Adaptive Clustering and Sampling for High-Dimensional and Multi-Failure-Region SRAM Yield Analysis

[1,2]Xiao Shi, [3]Hao Yan, [3]Jinxin Wang, [3]Xiaofen Xu, [3]Fengyuan Liu, [3]Longxing Shi, [1,2]Lei He
[1]State Key Lab of ASIC & System, Microelectronics Dept., Fudan University, China
[2]Electrical and Computer Engineering Dept., University of California, Los Angeles, CA, USA
[3]Electrical Engineering Dept., Southeast University, China
pokemoon2009@g.ucla.edu,yanhao@seu.edu.cn,jxwang1995@gmail.com,xxf_deborah@seu.edu.cn,liu_lfy@seu.edu.cn,lxshi@seu.edu.cn,lhe@ee.ucla.edu

ABSTRACT

Statistical circuit simulation is exhibiting increasing importance for memory circuits under process variation. It is challenging to accurately estimate the extremely low failure probability as it becomes a high-dimensional and multi-failure-region problem. In this paper, we develop an Adaptive Clustering and Sampling (ACS) method. ACS proceeds iteratively to cluster samples and adjust sampling distribution, while most existing approaches pre-decide a static sampling distribution. By adaptively searching in multiple cone-shaped subspaces, ACS obtains better accuracy and efficiency. This result is validated by our experiments. For SRAM bit cell with single failure region, ACS requires 3-5X fewer samples and achieves better accuracy compared with existing approaches. For 576-dimensional SRAM column circuit with multiple failure regions, ACS is 2050X faster than MC without compromising accuracy, while other methods fail to converge to correct failure probability in our experiment.

CCS CONCEPTS

• **Hardware → Yield and cost modeling**;

KEYWORDS

Process Variation; Failure Probability; SRAM; High Dimension; Failure Regions

ACM Reference Format:
Xiao Shi, Hao Yan, Jinxin Wang, Xiaofen Xu, Fengyuan Liu, Longxing Shi, Lei He. 2019. Adaptive Clustering and Sampling for High-Dimensional and Multi-Failure-Region SRAM Yield Analysis. In 2019 International Symposium on Physical Design (ISPD'19), April 14–17, 2019, San Francisco, CA, USA. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3299902.3309748

1 INTRODUCTION

As microelectronic devices shrink to deep submicrometer scale, circuit reliability has become an area of growing concern for efficient circuit sizing and design. Among various integrated circuits (ICs), SRAM circuits are highly duplicated with minimum size devices that require extremely small failure probability [1]. This failure probability is a rare event that deterministic analysis is infeasible.

In general, modern statistical circuit simulation methods are applied to model the stochastic behavior under process variation. Standard Monte Carlo (MC) method remains the gold standard, which repeatedly collects samples and evaluates circuit performance with transistor-level simulation. Although circuit simulation has been remarkably accelerated through the years [2], MC is extremely time-consuming under the "rare-event" scenario because millions of simulations are required to capture one single failure event.

Instead of sampling randomly with standard MC, more efficient approaches have been proposed to sample from the likely-to-fail region, which can be grouped into two major categories:
(1) **Classification:** Statistical Blockade (SB) [3] constructs a classifier to filter out the samples that unlikely to fail and only simulate the remaining samples. A safety margin is applied in [3] to decrease classification error. More recently, recursive SB [4] and REscope [5] improve the classifier to conditional classifier and SVM nonlinear classifier. However, training such classifiers is expensive in high dimension and the effectiveness deteriorates very quickly with extremely small failure probability.
(2) **Importance Sampling:** As a classic modification of MC method, Importance Sampling (IS) tries to build a distorted sampling distribution to assemble the failure region. For example, Mixture Importance Sampling (MixIS) [6] uses the mixture of a uniform distribution, the original distribution and a shifted distribution centered at the failure region as target distribution. Norm Minimization (MNIS) [7] and Spherical Sampling (SS) [8] methods spherically search the parametric space and then shift the sampling distribution toward the minimum L_2-norm point of a set of failure samples. In order to tackle multi-failure-region circuit cases, methods in [9, 10] attempt to construct multiple shift vectors and perform mixture importance sampling. The drawbacks of these approaches is that IS is inefficient to collect samples and evaluate yield rate, and the effectiveness is highly dependent on the quality of distorted sampling distribution.

Among others, Particle Filter [11] utilizes a resampling step followed by an IS step to accelerate failure region exploration and failure rate evaluation. Adaptive Impotance Sampling (AIS) [12] method takes one step forward. AIS develops an unbiased estimator along with the resampling iterations, which can eliminate the time consuming static IS step. However, these modified IS methods suffer from sample diversity degeneracy. In the multi-failure-region circuit case, the resampling scheme is more prone to converge to the region with higher importance, at the expense of neglecting less important

ISPD '19, April 14–17, 2019, San Francisco, CA, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6253-5/19/04…$15.00
https://doi.org/10.1145/3299902.3309748

regions. This property leads to biased sampling distribution, and it comes with smaller failure probability estimation.

In this paper, we present an accurate and efficient algorithm based on Adaptive Clustering and Sampling (ACS) method to estimate the failure rate of high-dimensional and multi-failure-region circuit cases. The basic idea of the algorithm is to cluster failure samples and build global sampling distribution at each iteration. Specifically, in clustering step, we propose a multi-cone clustering method, which partitions the parametric space and clusters failure samples. Then global sampling distribution is constructed from a set of weighted Gaussian distributions. Next, we calculate importance weight for each sample based on the discrepancy between sampling distribution and target distribution. Failure probability is updated at the end of each iteration. This clustering and sampling procedure proceeds iteratively until all the failure regions are covered.

Our main contribution is summarized in three aspects: first, we initialize our ACS algorithm with hyperspherical presampling method, which reduces dimension by sampling from a set of hyperspherical surfaces. Second, we propose an adaptive scheme to explore high-dimensional space and search for failure regions. As iteration continues, each partial IS estimator provides a better estimation, and global sampling distribution will tilt toward failure region. Moreover, our estimator is adapted parallelly in different directions, which can effectively improve sample diversity.

2 BACKGROUND

2.1 Rare Event Analysis

Let $f(x)$ denote multivariate probability density function (PDF) of circuit process variation x. Let Y be the observed performance metric, such as memory read/write time, amplifier gain, etc. This metric Y usually requires expensive transistor-level circuit simulation to evaluate.

In statistical circuit simulation, it is of great interest to estimate the probability of Y belonging to a subset S of the entire parametric space. For example, under circuit failure probability estimation scenario, we generally assume the performance specification is $Y \notin S$. On the contrary, a failure occurs when $Y \in S$. Thereby, we introduce indicator function $I(x)$ to identify pass/fail of Y:

$$I(x) = \begin{cases} 0, & if \quad Y \notin S \\ 1, & if \quad Y \in S \end{cases} \tag{1}$$

Therefore, the probability P_{fail} can be calculated as

$$P_{fail} = P(Y \in S) = \int I(x) \cdot f(x) dx \tag{2}$$

Note that the integral in equation (2) is intractable because $I(x)$ is unavailable in analytical form. Conventionally, Monte Carlo (MC) method enumerates a sample set $\{X_i\}_{i=1}^N$ according to $f(x)$ and evaluates their indicator values $\{I(X_i)\}_{i=1}^N$ to generate an unbiased estimation of $\hat{P}_{fail}$:

$$\hat{P}_{fail} = \hat{P}(Y \in S) = \frac{1}{N} \sum_{i=1}^{N} I(X_i) \xrightarrow{N \to +\infty} P(Y \in S) \tag{3}$$

2.2 Importance Sampling

When P_{fail} is an extremely small value, standard MC becomes inefficient because it requires millions of simulations to collect one single failure event. To avoid massive simulations, an intuitive idea is to sample from a "distorted" sampling distribution $g(x)$ that tilts toward the failure region S.

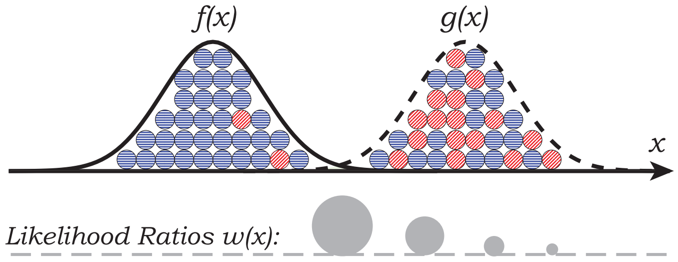

Figure 1: Scale illustration of likelihood ratios in mean-shift importance sampling

Figure 1 illustrates a one-dimensional example of IS method. It samples from a shifted distribution $g(x)$ which contains more failure samples. Failure probability can thus be calculated as:

$$P_{fail} = P(Y \in S) = \int I(x) \cdot \frac{f(x)}{g(x)} \cdot g(x) dx \tag{4}$$

$$= \int I(x) \cdot w(x) \cdot g(x) dx \tag{5}$$

If shifted PDF $g(x)$ is properly chosen, $\hat{P}_{IS,fail}$ can converge to failure probability P_{fail} with much smaller sample size because failure in $g(x)$ is not rare. Theoretically, the optimal sampling distribution $g^{opt}(x)$ is the failure event distribution, which can be expressed as:

$$g^{opt}(x) = \frac{I(x) \cdot f(x)}{P_{fail}} \tag{6}$$

However, $g^{opt}(x)$ cannot be calculated with (7) deterministically because the analytical form of $I(x)$ is unavailable and P_{fail} is unknown. In practice, people propose various approaches to build $g^{opt}(x)$ with optimal shift vector. For example, MNIS [7] shifts $f(x)$ to pass/fail boundary, HDIS [13] shifts to centroid of a cluster of failure samples, and HSCS [9] shifts to multiple centroids by clustering failure samples. However, static IS with predefined $g(x)$ suffers from two major drawbacks:

First, existing IS approaches consist of two stages: a presampling stage to calculate mean shift vector and importance sampling stage to evaluate P_{fail}. However, the complexity of constructing optimal shift vector increases exponentially with circuit dimension. A classic modification of static IS is applying adaptive scheme to search for failure regions.

Moreover, the likelihood ratio $w(x)$ is substantially biased when the discrepancy between $f(x)$ and $g(x)$ is huge. In this case, only a few samples with larger weight contribute to estimation, which results in higher variance and unstable performance.

3 ADAPTIVE CLUSTERING AND SAMPLING ALGORITHM

3.1 Algorithm Description

Algorithm 1 summarizes the main steps of proposed ACS method. The objective of this algorithm is to iteratively explore multiple failure regions in different directions and update failure rate at the end of each iteration. In the initialization step, we collect M samples $\{X_i^{(0)}\}_{i=1}^{M}$ by hyperspherical presampling, and group these samples into k clusters $\{C_j^{(0)}\}_{j=1}^{k}$ using our multi-cone clustering algorithm. At each iteration t, we construct a local sampling distribution $g_j^{(t-1)}(x)$ in each cluster $C_j^{(t-1)}$ based on the samples that assigned to it. Our global sampling distribution $g^{(t-1)}(x)$ is built from a weighted mixture of all the local sampling distributions $\{g_j^{(t-1)}(x)\}_{j=1}^{k}$. Next we generate M new samples $\{X_i^{(t)}\}_{i=1}^{M}$ from $g^{(t-1)}(x)$, and calibrate the cluster sets $\{C_j^{(t)}\}_{j=1}^{k}$. At the end of each iteration, we update failure probability estimation by averaging all the partial IS estimators $\hat{P}_{fail,t} = \frac{1}{tM}\sum_{l=1}^{t}\sum_{i=1}^{M} w_{i,l}$ up to present. This iteration proceeds iteratively until our estimation converge to certain confidence interval.

3.2 Hyperspherical Presampling

One major challenge in implementing our ACS algorithm is to generate some initial samples that can locate multiple failure regions. In practice, an effective initialization method can help improve sample diversity, convergence speed, and capability to explore parametric space. It is quite difficult to recover from a set of poor starting samples, and the adaptation may converge to a local optimum in the parametric space.

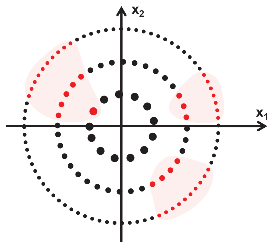

Figure 2: Incremental hyperspherical presampling (for finding initial samples to enable local exploration)

In order to develop a good initialization, as demonstrated in Figure 2, we implement a hyperspherical presampling procedure. This procedure starts with circuit nominal distribution, which indicates a unit hypersphere in parametric space. The radius of this unit hypersphere denotes the variance of circuit parameters. To cover multiple failure regions, we gradually increase the radius of hypersphere until M failure samples are captured. Our presampling method is a dimension reduction process by restricting the samples on a union of hyperspherical surfaces. This method can accelerate the exploration in the high dimensional space while maintaining sample diversity.

Algorithm 1: ACS Algorithm

```
Initialization:
1. Set iteration index $t = 0$, generate initial failure sample
   set $\{X_i^{(0)}\}_{i=1}^{M}$.
2. Assign failure samples $\{X_i^{(0)}\}_{i=1}^{M}$ to cluster set $\{C_j^{(0)}\}_{j=1}^{k}$.
   Define $N_j^{(0)}$ the number of failure samples in the cluster
   $C_j^{(0)}$, we note that $\sum_{j=1}^{k} N_j^{(0)} = M$.
Repeat
Update iteration index $t = t + 1$.
    1. Cluster sampling distribution:
    In each cluster:
    (a) Construct Gaussian distribution as sample proposal:
            $q_i^{(t-1)}(x) = N(X_i^{(t-1)}, \Sigma)$
        where $\Sigma$ is predefined covariance matrix.
    (b) Calculate probability density for $N_j^{(t)}$ failure samples:
            $\beta_{i,t-1} = f(X_i^{(t-1)})I(X_i^{(t-1)}) \qquad i = 1, \ldots, N_j^{(t)}.$
    (c) Construct local distribution $g_j^{(t-1)}(x)$:
            $g_j^{(t-1)}(x) = \frac{1}{\sum_{i=1}^{N_j} \beta_{i,t-1}} \sum_{i=1}^{N_j} \beta_{i,t-1} \cdot q_i^{(t-1)}(x)$
    2. Sample propagation:
    Generate M samples from global distribution $g^{(t-1)}(x)$:
        $X_i^{(t)} \sim g^{(t-1)}(x) = \frac{1}{\sum_{i=1}^{M} \beta_{i,t-1}} \sum_{j=1}^{k} \Big( \sum_{X_i \in C_j^{(t)}} \beta_{i,t-1} \Big) \cdot g_j^{(t-1)}(x)$
    3. Cluster calibration:
    Re-cluster current failure samples $\{X_i^{(t)}\}_{i=1}^{M}$ into new
    set$\{C_j^{(t)}\}_{j=1}^{k}$ and update $\{N_j^{(t)}\}_{j=1}^{k}$.
    4. Failure probability calculation:
    (a) Compute incremental importance weight:
            $w_{i,t} = \frac{\pi(x)}{g^{(t-1)}(x)} = \frac{f(x)I(x)}{g^{(t-1)}(x)} \qquad i = 1, \ldots, M.$
    (b) Update unbiased estimator using all samples up to present
        iteration:
            $\hat{P}_{fail,t} = \frac{1}{tM} \sum_{l=1}^{t} \sum_{i=1}^{M} w_{i,l}$
Until
Relative standard deviation (FOM): $\rho = \frac{\sqrt{\sigma^2_{\hat{P}_{fail}}}}{\hat{P}_{fail}} \leq 0.1$
```

In Algorithm 1, the number of initial failure samples, M, can be arbitrarily user specified. We note that there exists a trade-off between estimation accuracy and algorithm complexity: the larger M is, the higher estimation accuracy we can achieve while sacrificing simulation runtime. In our experiments, we collect 100 samples on each hypersphere until 10% or more samples fail.

3.3 Multi-Cone Clustering

After we collect a set of failure samples located on several discrete hyperspherical surfaces with different radius, we need to cluster these samples with the boundary of multiple disjoint failure regions. Conventional clustering algorithms apply techniques such as graph-based methods, density-based methods and boundary-based methods. They group sample points into optimal clusters by evaluating the Euclidean distance between sample pairs, as defined in (8).

$$EuclideanDistance(X^{(1)}, X^{(2)}) = \|X^{(1)} - X^{(2)}\| \tag{7}$$

However, clustering samples that are randomly distributed in high dimensional open space is challenging. As the dimension of parametric space increases, the ratio of Euclidean distance between nearest and farthest neighbors is closer to 1. In such cases, the nearest neighbor problem becomes ill-defined and qualitative clustering methods are unapproachable.

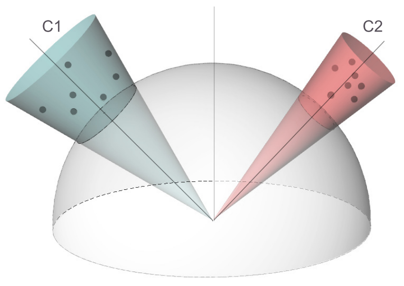

Figure 3: Partition the space into non-overlapping cones along radial directions.

Alternatively, at each iteration, we cluster the sample points based on their directions rather than Euclidean distance. We partition the high-dimensional space into a union of disjoint cones, while maintaining whole space coverage. For each cone-shaped subspace, we build a nonlinear mapping from sample space to feature space. As illustrated in Figure 3, all distinct sample points in each cone are projected to the unit hypersphere surface in the radial direction. We note that the location of images projected on the unit hypersphere describes the direction of sample points. The distance metric of each sample pair is evaluated by the cosine distance of direction vectors, as defined in (9).

$$CosineDistance(X^{(1)}, X^{(2)}) = 1 - \frac{X^{(1)} \cdot X^{(2)}}{\|X^{(1)}\|\|X^{(2)}\|} \tag{8}$$

As shown in Algorithm 2, a k-means based algorithm is implemented to cluster the failure samples $\{X_i\}_{i=1}^M$ according to their direction. The algorithm proceeds as follows. We first generate k unit length vectors $\{V_j\}_{j=1}^k$ as initial direction vectors arbitrarily, where k is a user specified parameter. Each sample X_i is then assigned to the closest cluster C_{λ_i} according to the cosine distance. Next, the direction vector of each cluster is updated to the average of associated sample vectors. This assignment and update procedure is repeated until cluster membership stabilizes.

Although k-means algorithm has advantages of its simplicity to implement and efficiency, it searches for clusters in a greedy fashion,

Algorithm 2: Multi-cone Clustering Algorithm

```
Input:
    Failure sample set: $\{X_i\}_{i=1}^M$
    Initial cluster number: $k$
Output:
    Cluster label for samples: $\{\lambda_i\}_{i=1}^M$
Initialization:
    Randomly initialize $k$ unit length direction vectors
    $\{V_j\}_{j=1}^k$ and corresponding empty clusters $\{C_j\}_{j=1}^k$.
Repeat
  (1) For each sample $X_i$, calculate cosine distance with all the
      direction vectors $\{V_j\}_{j=1}^k$:
          $CosineDistance(X_i, V_j) = 1 - \frac{X_i \cdot V_j}{\|X_i\|\|V_j\|}$.
      Update $\lambda_i = \mathrm{argmin}_j\, CosineDistance(X_i, V_j)$ and assign $X_i$
      to cluster $C_{\lambda_i}$.
  (2) For each cluster $C_j$, update its direction vector:
          $V_j = \frac{1}{|C_j|} \sum_{X \in C_j} X$.
      where $|C_j|$ denotes the number of samples in cluster $C_j$.
Until
    Sample labels $\{\lambda_i\}_{i=1}^M$ remain unchanged.
```

which makes it sensitive to bad initialization and outliers. In our experiment, we start from multiple set of initial direction vectors, and minimize sum of cosine distance as error function. Thus, our clustering result is more robust and more prone to converge to global optimum rather than local optimum.

We also note that the number of clusters, k, can be tuned to improve the effectiveness of our k-means algorithm. Larger k always improves the cluster cohesiveness by decreasing squared error, but this comes at the expense of higher computational cost. We explore this trade-off by utilizing different k values. In practice, k is fixed as $\sqrt{M}$, where M is the number of failure samples to be clustered.

3.4 ACS Estimator Analysis

3.4.1 *Unbiasedness of ACS Estimator*.

In this section, we prove that our ACS estimator $\hat{P}_{fail}$ is unbiased between iterations. The unbiasedness of estimator is evaluated by its expected value. It guarantees the estimation built from random measure $\{X_i^{(t)}, w_{i,t}\}_{i=1}^M$ is consistent, and it converges to failure probability P_{fail}.

$$E[\hat{P}_{fail}] = E[\frac{1}{tM} \sum_{l=1}^{t} \sum_{i=1}^{M} w_{i,l}] \tag{9}$$

$$= \frac{1}{tM} \sum_{l=1}^{t} \sum_{i=1}^{M} E[\frac{f(x)I(x)}{g^{(l-1)}(x)}] \tag{10}$$

$$= \frac{1}{tM} \sum_{l=1}^{t} \sum_{i=1}^{M} \int \frac{f(x)I(x)}{g^{(l-1)}(x)} g^{(l-1)}(x)dx \tag{11}$$

$$= \frac{1}{tM} \sum_{l=1}^{t} \sum_{i=1}^{M} \int f(x)I(x)dx = P_{fail} \tag{12}$$

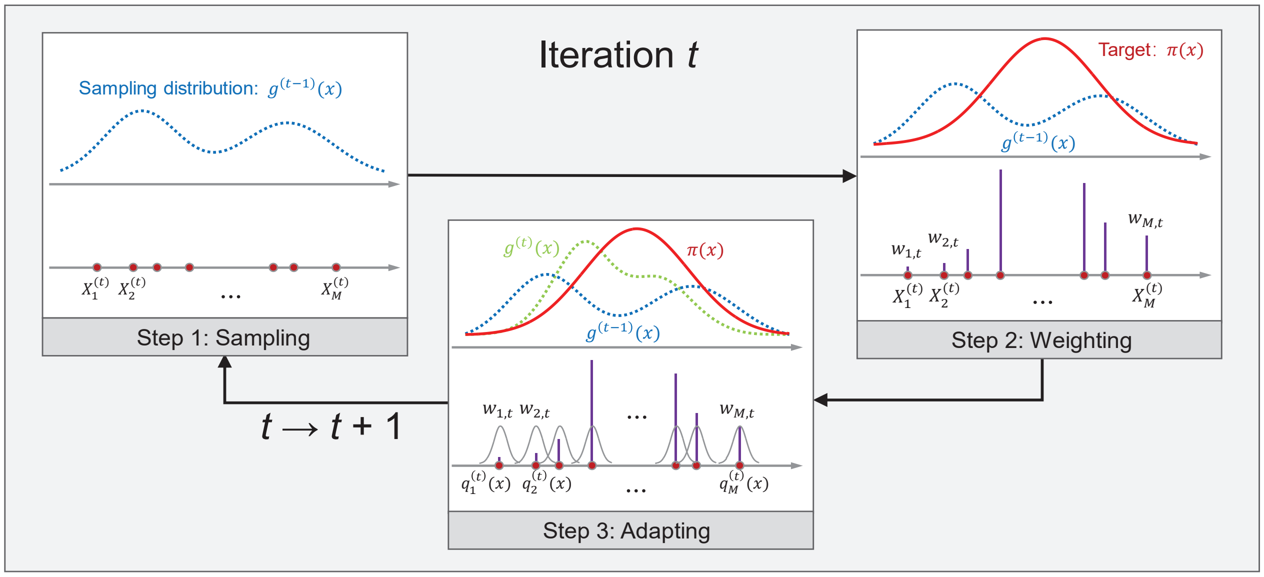
Iteration t
Sampling distribution: $g^{(t-1)}(x)$
$X_1^{(t)}$ $X_2^{(t)}$ ... $X_M^{(t)}$
Step 1: Sampling
Target: $\pi(x)$
$g^{(t-1)}(x)$
$w_{1,t}$ $w_{2,t}$ $w_{M,t}$
Step 2: Weighting
$g^{(t)}(x)$
$\pi(x)$
$q_1^{(t)}(x)$ $q_2^{(t)}(x)$ ... $q_M^{(t)}(x)$
Step 3: Adapting
$t \rightarrow t+1$

Figure 4: Flow diagram that shows the adaptation of ACS estimator. The target distribution are shown by solid lines, while the sampling distributions are plotted with dashed lines. The initial sampling distribution gradually tilts toward target distribution by reweighting sample proposals.

3.4.2 *Adaptation of ACS Estimator*.

The ACS methodology is based on an iterative process which proceeds parallelly in multiple disjoint clusters. With the adaptation of ACS estimator, our sampling distribution $g^{(t)}(x)$ gradually evolves to accurately approximate the target probability density $\pi(x)$. As demonstrated in Algorithm 1, this procedure consists of three main stages: generating samples from sampling distribution (sampling), calculation of the incremental importance weight for each of the samples (weighting) and updating the parameters to define the new sampling distribution for next iteration (adapting). Figure 4 shows the flow diagram of the stages in ACS.

The adaptive mechanism of ACS is driven by the uncertainty in the partial IS estimators, which can be quantified by their variance. To be specific, each sample forms a sample proposal $q_i^{(t)}(x)$ that can describe local features of the target distribution $\pi(x)$. In order to obtain a discrete probability distribution that approximates target distribution, we introduce incremental importance weight $w_{i,t} = \frac{\pi(x)}{g^{(t-1)}(x)}$, which quantifies the discrepancy between target distribution $\pi(x)$ and current sampling distribution $g^{(t-1)}(x)$. Then we average out all the sample proposals $q_i^{(t)}(x)$ based on their $w_{i,t}$ and generate new sampling distribution $g^{(t)}(x)$ for next iteration. This concept is very similar to the methodology in Kernel Density Estimation (KDE) method, where each $q_i^{(t)}(x)$ represents a kernel. As iteration continues, more observations will be added to target distribution $\pi(x)$ to make it closer to real failure region distribution. And our sampling distribution approximates the target distribution $\pi(x)$ through a random walk, as shown in Figure 4. With this ACS adaptation scheme, our sampling distribution can search to achieve full failure region coverage, and focus on the most important failure boundaries.

3.4.3 *Comparison with other IS estimators*.

In comparison with conventional static IS with deterministic mixture, our ACS algorithm has two primary advantages. First of all, our ACS estimator can spread out throughout the parametric space while maintaining sample diversity. To be specific, a major challenge for other AIS methodologies in statistics community is the diversity of samples is prone to degenerate as iteration proceeds. These approaches utilize a resampling scheme to sample with replication from a series of sample proposals. Resampling method tilts to approximate failure event distribution. However, this random walk is generated globally in a greedy fashion, which may focus on the major failure regions and neglect the minor ones. And it also yields higher computational complexity. Our ACS estimator, on the contrary, is completely parallelized in different directions, and the evaluation of failure probability is also performed in parallel. It accelerates the convergence of our estimation and all the failure regions on different directions are separately protected between iterations.

Moreover, ACS estimator outperforms other IS based estimator in terms of Effective Sample Size (ESS). ESS is defined as:

$$ESS = \frac{1}{\sum_{i=1}^{M} (\bar{w}_{i,t})^2} \tag{13}$$

where $\bar{w}_{i,t}$ is the normalized incremental importance weight for each sample in all iterations. It reflects the number of samples that contribute to corresponding estimator. Our ACS estimator bias the sampling distribution to the samples with larger $\bar{w}_{i,t}$. In the next iteration, more samples with larger weight will be generated. Thus $\bar{w}_{i,t}$ is naturally balanced and it is more likely to converge to optimal $w^* = \frac{1}{M}$, which maximizes ESS to M. In this case, each sample uniformly contributes to the estimator and the estimator is stabilized.

Table 1: Accuracy and efficiency comparison on 18-dimensional SRAM bit cell

	MC	HSCS	AIS	Proposed
Failure prob.(error)	1.24e-5(0%)	1.18e-5(4.6%)	1.32e-5(6.1%)	1.22e-5(1.5%)
Presampling # sim.	0	7000	3000	1100
Importance Sampling # sim.	1e7	8688	5111	1736
Total # sim. (speedup)	1e7(1X)	15688(637X)	8111(1233X)	2836(3526X)

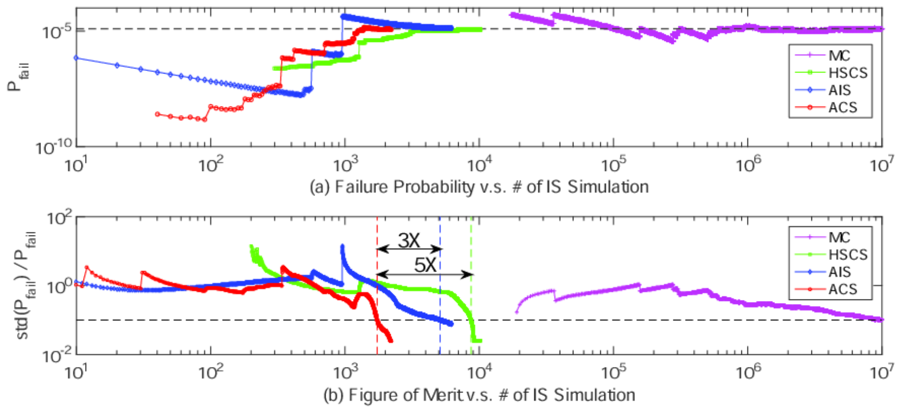

Figure 5: Evolution comparison of failure prob. and FOM on SRAM bit cell

4 EXPERIMENT RESULT

In this section, we first evaluate our proposed ACS method on a typical SRAM bit cell with 18 variables. More realistically, we verify ACS on high-dimensional SRAM column with 576 variables. We also implement different methods, including MC, HSCS [9] and AIS [12], and compare from both accuracy and efficiency perspective. The experiment environment is HSPICE with SMIC 40nm model.

4.1 Experiments on 6T SRAM Bit Cell

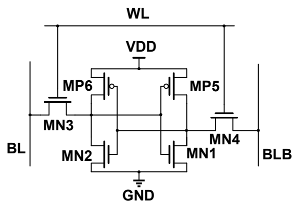

Figure 6: The schematic of typical 6T SRAM cell

Figure 6 shows the schematic of a typical 6-transistors SRAM bit cell. Four transistors $MN1$, $MP5$, $MN2$ and $MP6$ form two cross-coupled inventers and utilize two steady states '0' and '1' to store data in the memory cell. The other two access transistors MN3 and MN4 work as switches for read and write operation. In this experiment, we consider various failure mechanisms, including reading failure, writing failure and data retention failure. We will compare different methods (MC, HSCS, AIS, proposed) in terms of accuracy and efficiency.

4.1.1 *Accuracy Comparison.*

To verify the accuracy of proposed ACS algorithm, Figure of Merit (FOM), ρ, is applied to represent the accuracy convergence and confidence of estimation. The definition of FOM is:

$$\rho = \frac{\sqrt{\sigma^2_{\hat{P}_{fail}}}}{\hat{P}_{fail}} \tag{14}$$

where $\hat{P}_{fail}$ indicates failure probability and $\sigma_{\hat{P}_{fail}}$ indicates the standard deviation of $\hat{P}_{fail}$. We define one estimation has $(1-\epsilon)100\%$ accuracy with $(1-\delta)100\%$ confidence when $\rho < \epsilon\sqrt{log(1/\delta)}$. In our experiment, we draw a dashed line when ρ reaches 0.1 to indicate the 90% accuracy with 90% confidence, which is an extensively used ρ value in the literature [9, 12].

We compare the convergence of failure probability estimation and FOM calculation in Figure 5. We observe that the estimation of MC, HSCS, AIS and proposed ACS all converge when sufficient simulations are allowed. As illustrated in Table 1, the ground truth MC estimation is 1.24e-5 (4.37σ). Among these methods, the proposed ACS algorithm is most accurate with only 1.5% relative error, while the results of HSCS and AIS have 4.6% and 6.1% relative error, respectively.

4.1.2 *Efficiency Comparison.*

The efficiency of MC, HSCS, AIS and ACS is shown in Figure 5. We notice that ACS has the fastest convergence among these algorithms. It is attributed to our unbiased estimator, which is updated parallelly in disjoint clusters. It is far more efficient than static sampling distribution in HSCS and global resampling scheme in AIS. To be specific, as shown in Table 1, our ACS method converge to 90% confidence with only 1736 IS simulations, while HSCS and AIS need 8688 and 5111 IS simulations to obtain the same FOM value. In addition, the proposed ACS algorithm requires 1100 presampling simulations. In comparison, HSCS and AIS need 7000 and 3000 times, respectively. This result demonstrates that proposed method is less sensitive to initial states. In total, ACS algorithm can achieve 3526X speedup over MC, 5X over HSCS and 3X over AIS.

4.2 Experiments on SRAM Column Circuit

A simplified schematic of SRAM column consisted of 32 bit cells is shown in Figure 7. Compared with a single bit cell in the previous low-dimensional experiment, we consider the impact of peripheral circuit and generate a more accurate estimation of failure probability. The configuration in Figure 7 demonstrates the worst-case scenario of read operation, in which accessed bit $CELL$<0> stores "0" and other idle bits store "1". In this case, the leakage current through idle bits (storing complementary value to the accessed bit) increases read access time and impedes a successful read. In our experiment, we simulate various SRAM failures in reading, writing and standby mode. There are in total 576 variation parameters in this test case, which is a high-dimensional problem.

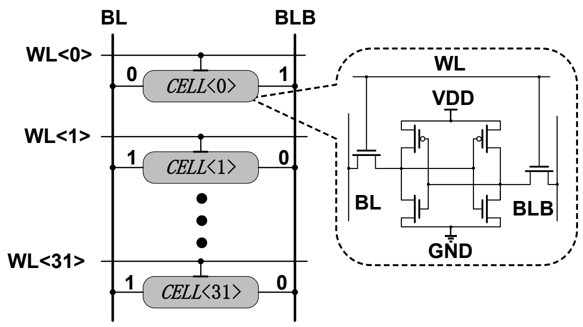

Figure 7: The schematic of 576-dimensional SRAM column circuit

4.2.1 *Comparison of Visualized Failure Regions.*

In order to investigate the different performance in high-dimensional circuit case, we project the sample points on two most important dimensions, $Vth0$ of $MN1$ and $Vth0$ of $MN3$. Figure 8 shows the visualized multiple failure regions in 2D parametric space for different methods. Based on MC sampling in Figure 8(a), three disjoint failure regions are clearly displayed. We notice that it contains two major failure regions and a minor one in the lower right corner. Here green dot denotes the mean value for original distribution on these two dimensions.

As shown in Figure 8(b), HSCS groups failure samples into clusters, and utilizes min-norm points to generate shifted Gaussian distributions that cover multiple failure regions. The location of min-norm points are depicted as black triangles in Figure 8(b). It is, however, less effective on high-dimensional circuit, as it is challenging to locate the min-norm points on the failure boundary. With biased min-norm points, we note that majority of captured samples are not generated in failure regions. Thus it requires more simulations in both presampling and IS steps. In Figure 8(c), AIS converges to wrong P_{fail} value because only two failure regions are notified. The minor failure region in the lower right corner disappears during iterations. It is due to the unbalanced weights for each sample. Samples with smaller weights are more prone to be eliminated in the global resampling procedure. The proposed ACS algorithm successfully searches for all three failure regions after a few iterations. And we notice that the whole sample set is focused on the most important failure boundary, which dominates the failure probability estimation. Therefore, ACS method outperforms other methods in terms of failure region exploration and convergence speed.

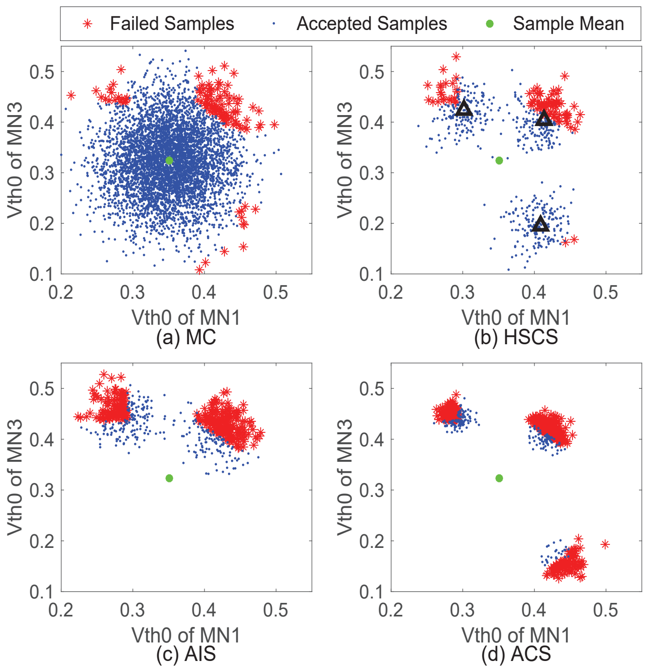

Figure 8: Multiple failure region coverage test (failed samples/ accepted samples/ sample mean are colored)

4.2.2 *Accuracy and Efficiency Comparison.*

In this section, we compare the evolution of the failure probability and FOM in Figure 9. The ground truth failure probability is estimated by brute-force MC. Among HSCS, AIS and ACS algorithms, only our ACS method is capable of converging to gold failure probability. HSCS converges to wrong failure rate with much slower speed because static deterministic mixture IS estimator cannot work in high-dimensional case. AIS also cannot generate correct failure rate estimation. For multi-failure-region SRAM column circuit, some less important failure regions are neglected by AIS global resampling process, which leads to smaller failure probability.

To be specific, as shown in Table 2, the gold failure probability estimated by MC is 1.6e-5 (4.3σ) with 10 million simulations. Only ACS algorithm succeeds in giving accurate estimation with 3.1% relative error. The number of simulations in presampling and IS steps are 2000 and 2878, respectively. Therefore, the proposed ACS algorithm can obtain 2050X speedup w.r.t MC.

Table 2: Accuracy and efficiency comparison on 576-dimensional SRAM column

	Monte Carlo	HSCS	AIS	Proposed
Failure prob.(error)	1.60e-5(0%)	9.82e-6(error)	2.23e-6(error)	1.55e-5(3.1%)
Presampling # sim.	0	18000	5000	2000
Importance Sampling # sim.	1e7	28699	11253	2878
Total # sim. (speedup)	1e7(1X)	46699	16253	4878(2050X)

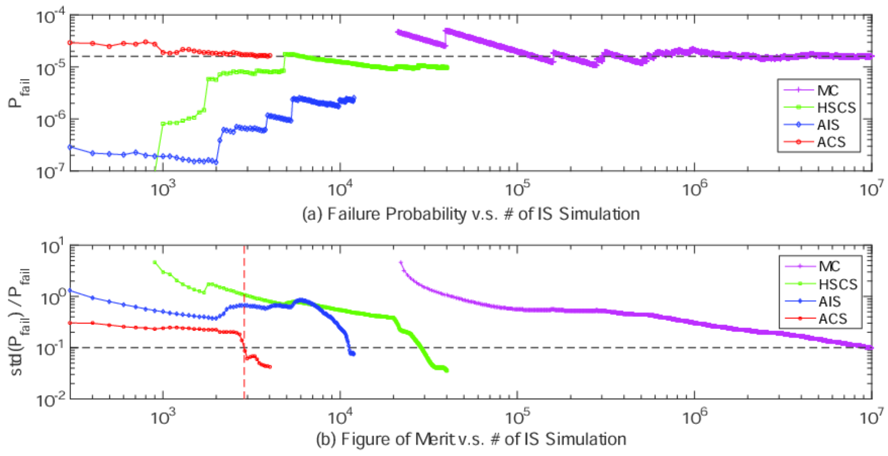

Figure 9: Evolution comparison of failure prob. and FOM on SRAM Column

5 CONCLUSION

In this paper, we present an Adaptive Clustering and Sampling method to efficiently estimate the rare-event failure probability of SRAM circuits. This method first applies hyperspherical presampling to generate a set of initial samples in high dimension. Next, an iterative clustering and sampling scheme is performed to search for failure regions and update estimation. The experiments demonstrate that proposed algorithm can provide extremely high accuracy and efficiency. Experiments on SRAM bit cell indicate that ACS achieves 3526X speedup over MC and 3-5X over other state-of-the-art methods. On SRAM column circuit in high dimension and with multiple failure regions, ACS is 2050X faster than MC method, while other IS based approaches fail to provide reasonable accuracy.

6 ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grant NSFC 61874152.

REFERENCES

[1] Dennis Sylvester, Kanak Agarwal, and Saumil Shah. Variability in nanometer cmos: Impact, analysis, and minimization. *Integration, the VLSI Journal*, 41(3):319–339, 2008.
[2] L He, Fang Gong, Rahul Krishnan, and Hao Yu. Exploiting parallelism by data dependency elimination: A case study of circuit simulation algorithms. *IEEE Design & Test of Computers*, page 1, 2012.
[3] Amith Singhee and Rob A Rutenbar. Statistical blockade: a novel method for very fast monte carlo simulation of rare circuit events, and its application. In *Design, Automation, and Test in Europe*, pages 235–251. Springer, 2008.
[4] Amith Singhee, Jiajing Wang, Benton H Calhoun, and Rob A Rutenbar. Recursive statistical blockade: An enhanced technique for rare event simulation with application to sram circuit design. In *VLSI Design, 2008. VLSID 2008. 21st International Conference on*, pages 131–136. IEEE, 2008.
[5] Wei Wu, Wenyao Xu, Rahul Krishnan, Yen-Lung Chen, and Lei He. Rescope: High-dimensional statistical circuit simulation towards full failure region coverage. In *Proceedings of the 51st Annual Design Automation Conference*, pages 1–6. ACM, 2014.
[6] Rouwaida Kanj, Rajiv Joshi, and Sani Nassif. Mixture importance sampling and its application to the analysis of sram designs in the presence of rare failure events. In *Design Automation Conference, 2006 43rd ACM/IEEE*, pages 69–72. IEEE, 2006.
[7] Lara Dolecek, Masood Qazi, Devavrat Shah, and Anantha Chandrakasan. Breaking the simulation barrier: Sram evaluation through norm minimization. In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 322–329. IEEE Press, 2008.
[8] Masood Qazi, Mehul Tikekar, Lara Dolecek, Devavrat Shah, and Anantha Chandrakasan. Loop flattening & spherical sampling: Highly efficient model reduction techniques for sram yield analysis. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 801–806. European Design and Automation Association, 2010.
[9] Wei Wu, Srinivas Bodapati, and Lei He. Hyperspherical clustering and sampling for rare event analysis with multiple failure region coverage. In *on International Symposium on Physical Design*, pages 153–160, 2016.
[10] Mengshuo Wang, Changhao Yan, Xin Li, Dian Zhou, and Xuan Zeng. High-dimensional and multiple-failure-region importance sampling for sram yield analysis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(3):806–819, 2017.
[11] Hiromitsu Awano, Masayuki Hiromoto, and Takashi Sato. Efficient aging-aware sram failure probability calculation via particle filter-based importance sampling. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 99(7):1390–1399, 2016.
[12] Xiao Shi, Jun Yang, Fengyuan Liu, and Lei He. A fast and robust failure analysis of memory circuits using adaptive importance sampling method. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2018.
[13] Wei Wu, Fang Gong, Gengsheng Chen, and Lei He. A fast and provably bounded failure analysis of memory circuits in high dimensions. In *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*, pages 424–429. IEEE, 2014.