Enhanced Statistical Blockade Approaches for Fast Robustness Estimation and Compensation of Nano-CMOS Circuits

Luo Sun[1], Jimson Mathew[2], Dhiraj K. Pradhan[3], and Saraju P. Mohanty[4]
Department of Computer Science, University of Bristol, Bristol, UK.[1,2,3]
NanoSystem Design Laboratory (NSDL), University of North Texas, Denton, TX 76203, USA.[4]
E-mail ID: sun@compsci.bristol.ac.uk[1], jimson@compsci.bristol.ac.uk[2],
pradhan@compsci.bristol.ac.uk[3], and saraju.mohanty@unt.edu[4]

Abstract

The challenges to design engineers have been complicated due to the introduction of nanoscale process variation into the design phase. One of the ways to analyze the circuit behaviors under process variation is to determine the rare events that may be originated due to such process variation. A method called Statistical Blockade (SB) had been investigated to estimate the rare events statistics especially for high-replication circuits [1]. An enhanced statistical blockade method is proposed in this paper which is shown to be much faster compared to the traditional exhaustive Monte Carlo simulation. In SB, the classification threshold determination is quite important for different tail regions which is related to the number of rare events simulation. This paper presents the values of classification threshold t_c for different tail regions of typical circuits and the training samples size n required for corresponding t_c. It offers both fastest speed of simulation and highest accuracy for the proposed Statistical Blockade method. It is also proven that the obtained t_c can be used for all the technology corners. The enhanced statistical blockade method thus performs fast estimate the robustness for different designs. In the proposed method, the tail part of the whole distribution is used in estimation; thereby saving time. It shows 7.3$\times$ faster than traditional evaluation methods. Furthermore, for the design which is proved to be robust even in worst-case, the optimal body bias voltage is applied to improve the performance and power while reducing the variability with Adaptive Body Bias (ABB) technique.

Index Terms

Statistical Blockade Method, Robustness Circuits, Nanoscale CMOS, Arithmetic Circuits, Monte Carlo, Rare Event

I. Introduction

The CMOS technology continues to scale down to nanoscale domain achieve higher performance and higher level of integration. The impact of process variations on performance has been increasing with each semiconductor technology generation. The technology scaling has resulted in significant deviations from the nominal values of transistor parameters, such as channel length and threshold voltage [2]. For example, variation in gate length increases from 35% in $130\ nm$ technology to almost 60% in $65\ nm$ technology. The parameter variations result in the large variation in leakage and performance of the designed circuit. Traditional corner model based analysis and design approaches provide guard-bands for parameter variations. Therefore, these approaches are prone to introducing pessimism in the design [3]. Computer-Aided Design (CAD) tools have traditionally use for handling corner analysis, under the assumption of fixed or deterministic circuit parameters. However, in nano-CMOS circuits small variations due to inaccuracies in the manufacturing process can cause large relative variations in the behavior of the circuit. For example, 10,000 runs of a Monte Carlo simulation for delay variation in a 16-bit adder for 20% variation in threshold voltage and gate oxide thickness is shown in Fig. 1.

The process variations in the nano-CMOS technology are classified into the following two types: inter-die variations are the variations from die to die and intra-die variations correspond to variability within a single chip. Inter-die variations affect all the devices on the same chip in the same way, e.g., making the transistor gate lengths of devices on the same chip all larger or all smaller. On the other hand, the intra-die variations may affect different devices differently on the same chip, e.g., making some devices have smaller transistor gate lengths and others larger transistor gate lengths [4]. In the past, the inter-die variations dominated intra-die variations, so that the latter could be safely neglected. However, in modern nanoscale technologies, intra-die variations are rapidly and steadily growing and can significantly affect the variability of performance parameters on a chip. The increase in intra-chip parameter variations is due to the effects such as microloading in the etch, variation in photoresist thickness, and optical proximity effects. Intra-die variation is spatially correlated. It is locally layout dependent and circuit specific, i.e., devices with similar layout patterns and proximity structures tend to have similar characteristics. It is globally location dependent, i.e., devices located close to each other are more likely to have the similar characteristics than those placed far away [4].

Corresponding Author: **Luo Sun**, Department of Computer Science, University of Bristol, Bristol, UK. Phone : +44 798 842 3985, Email : sun@compsci.bristol.ac.uk.

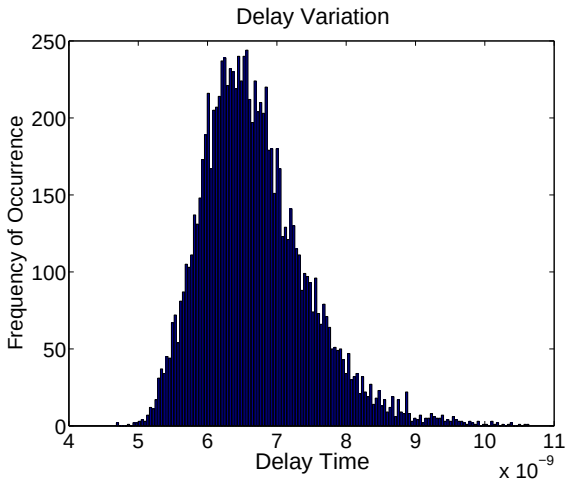

Fig. 1. 10,000 MC Simulation for Delay variation in 16-bit adder for 20% variation in gate oxide thickness and threshold voltage

The **novel contributions of this paper** to the state-of-the art are as follows:

1) The Statistical Blockade [1] is investigated as a faster Monte Carlo technique.
2) The enhanced Statistical Blockade algorithms are proposed to obtain the minimal classification threshold t_c for different tail portions and optimal training samples size for the corresponding t_c. These guarantee the fastest speed and highest accuracy of the prediction.
3) The obtained minimal classification threshold t_c of each tail part can be applied to all the technology corners for both high-performance applications and low-power applications.
4) A novel method is proposed for estimating the robustness of different nano-CMOS Arithmetic Circuits. This method is combined with an implementation of Statistical Blockade to achieve significant reduction in the computational costs.
5) After picking up the design with worst tolerant ability under process variation, a method is used to compensate the yield loss of the design with Adaptive Body Bias (ABB) technique.

The rest of this paper is organized as follows. Section II describes the related previous related research. In Section III, the basic definitions and notations of Statistical Blockade method is presented. Section IV describes the improved Statistical Blockade algorithms for minimal threshold t_c and optimal training samples size n. In Section V, a novel method based on improved Statistical Blockade algorithms is proposed to evaluate the design robustness. Section VI shows the basic simulation approach. Section VII shows the experimental results. The conclusions are drawn in Section VIII.

II. Related Prior Research

Recently developed statistical static timing analysis (SSTA) tools have recognized these unavoidably random aspects of manufacturing variations and have attempted to account for them using simple models that are computationally inexpensive. In [5], the authors use a linear model for the gate delay as a function of varying gate parameters. In [4], [6], the authors recognize the impact of spatial correlation between gates and use simple grid-based models to represent this correlation. Also use Principal Components Analysis (PCA) to extract uncorrelated parameters from this correlation model, and builds the gate timing models using these uncorrelated components. Several analytical and semi-analytical approaches have been suggested to model the behavior of SRAM cells and digital circuits in the presence of process variations. All suffer from approximations necessary to make the problem tractable Monte Carlo analysis (MC) is one of the standard techniques used for statistical modeling. Standard Monte Carlo techniques are, by construction, most efficient at sampling the statistically likely cases. When used for simulating statistically unlikely or rare events, these techniques are extremely slow. In [1], authors proposed Statistical Blockade (SB) as a Monte Carlo technique that allows us to efficiently filter to block unwanted samples insufficiently rare in the tail distributions. The method imposes almost no a priori limitations on the form of the statistics for the process parameters, device models, or performance metrics. The key observation behind Statistical Blockade is that generating each sample is not expensive as the parameters for a circuit are created on the fly. The current archival journal paper is based on a shorter conference publication with significantly more materials [7].

III. The Statistical Blockade Method

The statistical blockade method has been used to efficiently generate samples in the tail of the distribution of the performance metric of a circuit [1]. Standard Monte Carlo that generates samples using the complete distribution is not suitable for this purposes. A flow chart for the Statistical Blockade method is shown in Fig. 2.

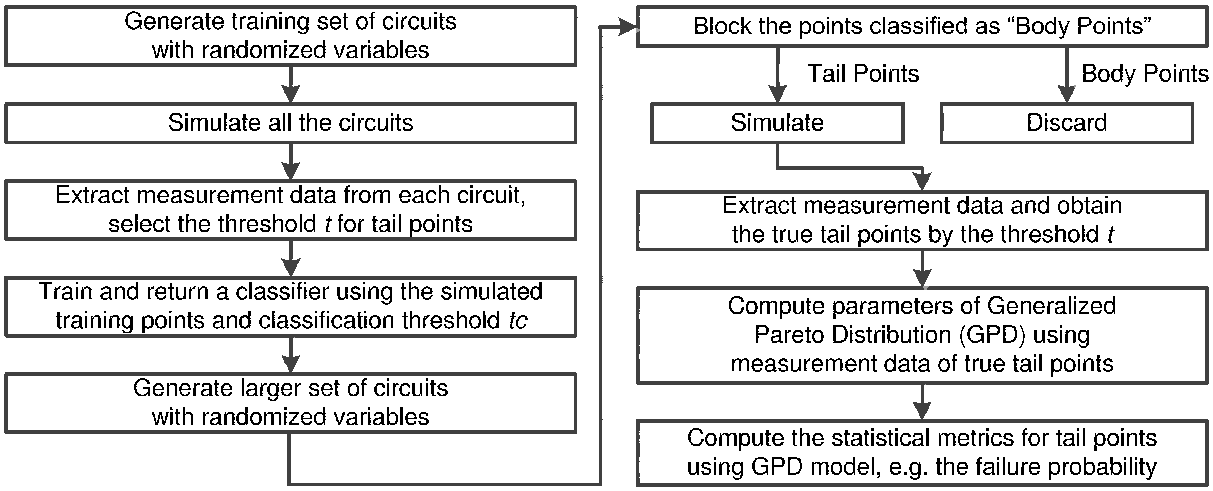

Fig. 2. Flow chart for the Statistical blockade method.

The key idea is to define a region in the design parameter space that results in the circuit performance values greater than threshold (t). Then only those Monte Carlo samples that lie in the tail region are simulated while blocking the body samples. For example, if the tail threshold is the 99% point of the distribution then only one out of 100 simulations is useful; thus resulting in an immediate speedup of $100\times$ over standard Monte Carlo. A small Monte Carlo sample set (e.g. 1000 points) is used to train a classifier to identify the tail points. However, it is difficult to classify all the tail points to build the accurate tail model for high-dimensional data. Thus, a relaxed boundary of the classifier which is denoted as classification threshold (t_c) is applied to block most unwanted points. A specific example, t_c is the 97^{th} percentile to extract 1% tail samples (i.e. t = 99%), based on empirical analysis of the tradeoff between classifier accuracy, simulation time, and tail model fit [8].

The classifier used in the current paper is named Support Vector Machine (SVM) [9], which is implemented using LIBSVM [10], SVM$^{\text{light}}$ [11] or WEKA [12]. The simulated data are divided into two classes: body region and tail region. Therefore, the C-Support Vector Classifier is chosen [13], [14]. The four basic kernels of SVM are as follows: linear, polynomial, radial basis function (RBF), and sigmoid. The RBF kernel is a good choice as it can non-linearly map samples into a higher dimensional space. One important modification for the classifier is the increment of the tail weight. The number of body points is much more than the tail samples. The classification error is very low as long as all the body points have been classified correctly no matter whether the tail points are misclassified. So the classifier is biased to the body points by default. Thus, the classification error should be reversed by increasing the weight of the tail. Training of an aforementioned classifier using a small set of 2-dimensional samples ($n = 1000$) by varying threshold voltages for a NAND gate is shown in Fig. 3. Then the randomly generated samples are classified by the trained SVM classifier into two categories shown in Fig. 4.

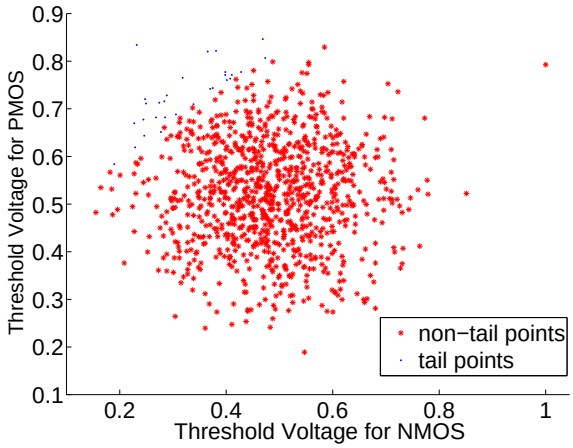

Fig. 3. Initial Training Samples for a Classifier.

The tail circuits do not follow the Gaussian statistics after the classification is performed. An important observation that can be used is that the conditional distribution of the events in the tail region trend toward a generalized Pareto distribution (GPD)

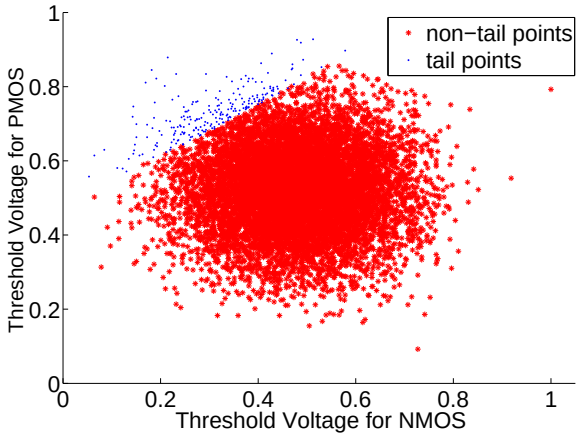

Fig. 4. Classification of Monte Carlo Generated Samples.

from the extreme value theory (EVT) [15],. Thus, the fitting of a generalized Pareto distribution (GPD) is required to build the tail model for calculating the coefficients of GPD, shape parameter α, and scale parameter β. The tail model is the cumulative distribution function (CDF) of the GPD. Statistical blockade filtering and generalized Pareto distribution (GPD) modeling is performed using the following steps [16]:

1) Perform initial sampling to generate data to build a classifier. This initial sampling can be standard MC or importance sampling. Also estimate t and $t_c < t$ from this data.
2) Build a classifier using the classification threshold t_c.
3) Generate more samples using MC, following the CDF F, but simulate only those that are classified as tail points. Update the estimate of t.
4) Fit GPD model to the simulated tail points.

There are significant practical problems with the technique proposed originally. Extensions to make statistical blockade practically usable in common scenarios is proposed in [17]. In statistical blockade, the classification threshold selection becomes very important for different tail regions which is related to the number of rare events simulation. The classification threshold and number of training samples may result in inaccurate tail model and slow simulation speed. Thus, the following research questions need to be answered for an efficient statistical blockade method that can be practically useful:

1) What is the minimal value of classification threshold that can ensure all the true tail points be covered in a loosely defined boundary?
2) What is the minimum number of initial sampling required to cover the given tail points?

An effective way to the problem of finding the minimum threshold of classifier t_c for the given tail part and the minimum number of samples n required for corresponding classification threshold will be described in the next section. There are two main problems focussed:

1) One is the appropriate classification threshold t_c for different tail regions - 3%, 2% and 1%.
2) The other one is the minimal training samples size for given classification threshold t_c.

IV. Improved Statistical Blockade Algorithms

The proposed steps to measure the threshold t_c of the classifier for the corresponding tail region is shown in Algorithm 1. The training samples size used is 2000. The function $\mathrm{F}_{sim}(N)$ runs 10,000 Monte Carlo simulation. Y is a vector of output values; e.g. delay of an adder. The function percentile (Y, t) calculates the value of delay V for the corresponding tail threshold t. The function size (Y_{tail}) is used to count the number of tail points after Y compared with V. These steps are also used to select tail points from full Monte Carlo simulation by empirical method. If threshold t is defined as 99% point in 10,000 points, tail count should be approximately 100. The values of V and S for different tail thresholds t in the test case of an 8-bit adder is listed in Table I.

TABLE I

The Tail Points Size and Delay of Corresponding Threshold for Different Tail Regions in 10,000 MC Points of 8-bit Adder.

Tail Threshold (t)	97%	98%	99%
Delay for Corresponding Threshold (V)	6.1180ns	6.1740ns	6.2890ns
Tail Points Size (S)	299	204	100

Algorithm 1 For choosing minimal classification threshold t_c for given tail threshold t.

1: Assume: training sample size n_0 (e.g., n_0 = 2,000); total sample size N (e.g., N = 10,000). // Generate 10,000 MC points in statistical parameter space.
2: Y = F$_{sim}$(N) // Simulate 10,000 MC samples.
3: V = Percentile(Y, t) // Find the value V for the tail threshold t according to the measurement data.
4: Y_{tail} = $\{Y_i \in Y$: $Y_i > V\}$ // Count the number of the tail.
5: S = Size(Y_{tail}) // S is the tail size of the 10,000 MC samples.
6: **for all** $t_c = t$ **do**
6: // Initially let t_c equal to t, therefore no safety margin given.
7: x = MonteCarlo(n_0) // Randomly pick 2000 MC samples up from 10,000 points as training set.
8: y = F$_{sim}$(x) // Simulate 2000 training samples.
9: V_c = Percentile(y, t_c) // Find the value V_c for the tail threshold t_c.
10: C = BuildClassifier(x, y, V_c) // Train a classifier C using training set x and classification threshold value V_c.
11: y_{tail} = F$_{sim}$(Filter(C, N)) // Extract the tail points by the classifier C from the same 10,000 MC samples.
12: $y_{tail,true}$ = $\{y_i \in y_{tail}$: $y_i > V\}$ // Select the true tail points.
13: S_c = Size($y_{tail,true}$) // S_c is the number of the classified tail points from 10,000 samples.
14: **if** $S_c < S$ **then**
14: // If $S_c < S$, it means the classifier C is not able to extract all the tail points.
15: $t = t$ -1% // Increase the safety margin by 1% each cycle.
16: Repeat
17: **else**
18: $t_c = t$
19: **end if**
20: **end for**

The step-7 to step-12 of the algorithm is the typical Statistical Blockade method [18]. The function Monte Carlo (n_0) randomly picks up 2000 training samples from N. For an 8-bit adder, x is a 2000 $\times$ 4 matrix, since there are 4 variables to be varied. Each row of matrix x is a point in 4 dimensions. y is a vector of output values from simulation of training samples. V_c is the value of delay for classification threshold t_c. It is computed from y. The function BuildClassifier (x, y, V_c) trains and returns the SVM classifier C using training data package y and classification threshold t_c. First it is assumed that $t_c = t$. The function Filter(C, N) filters out the non-tail points in N by the classifier C. Only the tail points classified by C will be simulated using the function F$_{sim}$(Filter(C, N)). Since classification threshold t_c is usually smaller than the real tail threshold t, there are some safety margin with t_c. The current tail points are not the real tail points we want. Then the false tail points are eliminated by comparing with V, the value of output metric for the given tail threshold t in full MC simulation N. Finally, the tail points size S_c is obtained by Statistical Blockade method from the same 10,000 samples. S_c is compared with the real size of tail points S which is obtained in step-5. If $S_c < S$, that means not all the tail points can be collected using t_c as the classification threshold. Then the safety margin for t_c is increased. For each cycle, safety margin is increased 1% till S_c is equal to S to get all tail points. At last, t_c is the minimal classification threshold for the corresponding tail region.

The minimal number of training samples n for the fixed classification threshold t_c is chosen using Algorithm 2. The number of training samples can not be less than n for a given t_c when SB method is applied to build the tail model. Otherwise the tail model will not be accurate enough, since not all the tail points are considered. The full tail points size S is obtained from Step-2 to Step-5 in Algorithm 2 which are the same with Algorithm 1. The cycle (Step-6 $\sim$ Step.15) is used to collect values of S_i (i = 1, 2, ... , 20) when the training sample size n is 100, 200, 300, ... , 2000, respectively ($n = i \times 100$). S_i is the number of selected tail points when SB method is applied. The function Find($S_i = S$ and $S_{i+1} = S_{i+2} = ... = S_{20} = S$) gets the value of i when $S_i = S_{i+1} = S_{i+2} = ... = S_{20} = S$. It implies that full tail points S can be classified through classifier C when initial training sample size n is $i \times 100$. And with increasing of training samples size n, the number of extracted tail points ($S_{i+1} \sim S_{20}$) remain stable - all equal to S. Finally the minimal number of training samples n for the fixed threshold t_c is $n = i \times 100$.

V. The Intelligent Statistical Blockade Approach For Tolerant Ability Estimation

In this section, an intelligent method with improved Statistical Blockade algorithm is proposed to estimate tolerant ability of different FA cells based 16-bit ripple-carry adders (RCAs).

A. The Example Arithmetic Circuit: Full Adder For Alternative Logic Styles

There are varieties of static CMOS logic styles which have been proposed to implement 1-bit full adder cells [19]. These designs can be mainly divided into two categories: complementary CMOS logic and pass transistor logic circuits.

A complementary CMOS FA (C-CMOS) cell is shown in Fig. 5(a). It is based on the regular CMOS structure with PMOS pull-up and NMOS pull-down. The advantage of C-CMOS FA is its robustness against the supply voltage scaling and transistors sizing. The FA cell in Fig. 5(b) is the complementary pass transistor logic full adder cell (CPL). The difference between pass transistor logic and C-CMOS logic is the source of the pass transistor connects to the input signal instead of connecting to

Algorithm 2 For choosing minimal number of training samples n for the fixed threshold t_c.

Assume: Initial training sample size n_0 (e.g., n_0 = 100); total sample size N (e.g., N = 10,000). // Generate 10,000 MC points in statistical parameter space.
Y = F$_{sim}$(N) // Simulate 10,000 MC samples.
V = Percentile(Y, t) // Find the value V for the tail threshold t according to the measurement data.
Y_{tail} = $\{Y_i \in Y: Y_i > V\}$ // Count the number of the tail.
S = Size(Y_{tail}) // S is the tail size of the 10,000 MC samples.
for all (i = 1; $n_0 \leq$ 2000; i++) **do**
    // Initially set 100 training points.
    x = MonteCarlo(n_0) // Randomly pick n_0 MC samples up from 10,000 points as training set.
    y = F$_{sim}$(x) // Simulate these training samples.
    V_c = Percentile(y, t_c) // Find the value V_c for the tail threshold t_c.
    C = BuildClassifier(x, y, V_c) // Train a classifier C using training set x and classification threshold value V_c.
    y_{tail} = F$_{sim}$(Filter(C, N)) // Extract the tail points by the classifier C from the same 10,000 MC samples.
    $y_{tail,true}$ = $\{y_i \in y_{tail}: y_i > V\}$ // Select the true tail points.
    S_i = Size($y_{tail,true}$) // S_i is the number of the classified tail points from 10,000 samples.
    n_0 = n_0 + 100 // Increase the training set size by 100 each cycle.
end for
Find(S_i = S and S_{i+1} = S_{i+2} = ... = S_{20} = S) // Obtain the value of i.
n = i * 100 // The minimum number of training samples is equal to i * 100.

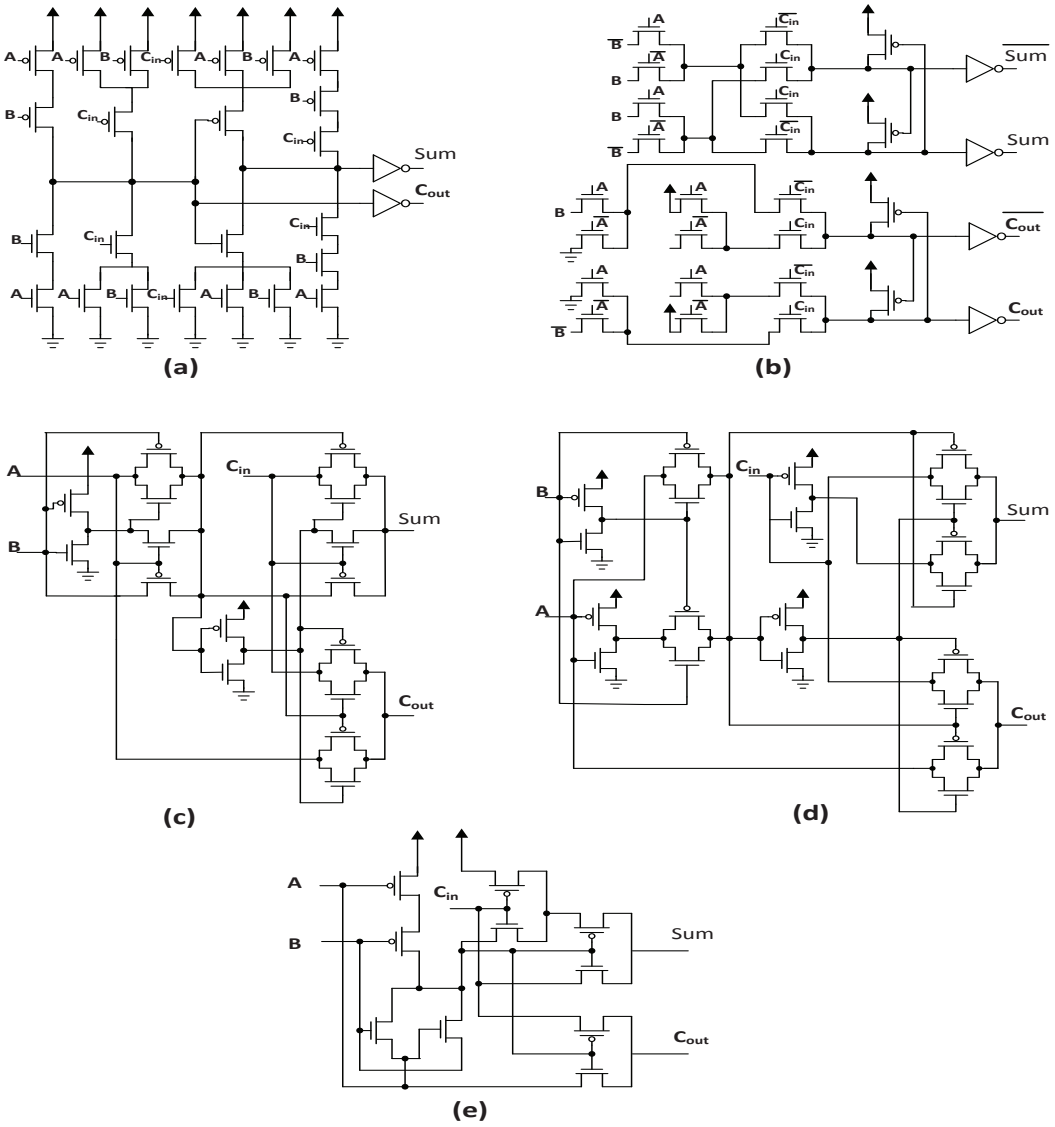

Fig. 5. Full Adder cells of different logic styles. (a) C-CMOS (b) CPL (c) TFA (d) TGA (e) 10T.

the power lines. A single pass transistor can implement the logic function resulting in smaller count transistors and smaller input load. However, the pass transistor logic meets a threshold voltage drop problem. Thus, the inverters have to be employed to ensure the drivability. Fig. 5(c) shows a transmission-function full adder (TFA) by using 16 transistors. Fig. 5(d) is a 20 transistors transmission gate full adder (TGA). Both of them belong to pass transistor logic circuit. This type design employs PMOS and NMOS in parallel and the signal can pass though the transistors when they are on simultaneously. Therefore, this design has nothing to do with the voltage drop problem. Moreover, less number transistors designs have been proposed as well, for example 10-transistor full adder as shown in Fig. 5(e).

B. The Proposed SB Method Using Improved Algorithms

The steps of the proposed statistical blockade approach using and improved approach are shown in Algorithm 3:

Algorithm 3 Intelligent Statistical Blockade for Robustness Estimation.

1: Compute the delay times of 5 RCAs with their nominal values for a single run.
2: Make the delay times to the target value through transistor sizing.
3: Use improved SB method to pick up 1% tail points from $10,000$ generated MC points and simulate the extracted tail points.
4: Extract true tail points to the Conditional Cumulative Distribution (CDF).
5: Calculate the coefficients α and β for Generalized Pareto Distribution (GPD) approximation $G_{\alpha,\beta}$ with the above.
6: Compare the CDFs of 5 RCAs for the robust abilities estimation.

In the 1^{st} step of proposed algorithm, it shows how to adjust the delay time using transistor sizing approach:

1) Set all the transistors (NMOS and PMOS) to the minimum size. ($45\ nm$ for channel length and $90\ nm$ for channel width in $45\ nm$ BPTM)
2) Simulate 5 16-bit RCA designs for a single run with nominal values of variables to measure the initial delays. Set a target delay time according to the measured nominal delays.
3) Figure the transition with the highest delay (from 1^{st} bit carry in to 16^{th} bit carry out) and mark the transistors in that are involved.
4) Change the channel width of the transistors which are involved in the critical path to adjust the 5 initial delay times to the target value.

The channel widths of the marked transistors in the critical are listed in the Table II for 5 RCAs.

TABLE II
CHANNEL WIDTH OF TRANSISTORS INVOLVED IN CRITICAL PATH AFTER TRANSISTOR SIZING.

RCA	C-CMOS	CPL	TFA	TGA	10T
Channel Width (nm)	83.2	90	239.3	206.3	178.9

In the 2^{nd} step, the Support Vector Machine (SVM) classifier presented in [20] is considered as the classification tool in SB implementation. To achieve the fastest simulation speed and highest accuracy, the algorithm proposed in Section IV is applied to find out the minimal classification threshold t_c and the optimal number of training samples n. The SB method with t_c and n is used to extract 1% tail points from the generalized MC points. Then the true tail points can be selected after the simulation of extracted tail points. The values of t_c and n are shown in Table III.

TABLE III
MINIMUM CLASSIFICATION THRESHOLD t_c AND OPTIMAL TRAINING SAMPLE SIZE n FOR 1% TAIL SELECTION

RCA	C-CMOS	CPL	TFA	TGA	10T
Minimum Classification Threshold t_c	2%	2%	2%	2%	2%
Optimal Training Sample Size n	700	900	1300	1300	1100

In Step-3, the maximum likelihood estimation (MLE) is used to fit GPD to the 1% tail data to build the CDF tail model. For fitting the GPD model, two GDP coefficients α and β need to be computed by MATLAB function. After building the tail models, the comparison can be done in Step-4 and the detailed analysis will be described in the following section.

VI. EXPERIMENTAL RESULTS OF THE PROPOSED ALGORITHMS AND METHOD

In this section, the details of the experiments and the experimental results are presented.

A. Experiments And Results For The Improved Statistical Blockade Algorithms

In this experiment, an 8-bit adder in 45nm BPTM is considered as the test case. Monte Carlo simulation of 10,000 runs were performed with global, Gaussian distributed threshold voltage and gate oxide thickness with 20% variation. The metric measured is the delay time for the 8-bit adder.

Fig. 6 shows the number of 1% tail points (tail threshold t = 99%) collected in cases t_c is equal to 99%, 98% and 97% with training sample size increasing from 100 to 2000 for the 8-bit adder. The full 1% tail size of 10,000 samples is 100 as in Table I. From Fig. 6, it is evident that it is unable to pick up all the tail points when t_c = 99% no matter how many samples have been used for training. In case t_c = 98%, 100 tail points can be obtained as well as t_c = 3%. The principle of how to choose t_c in Algorithm 1 is to make the safety margin as small as possible. Otherwise the number of the simulation for tail points will increase significantly. For example, for a million MC points, if the selected t_c has 1% more safety margin, it means about 10,000 more simulations will be needed. Thus, the best value of t_c is 98% to select 1% tail points.

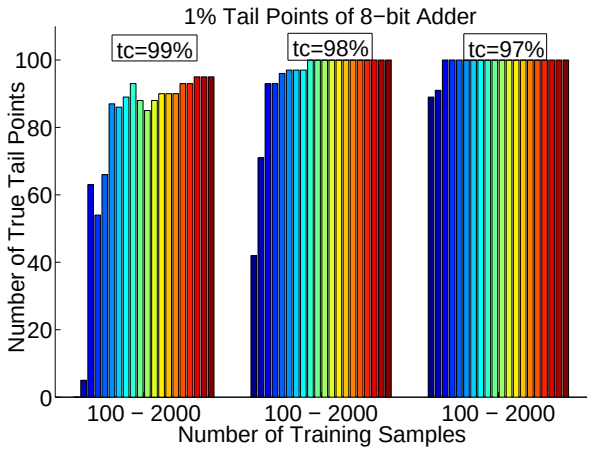

Fig. 6. True Tail Points obtained with 100∼2000 Training Samples for Different Classification Threshold.

After t_c is chosen, the minimal training sample size for t_c = 98% is needed. Fig. 7 indicates when full tail points can be extracted in cases t_c = 99%, 98% and 97%. All 100 tail points are collected from the points (arrow in Fig. 7), and the following extracted tail points sizes all equal to full tail size - 100 in case t_c = 98%. The training sample size in that point is 900. So the minimal number of training points is 900. The overall results are in Table IV.

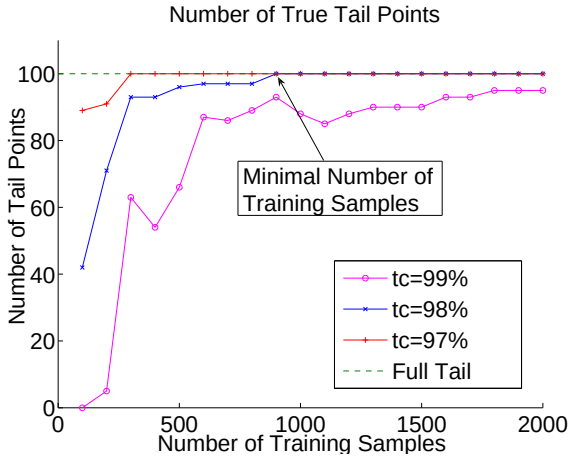

Fig. 7. Method for Obtaining Minimal Training Sample Size with Different Classification Threshold.

The conditional Cumulative Distribution (CDFs) of tail points between empirical method and statistical blockade using the values from Table IV is compared in Fig. 8. It shows a good match of two CDFs. The empirical method needs 10,000 Monte Carlo runs. In statistical blockade method, 344 tail points are picked up after classification (t_c = 98%). Another 900 runs are used to train the classifier. Totally it needs 1244 (344 + 900) Monte Carlo simulations. Note that, if t_c = 97% is chosen here, the total number of simulation, 865 (565 + 300), is even fewer. Since it is only for 10,000 Monte Carlo samples, which is impossible in industry. For large quantities of Monte Carlo points, the simulation size of t_c = 98% is significantly fewer than t_c

TABLE IV

MINIMAL CLASSIFICATION THRESHOLD AND MINIMAL NUMBER OF TRAINING SAMPLES REQUIRED FOR 1% TAIL OF 8-BIT ADDER.

Tail Threshold (t)	99%
Minimal Classification Threshold (t_c)	98%
Minimal Training Samples Size (n)	900

= 97%. For 10,000 Monte Carlo simulation, it shows 10× speedup compared with empirical method. While it can demonstrate much higher speedup for millions or billions of Monte Carlo samples.

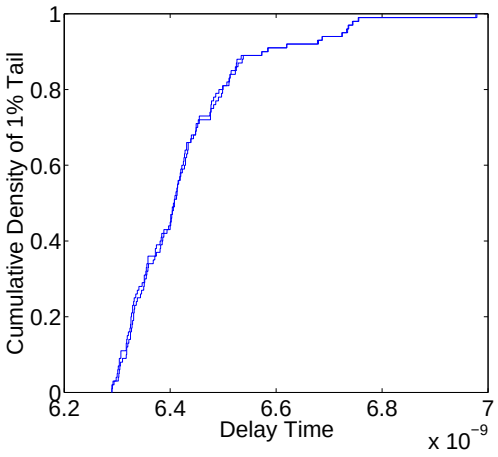

Fig. 8. Adder Delay Analysis: Empirical versus Statistical Blockade.

Then values of t_c and n are obtained by repeating above processes for 2% and 3% tails for an 8-bit adder. The t_c and n for the test case are indicated in Table V. The minimum classification threshold t_c is 98% and at least 900 training samples are needed to build 1% tail model (tail threshold t_c = 99%). For extracting 2% tail, t_c is no larger than 97% and 800 samples should be used for training. When tail threshold t is 97%, the appropriate t_c is 96% and minimum initial training samples size is 1200.

TABLE V

MINIMAL CLASSIFICATION THRESHOLD AND MINIMAL NUMBER OF TRAINING SAMPLES REQUIRED FOR DIFFERENT TAILS OF 8 BITS ADDER.

Blocks	Tail Threshold (t)	99%	98%	97%
8-bit Adder	Minimal Classification Threshold t_c	98%	97%	96%
	Minimal Training Sample Size n	900	800	1200

Then the improved statistical blockade algorithms are tested by another typical high-replicated circuit, 6-transistor SRAM shown in Fig. 9 to check if the obtained t_c and n can be used for all technology corners. The 6-transistor SRAM is designed in 45nm, 32nm, 22nm and 16nm models for both high-performance applications and low-power applications. The values of t_c and n for each technology corner is shown in Table VI. As can be seen from Table VI, the classification thresholds t_c of different tail regions, 1%, 2% and 3%, are the same for all the listed technology corners. However, the number of training samples n required for corresponding classification threshold are fluctuated.

B. Experimental Evaluation For Proposed Statistical Blockade Approach

The test platform [21] for the delay measurement of five 16-bit RCAs is shown in Fig. 10. The shown 4-bit ripple-carry adder in Fig. 10 is cascaded by 4 FA cells, with the carry output of the current full adder connecting to the next bit full adder input in the chain. It starts from X_0 and Y_0 which represent the least significant bits of the numbers to be added and the output is $sum0$, $sum1$, $sum2$ and $sum3$. The 16-bit RCA is based on the same principle.

The buffers which consist of two cascaded inverters are connected to the input signals and outputs. The inputs are fed from buffers to give more realistic signals. The outputs are loaded with buffers to provide proper loading conditions to ensure the fairness of the comparison. The attached buffers for the outputs can be also used to solve the threshold voltage drop problem caused by the pass transistor logic, thus, to enhance the driven capability.

During the Monte Carlo simulation, the variables to be varied are threshold voltage (V_{th}) and gate oxide thickness (T_{ox}) for NMOS and PMOS transistors. Thus, there are 4 variables involved in the simulation. The output metric to be measured is

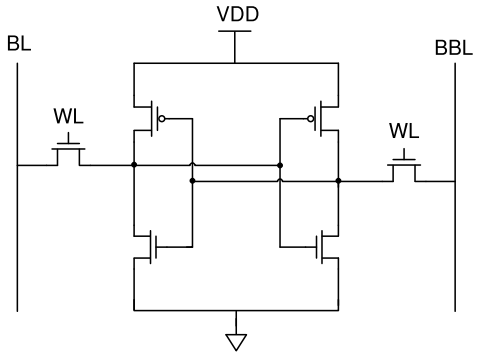

Fig. 9. Schematic of a 6T SRAM cell

TABLE VI
MINIMAL CLASSIFICATION THRESHOLD AND MINIMAL NUMBER OF TRAINING SAMPLES REQUIRED FOR DIFFERENT TECHNOLOGY CORNERS OF 6T SRAM.

Technology Corners	Tail Threshold (t)	99%	98%	97%
45nm HP	Minimal Classification Threshold t_c	98%	97%	96%
	Minimal Training Sample Size n	800	900	800
32nm HP	Minimal Classification Threshold t_c	98%	97%	96%
	Minimal Training Sample Size n	1000	900	800
22nm HP	Minimal Classification Threshold t_c	98%	97%	96%
	Minimal Training Sample Size n	900	1100	1000
16nm HP	Minimal Classification Threshold t_c	98%	97%	96%
	Minimal Training Sample Size n	1200	1300	1100
45nm LP	Minimal Classification Threshold t_c	98%	97%	96%
	Minimal Training Sample Size n	700	800	900
32nm LP	Minimal Classification Threshold t_c	98%	97%	96%
	Minimal Training Sample Size n	900	800	900
22nm LP	Minimal Classification Threshold t_c	98%	97%	96%
	Minimal Training Sample Size n	1000	900	1100
16nm LP	Minimal Classification Threshold t_c	98%	97%	96%
	Minimal Training Sample Size n	1100	1300	1200

the delay time (Td) of the 16-bit RCA. In each run, the delay time is measured from 50% of input voltage swing (carry in of 1^{st} bit) to 50% of output voltage swing (carry out of 16^{th} bit). The variations of the variable parameters are considered to be Gaussian distributed with 10% variation and a deviation of $\pm 3\sigma$ around their nominal values.

In statistical blockade based estimation, 1% tail points have been extracted from 10,000 generated Monte Carlo points. Fig. 11 is the distribution of the delay time for 1% true tail points of 16-bit C-CMOS RCA. As shown in the distribution, the threshold of delay for 1% is 7.2 ns. i.e., for 1% tail, the threshold of 99% delays is 7.2 ns. It also can be seen clearly that the tail points are no longer Gaussian distributed. The extracted tail data are required to fit of GPD, since GPD is a highly left skewed distribution and especially suitable for the tail data fitting.

After the simulation of extracted tail points, the true ones will be selected and used to compute the coefficients α and β of GPD to build the CDF tail model. The coefficients are calculated by the MATLAB function. The conditional CDFs of 1% tail points plotted by empirical method and statistical blockade method are compared in Fig. 12. It shows a good match between two CDFs. The value of horizontal axis is the exceedance of delay, i.e., the values of the delays for 1% tail points. For the tail points which delay time is 7.4 ns, the exceedance is 0.2 ns. The vertical axis is the predicted possibilities for the tail points from the model.

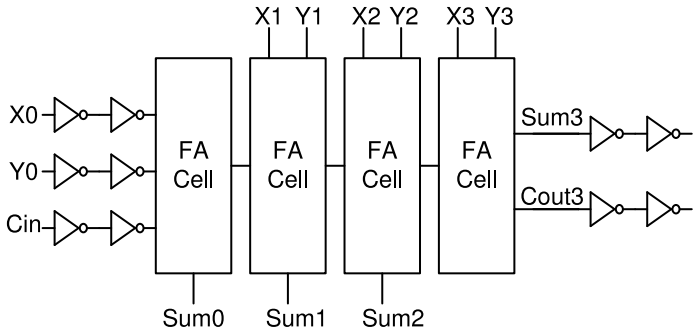

Fig. 10. Test Bench used to perform the experimental evaluations.

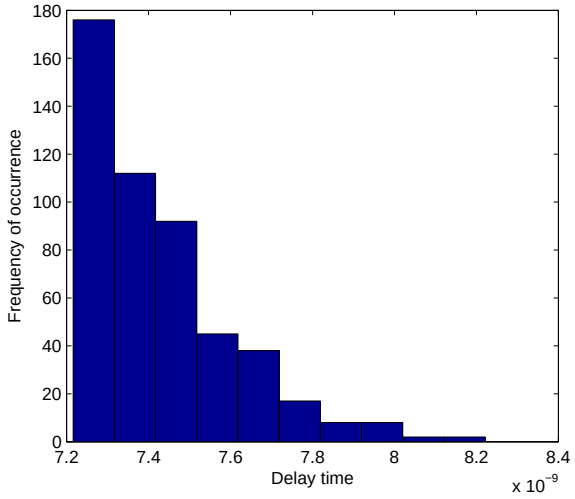

Fig. 11. Delay distribution of 1% tail for 16-bit C-CMOS.

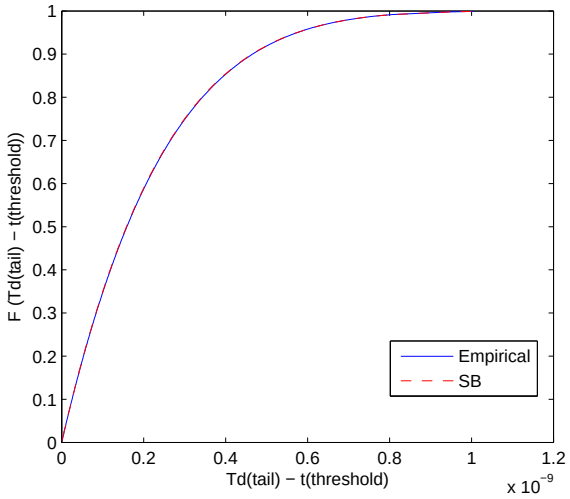

Fig. 12. CDFs for tail points: Empirical Method vs. SB approach.

The comparison of CDFs between 16-bit C-CMOS and 16-bit 10-transistor RCA is shown in Fig. 13. The tail model with full curve is C-CMOS FA based 16-bit RCA and the dashed curve is 10-transistor based. As shown in the diagram, for the tail points which exceedance of delay time is less than or equal to $0.2\ ns$, the possibilities are 60% for C-CMOS RCA and 43% for 10-transistor. For the tail points which exceedance is over $0.2\ ns$, their possibilities are 40% and 57%, respectively. Thus, the delay distribution of 10-transistor RCA has a longer tail than C-CMOS and the distribution of C-CMOS is more tighten. Obviously, for a given delay threshold, the 10-transistor RCAs yield loss due to violation of timing requirement is higher than C-CMOS. Then, we can say the C-CMOS based 16-bit RCA is robust better than 10-transistor based RCA.

The improved SB method is applied to pick up 1% tail points to build the CDF tail models for 5 considered RCAs. Fig. 14 shows the comparison of 5 tail models. Through the analysis of the tail models, the sequence of tolerant ability is: C-CMOS $>$ TFA $>$ TGA $>$ 10T $>$ CPL.

To prove the correctness of SB method based estimation, two other variability estimation approaches have been applied to the same 5 RCAs. First, the full MC simulation based robustness estimation has been done by computing the standard deviation and variance of the simulated 10,000 MC points. The standard deviation and variance for the 5 designs is listed in Table VII. As can be seen from the table, the result is the same with the proposed SB method based estimation.

The approach proposed in [22] has been used to show the robustness under process variation of the 5 different RCAs. The impact of process variability was evaluated through MC simulations performed on 10,000 samples under different supply voltages V_{DD}. The ratio between the maximum spread 3σ and the mean value μ ($i.e.$, $3\sigma/\mu$) was considered as a measure of tolerant ability of each design. As can be easily observed in Fig. 15, the delay variability is reduced for higher power

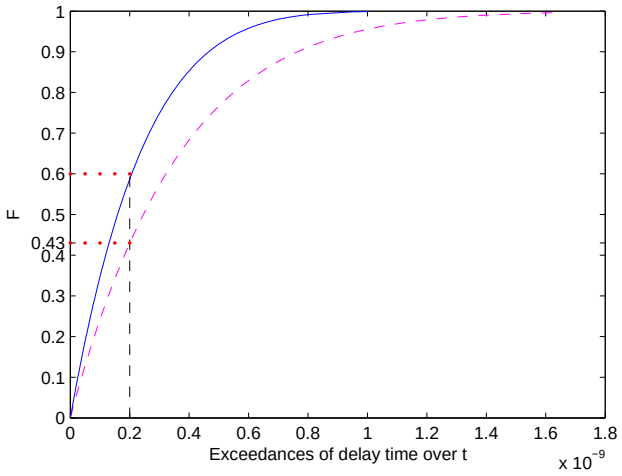

Fig. 13. Comparison of tail model between 16-bit C-CMOS and 10T.

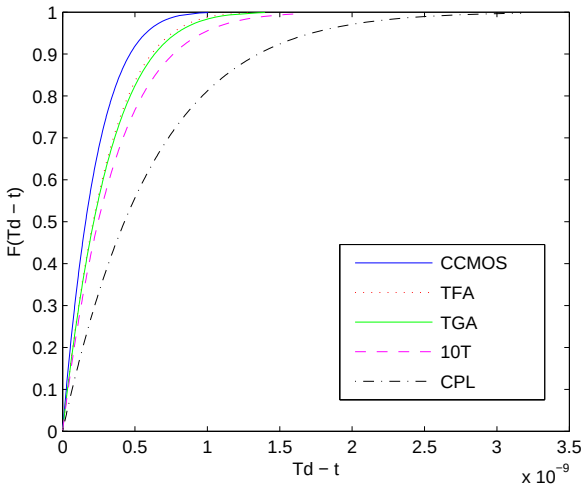

Fig. 14. The tail models (CDFs) for 5 RCAs.

supply voltages. The CCMOS based RCA is the less PV delay sensitive circuit. (its delay variability ranges from 5.6%@0.8V to 3.5%@1.2V). In contrast, the CPL based RCA is the most PV delay sensitive structure. Its delay variability ranges from 9.7%@0.8V to 5.2%@1.2V. This is resulting from 1.50× to 1.73× more delay sensitive with respect to CCMOS based RCA structure. Also, the estimation result is the same as the two previous results.

For the above mentioned approaches, both of them require totally 50,000 MC simulations for the robustness estimation of 5 RCAs (10,000 simulation for each RCA). And for proposed SB based estimation method, the number of simulations for each design is the sum of training sample size listed in Table III and the number of extracted tail points through classification shown in Table VIII.

From Table VIII, the total number of MC simulations required by proposed SB based estimation method is 6845. It is up to 7.3× faster than the other two methods. In other words, it only takes 13.89% running time of two aforementioned approaches.

TABLE VII
STANDARD DEVIATION AND VARIANCE FOR 5 RCAS

RCA	C-CMOS	CPL	TFA	TGA	10T
Standard Deviation	3.6002 e-010	7.7530 e-010	4.6593 e-010	5.7400 e-010	6.6522 e-010
Variance	1.2962 e-019	6.0109 e-019	2.1709 e-019	3.2947 e-019	4.4251 e-019

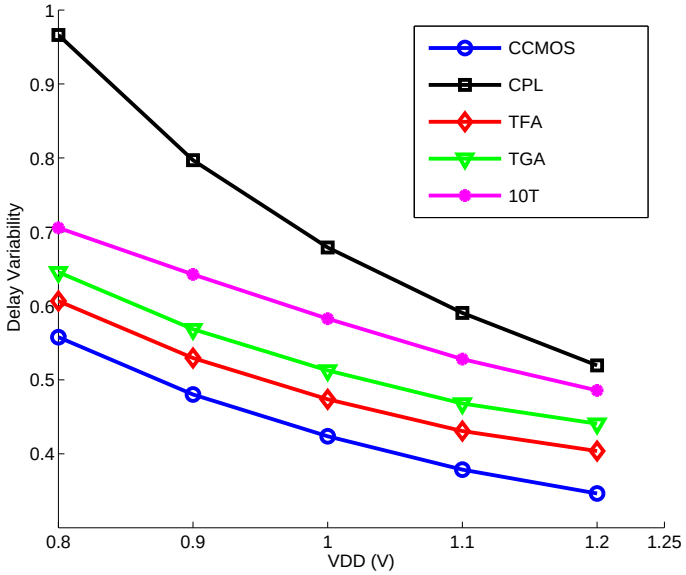

Fig. 15. Delay Variability for 5 RCAs with respect to V_{DD}.

TABLE VIII
NUMBER OF MC SIMULATIONS REQUIRED BY PROPOSED SB BASED ESTIMATION APPROACH

RCA	C-CMOS	CPL	TFA	TGA	10T
Extracted tail Points	321	313	295	306	310
Total MC Size	1021	1213	1595	1606	1410

VII. ADAPTIVE BODY BIAS (ABB) FOR YIELD COMPENSATION

Through the comparison of robust ability for different designs, the design which is robust worst has been picked up. How to reduce the variability and improve performance subject to a power constraint, and thus, improve the yield loss for a chosen design?

Adaptive Body Bias (ABB) is used to compensate for parameter variation, which reduce variability and improve performance and power consumption. The bias for PMOS transistor trends to be forward biased to improve the circuit performance with the threshold voltage increasing. While the bias for NMOS transistor is reverse biased to reduce the leakage current, thus reduce the power consumption with the threshold voltage decreasing. However, ABB has its own drawback: it requires an additional on-power distribution networks for body voltage, and thus, occupy additional silicon area.

In above experiments, CPL RCA is proved to be the robust worst. Thus, ABB is applied to improve its tolerant ability under process variation and also enhance the circuit performance. All the bulks of PMOS transistors in 16-bit CPL PCA are connected to the body bias voltage V_p and all the NMOS bulks are connected to V_n. In this paper, the optimum combination for V_p and V_n has been selected to achieve the best performance and power consumption for a trade off and improve its robustness. [23] suggests the range of the body bias voltage being $\pm 20\%$ of the nominal supply voltage (1.1V) mainly due to the fact that we would like to provide enough noise margin for body bias voltage signal in order not to allow the possibility of crossing the forward threshold of the transistor junctions by the body bias voltage. The exhaustive combinations for V_p and V_n has been simulated, then the optimum pair has been found out as V_p = 1.3V and V_n = -0.2V. As shown in the Table IX, the circuit performance and power consumption has been improved with less variability when optimal body bias voltage has been applied.

TABLE IX
COMPARISON BETWEEN WITH AND WITHOUT ABB FOR CIRCUIT PERFORMANCE AND POWER CONSUMPTION

V_p=1.3V V_n=-0.2V	Mean (Delay)	Std (Delay)	Var (Delay)	Mean (Power)	Std (Power)	Var (Power)
Without ABB	6.5909 e-0009	4.6618 e-010	2.1732 e-019	7.1623 e-005	4.6007 e-006	2.1167 e-011
With ABB	6.3544 e-0009	4.6255 e-010	2.1395 e-019	6.9692 e-005	4.3726 e-006	1.9128 e-011

VIII. Conclusions

Statistical Blockade is an efficient approach for rare events analysis. However, the classification threshold and number of training samples are still major issues which may result in inaccurate tail model and slow simulation speed. This paper presented an effective way to the problem of finding the minimum threshold of classifier t_c for the given tail part and the minimum number of samples n required for corresponding classification threshold. Based on proposed improved algorithms, the values of t_c and n are derived for high-replication block. When the values are applied to Statistical Blockade method, the resulting tail model shows a good match with empirical model. It offers both the fastest speed of simulation and highest accuracy for Statistical Blockade. The experimental results also indicates that the obtained minimum classification threshold t_c for 3%, 2% and 1% tails can be applied to all the latest technology corners from 45nm to 16nm in both HP and LP models. Additionally, a novel method combined with improved SB has been proposed to fast evaluate the tolerant ability under process variation for different designs. It can make computation time much shorter compared with full MC simulation method. Furthermore, the circuit delay and power consumption as well as its variability has been reduced for the chosen design using ABB technique by applying optimum body bias voltage. However, it may not significantly improve the yield loss of the products since there is still a distribution tail which may not meet the accepted requirement. Thus, ABB technique can be explored further to compensate the yield loss.

References

[1] A. Singhee and R. A. Rutenbar, "Statistical blockade: a novel method for very fast monte carlo simulation of rare circuit events, and its application," in *DATE '07: Proceedings of the conference on Design, automation and test in Europe*, 2007, pp. 1379–1384.

[2] S. P. Mohanty, "Unified challenges in nano-cmos high-level synthesis," in *Proc. 22nd International Conf. VLSI Design*, 2009, pp. 531–531.

[3] A. Srivastava, D. Sylvester, and D. Blaauw, *Statistical Analysis and Optimization for VLSI: Timing and Power*. New York: Springer, 2005.

[4] H. Chang and S. Sapatnekar, "Statistical timing analysis under spatial correlations," in *DAC Conference Proceedings*, Sept. 2005, pp. 1467–1482.

[5] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan, "First-order incremental block-based statistical timing analysis," in *DAC Conference Proceedings*, 2004, pp. 331–336.

[6] V. Khandelwal and A. Srivastava, "A general framework for accurate statistical timing analysis considering correlations," in *DAC Conference Proceedings*, 2005, pp. 89–94.

[7] L. Sun, J. Mathew, D. K. Pradhan, and S. P. Mohanty, "An intelligent statistical blockade method for fast robustness estimation and compensation of nano-cmos arithmetic circuits," in *Proceedings of the 2nd IEEE International Symposium on Electronic System Design (ISED)*, 2011, pp. accepted on 2011–08–25.

[8] A. Singhee and R. A. Rutenbar, "Statistical blockade: Very fast statistical simulation and modeling of rare circuit events, and its application to memory design," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 28, no. 8, pp. 1176–1189, Aug. 2009.

[9] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery 2*, pp. 121–167, 1998.

[10] C. C. Chang and C. J. Lin, *LIBSVM: a Library for Support Vector Machines*. http://www.csie.ntu.edu.tw/ cjlin/libsvm/, 2001.

[11] T. Joachims, *Making Large-Scale SVM Learning Practical*. Universität Dortmund: LS8-Report, 24, 1998.

[12] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. San Francisco: Morgan Kaufmann, 2005.

[13] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*. ACM Press, 1995, pp. 273–297.

[14] C. Cortes and V. Vapnik, "Support-vector networks," in *Machine Learning*, 1992, pp. 144–152.

[15] A. A. Balkema and L. de Haan, "Residual life time at great age," *Ann. Prob.*, vol. 2, no. 5, pp. 792–804, 1974.

[16] J. Wang, A. Singhee, and R. A. Rutenbar, "Two fast methods for estimating the minimum standby supply voltage for large srams," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 29, no. 12, pp. 1908–1920, Dec. 2010.

[17] A. Singhee, J. Wang, H. Calhoun, and R. A. Rutenbar, "Recursive statistical blockade: An enhanced technique for rare event simulation with application to sram circuit design," in *Proc. International Conf. VLSI Design*, 2008, pp. 131–136.

[18] A. Singhee and R. Rutenbar, *Novel Algorithms for Fast Statistical Analysis of Scaled Circuits*. Springer Science: Lecture Notes in Electrical Engineering 46, 2009.

[19] C. Chang, J. Gu, and M. Zhang, "A review of 0.18-μm full adder performances for tree structured arithmetic circuits," *IEEE Trans. Very Large Scale Integration System*, vol. 13, no. 6, pp. 686–695, June 2005.

[20] L. Sun, J. Mathew, D. K. Pradhan, and S. P. M. Mohanty, "Algorithms for rare event analysis in nano-cmos circuits using statistical blockade," in *Proc. International SoC Design Conference*, 2010, pp. 162–165.

[21] A. M. Shams, T. K. Darwish, and M. A. Bayoumi, "Performance analysis of low-power 1-bit cmos full adder cells," *IEEE Trans. Very Large Scale Integration System*, vol. 10, no. 1, pp. 20–29, Feb. 2002.

[22] B. Wong, F. Zach, V. Moroz, A. Mittal, G. Starr, and A. Kahng, *Nano-CMOS Design for Manufacturability*. NJ, USA: John Wiley & Sons: Hoboken, 2009.

[23] J. W. Tschanz, J. T. Kao, S. G. Narendra, R. Nair, D. A. Antoniadis, A. P. Chandrakasan, and V. De, "Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 11, pp. 1396–1402, Nov. 2002.