ECRIPSE: An Efficient Method for Calculating RTN-Induced Failure Probability of an SRAM Cell

Hiromitsu Awano, Masayuki Hiromoto and Takashi Sato
Graduate School of Informatics, Kyoto University
Yoshida-honmachi, Sakyo-ku, Kyoto, Japan 606–8501
Email: paper@easter.kuee.kyoto-u.ac.jp

***Abstract*—Failure rate degradation of an SRAM cell due to random telegraph noise (RTN) is calculated for the first time. ECRIPSE, an efficient method for calculating the RTN-induced failure probability of an SRAM cell, has been developed to exhaustively cover a large number of possible bias-voltage combinations on which RTN statistics strongly depend. In order to shorten computational time, the Monte Carlo calculation of a single gate-bias condition is accelerated by incorporating two techniques: 1) construction of an optimal importance sampling using particles that move about the "important" regions in a variability space, and 2) a classifier that quickly judges whether the random samples are in failure regions or not. We show that the proposed method achieves at least $15.6\times$ speed-up over the state-of-the-art method. We then integrate an RTN model to modulate failure probability. In our experiment, RTN worsens failure probability by six times than that calculated without the effect of RTN.**

I. Introduction

Random telegraph noise (RTN), which is observed as temporal changes in threshold voltages of MOS transistors, is an increasing concern since its impact on circuit operations drastically increases as transistor sizes shrink. It is reported that the threshold voltage (V_{TH}) variation caused by RTN reaches as large as 70 mV at a 22 nm technology node [1].

Static random access memory (SRAM) is considered to be very susceptible to the RTN-induced V_{TH} variation for two reasons. One is that an SRAM cell is designed using the smallest possible transistors to save chip area. Each transistor used in an SRAM cell should be carefully balanced to ensure the correct operation of the cell. However, smaller transistors are very sensitive to disturbances induced by RTN. The other one is that extremely low failure probability is required for an SRAM cell. It is common for a modern microprocessor to include tens of mega bytes of on-chip cache. Hence, even a small V_{TH} variation is not negligible to ensure the correct operation of an entire cache array.

The impact of RTN must be evaluated to optimize the circuit design. Ye et al. proposed a simulation methodology of the RTN-induced V_{TH} variation using RC filters and comparator circuits [2]. Aadithya et al. proposed an accurate simulation method that can estimate the impact of non-stationary RTN [3]. Those methods analyze the impact of RTN in time domain. Due to their very high computational cost, it is difficult for them to apply for failure probability calculations of an SRAM cell.

The failure probability calculation of an SRAM cell is a difficult problem even if we do not take into account the impact of RTN. A naive Monte Carlo method, which directly generates random samples in a variability space, cannot be applied. Only a few samples fall into a failure region because of low failure rate requirement of the cell. Importance sampling techniques are definitely required to solve this problem [4]–[6]. In the importance sampling, samples are drawn from a biased probability distribution called "alternative distribution." The alternative distribution is selected so that more number of random samples, which have large contributions to the Monte Carlo approximation, is drawn from the distribution. Application of the importance sampling to a failure probability calculation of an SRAM cell is not a trivial task since its performance depends strongly on the selection of the alternative distribution. The mean-shift methods in which the alternative distribution is approximated with a distribution whose mean is shifted to the most probable failure point [**?**]. Some methods also have been proposed to estimate the optimal alternative distiburion directly [7], [8], which are considered to be more accurate than the mean-shift methods.

There are two major factors which make the consideration of RTN difficult. One is an increase of the number of parameters. In order to calculate a failure probability, we first draw N random samples to simulate fabrication-process-induced variabilities. Then, for each drawn sample, M random samples that simulate RTN-induced variabilities must be drawn. The number of total random samples increases to $N \times M$. Its simulation takes very long time even if we utilize the importance sampling methods. The other difficulty is the treatment of a gate bias dependence. Major statistics of RTN, such as time constants that represent the average duration in which V_{TH} is in high or low level, are highly susceptible to the gate bias voltage. In order to guarantee that an SRAM cell satisfies given specifications, multiple failure probability calculations with different gate bias conditions are required.

In this paper, we propose a novel SRAM failure probability calculation method which can take the impact of RTN into account. We first accelerate a failure probability calculation in a single gate bias condition using an ensemble of particles moving around variability space, which enables us to automatically and efficiently select the optimal alternative distribution. This idea, called a "particle filter," is first developed in statistical community to enable an on-line estimation of a probability distribution [9], [10]. In a recent study, the particle filter is applied to estimate the optimal alternative distribution for a failure probability calculation of an SRAM cell [8]. While our approach is based on [8], we enhance the method so that it suits the failure probability calculation to efficiently consider multiple bias conditions via two new methods: a binary classifier and a two-stage Monte Carlo.

By utilizing a binary classifier, random samples are judged either failure or pass without executing time-consuming transistor-level simulations. Since the classifier is based on a linear model, the time required for the classifications is negligibly small. The reduction of the total calculation time is significant even though the time to train the classifier is newly introduced. An adoption of a two-stage Monte Carlo flow further reduces the calculation time while maintaining the accuracy. In the first stage, a rough estimation of the optimal

978-3-9815370-4-8/DATE15/©2015 EDAA

alternative distribution is obtained using a small number of random samples. Then, in the second stage, a failure probability is accurately calculated using the samples generated from the alternative distribution. With the above modifications, our method achieves $15.6\times$ speed-up compared to the state-of-the-art failure probability calculation method [8]. We then integrate the gate bias dependent RTN model into the extended method so as to take into account the impact of RTN.

Followings are the key contributions of this paper.

- To the best of our knowledge, this paper first focuses on the impact of RTN on a failure probability of an SRAM cell.
- A simulation method that accelerates the failure rate estimation by at least $15.6\times$ over the state-of-the-art method [8] has been proposed. We then integrate the gate bias dependent RTN model into the extended method and realize the RTN-induced failure probability calculation of an SRAM cell.
- The relationship between a failure probability of an SRAM cell and gate bias conditions is studied for the first time, which has become possible only with the proposed estimation method. Through numerical experiments, we show that the failure probability estimation becomes too optimistic unless the impact of RTN is taken into account. We also show that, in our experimental setup, the optimistic-case condition for the failure probability estimation is when a cell stores "0" and "1" with equal probability

The rest of this paper is constructed as follows. In Section II, we explain background that forms the basis of our method. In Section III, we explain the details of the proposed method. Then in Section IV, we describe experimental procedures and its results. Finally in Section V, conclusion remarks are provided.

II. Background

A. Failure probability calculation

The general failure probability calculation is written using a following integral:

$$P_{\text{fail}} = \int I(\boldsymbol{x}) P(\boldsymbol{x}) d\boldsymbol{x}. \tag{1}$$

Here, P_{fail} is the failure probability and $\boldsymbol{x}$ is the random variable such as V_{TH} in a D-dimensional variability space, which corresponds to the process variability of transistors. $P(\boldsymbol{x})$ is a probability distribution function (PDF) over the process variability. The PDF is typically represented as a multivariate Gaussian distribution. Without loss of generality, we assume that the random variables x_d which are elements of $\boldsymbol{x}$ are mutually independent since any set of random variables can be uncorrelated using a transformation called "whitening." $I(\boldsymbol{x})$ is an indicator function that returns one only when the realization of a circuit, which corresponds to $\boldsymbol{x}$, can not meet required specifications.

Since the indicator function does not, in general, have an analytical form, we rely on a Monte Carlo approximation of (1). In the Monte Carlo approximation, the above integral is calculated using random samples drawn from $P(\boldsymbol{x})$:

$$\widehat{P}_{\text{fail}} \approx \frac{1}{N} \sum_{i=1}^{N} I(\boldsymbol{x}_i), \tag{2}$$

where $\boldsymbol{x}_i \sim P(\boldsymbol{x})$. A naive Monte Carlo method in (2) cannot be applied to the calculation of (1) in low failure probability problems since very few or no samples fall inside the failure region in a practical time.

In order to improve the sampling efficiency, importance sampling techniques is developed. The key idea of an importance sampling is to calculate (1) using samples drawn from an alternative distribution $Q(\boldsymbol{x})$. The following equation is obtained by modifying (1):

$$P_{\text{fail}} = \int I(\boldsymbol{x}) \frac{P(\boldsymbol{x})}{Q(\boldsymbol{x})} Q(\boldsymbol{x}) d\boldsymbol{x}. \tag{3}$$

The Monte Carlo approximation of (3) using samples drawn from $Q(\boldsymbol{x})$ is obtained as:

$$\widehat{P}_{\text{fail}} \approx \frac{1}{N} \sum_{i=1}^{N} I(\boldsymbol{x}_i) \frac{P(\boldsymbol{x}_i)}{Q(\boldsymbol{x}_i)}, \tag{4}$$

where $\boldsymbol{x}_i \sim Q(\boldsymbol{x})$. The alternative distribution $Q(\boldsymbol{x})$ is selected so that larger number of failure samples are drawn from $Q(\boldsymbol{x})$ than the original PDF. The optimal alternative distribution is

$$Q_{\text{opt}}(\boldsymbol{x}) \propto I(\boldsymbol{x}) P(\boldsymbol{x}). \tag{5}$$

If we can draw samples from $Q_{\text{opt}}(\boldsymbol{x})$, the perfect approximation of (3) is achieved with only few samples since $I(\boldsymbol{x})P(\boldsymbol{x})/Q_{\text{opt}}(\boldsymbol{x})$ becomes constant. Namely, instead of using samples drawn from the original PDF, the integral is calculated using samples whose contributions to the summation are large. Therefore, in order to improve the efficiency, we have to select $Q(\boldsymbol{x})$ whose shape is similar to $Q_{\text{opt}}(\boldsymbol{x})$. However, this is not a trivial task since we do not know the exact shape of the indicator function $I(\boldsymbol{x})$. We here introduce a particle filtering technique to enable an automatic estimation of the optimal alternative distribution.

B. Particle filter

Particle filter is an on-line estimator of non-Gaussian distributions [9], [10]. A probabilistic density is approximated using the density of particles which move around the D-dimensional process variability space. The position of particles are updated iteratively using the following algorithm.

Prediction: The candidates of particles at next iteration step are drawn from a proposal distribution $q(\boldsymbol{x})$. A usual choice of $q(\boldsymbol{x})$ is the mixture of Gaussian distributions with each component centered at the previous particles so that the regions where previous particles existed are more likely to be visited.

Measurement: The weights which represent the goodness of fit of each candidate particle are calculated. In the context of the failure probability calculation of an SRAM cell, the weight is calculated as $I(\boldsymbol{x}) \cdot P(\boldsymbol{x})$. Therefore, large weights are assigned to particles that are in the failure region and are close to the origin of the variability space.

Resampling: The particles are resampled from the candidate particles according to the probabilities proportional to the weight assigned in the **Measurement** step. Hence, candidate particles which have larger weight attain more number of copies while particles with smaller weights are more likely to be eliminated. This step enables particle density to follow the weight distributions. Since we here calculate the weight as $I(\boldsymbol{x}) \cdot P(\boldsymbol{x})$, the population of particles become gradually closer to the distribution of $I(\boldsymbol{x}) \cdot P(\boldsymbol{x})$ by repeating the above procedures. Fig. 1 depicts the particles tracking the target PDF. The approximation of the optimal alternative distribution can be obtained as the distribution of particles.

C. Support vector machine

Support vector machine (SVM) is a supervised training model for binary classification [11]. Given a set of training examples which consists of feature vectors and corresponding labels, SVM learns a classification model which assigns a new feature vector into one of two classes.

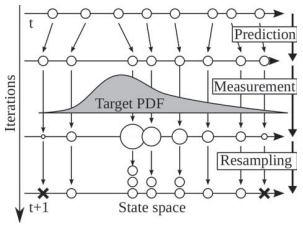


Fig. 1. Particle filter.

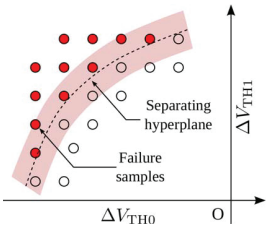


Fig. 2. Support vector machine.

SVM assumes the following linear classification model:

$$y = \sum_i w_i f_i. \tag{6}$$

Here, w_i are coefficients of particular feature quantities and f_i is the i-th element of the feature vector $\boldsymbol{f}$. The signature of y represents the class label of the feature vector. In other words, SVM learns a hyper plane in a feature space, which separates training examples into two classes as shown in Fig. 2. A good separation for a new feature vector is achieved when the distances between training examples and the hyper-plane are the largest.

SVM-based classifier was first applied to a failure probability calculation in [12]. As we have seen, the drawback of the naive Monte Carlo is that very few failure samples are drawn from the original PDF. Therefore, the authors of [12] used SVM-based classifier as a blockade so that they can skip transistor-level simulations of samples which obviously fall outside the failure region.

The difference between our approach and [12] is that we combine the classifier with the importance sampling. In the context of a failure probability calculation, the variability space is not equally important, i.e. $I(\boldsymbol{x}) \cdot P(\boldsymbol{x})$ represents the importance of the corresponding region. Misclassifications of random samples which are rarely drawn from the alternative distribution have almost no impact on the failure probability calculation. Note that the labels of the training examples are obtained from transistor-level simulations. If we can modify the distribution so that the large number of training examples are available around the important regions, we can improve the efficiency and further reduce the number of transistor-level simulations required to achieve predefined classification performance. Let us recall that the estimated alternative distribution reflects the importance of the variability space. Therefore, by training SVM-based classifier using the samples drawn from the estimated alternative distribution, we can minimize the number of training examples and the number of transistor-level simulations while maintaining the accuracy of the classification.

D. RTN-induced V_{TH} fluctuation model

Fig. 3(a) shows the physical mechanism of RTN. A carrier is captured to or emitted from an electrically active interface trap located inside the gate oxide. This causes the change of surficial potential, which results in the fluctuation of V_{TH}. Fig. 3(b) shows the corresponding V_{TH} fluctuations. When the defect captures the carrier, V_{TH} becomes high and vice versa. The average duration in which V_{TH} is in the higher level is denoted as mean time to emission (τ_e) and that V_{TH} is in the lower level is denoted as mean time to capture (τ_c). The τ_e and τ_c are highly susceptible to the gate bias conditions.

The τ_e and τ_c of a transistor under the switching bias condition with duty ratio α can be represented as follows [13]:

$$\tau_c = \alpha \tau_c^{\mathrm{ON}} + (1-\alpha)\tau_c^{\mathrm{OFF}} \tag{7}$$

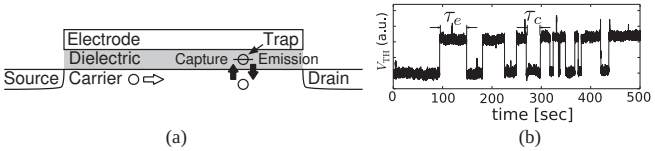


Fig. 3. (a) The origin of RTN and (b) corresponding V_{TH} shift.

and

$$\tau_e = \alpha \tau_e^{\mathrm{ON}} + (1-\alpha)\tau_e^{\mathrm{OFF}}. \tag{8}$$

Here, τ_c^{ON} and τ_e^{ON} are τ_c and τ_e when the transistor is in the "ON" state and τ_c^{OFF} and τ_e^{OFF} are τ_c and τ_e when the transistor is in the "OFF" state. If there are multiple defects, the total V_{TH} shift can be represented as the summation of V_{TH} shifts caused by each trap as

$$\Delta V_{\mathrm{TH}} = \frac{q \cdot N_{\mathrm{eff}}}{C_{\mathrm{ox}} LW}. \tag{9}$$

Here, L and W are the channel length and channel width and C_{ox} is the unit area gate capacitance. q is the elementary charge and N_{eff} is the number of defects which capture the carriers. The probability distribution of N_{eff} can be represented as a following Poisson distribution:

$$N_{\mathrm{eff}} = Pois\left(\frac{\tau_c}{\tau_c + \tau_e}\lambda\right). \tag{10}$$

Here, λ is the total number of defects located in the transistor.

III. Proposed Method

A. Basics of RTN-induced failure probability calculation

Let us modify (1) so that it can take the impact of RTN into account:

$$P_{\mathrm{fail}} = \iint I(\boldsymbol{x}_{\mathrm{RDF}}, \boldsymbol{x}_{\mathrm{RTN}}) P_{\mathrm{all}}(\boldsymbol{x}_{\mathrm{RDF}}, \boldsymbol{x}_{\mathrm{RTN}}) d\boldsymbol{x}_{\mathrm{RDF}} d\boldsymbol{x}_{\mathrm{RTN}}. \tag{11}$$

Here, $\boldsymbol{x}_{\mathrm{RDF}}$ and $\boldsymbol{x}_{\mathrm{RTN}}$ are variables that originate from the fabrication process such as random dopant fluctuation (RDF) and that originate from RTN. $P_{\mathrm{all}}(\boldsymbol{x}_{\mathrm{RDF}}, \boldsymbol{x}_{\mathrm{RTN}})$ is a joint PDF of $\boldsymbol{x}_{\mathrm{RDF}}$ and $\boldsymbol{x}_{\mathrm{RTN}}$. Equation (11) can be modified as follows:

$$P_{\mathrm{fail}} = \int P_{\mathrm{fail}}^{\mathrm{RTN}}(\boldsymbol{x}_{\mathrm{RDF}}) P_{\mathrm{RDF}}(\boldsymbol{x}_{\mathrm{RDF}}) d\boldsymbol{x}_{\mathrm{RDF}} \tag{12}$$

and

$$P_{\mathrm{fail}}^{\mathrm{RTN}}(\boldsymbol{x}_{\mathrm{RDF}}) = \int I(\boldsymbol{x}_{\mathrm{RDF}}, \boldsymbol{x}_{\mathrm{RTN}}) P_{\mathrm{RTN}}(\boldsymbol{x}_{\mathrm{RTN}} | \boldsymbol{x}_{\mathrm{RDF}}) d\boldsymbol{x}_{\mathrm{RTN}}. \tag{13}$$

Here, $P_{\mathrm{RDF}}(\boldsymbol{x}_{\mathrm{RDF}})$ is a PDF over a fabrication-process-induced variability space and it can be represented as a following multivariate standard normal distribution:

$$P_{\mathrm{RDF}}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{0}, \boldsymbol{1}) = (2\pi)^{-D/2} \exp\left(-\frac{1}{2}\sum_{d=1}^{D} x_d^2\right). \tag{14}$$

$P_{\mathrm{RTN}}(\boldsymbol{x}_{\mathrm{RTN}}|\boldsymbol{x}_{\mathrm{RDF}})$ is a conditional PDF over an RTN-induced variability space given $\boldsymbol{x}_{\mathrm{RDF}}$. We here assume that RTN and RDF are independent for simplicity. Therefore, $P_{\mathrm{RTN}}(\boldsymbol{x}_{\mathrm{RTN}}|\boldsymbol{x}_{\mathrm{RDF}})$ equals to $P_{\mathrm{RTN}}(\boldsymbol{x}_{\mathrm{RTN}})$. Note that this assumption does not limit the ability of the proposed method since the proposed method is based on a Monte Carlo approach and we do not require any limitations on the PDF. As we have seen, the RTN-induced V_{TH} shifts can be represented as (9) and (10). Note that we can not represent $P_{\mathrm{RTN}}(\boldsymbol{x}_{\mathrm{RTN}})$ using well known distributions since $\boldsymbol{x}_{\mathrm{RTN}}$ is a random variable drawn from the Poisson distribution multiplied by a constant,

Algorithm 1: Summary of the proposed method

(1) Initial sample selection: Arrange initial particles near the failure boundary.
Repeat steps (2), (3) and (4) untile a sufficient convergence is achieved.
(2) Prediction: Draw candidate particles from the proposal distribution in (15).
(3) Measurement: Calculate weights of each particle according to (16)
(4) Resampling: Resample particles according to the probability proportional to the weights.
Construct alternative distribution (18) for importance sampling using the samples in step (4)
(5) Importance sampling: Draw random samples from the alternative distribution to calculate the failure probability according to (19).

which does not necessary to follow a Poisson distribution.

The proposed method consists of two-stages. In the first stage, the optimal alternative distribution required for the importance sampling is estimated. In order to achieve this goal, we utilize the particle filter in which the alternative distribution is approximated as the density of particles that move in the D-dimensional process variability space. Then, in the second stage, the failure probability is calculated using the estimated alternative distribution. The summary of the proposed method is shown in Algorithm 1. The steps from (2) to (4) correspond to the first stage and the step (5) corresponds to the second stage. In the next subsection, we describe the details of each step.

B. Failure probability calculation

(1) Initial sample selection: Random samples on the surface of a D-dimensional unit sphere are generated. Then, toward the generated radial directions, the boundary of the failure region is searched using bi-section algorithm and allocate candidates of initial particles near the boundary as shown in Fig. 4(a).

The initialization step is conducted for only the first failure probability calculation. The same initial samples are shared among the other calculations with different gate bias conditions to reduce the calculation time.

(2) Prediction step: The candidate particles at next time step $\{\hat{\boldsymbol{x}}_{\mathrm{RDF}}^{(t+1,i)},\ i = 1, 2, \ldots, N\}$ are drawn from a following predictive distribution:

$$\hat{\boldsymbol{x}}_{\mathrm{RDF}}^{(t+1,i)} \sim \frac{1}{N}\sum_{j=1}^{N} \mathcal{N}(\hat{\boldsymbol{x}}_{\mathrm{RDF}}^{(t+1,i)} | \boldsymbol{x}_{\mathrm{RDF}}^{(t,j)}, \boldsymbol{\sigma}). \tag{15}$$

Here, N is the number of particles and $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\sigma})$ is a D-dimensional Gaussian distribution. $\boldsymbol{x}_{\mathrm{RDF}}^{(t,j)}$ is the j-th particle at t-th iteration and $\boldsymbol{\sigma}$ is a diagonal covariance matrix.

(3) Measurement: For $\widehat{\boldsymbol{x}}_{\mathrm{RDF}}^{(t+1,i)}$, corresponding weights are calculated. Here, weight is defined as

$$w(\widehat{\boldsymbol{x}}_{\mathrm{RDF}}^{(t+1,i)}) = P_{\mathrm{fail}}^{\mathrm{RTN}}(\widehat{\boldsymbol{x}}_{\mathrm{RDF}}^{(t+1,i)}) P_{\mathrm{RDF}}(\widehat{\boldsymbol{x}}_{\mathrm{RDF}}^{(t+1,i)}). \tag{16}$$

$P_{\mathrm{fail}}^{\mathrm{RTN}}(\widehat{\boldsymbol{x}}_{\mathrm{RDF}}^{(t+1,i)})$ is calculated using random samples $\{\boldsymbol{x}_{\mathrm{RTN}}^{(m)}, m = 1, 2, \ldots, M\}$ drawn from the Poisson distribution in (9) as follows:

$$\widehat{P}_{\mathrm{fail}}^{\mathrm{RTN}}(\widehat{\boldsymbol{x}}_{\mathrm{RDF}}^{(t+1,i)}) \approx \frac{1}{M}\sum_{m=1}^{M} I(\widehat{\boldsymbol{x}}_{\mathrm{RDF}}^{(t+1,i)}, \boldsymbol{x}_{\mathrm{RTN}}^{(m)}). \tag{17}$$

Here, M is the number of random samples used for the approximation. Note that the Monte Carlo calculation converges fast since we have already located $\widehat{\boldsymbol{x}}_{\mathrm{RDF}}^{(t+1,i)}$ near the failure boundary.

For the computation of $I(\boldsymbol{x})$, transistor-level simulations are required. $N \times M$ samples need to be simulated for the weight calculations of all particles. We reduce the number of required transistor-level simulations with the help of the SVM-based classifier. First, K training examples are randomly selected among $N \times M$ samples and class labels of the selected samples are obtained using transistor-level simulations. Then, the classifier is trained using the K training examples. Finally, the rest of samples are classified using the trained classifier. Therefore, the number of transistor-level simulations can be reduced from $N \times M$ to K.

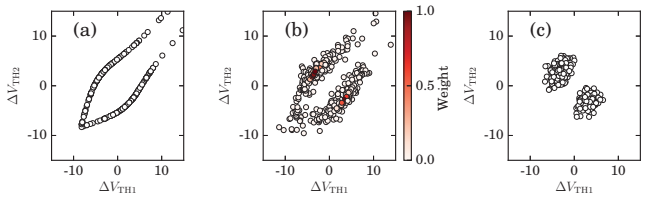


Fig. 4. An example of particle filter based failure region tracking. (a) Particles after initialization step, (b) after prediction and weight calculation steps and (c) after resampling step.

We use a polynomial transform of the D-dimensional variability vector $\boldsymbol{x}$ as feature quantities $\boldsymbol{f}$ in (6) in order to construct a non-linear classification model. For example, if the input vector is a two-dimensional vector $[x_1, x_2]$ and the degree of the polynomial transform D_{poly} is two then the feature vector is $[1, x_1, x_2, x_1x_2, x_1^2, x_2^2]$. In this implementation, D_{poly} is set to be four.

Samples which exist near the separating hyper-plane, i.e. colored region in Fig. 2, may likely to be misclassified. Therefore, such samples should better be classified using transistor-level simulations. However, the weights of particles do not have direct impacts on the failure probability calculation. Instead, it affects the estimation of the optimal alternative distribution and the efficiency of the importance sampling. Therefore, a rough approximation of $I(\boldsymbol{x})$ is sufficient in this step. Hence, we can safely skip the transistor-level simulations and classify all of the $N \times M - K$ samples using the trained classifier to reduce the calculation time.

Fig. 4(b) shows particles after the prediction and measurement steps. The colors of points represent the weights assigned to each particle. We notice that the particles near the origin, i.e. they are more likely to occur, are assigned large weights.

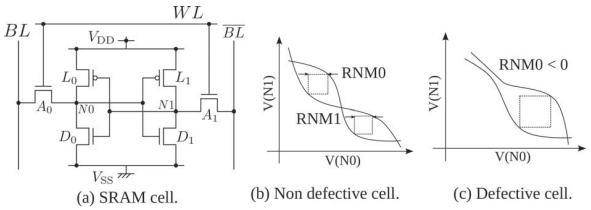
(4) Resampling: Particles at time step $t+1$ are randomly selected from $\widehat{\boldsymbol{x}}_{\mathrm{RDF}}^{(t+1,i)}$ according to the probability that is proportional to the weights assigned to the candidate particles. An example result of the resampling step is shown in Fig. 4(c).

Steps (2) to (4) are repeated to search failure region and to construct the alternative distribution. In our experiment, ten times of repetition is enough for achieving sufficient convergence.

While particle filters drastically speed up the estimation of the alternative distribution, they have drawback that the particles degenerate to a single point in the variability space as the number of resampling increases. In the context of a failure probability calculation of an SRAM cell, there are two major failure regions since an SRAM cell has a symmetric structure. However, due to small differences in weights assigned to particles, the particles concentrate to one of the two regions as the number of iteration increases. This leads to an under estimation of the failure probability.

In the proposed method, we utilize multiple particle filters. The resampling of particles is conducted for each particle filter in order to avoid the concentration of particles. In the example in Fig. 4(c), two major failure regions are tracked by different particle filters.

(5) Importance sampling: Finally, in the second stage, the failure probability is calculated using importance sampling. In order to optimize alternative distribution, outcome of the previous stage is used. Specifically, the distribution of the



(a) SRAM cell. (b) Non defective cell. (c) Defective cell.

Fig. 5. (a) The schematics of the SRAM cell and (b) examples of static noise margin for a non-defective and (c) a defective cell.

particles in step (4) is very close to the optimal distribution, hence it is approximated as:

$$\widehat{Q}(\boldsymbol{x}_{\mathrm{RDF}}) \approx \frac{1}{N}\sum_{i=1}^{N}\mathcal{N}(\boldsymbol{x}_{\mathrm{RDF}}|\boldsymbol{x}_{\mathrm{RDF}}^{(t,i)},\boldsymbol{\sigma}). \tag{18}$$

Then, the failure probability is calculated using random samples $\{\boldsymbol{x}_{\mathrm{IS}}^{(k)},\ k=1,2,\ldots,N_{\mathrm{IS}}\}$ drawn from $\widehat{Q}(\boldsymbol{x}_{\mathrm{RDF}})$ as follows:

$$\widehat{P}_{\mathrm{fail}} \approx \frac{1}{N_{\mathrm{IS}}}\sum_{k=1}^{N_{\mathrm{IS}}} P_{\mathrm{fail}}^{\mathrm{RTN}}(\boldsymbol{x}_{\mathrm{IS}}^{(k)})P_{\mathrm{RDF}}(\boldsymbol{x}_{\mathrm{IS}}^{(k)})/\widehat{Q}(\boldsymbol{x}_{\mathrm{IS}}^{(k)}). \tag{19}$$

Here, N_{IS} is the number of random samples used for the approximation. The calculation of the indicator function is again needed in the evaluation of $P_{\mathrm{fail}}^{\mathrm{RTN}}(\boldsymbol{x}_{\mathrm{IS}}^{(k)})$. We again use the classifier to reduce the number of simulations. Contrary to the classification in the first stage, classification accuracy in the second stage has a direct impact on the accuracy of the failure probability calculation. Therefore, the samples which lie close to the separating hyper-plane go through transistor-level simulations to obtain correct labels. The simulated samples are used to incrementally train the classifier and to increase the classification performance.

IV. Experiment

A. Experimental setup

Fig. 5 shows the circuit schematic of an SRAM cell. In the experiment, failure samples are defined as samples which have negative read noise margin (RNM). RNM is a stability measure of the cell, which can be computed as the maximum side of square embedded within the opening of the butterfly curve [14] of the cell. Fig. 5(b) and (c) show two examples for defective and non-defective cells. The mismatch of driving abilities among transistors result in negative noise margin, which causes the read failure.

We here deal with only V_{TH} variability but other components such as variability of channel size can easily be taken into account in the same way. The variability of V_{TH} originated from a fabrication-process-induced variability is assumed to be represented as a following Gaussian distribution:

$$\Delta V_{\mathrm{TH}}^{\mathrm{RDF}} \sim \mathcal{N}\left(\Delta V_{\mathrm{TH}}^{\mathrm{RDF}}\middle|0,\frac{A_{V_{\mathrm{TH}}}}{\sqrt{L\cdot W}}\right). \tag{20}$$

Here, L and W are the channel length and width. $A_{V_{\mathrm{TH}}}$ is Pelgrom coefficient that is assumed to be equal for pMOS and nMOS transistors in this experiment. The defect density is set to be $\lambda = 4\times10^{-3}\,\mathrm{nm}^{-2}$. This means that the smallest transistor ($W/L = 30\,\mathrm{nm}/16\,\mathrm{nm}$) contains 1.92 defects on average. The 16 nm high-performance model from predictive technology model (PTM) [15] is used as a transistor model. The parameters used in the experiments are summarized in Table I.

B. Experimental results

We first compare the proposed method with one of the state-of-the-art method proposed in [8]. Note that in this experiment, we only take into account the fabrication-process-induced variability since the conventional method can not take into account the RTN-induced variability. Fig. 6(a) shows the calculated failure probability of the SRAM cell and the number of transistor-level simulations required using the conventional method (black line) and using the proposed method (red line). The filled regions represent the 95% confidence intervals. We can see that the proposed method converges with significantly smaller number of simulations than the conventional method. Fig. 6(b) shows the relative error defined as the ratio of the 95% confidence interval to the estimated failure probability. In this experiment, the proposed method required 36 times less the number of transistor-level simulations than the conventional method [8] to achieve relative error of 1%. The calculation time required to achieve that accuracy is about 620 seconds. This includes the times for classifier training and sample classification. The conventional method requires about 9 700 seconds to achieve equal accuracy, which corresponds to about 15.6× speed-up.

TABLE I. Experimental conditions.

	Load (L_i)	Driver (D_i)	Access (A_i)
$A_{V_{\mathrm{TH}}}$ [mV·nm]	5×10^2		
Channel length [nm]	16		
Channel width [nm]	60	30	30
t_{ox} [nm]	0.95		
λ [nm^{-2}]	4×10^{-3}		
τ_e^{on}	1.2		
τ_e^{off}	0.1		
τ_c^{on}	0.01		
τ_c^{off}	0.12		

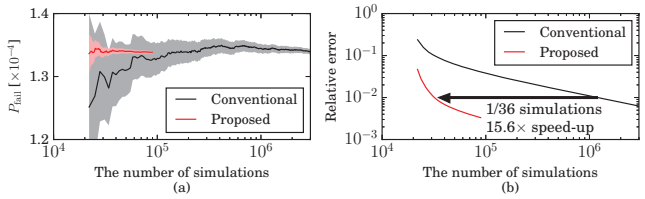


Fig. 6. The comparison of the proposed method and the conventional method [8].

We then validate the accuracy of proposed method when both the fabrication-process-induced and the RTN-induced variability are taken into account using the result of naive Monte Carlo method as the reference. Fig. 7(a) shows the

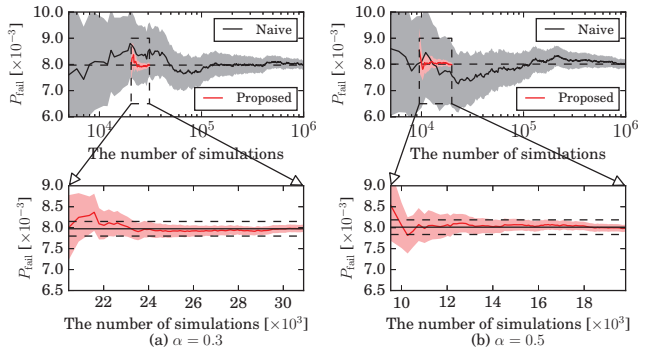


Fig. 7. The comparison of the proposed method and the naive Monte Carlo. The duty cycle is set to be (a) 0.3 and (b) 0.5. The 95% confidence interval is indicated as colored region.

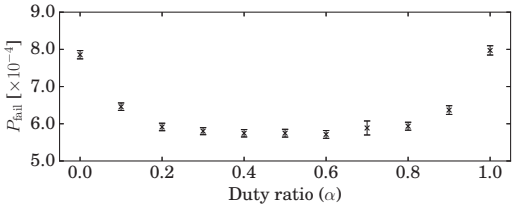


Fig. 8. The relationship between failure probability and duty ratio α.

calculated failure probability of the SRAM cell. Here, duty ratio α is set to be 0.3. The black and red lines show the failure probabilities calculated using the naive Monte Carlo and the proposed method, respectively. The filled regions correspond to 95% confidence interval. The supply voltage is dropped to 0.5 V so that the naive Monte Carlo method can obtain a good convergence. The lower part of Fig. 7(a) is the magnified view of the estimation result of the proposed method. The solid horizontal line and the dashed lines represent the calculated failure probability and 95% confidence interval obtained from 10^6 Monte Carlo trials, respectively. From Fig. 7(a), the proposed method required approximately 24k simulations to achieve equal accuracy as that of 10^6 Monte Carlo trials. This corresponds to $40\times$ speed-up. Note again that the failure probability is adjusted to be high. In more realistic situations where failure probability of an SRAM cell is very low, calculation using the naive Monte Carlo is impossible.

The comparison of two methods are conducted for the other gate bias settings. Fig. 7(b) shows the calculated failure probability when the duty ratio is set to be 0.5. In this case, the same initial particles are used as that of previous calculation. We can see that roughly half of the number of transistor-level simulations is sufficient to achieve equal accuracy as that shown in Fig. 7(a).

Based upon the developed method, it becomes possible to study the effect of RTN. We here investigate how duty ratio α affects the failure probability of the SRAM cell for the first time. Fig. 8 shows the relationship between failure probabilities and duty ratios calculated using the proposed method. The sample points and error-bars show the calculated failure probability and 95% confidence intervals, respectively. In this example, the failure probability reaches a minimum when the duty ratio equals to 0.5, i.e. the cell stores "0" and "1" at the same probability. We also notice that Fig. 8 displays almost bilateral symmetry. This comes from a symmetric structure of an SRAM cell. The failure probability calculated without the impact of RTN was 1.33×10^{-4}, which indicates that the conventional methods that ignore the impact of RTN give a six times optimistic estimation. The consideration of the impact of RTN is definitely required to realize accurate reliability design of modern circuits.

The total number of simulations required to obtain Fig. 8 was about 2×10^5 with our method. Let us recall that the failure probabilities in Fig. 7 were $10\times$ higher than that of Fig. 8 and that 10^6 samples were required for naive Monte Carlo. Considering the square root dependence of the accuracy improvement to the number of samples, the naive Monte Carlo method would require at least 1.1×10^9 ($= 10^8 \times 11$ bias conditions) samples to obtain Fig. 8. Hence, the proposed method achieves over $5500\times$ speed-up compared to the naive method.

V. Conclusion

In this paper, we proposed a novel SRAM failure probability calculation method which can take into account the impact of RTN. What makes the consideration of RTN difficult is that the statistics of RTN has gate bias dependence and multiple failure probability calculations are required to ensure an SRAM cell reliability. We introduced particle filters in order to estimate the optimal alternative distribution for importance sampling. Combined with a classifier to reduce the number of required transistor-level simulations, the proposed method achieved $15.6\times$ speed-up compared to one of the state of the art method [8]. The relationship between gate bias conditions and the failure probability of the SRAM cell was presented for the first time. In our experiment, the failure probability of the SRAM cell calculated with consideration of the impact of RTN is about six times higher than that calculated without the impact of RTN, which indicates that the impact of RTN must be taken into account for ensuring the modern circuit reliability.

Acknowledgment

This work was partially supported by a Grant-in-Aid for JSPS Fellows and MEXT/JSPS KAKENHI Grant No. 26280014. The authors also acknowledge support from VDEC with the collaboration with Synopsys Corporation.

References

[1] N. Tega, H. Miki, F. Pagette, D. Frank, A. Ray, M. Rooks, W. Haensch, and K. Torii, "Increasing threshold voltage variation due to random telegraph noise in FETs as gate lengths scale to 20 nm," in *Symp. on VLSI Technology*, 2009, pp. 50–51.

[2] Y. Ye, C.-C. Wang, and Y. Cao, "Simulation of random telegraph Noise with 2-stage equivalent circuit," in *Int. Conf. on Comput.-Aided Design*, 2010, pp. 709–713.

[3] K. Aadithya, S. Venogopalan, A. Demir, and J. Roychowdhury, "MUSTARD: A coupled, stochastic/deterministic, discrete/continuous technique for predicting the impact of Random Telegraph Noise on SRAMs and DRAMs," in *Design Automation Conf.*, 2011, pp. 292–297.

[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," in *Design Automation Conf.*, 2006, pp. 69–72.

[5] J. Jaffari and M. Anis, "Adaptive sampling for efficient failure probability analysis of SRAM cells," in *Int. Conf. on Comput.-Aided Design*, 2009, pp. 623–630.

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

[7] C. Dong and X. Li, "Efficient SRAM Failure Rate Prediction via Gibbs Sampling," in *Design Automation Conf.*, 2011, pp. 200–205.

[8] K. Katayama, S. Hagiwara, H. Tsutsui, H. Ochi, and T. Sato, "Sequential importance sampling for low-probability and high-dimensional SRAM yield analysis," in *Int. Conf. on Comput.-Aided Design*, 2010, pp. 703–708.

[9] N. Gordon, D. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *Radar and Signal Processing, IEE Proceedings F*, vol. 140, no. 2, pp. 107–113, 1993.

[10] G. Kitagawa, "Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models," *J. Comput. and Graphical Stat.*, vol. 5, no. 1, pp. 1–25, 1996.

[11] C. Cortes and V. Vapnik, "Support-Vector Networks," *Mach. Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[12] 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 Integr. Circuits Syst.*, vol. 28, no. 8, pp. 1176–1189, 2009.

[13] X. Chen, Y. Wang, Y. Cao, and H. Yang, "Statistical analysis of random telegraph noise in digital circuits," in *Asia and South Pacific Design Automation Conf.*, 2014, pp. 161–166.

[14] E. Seevinck, F. List, and J. Lohstroh, "Static-noise margin analysis of MOS SRAM cells," *IEEE J. Solid-State Circuits*, vol. 22, no. 5, pp. 748–754, 1987.

[15] Nanoscale Integration and Modeling (NIMO) Group, ASU, "Predictive Technology Model (PTM)," http://ptm.asu.edu/.