k-1}} \sum_{l=1}^{L_k} I_{F_k}\left[y\left(\mathbf{x}^{(k,t,l)}\right)\right], \tag{33}$$

where $I_{Fk}[y(\mathbf{x})]$ represents the indicator function

$$I_{F_k}\left[y(\mathbf{x})\right] = \begin{cases} 1 & y(\mathbf{x}) \in F_k \\ 0 & y(\mathbf{x}) \notin F_k \end{cases}. \tag{34}$$

From Step 7~15 of Algorithm 1, we can observe that $\{\mathbf{x}^{(k,\,t,\,l)};\ l = 1, 2, \cdots, L_k\}$, where $t \in \{1, 2, \cdots, T_{k-1}\}$, are strongly correlated. Alternatively speaking, the MCMC samples $\{\mathbf{x}^{(k,\,t,\,l)};\ t = 1, 2, \cdots, T_{k-1},\ l = 1, 2, \cdots, L_k\}$ used to calculate P_k^{SUS} in (33) are not independent and, hence, cannot be considered as i.i.d. samples. For this reason, we cannot directly apply CLT [20] to derive the distribution for the estimator P_k^{SUS} in (33).

To address this issue, we define a set of new random variables

$$s^{(k,t)} = \frac{1}{L_k} \cdot \sum_{l=1}^{L_k} I_{F_k}\left[y\left(\mathbf{x}^{(k,t,l)}\right)\right] \quad (t = 1, 2, \cdots, T_{k-1}). \tag{35}$$

Studying (35) reveals an important observation that $s^{(k,\,t)}$, where $t \in \{1, 2, \cdots, T_{k-1}\}$, only depends on the t-th Markov chain $\{\mathbf{x}^{(k,\,t,\,l)};\ l = 1, 2, \cdots, L_k\}$. Since different Markov chains are created from different initial samples $\{\mathbf{x}^{(k,\,t,\,1)};\ t = 1, 2, \cdots, T_{k-1}\}$, the random variables $\{s^{(k,\,t)};\ t = 1, 2, \cdots, T_{k-1}\}$ in (35) are almost statistically independent. Furthermore, since all the initial samples $\{\mathbf{x}^{(k,\,t,\,1)};\ t = 1, 2, \cdots, T_{k-1}\}$ follow the same conditional PDF $p(\mathbf{x} \mid \mathbf{x} \in \Omega_{k-1})$ and all the Markov chains are generated by following the same procedure (i.e., Step 7~15 of Algorithm 1), the random variables $\{s^{(k,\,t)};\ t = 1, 2, \cdots, T_{k-1}\}$ should be identically distributed. For these reasons, we can consider $\{s^{(k,\,t)};\ t = 1, 2, \cdots, T_{k-1}\}$ as a set of i.i.d. random variables.

Based on (35), P_k^{SUS} in (33) can be re-written as

$$P_k^{SUS} = \frac{1}{T_{k-1}} \cdot \sum_{t=1}^{T_{k-1}} s^{(k,t)}. \tag{36}$$

Since $\{s^{(k,\,t)};\ t = 1, 2, \cdots, T_{k-1}\}$ are i.i.d. random variables, P_k^{SUS} approximately follows a Normal distribution according to CLT

$$P_k^{SUS} \sim N\left[\mathrm{MEAN}\left(P_k^{SUS}\right), \mathrm{VAR}\left(P_k^{SUS}\right)\right], \tag{37}$$

where MEAN(•) and VAR(•) denote the mean and variance of a random variable respectively. From (33) and (34), MEAN(P_k^{SUS}) can be easily calculated as

$$\mathrm{MEAN}\left(P_k^{SUS}\right) = P_k, \tag{38}$$

where P_k is defined in (12). Based on (36) where $\{s^{(k,\,t)};\ t = 1, 2, \cdots, T_{k-1}\}$ are i.i.d. random variables, VAR(P_k^{SUS}) can be computed as

$$\mathrm{VAR}\left(P_k^{SUS}\right) = \frac{1}{T_{k-1}^2} \cdot \sum_{t=1}^{T_{k-1}} \mathrm{VAR}\left[s^{(k,t)}\right] = \frac{1}{T_{k-1}} \cdot v_{s,k}, \tag{39}$$

where $v_{s,k}$ represents the variance of the random variables $\{s^{(k,\,t)};\ t = 1, 2, \cdots, T_{k-1}\}$, and it can be approximated by the sample variance of $\{s^{(k,\,t)};\ t = 1, 2, \cdots, T_{k-1}\}$

$$v_{s,k} \approx \frac{1}{T_{k-1} - 1} \cdot \sum_{t=1}^{T_{k-1}} \left[s^{(k,t)} - P_k^{SUS}\right]^2. \tag{40}$$

Substituting (38), (39) and (40) into (37) yields

$$P_k^{SUS} \sim N(P_k, v_k) \quad (k = 2, 3, \cdots, K), \tag{41}$$

where P_k is defined in (12) and

$$v_k \approx \frac{1}{(T_{k-1} - 1) \cdot T_{k-1}} \cdot \sum_{t=1}^{T_{k-1}} \left[s^{(k,t)} - P_k^{SUS}\right]^2. \tag{42}$$

To further derive the distribution for P_F^{SUS} in (23) based on (31) and (41), we take logarithm on both sides of (23) because it is much easier to handle summation than multiplication

$$\log\left(P_F^{SUS}\right) = \sum_{k=1}^{K} \log\left(P_k^{SUS}\right). \tag{43}$$

To derive the distribution of $\{\log(P_k^{SUS});\ k = 1, 2, \cdots, K\}$, we approximate the nonlinear function log(•) by the first-order Taylor expansion around the mean value P_k of the random variable P_k^{SUS}

$$\log\left(P_k^{SUS}\right) \approx \log\left(P_k\right) + \frac{P_k^{SUS} - P_k}{P_k} \approx \log P_k + \frac{P_k^{SUS} - P_k}{P_k^{SUS}}. \tag{44}$$

According to the linear approximation in (44), $\log(P_k^{SUS})$ follows a Normal distribution

$$\log\left(P_k^{SUS}\right) \sim N\left[\log\left(P_k\right), v_{\log,k}\right] \quad (k = 1, 2, \cdots, K), \tag{45}$$

where

$$v_{\log,k} = v_k \Big/ \left(P_k^{SUS}\right)^2. \tag{46}$$

Since $\log(P_F^{SUS})$ is the summation of several "approximately" Normal random variables $\{\log(P_k^{SUS});\ k = 1, 2, \cdots, K\}$, $\log(P_F^{SUS})$ also approximately follows a Normal distribution [20]

$$\log\left(P_F^{SUS}\right) \sim N\left\{\text{MEAN}\left[\log\left(P_F^{SUS}\right)\right], \text{VAR}\left[\log\left(P_F^{SUS}\right)\right]\right\}. \quad (47)$$

Based on (10), (43), and (45), MEAN[$\log(P_F^{SUS})$] can be expressed as

$$\text{MEAN}\left[\log\left(P_F^{SUS}\right)\right] = \sum_{k=1}^{K}\log\left(P_k\right) = \log\left(\prod_{k=1}^{K}P_k\right) = \log\left(P_F\right), \quad (48)$$

and VAR[$\log(P_F^{SUS})$] can be calculated as

$$\begin{aligned}\text{VAR}\left[\log\left(P_F^{SUS}\right)\right] &= \text{VAR}\left[\sum_{k=1}^{K}\log\left(P_k^{SUS}\right)\right] \\ &= \sum_{k=1}^{K}v_{\log,k} + 2\cdot\sum_{i=1}^{K-1}\sum_{j=i+1}^{K}\text{COV}\left[\log\left(P_i^{SUS}\right),\log\left(P_j^{SUS}\right)\right]\end{aligned}, \quad (49)$$

where COV(•, •) denotes the covariance of two random variables.

When applying MCMC, we often observe that an MCMC sample is strongly correlated to its adjacent sample. However, the correlation quickly decreases as the distance between two MCMC samples increases. Therefore, we can assume that the samples used to estimate $\log(P_i^{SUS})$ are weakly correlated to the samples used to estimate $\log(P_j^{SUS})$, if the distance between i and j is greater than 1 (i.e., $|i - j| > 1$). Based on this assumption, Eq. (49) can be approximated as

$$\begin{aligned}&\text{VAR}\left[\log\left(P_F^{SUS}\right)\right] \\ &\approx \sum_{k=1}^{K}v_{\log,k} + 2\cdot\sum_{k=1}^{K-1}\text{COV}\left[\log\left(P_k^{SUS}\right),\log\left(P_{k+1}^{SUS}\right)\right]\end{aligned}. \quad (50)$$

Accurately estimating the covariance between $\log(P_k^{SUS})$ and $\log(P_{k+1}^{SUS})$ is nontrivial. Here, we derive an upper bound for COV[$\log(P_k^{SUS})$, $\log(P_{k+1}^{SUS})$] [20]

$$\begin{aligned}&\text{COV}\left[\log\left(P_k^{SUS}\right),\log\left(P_{k+1}^{SUS}\right)\right] \le \sqrt{v_{\log,k}\cdot v_{\log,k+1}} \\ &\left(k = 1, 2, \cdots, K-1\right)\end{aligned}. \quad (51)$$

Substituting (51) into (50) yields

$$\text{VAR}\left[\log\left(P_F^{SUS}\right)\right] \le \sum_{k=1}^{K}v_{\log,k} + 2\cdot\sum_{k=1}^{K-1}\sqrt{v_{\log,k}\cdot v_{\log,k+1}} = v_{\log,SUS}. \quad (52)$$

In this paper, we approximate VAR[$\log(P_F^{SUS})$] by its upper bound $v_{\log,SUS}$ defined in (52) to provide a conservative estimation for the CI. Based on (48) and (52), Eq. (47) can be re-written as

$$\log\left(P_F^{SUS}\right) \sim N\left[\log\left(P_F\right), v_{\log,SUS}\right]. \quad (53)$$

According to (53), we can derive the CI for any given confidence level. For instance, the 95% CI is expressed as

$$\begin{aligned}&\left[\exp\left(\log\left(P_F^{SUS}\right) - 1.96\cdot\sqrt{v_{\log,SUS}}\right),\right. \\ &\left.\exp\left(\log\left(P_F^{SUS}\right) + 1.96\cdot\sqrt{v_{\log,SUS}}\right)\right]\end{aligned}. \quad (54)$$

The aforementioned CI estimation accurately captures the uncertainty of our proposed estimator P_F^{SUS}, as will be demonstrated by several numerical examples in Section 5.

5. NUMERICAL EXAMPLES

In this section, we demonstrate the efficacy of SUS by several circuit examples. For testing and comparison purposes, four different algorithms are implemented: (i) brute-force MC, (ii) minimum-norm importance sampling (MNIS) [10], (iii) SSS [15], and (iv) SUS. In our experiments, the brute-force MC is applied to generate the "golden" failure rate which is used to evaluate the accuracy of the other three approaches. As described in [10], MNIS consists of two stages: (i) 2000 transistor-level simulations are first used to search the variation space, and (ii) a shifted Normal PDF is then constructed to perform importance sampling and estimate the rare failure rate. When implementing SSS, six scaling factors (i.e., 1.5, 2.3, 3.1, 3.9, 4.7 and 5.5) are empirically chosen. Finally, for our proposed SUS, the 1-D Normal distribution in (24) is used as the transition PDF in Step 9 of Algorithm 1 and the parameter P_{OBJ} in (29) is empirically set to 0.1.

5.1 SRAM Read Current

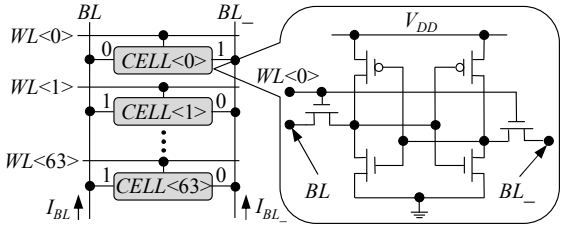


Figure 2. The simplified schematic is shown for an SRAM column consisting of 64 bit-cells designed in a 45nm CMOS process.

Figure 2 shows the simplified schematic of an SRAM column consisting of 64 bit-cells designed in a 45nm CMOS process. In this example, our PoI is the read current I_{READ}, which is defined as the difference between the bit-line currents I_{BL} and $I_{BL\_}$ (i.e., $I_{READ} = I_{BL} - I_{BL\_}$). The PoI I_{READ} directly impacts the SRAM read delay and, hence, is an important performance metric. If I_{READ} is greater than a pre-defined specification, we consider the SRAM circuit as "PASS". Otherwise, it is considered as "FAIL". To mimic the worst-case scenario for read operation, we maximize the impact of leakage current by storing "0" in *CELL*<0> and "1" in all other bit-cells. For process variation modeling, the local V_{TH} mismatch of each transistor is considered as an independent Normal random variable. In total, we have 384 independent random variables (i.e., 64 bit-cells × 6 transistors per bit-cell = 384). It, in turn, serves as a good example to demonstrate the efficacy of SUS in a high-dimensional (i.e., $M = 384$) variation space.

Table 1. Failure rate P_F and 95% CI [P_F^L, P_F^U] estimated by MNIS, SSS, and SUS ("golden" failure rate = 10^{-6})

# of Sims		3600	4200	4800	5400	6000
MNIS [10]	P_F^L	0	0	0	0	0
	P_F	9.2×10^{-11}	6.8×10^{-11}	5.4×10^{-11}	2.1×10^{-10}	1.8×10^{-10}
	P_F^U	2.5×10^{-10}	1.8×10^{-10}	1.4×10^{-10}	4.7×10^{-10}	4.0×10^{-10}
SSS [15]	P_F^L	9.9×10^{-11}	3.2×10^{-11}	1.3×10^{-10}	2.8×10^{-10}	3.8×10^{-9}
	P_F	5.3×10^{-7}	1.6×10^{-7}	8.5×10^{-7}	9.6×10^{-7}	1.3×10^{-6}
	P_F^U	5.8×10^{-4}	5.0×10^{-5}	5.3×10^{-4}	5.2×10^{-4}	1.4×10^{-4}
SUS	P_F^L	2.1×10^{-7}	1.2×10^{-7}	1.9×10^{-7}	1.7×10^{-7}	2.3×10^{-7}
	P_F	1.2×10^{-6}	7.6×10^{-7}	9.9×10^{-7}	8.9×10^{-7}	1.0×10^{-6}
	P_F^U	7.4×10^{-6}	4.7×10^{-6}	5.1×10^{-6}	4.7×10^{-6}	4.5×10^{-6}

Based on the brute-force MC with 10^7 samples, the "golden" failure rate is equal to 10^{-6} in this example. Table 1 summarizes the failure rate and the 95% CI estimated by MNIS, SSS, and SUS with different numbers of simulations. From Table 1, we have several important observations. First, the estimated failure rate from MNIS is strongly biased. We believe that the shifted Normal

PDF used by MNIS for importance sampling does not capture the most important failure region and, hence, the estimated failure rate is substantially less than the "golden" failure rate. More importantly, the 95% CI estimated by MNIS cannot cover the "golden" failure rate, implying that MNIS also fails to assess the accuracy of its estimated failure rate. This is an important limitation of MNIS since the user cannot reliably know the actual "confidence" of the estimator in practice.

Second, even though the 95% CI estimated by SSS can overlap with the "golden" failure rate, its range is large and, hence, the estimation is inaccurate. Because SSS relies on extrapolation to estimate the failure rate based on a set of probability values associated with different scaling factors, the estimation errors for these intermediate probabilities can be "amplified" due to the extrapolation operation. For this reason, the failure rate estimated by SSS can be inaccurate and, consequently, its CI covers a large range.

Third, for our proposed SUS method, the estimator is almost unbiased and the 95% CI is relatively small compared to MNIS and SSS. It, in turn, demonstrates that SUS can achieve substantially better accuracy than the traditional methods (i.e., MNIS and SSS) in this SRAM example.

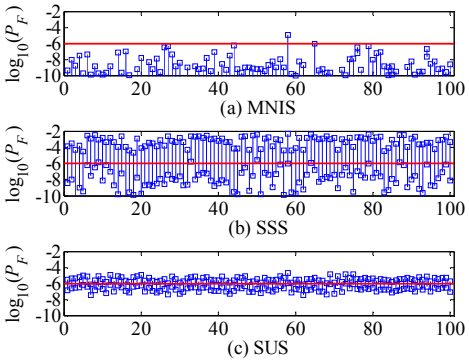


Figure 3. The 95% CIs are estimated from 100 repeated runs with 6000 simulations in each run for three different methods: (a) MNIS, (b) SSS, and (c) SUS. The red line represents the "golden" failure rate.

Finally, we further verify the accuracy of the 95% CI estimated by three different methods (i.e., MNIS, SSS and SUS). To this end, we repeatedly run each method for 100 times with 6000 simulations in each run. Figure 3 shows the 100 estimated 95% CIs for each method, where each blue bar represents the CI of a single run, and the red line represents the "golden" failure rate. To clearly plot these CIs, the y-axis of Figure 3 is displayed in logarithmic scale over the range $[10^{-10}, 10^{-2}]$. If this range cannot cover a complete CI, we only show a portion of the CI that is inside the range.

Studying Figure 3 reveals several important observations. First, only a single CI estimated from 100 repeated runs by MNIS can cover the "golden" failure rate. It, again, demonstrates the fact that the confidence level of MNIS cannot be accurately assessed by its estimated CI. Second, there are 94 and 97 CIs out of 100 runs that cover the "golden" failure rate for SSS and SUS respectively. It demonstrates that the CIs estimated by both SSS and SUS accurately measure the estimation error and can appropriately indicate the conference level of the estimator. Third, the CIs of SUS are substantially tighter than those of SSS, implying that SUS is more accurate than SSS in this example.

5.2 DFF Delay

We consider a DFF circuit designed in a 45nm CMOS process, as shown in Figure 4. It consists of 20 transistors and the random mismatch of each transistor is modeled by 10 independent Normal random variables. Compared to the SRAM example in Section 5.1, a detailed statistical model is used to capture the random mismatch for the DFF circuit and, hence, additional random variables are needed to model the mismatch of each transistor. In total, there are 200 independent random variables in this example. Our PoI is the data transfer delay $D_{CLK \to Q}$ from the clock *CLK* to the output Q. If the delay $D_{CLK \to Q}$ belongs to a pre-defined interval $[D_{LOW}, D_{UP}]$, we consider the DFF circuit as "PASS". Otherwise, the DFF circuit is considered as "FAIL".

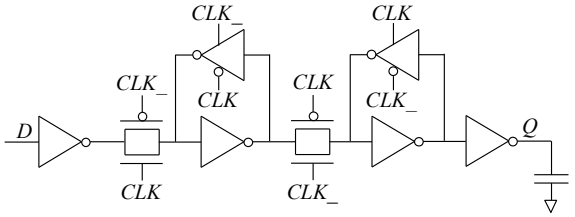


Figure 4. The simplified circuit schematic is shown for a DFF circuit designed in a 45nm CMOS process.

Table 2. Failure rate P_F and 95% CI $[P_F^L, P_F^U]$ estimated by MNIS, SSS, and SUS ("golden" failure rate = 4.8×10^{-6})

# of Sims		3300	3850	4400	4950	5500
MNIS [10]	P_F^L	0	0	6.9×10^{-9}	3.6×10^{-8}	1.8×10^{-7}
	P_F	1.2×10^{-6}	9.2×10^{-7}	7.5×10^{-7}	6.4×10^{-7}	7.3×10^{-7}
	P_F^U	2.5×10^{-6}	1.9×10^{-6}	1.5×10^{-6}	1.2×10^{-6}	1.3×10^{-6}
SSS [15]	P_F^L	2.4×10^{-8}	2.1×10^{-8}	1.3×10^{-8}	1.1×10^{-8}	3.2×10^{-8}
	P_F	2.4×10^{-6}	1.5×10^{-6}	7.8×10^{-7}	5.4×10^{-7}	1.2×10^{-6}
	P_F^U	1.8×10^{-4}	8.7×10^{-5}	3.9×10^{-5}	2.3×10^{-5}	3.9×10^{-5}
SUS	P_F^L	1.3×10^{-6}	4.9×10^{-7}	1.6×10^{-6}	1.4×10^{-6}	2.3×10^{-6}
	P_F	6.0×10^{-6}	2.2×10^{-6}	6.1×10^{-6}	5.0×10^{-6}	7.4×10^{-6}
	P_F^U	2.9×10^{-5}	9.9×10^{-6}	2.3×10^{-5}	1.7×10^{-5}	2.4×10^{-5}

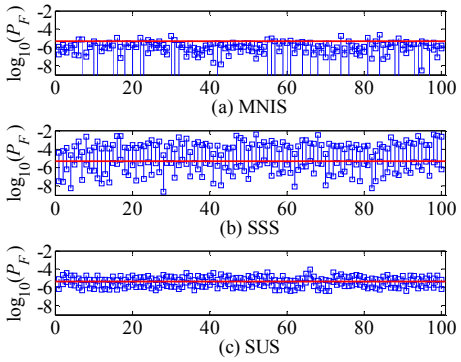


Figure 5. The 95% CIs are estimated from 100 repeated runs with 5500 simulations in each run for three different methods: (a) MNIS, (b) SSS, and (c) SUS. The red line represents the "golden" failure rate.

We first apply the brute-force MC with 5×10^6 random samples and the estimated "golden" failure rate is equal to 4.8×10^{-6} in this example. Next, we apply three different algorithms (i.e., MNIS, SSS and SUS) to estimate the failure rate

with different numbers of simulations. Table 2 compares the estimation accuracy of these three methods. Similar to the SRAM example in Section 5.1, neither MNIS nor SSS can accurately estimate the failure rate even with 5500 simulations. On the other hand, our proposed SUS method can accurately estimate the failure rate with a tight confidence interval, even if the number of simulations is as small as 3300. This DFF example again demonstrates the superior accuracy of our proposed SUS method over other traditional methods (i.e., MNIS and SSS) in a high-dimensional variation space.

Next, to further verify the accuracy of the 95% CI estimated by MNIS, SSS and SUS, we repeatedly run each method for 100 times with 5500 simulations in each run. The 100 estimated CIs for these three methods are plotted in logarithmic scale in Figure 5, where the range of the y-axis is set to $[10^{-9}, 10^{-2}]$. In this example, there are 17, 95 and 98 CIs out of 100 runs that cover the "golden" failure rate for MNIS, SSS, and SUS respectively. From these results, we observe that while MNIS fails to accurately estimate its CIs, both SSS and SUS successfully estimate the CIs and, hence, their confidence levels are accurately measured. Furthermore, the CIs associated with SUS are significantly tighter than those of SSS. It, in turn, demonstrates that our proposed SUS method is superior to the traditional SSS method in this example.

6. CONCLUSIONS

In this paper, a novel SUS approach is proposed to accurately analyze the rare failure events for nanoscale ICs in a high-dimensional variation space. The key idea of SUS is to express the rare failure probability as the product of several large conditional probabilities by introducing a number of intermediate failure events. A Markov chain Monte Carlo algorithm (i.e., the modified Metropolis algorithm) is used to accurately estimate the intermediate conditional probabilities and, eventually, the rare failure rate of a given circuit. In addition, a statistical methodology is further developed to reliably estimate the confidence interval of SUS. Two circuit examples designed in nanoscale technologies demonstrate that SUS offers superior estimation accuracy over the traditional techniques (i.e., MNIS and SSS) when hundreds of random variables are used to model process variations.

7. ACKNOWLEDGEMENTS

This work has been supported in part by the National Science Foundation under contract CCF–1148778.

8. REFERENCES

[1] B. Calhoun, Y. Cao, X. Li, K. Mai, L. Pileggi, R. Rutenbar and K. Shepard, "Digital circuit design challenges and opportunities in the era of nanoscale CMOS," *Proc. IEEE*, vol. 96, no. 2, pp. 343-365, Feb. 2008.

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

[3] R. Heald and P. Wang, "Variability in sub-100nm SRAM designs," *IEEE ICCAD*, pp. 347-352, 2004.

[4] 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," *IEEE DAC*, pp. 69-72, 2006.

[5] R. Topaloglu, "Early, accurate and fast yield estimation through Monte Carlo-alternative probabilistic behavioral analog system simulations," *IEEE VTS*, pp. 137-142, 2006.

[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," *IEEE ASP-DAC*, pp. 754-761, 2008.

[7] L. Dolecek, M. Qazi, D. Shah and A. Chandrakasan, "Breaking the simulation barrier: SRAM evaluation through norm minimization," *IEEE ICCAD*, pp. 322-329, 2008.

[8] J. Wang, S. Yaldiz, X. Li and L. Pileggi, "SRAM parametric failure analysis," *IEEE DAC*, pp. 496-501, 2009.

[9] 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. on CAD*, vol. 28, no. 8, pp. 1176-1189, Aug. 2009.

[10] 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," *IEEE DATE*, pp. 801-806, 2010.

[11] R. Fonseca, L. Dilillo, A. Bosio, P. Girard, S. Pravossoudovitch, A. Virazel and N. Badereddine, "A statistical simulation method for reliability analysis of SRAM core-cells," *IEEE DAC*, pp. 853-856, 2010.

[12] K. Katayama, S. Hagiwara, H. Tsutsui, H. Ochi and T. Sato, "Sequential importance sampling for low-probability and high-dimensional SRAM yield analysis," *IEEE ICCAD*, pp. 703-708, 2010.

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

[14] R. Kanj, R. Joshi, Z. Li, J. Hayes and S. Nassif, "Yield estimation via multi-cones," *IEEE DAC*, pp. 1107-1112, 2012.

[15] 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," *IEEE ICCAD*, pp. 478-485, 2013.

[16] S. Au and J. Beck, "Estimation of small failure probabilities in high dimensions by subset simulation," *Probabilistic Eng. Mechanics*, vol. 16, no. 4, pp. 263-277, Oct. 2001.

[17] A. Guyader, N. Hengartner and E. Matzner-Løber, "Simulation and estimation of extreme quantiles and extreme probabilities," *Appl. Math. Optimization*, vol. 64, no. 2, pp. 171-196, Oct. 2011.

[18] F. Cérou, P. Moral, T. Furon and A. Guyader, "Sequential Monte Carlo for rare event estimation," *Stat. Computing*, vol. 22, no. 3, pp. 795-808, May 2012.

[19] C. Bishop, *Pattern Recognition and Machine Learning*, Prentice Hall, 2007.

[20] A. Papoulis and S. Pillai, *Probability, Random Variables and Stochastic Process*, McGraw-Hill, 2001.