Algorithms for Rare Event Analysis in Nano-CMOS Circuits Using Statistical Blockade

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— **Accurate and fast characterization of the process variations of nano-CMOS circuits is becoming increasingly important for design for manufacturing (DFM) with highest yield. One of the ways to understand the circuit behavior under the process variations is to analyze the rare events that may happen due to such process variations. The Statistical Blockade (SB) is a approach for such rare events analysis. In SB, the classification threshold selection becomes very important for different tail regions which is related to the number of rare events simulation. This paper presents the values of classification threshold for different tail regions of typical circuits. It is shown that a given classifier requires different number of training samples depending on classification thresholds.**

I. Introduction

The scaling down of CMOS technology has resulted in significant deviations from the nominal values of transistor parameters, such as channel length and threshold voltage [1]. For example, variation in gate length increases from 35% in a 130 nm technology to 60% in a 65 nm technology which results in large variations in the performance of the designed circuit. Traditional corner based analysis and design approaches provide guard-bands for parameter variations and therefore, are prone to introducing pessimism in the design [2]. However, in the nano-CMOS circuits small variations due to inaccuracies in the manufacturing process can cause large relative variations in the behavior of the circuit [3].

Process variations can be classified into two as follows: (1) inter-die variations are the variations from die to die and (2) intra-die variations correspond to variability within a single chip [4], [1]. In modern nano-CMOS, intra-die variations are rapidly and steadily growing and can significantly affect the variability of performance parameters on a chip.

The existing statistical static timing analysis (SSTA) tools have recognized these unavoidably random aspects of manufacturing variations. For example, a linear model is used in [5] for the gate delay as a function of varying gate parameters. In [4], [6], simple grid-based models are used to represent the spatial correlation between gates correlation. The Principal Components Analysis (PCA) techniques are used to extract uncorrelated parameters from this correlation model. Standard Monte Carlo (MC) techniques are most efficient for sampling the statistically likely cases due to nature of its construction. However, these techniques are extremely slow when used for simulating statistically unlikely or rare events. For example, on an average 100 million circuit simulations are required to simulate 5 rare events [7]. In [8], Statistical Blockade (SB) is proposed as a MC technique that allows to efficient filtering of unwanted samples. However, it is difficult to classify all the tail points to build the accurate tail model for high-dimensional data. So usually there is a relaxed boundary for the classifier to block most unwanted points. The tail threshold of whole distribution is denoted as t. The classification threshold is denoted as t_c. In particular, t_c is defined as 97% to extract 1% tail samples (i.e. t = 99%) [8].

II. Contributions of this Paper

A detailed analysis of Statistical Blockade as a Monte Carlo technique is presented. This paper presents solutions for the following questions:

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?

After obtaining basic MC analysis, SB is used for analyzing rare events. This helps to obtain the minimum classification threshold t_c required for a given specification. Furthermore, initial training sampling depends strongly on the value of t_c and the portion of the tails that is intended to extract.

III. The Rare Event Statistics

The SB approach is used to efficiently generate samples in the tail of the distribution of the performance metric of a circuit [8]. Standard MC is not suitable as it generates samples that follow the complete distribution. The tail portion is the point of interest for a designer for worst case power consumption. In other words, the problem that is of interest in this paper is the rare event statistics. If the tail threshold is the 99% point of the distribution then only one out of 100 simulations is useful.

SB applies "Extreme Value Theory" (EVT) to circuit analysis by blocking out the MC points which are unlikely to fall in the tail region. The classifier used in this research is called Support Vector Machine (SVM) [9], which is implemented as: LIBSVM [10], SVM$^{\text{light}}$ [11] or WEKA [12]. The simulation data are divided into two classes: (1) body region and (2) tail region. Therefore, the C-Support Vector Classifier is chosen [13], [14]. There are 4 basic kernels of SVM as follows: (1) linear, (2) polynomial, (3) radial basis function (RBF), and (4) 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. 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 is reversed by increasing the weight of the tail.

2-dimensional data is used to train a SVM classifier in a NAND Gate for the varying threshold voltages. The randomly generated samples which are classified by the SVM classifier into 2 classes is shown in Fig. 1. The two main problems in SB method focussed are: (1) One is the appropriate classification threshold t_c for different tail regions - 3%, 2% and 1% of typical high-replication blocks. (2) The other one is the minimal training samples size for given classification threshold t_c.

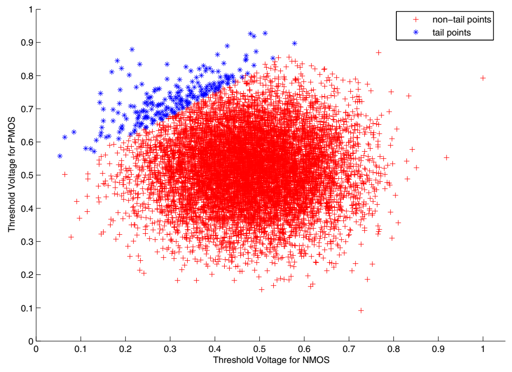

Fig. 1. Classification of Monte Calro Generated Samples.

IV. Proposed Algorithms for Rare Event Anlysis

The method 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 $\text{F}_{sim}(N)$ runs 10,000 MC simulation. Y is a vector of output values; e.g. delay for an 8-bit adder. The function Percentile(Y, t) calculates the value of delay V for the corresponding tail threshold t. Then function Size(Y_{tail}) is used to count the number of tail points after Y compared with V. The above steps are also used to select tail points from full MC simulation by empirical method. If threshold t is defined as 99% point in 10,000 points, tail count should be approximately 100. Table I lists the values of V and S for different tail thresholds t in the test case of an 8-bit adder.

The algorithm from step-7 to step-12 is SB method [3]. 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 SB 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.

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).
2:  $Y$ = $\text{F}_{sim}(N)$
3:  $V$ = Percentile($Y$, $t$)
4:  $Y_{tail} = \{Y_i \epsilon Y$: $Y_i > V\}$
5:  $S$ = Size($Y_{tail}$)
6:  for all $t_c = t$ do
7:      $x$ = MonteCarlo($n_0$)
8:      $y$ = $\text{F}_{sim}(x)$
9:      $V_c$ = Percentile($y$, $t_c$)
10:     $C$ = BuildClassifier($x$, $y$, $V_c$) // $C$ is a classifier
11:     $y_{tail}$ = $\text{F}_{sim}$(Filter($C$, $N$))
12:     $y_{tail,true} = \{y_i \epsilon y_{tail}$: $y_i > V\}$
13:     $S_c$ = Size($y_{tail,true}$)
14:     if $S_c < S$ then
15:         $t = t$ -1%
16:         Repeat
17:     else
18:         $t_c = t$
19:     end if
20: end for
```

TABLE I
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

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.

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

```
1:  Assume: Initial training sample size $n_0$ (e.g., $n_0$ = 100); total
    sample size $N$ (e.g., $N$ = 10,000).
2:  $Y$ = $F_{sim}(N)$
3:  $V$ = Percentile($Y$, $t$)
4:  $Y_{tail}$ = $\{Y_i \epsilon Y: Y_i > V\}$
5:  $S$ = Size($Y_{tail}$)
6:  for all (i = 1; $n_0 \leq$ 2000; i++) do
7:      $x$ = MonteCarlo($n_0$)
8:      $y$ = $F_{sim}(x)$
9:      $V_c$ = Percentile($y$, $t_c$)
10:     $C$ = BuildClassifier($x$, $y$, $V_c$)
11:     $y_{tail}$ = $F_{sim}$(Filter($C$, $X$))
12:     $y_{tail,true}$ = $\{y_i \epsilon y_{tail}: y_i > V\}$
13:     $S_i$ = Size($y_{tail,true}$)
14:     $n_0$ = $n_0$ + 100
15: end for
16: Find($S_i$ = $S$ and $S_{i+1}$ = $S_{i+2}$ = ... = $S_{20}$ = $S$)
17: $n$ = $i$ * 100
```

V. Experimental Results

The experiments are focused on six test cases: an 8-bit adder, an 8-bit subtracter, a 16-bit MUX, a 16-bit comparator, a 4-bit multiplier and a 3-stage single-ended voltage controlled ring oscillator. All of them are realized for 45nm CMOS. MC simulations of 10,000 runs were performed with global, Gaussian distributed threshold voltage and gate oxide thickness with 20% variation. The metric measured is power for VCO and delay (e.g. Fig. 2 for an 8-bit adder) for the rest of test cases.

Fig. 3 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. 3, 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

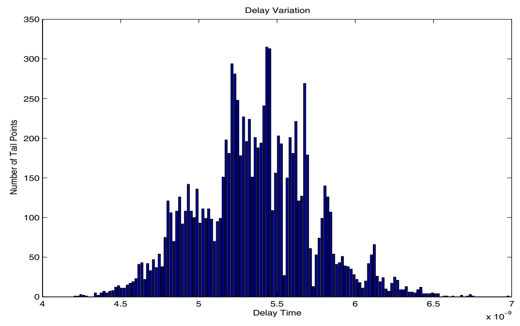

Fig. 2. 10,000 MC simulation for delay variation in 8-bit adder for 20% variation in gate oxide thickness and threshold voltage.

means about 10,000 more simulations will be needed. Thus, the best value of t_c is 98% to select 1% tail points.

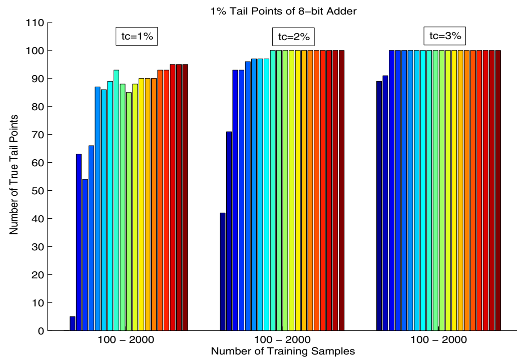

Fig. 3. True Tail Points obtained with 100$\sim$2000 Training Samples for Different Classification Threshold.

After t_c is chosen, the minimal training sample size for t_c = 98% is needed. Fig. 4 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. 4), 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 II.

TABLE II
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

Fig. 5 compares the conditional Cumulative Distribution (CDFs) of tail points between empirical method and statistical blockade using the values from Table II. It shows a good match of two CDFs. The empirical method needs 10,000 MC runs. In SB method, 344 tail points are picked up after classification (t_c = 98%). Another 900 runs are used to train the classifier.

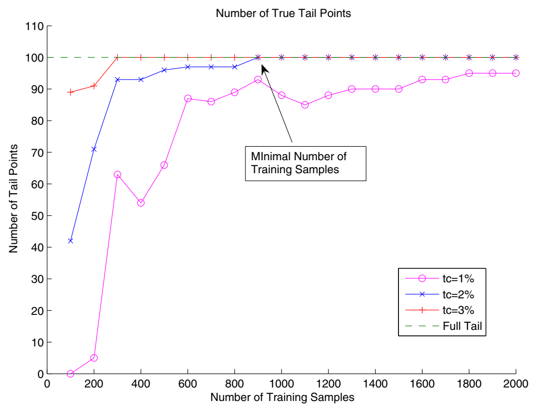

Fig. 4. Method for Obtaining Minimal Training Sample Size with Different Classification Threshold.

TABLE III

MINIMAL CLASSIFICATION THRESHOLD AND MINIMAL NUMBER OF TRAINING SAMPLES REQUIRED FOR DIFFERENT TAILS OF TEST CASES.

Blocks	Tail Threshold (t)	99%	98%	97%
8-bit	Minimal Classification Threshold t_c	98%	97%	96%
Adder	Minimal Training Sample Size n	900	800	1200
8-bit	Minimal Classification Threshold t_c	98%	97%	96%
sub.	Minimal Training Sample Size n	900	600	1400
	Minimal Classification Threshold t_c	98%	97%	97%
MUX	Minimal Training Sample Size n	800	1300	1500
	Minimal Classification Threshold t_c	98%	96%	94%
Comp.	Minimal Training Sample Size n	1200	1400	600
	Minimal Classification Threshold t_c	97%	96%	95%
Mult.	Minimal Training Sample Size n	1500	1500	1000
	Minimal Classification Threshold t_c	98%	95%	94%
VCO	Minimal Training Sample Size n	800	1400	1500

Totally it needs 1244 (344 + 900) MC 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 MC samples, which is impossible in industry. For large quantities of MC points, the simulation size of t_c = 98% is significantly fewer than t_c = 97%. For 10,000 MC simulation, it shows 10X speedup compared with empirical method. While it can demonstrate much higher speedup for millions or billions of MC samples.

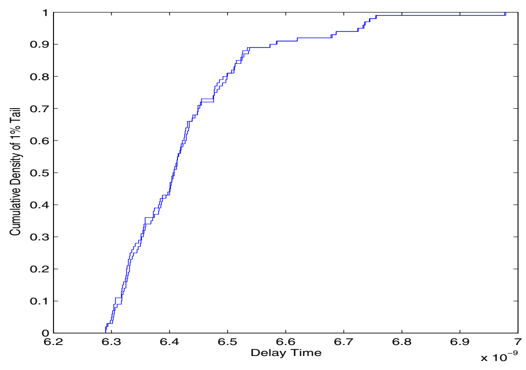

Fig. 5. Adder Delay Analysis: Empirical vs. Statistical Blockade.

Then values of t_c and n are obtained by repeating above processes for 2% and 3% tails for a 8-bit adder and other blocks. The t_c and n for six different blocks are indicated in Table III. For 8-bit adder, 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.

VI. CONCLUSIONS

Statistical Blockade is an efficient method 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 algorithms, the values of t_c and n are derived for six commonly used, high-replication blocks. 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.

VII. ACKNOWLEDGMENTS

The work was partially funded by the Engineering and Physical Science Research Council (EPSRC), United Kingdom, under Grant No: EP/G032904/1 and EP/F030991/1.

REFERENCES

[1] S. P. Mohanty, "Unified Challenges in Nano-CMOS High-Level Synthesis," in *Proc. 22nd International Conf. VLSI Design*, 2009, pp. 531–531.

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

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

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

[5] C. Visweswariah, et al., "First-order incremental block-based statistical timing analysis," in *Proc. DAC*, 2004, pp. 331–336.

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

[7] A. Singhee, et al., "Recursive statistical blockade: An enhanced technique for rare event simulation with application to SRAM circuit design," *Proc. International Conf. VLSI Design*, pp. 131–136, 2008.

[8] 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 *Proc. DATE*, 2007, pp. 1379–1384.

[9] T. Joachims, *Making Large-Scale SVM Learning Practical*. Universität Dortmund: LS8-Report, 24, 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," *MIT Press*, 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 *Proc. 5th Annual ACM Workshop on Computational Learning Theory*, 1995, pp. 273–297.

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