Statistical Blockade: Very Fast Statistical Simulation and Modeling of Rare Circuit Events and Its Application to Memory Design

Amith Singhee, *Member, IEEE*, and Rob A. Rutenbar, *Fellow, IEEE*

***Abstract*—Circuit reliability under random parametric variation is an area of growing concern. For highly replicated circuits, e.g., static random access memories (SRAMs), a rare statistical event for one circuit may induce a not-so-rare system failure. Existing techniques perform poorly when tasked to generate both efficient sampling and sound statistics for these rare events. Statistical blockade is a novel Monte Carlo technique that allows us to efficiently filter—to block—unwanted samples that are insufficiently rare in the tail distributions we seek. The method synthesizes ideas from data mining and extreme value theory and, for the challenging application of SRAM yield analysis, shows speedups of 10–100 times over standard Monte Carlo.**

***Index Terms*—Design automation, extreme values, memories, Monte Carlo methods, simulation, statistics, yield estimation.**

I. Introduction

CIRCUIT RELIABILITY under statistical process variation is an area of growing concern, as transistors scale deeply into the nanoscale regime. Designs that add excess safety margin or rely on simplistic assumptions about "worst case" corners no longer suffice. Worse, for critical circuits such as static random access memories (SRAMs) and flip-flops, replicated across ten thousand to ten million instances on a large design, we have the new problem that statistically rare events are magnified by the sheer number of these elements. In such scenarios, an exceedingly rare event for one circuit may induce a not-so-rare failure for the entire system.

Consider the case of a 1-Mb SRAM array, which has one million "identical" instances of an SRAM cell. These instances are designed to be identical, but due to manufacturing variations, they usually differ. Suppose we desire a *chip* yield of 99%, i.e., no more that one chip per 100 should fail. This chip yield requirement translates to a cell failure probability requirement: no more than 10.05 cells per billion should fail. In other words, the required *cell* yield is approximately 99.999999%. This failure probability is the same as for a 5.6 σ point on the standard normal distribution. If we want to estimate the yield of such an SRAM cell in the design phase, a standard Monte Carlo approach would require at least 100 million SPICE simulations on average to obtain just one failing sample point. Even then, the estimate of the yield or failure probability will be suspected because of the lack of statistical confidence, the estimate being computed using only one failing example. Such a large number of simulations are utterly intractable. This example clearly illustrates the widespread problem with designing robust memories in the presence of process variations: We need to simulate *rare* or *extreme* events and estimate the statistics of these rare events. The problem of simulating and modeling rare events stands for any circuit that has a large number of identical replications on the same chip, as in DRAM arrays and nonvolatile memories. We term such circuits as *high-replication circuits* (HRCs).

Note that systematic variations (e.g., proximity-based lithographic effects) can be well accounted for in SRAM cells, because they are typically small in size: The ubiquitous 6T SRAM cell contains only six transistors. What really cause significant variation then are the random interdevice variation sources, such as random dopant fluctuation (RDF) [1] and random poly-Si crystal orientation [1]. Furthermore, the impact of variations is roughly inversely proportional to the square root of the transistor area [2], and the transistors in SRAM cells tend to be of minimum size. Hence, SRAM cells are particularly susceptible to these random variations, increasing the need for an efficient yield estimation technique for these cells.

Memory designers have typically side stepped the problem of yield estimation by using multiple process and environmental corners with large safety margins. This approach, of course, is unreliable since it does not account for the actual statistics of the SRAM cell performance metrics. Worse, it usually results in significant overdesign, which translates to a waste of chip area and power. Monte Carlo simulation would be the ideal technique for reliably estimating the yield, but as we saw, it can be prohibitively expensive for HRCs.

One avenue of attack is to abandon Monte Carlo. Several analytical and semianalytical approaches have been suggested to model the behavior of SRAM cells [3]–[5] and digital circuits [6] in the presence of process variations. All suffer from approximations that are necessary to make the problem tractable or apply to a specific performance metric. Mukhopadhyay *et al.*

Manuscript received August 7, 2008; revised December 9, 2008. Current version published July 17, 2009. This work was supported in part by the Semiconductor Research Corporation (SRC) and in part by C2S2, the Focus Center for Circuit and System Solutions, which is one of the five research centers funded under the Focus Center Research Program, which is an SRC program. This paper was recommended by Associate Editor C. V. Kashyap.

A. Singhee is with IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 USA (e-mail: asinghe@us.ibm.com).

R. A. Rutenbar is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2009.2020721

[4] and Mahmoodi *et al.* [6] assume a linear relationship between the statistical variables and the performance metrics [e.g., static noise margin (SNM)] and assume that the statistical process parameters are normally distributed. These assumptions result in a normal distribution assumption for the performance metric too, which can suffer from gross errors, particularly while modeling rare events: We shall see examples in the result section. When the distribution varies significantly from Gaussian, Mukhopadhyay *et al.* [4] choose an F-distribution in an *ad hoc* manner. Bhavnagarwala *et al.* [3] present a complex analytical model limited to a specific long-channel transistor model (the transregional model) and further limited to only SNM analysis for the 6T SRAM cell. Calhoun and Chandrakasan [5] offer a more sophisticated model, but again, it models only the SNM for subthreshold SRAM cells under assumptions of independence and identical distribution of the upper and lower SNMs, which may not always be valid. All these methods are specific to either one circuit, one device model, or one performance metric.

A more general approach is the *most probable point* (MPP) method [7], also known as the *worst case distance* method in the electronic design automation community. This is a two-step technique: 1) Estimate the most likely failure point (the MPP) in the statistical variable space using an optimization formulation and 2) estimate the failure region boundary with a predefined function form passing through this MPP. The failure region, estimated by this boundary, can then be integrated over the probability distributions of the variables to obtain the failure probability. Although it is more flexible than the previously described methods, the MPP method suffers from some practical shortcomings. First, an efficient implementation would require gradient and, possibly, curvature information, which are often not easily available. Second, the accuracy is limited by the error in the MPP search and the fit of the predefined function form to the actual failure boundary. If the boundary approximation is bad, there will be an unavoidable error in the yield computation. Lastly, the optimizer may suffer from undesirable convergence issues when applied to objectives computed from nonlinear circuit simulation.

A different avenue of attack is to modify the Monte Carlo strategy. Hocevar *et al.* [8] show how importance sampling can be used to predict failure probabilities. Recently, Kanj *et al.* [9] applied an efficient formulation of these ideas for modeling rare failure events of single 6T SRAM cells, based on the concept of *mixture importance sampling* (MixIS) from [10]. The approach uses real SPICE simulations with no approximating equations. However, the method only estimates the failure (exceedance) probability of a *single* threshold value of the performance metric. A rerun is needed to obtain probability estimates for another failure threshold: No complete model of the tail of the distribution is computed. The shape of the tail can provide further insight to the designer about the sensitivity of the yield to the performance threshold. Knowledge of the tail also allows a desirable yield-centric design flow: The designer can answer questions like "What is the minimum SRAM cell supply voltage needed for a yield level of at least 6 σ?" This allows the design flow to target a yield level rather than a performance specification. MixIS also combines all performance metrics to compute a failure probability, given fixed thresholds. Hence, there is no way to obtain separate probability estimates for each metric, other than a separate run per metric.

In this paper, we develop a novel, general, and efficient Monte Carlo method that addresses both of the problems previously mentioned: very fast generation of 1) rare event samples and 2) sound models of the rare event (distribution tail) statistics for any performance metric. It imposes almost no *a priori* limitations on the statistics of the statistical parameters or on the device models or performance metrics. The method is conceptually simple and employs ideas from two rather nontraditional sources: *extreme value theory* (EVT) and *machine learning*.

To obtain both samples and statistics for rare events, we need to generate and evaluate an intractable number of Monte Carlo samples. *Generating* each sample is neither challenging nor expensive: We are merely creating the parameters for a circuit. *Evaluating* the sample is expensive, because we simulate it. What if we could quickly *filter* these samples and *block* those that are unlikely to fall in the low-probability tails we seek? Many samples could be generated, but very few are simulated. We show how to exploit ideas from machine learning [11] to build *classifier* structures, from a small set of Monte Carlo training samples, to create the necessary blocking filter. Given these samples, we show how to use the rigorous mathematics of EVT [12] (the theory of the limiting behavior of sampled maxima and minima) to build sound models of these tail distributions. It is this essential "blocking" activity of the filter that gives the technique its name: *statistical blockade*. An initial version of this paper was presented in [13]. An expanded version of this material is to appear in [14].

Techniques from the world of machine learning have begun to make their way into the circuit computer-aided design community, e.g., [15]. However, this is not the case for EVT, which offers a deep and useful set of tools for these problems. EVT [12] is a branch of probability that studies and optimally quantifies the statistics of, as the name suggests, extreme or rare events. It has found wide statistical application in fields such as hydrology [16], insurance [17], and finance [17] among several others: Wherever there is a need to estimate the probability of rare events. One of the most consequential applications, and, indeed, one of the driving forces for the development of the theory of extremal statistics, was the Dutch dike project following the disastrous North Sea flood of 1953 that claimed over 1800 human lives. One aspect of the postflood response was to determine appropriate heights for the sea dikes in the Netherlands, such that the probability of a flood in a year is reduced to some very small amount (e.g., 10^{-4}). Our problem of estimating extreme values of the SRAM SNM, using a limited number of Monte Carlo samples, is similar in flavor to the dike height problem (if not in impact on the human condition). Hence, we can employ the same technical tools from EVT for our problem. However, the circuit yield estimation problem is more "extreme" in the sense of the failure probabilities to be estimated: often 10^{-8} to 10^{-9} or smaller. We shall show how to filter an intractable set of Monte Carlo samples down to a practical set of simulation points and use EVT to obtain sound statistics for the tail distributions that govern the statistics of these filtered samples.

The remainder of this paper is organized as follows. As background, we begin in Section II by developing the mathematical relations that link the yield of a complete memory array to the yield of one constituent memory cell; this allows us to proceed in subsequent sections with a cell-level analysis of memory yield. In Section III, we define the rare event statistics problem and review relevant results from EVT. We highlight the applicable limit theorems for the distributions of rare events and then show how we can use these results for statistical inference from data. Section IV introduces our novel framework for fast sampling of rare events and for modeling their statistics and discusses some practical implementation issues. We present experimental results demonstrating the effectiveness of statistical blockade on relevant circuit test cases in Section V. Finally, Section VI offers concluding remarks.

II. Memory Array Yield and Cell Yield

Memory yield requirements are usually specified for the entire array. For example, the designer may be comfortable with an array failure probability of one in a thousand, i.e., $P_{f,\text{arr}} < 10^{-3}$. However, how does this translate to a yield requirement for each atomic bit-level memory cell? What is the maximum cell failure probability $P_{f,\text{cell}}$ allowed to satisfy this array failure probability requirement? In this section, we answer this question, given that most high-replication cells are memory cells (e.g., SRAM cells). The answer is essential for any practical statistical yield analysis of memories.

A. Computing Cell Yield From Array Yield

Let us talk in terms of the failure probability rather than the yield to avoid the use of confusing negatives in our discussion. Note that the relation between yield and failure probability (P_f) is simply

$$Yield = 1 - P_f. \tag{1}$$

Hence, they are equivalent. $P_{f,\text{arr}}$, which is the array failure probability, is the probability of one or more cells failing in any one array. Hence, if we manufacture ten arrays, and six of them have a total of eight faulty cells, our array failure probability is (approximately) 6/10 and not 8/10. Another equivalent way to define $P_{f,\text{arr}}$ is the probability of the worst cell in an array being faulty. Hence, if we are measuring some performance metric y (e.g., SNM), we are really interested in the statistics of the worst value of y from among N values, where N is the number of cells in our array. Let us define this worst value as M_N, and let us assume that, by worst, we mean the maximum, i.e., large values of y are bad. If, for some case, small values of y are bad, then we can use the maximum of $-y$ as the worst value. Hence

$$M_N = \max(Y_1, Y_2, \ldots, Y_N) \tag{2}$$

where $Y_1, \ldots, Y_N$ are the measured performance metric values for the N cells in an array. Also suppose that the failure threshold for y is y_f, i.e., any cell with $y > y_f$ is defined as failing. Then, $P_{f,\text{arr}}$ can be written as

$$P_{f,\text{arr}} = P(M_N > y_f) = 1 - P(M_N \leq y_f). \tag{3}$$

Now, $P(M_N \leq y_f)$ is the probability of the worst case cell passing, or, equivalently, all cells in the array passing[1]

$$P(M_N \leq y_f) = P(Y_1 \leq y_f, \ldots, Y_N \leq y_f) = [P(Y \leq y_f)]^N. \tag{4}$$

Since cell failure probability is

$$P_{f,\text{cell}} = P(Y > y_f) \tag{5}$$

we can combine (3)–(5) to obtain

$$P_{f,\text{arr}} = 1 - (1 - P_{f,\text{cell}})^N \tag{6}$$

$$P_{f,\text{cell}} = 1 - (1 - P_{f,\text{arr}})^{\frac{1}{N}}. \tag{7}$$

B. Incorporating Redundancy

One common approach to improve fault tolerance in memories is to have redundant columns in the array, which are used to replace defective columns in the chip: See, for example, [19]. In the simplest case, if there is one redundant column, the chip can tolerate one failing cell. The presence of redundant bits, of course, changes the statistics of the array failure probability in relation to the cell failure probability. We will now develop a simple model to allow the computation of process-induced cell failure probability $P_{f,\text{cell}}$, given the maximum tolerable array failure probability $P_{f,\text{arr}}$, in the presence of redundancy in the array.

Let there be r redundant bits in an array. For the case of column redundancy, this would mean that we have r redundant columns. If each column has n_c cells, then we can tolerate up to rn_c faulty cells, depending on their spatial configuration. However, the probability of having two or more faulty cells in the same column can be shown to be negligibly small, given the fact that cell failure probability is extremely low in practice, and column sizes are reducing as we move to advanced technologies. Hence, we consider only the case where all faulty cells are in different columns. This reasonable approximation helps keep tedious mathematics away, without losing the core insights of the derivations. In this case, the array would operate without faults with up to r faulty cells in it, and the array failure probability is the probability of having more than r faulty cells in it. If the probability of having k faulty cells in an array is $P_{f,k}$, then we can write the array failure probability $P_{f,\text{chip}}$ as

$$P_{f,\text{arr}} = 1 - \sum_{k=0}^{r} P_{f,k} \tag{8}$$

where

$$P_{f,k} = {}^{N}C_k P_{f,\text{cell}}^k (1 - P_{f,\text{cell}})^{N-k}. \tag{9}$$

Using these relations, we can relate array failure probability with cell failure probability in the presence of redundancy.

[1]Here, we have assumed that failure of a cell is independent of the failure of other cells in the array. This is reasonable because of the following reasons: 1) Parametric SRAM cell failure is induced primarily due to local mismatch induced asymmetry; 2) local mismatch tends to be dominated by random dopant fluctuation [18]; and 3) random dopant fluctuation is independent from device to device. However, similar relations between array and cell failure probabilities incorporating correlation may easily be used in the proposed statistical blockade framework.

However, (9) is inconvenient because, for large N (one million or more), the ${}^{N}C_{k}$ term can be inconveniently large, and the $P_{f,\text{cell}}^{k}$ term can be inconveniently small for floating point arithmetic. A more convenient relation is given by the popular Poisson yield model.

C. Poisson Yield Model

Let us define λ as the expected number of faulty cells in every N cells we manufacture

$$\lambda = P_{f,\text{cell}} N. \tag{10}$$

Then, we can write (9) as

$$P_{f,k} = \frac{N!}{(N-k)!N^k} \times \left(1-\frac{\lambda}{N}\right)^{N} \times \left(1-\frac{\lambda}{N}\right)^{-k} \times \frac{\lambda^k}{k!}. \tag{11}$$

Note that the array size N is very large compared to all relevant values of k and λ. For large values of N, we can approximate the first three product terms in this equation, to obtain

$$P_{f,k} \approx 1 \times \mathrm{e}^{-\lambda} \times 1 \times \frac{\lambda^k}{k!} = \frac{\lambda^k \mathrm{e}^{-\lambda}}{k!}. \tag{12}$$

Hence, $P_{f,k}$ closely follows the well-known Poisson distribution for large N. Now, we can write the array failure probability of (8) as

$$P_{f,\text{arr}} \approx 1 - \sum_{k=0}^{r} \frac{(P_{f,\text{cell}}N)^k \mathrm{e}^{-P_{f,\text{cell}}N}}{k!}. \tag{13}$$

Hence, given a specification for array yield, we can compute the required cell yield with this equation. Conversely, for a given cell design, if we can estimate the cell yield, then we can compute the corresponding array yield. In the rest of this paper, we develop a general, yet efficient, method to estimate the failure statistics of any given cell design.

III. EVT: Modeling Rare Event Statistics

EVT provides us with the mathematical tools to build models of the tails of distributions. It is the theory of the limiting behavior of sampled maxima and minima of distributions and can be thought of roughly as playing the same role for the extrema of distributions as the celebrated central limit theorem [17] plays for the means of distributions. We review the essential elements of EVT in this section. For clarity, we again use SRAM circuit behaviors as our guiding examples.

A. Problem

Suppose we want to model the rare event statistics of the write time of an SRAM cell. Fig. 1 shows an example of the distribution of the write time. We see that it is skewed to the right with a *heavy* right tail. A typical approach is to run a Monte Carlo with a small sample size (e.g., 1000) and fit a standard analytical distribution to the data, for example, a normal or lognormal distribution. Such an approach can be accurate for fitting the "body" of the distribution but will be grossly inaccurate in the tail of the distribution: The skewness

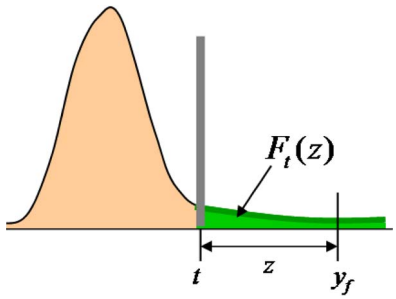


Fig. 1. Possible skewed distribution for some SRAM metric (e.g., write time).

of the actual distribution or the heaviness of its tail will be difficult to match. As a result, any prediction of the statistics of rare events, lying far in the tail, will be very inaccurate.

Let F denote the cumulative distribution function (CDF) of the write time y, and let us define a tail threshold t to mark the beginning of the tail (e.g., the ninety-ninth percentile). Let z be the excess over the threshold t. We can write the *conditional* CDF of the tail as

$$F_t(z) = P(Y - t \le z | Y > t) = \frac{F(z+t) - F(t)}{1 - F(t)} \tag{14}$$

and the overall CDF as

$$F(z+t) = (1 - F(t))\, F_t(z) + F(t). \tag{15}$$

If we know $F(t)$ and can estimate the conditional CDF of the tail $F_t(z)$ accurately, we can accurately estimate rare event statistics. For example, the yield for some extreme threshold $y_f > t$ is given as

$$F(y_f) = (1 - F(t))\, F_t(y_f - t) + F(t) \tag{16}$$

and the corresponding failure probability $\bar{F}(y_f) = 1 - F(y_f)$ is given as

$$\bar{F}(y_f) = (1 - F(t))\,(1 - F_t(y_f - t))\,. \tag{17}$$

$F(t)$ can be accurately estimated using a few thousand simulations, since t is not too far out in the tail. Then, the problem here is to efficiently estimate the conditional tail CDF F_t as a simple analytical form, which can then be used to compute statistical metrics such as (16) and (17) for rare events. Of course, here, we assume that any threshold y_f of interest will be far into the tail, such that $y_f \gg t$. This is easily satisfied for any real HRC scenario, for example, our 1-Mb cache example from Section I. We also assume that the extreme values of interest lie only in the *upper* tail of the distribution. This is without any loss of generality, because any lower tail can be converted to the upper tail by replacing $y = -y$. This same approach of fitting a *generalized Pareto distribution* (GPD) to the exceedances over some threshold has been developed and widely applied by hydrologists under the name of the *peaks over threshold* method [17]. In their case though, the data are from historical record and not synthetically generated. We now look at some results from EVT, which are directly applicable to the problem of estimating the tail CDF.

B. EVT: Modeling Tail Distributions

An important distribution in the theory of extreme values if the GPD, which has the following CDF:

$$G_{\xi,\beta}(z) = \begin{cases} 1 - \left(1 - \xi\frac{z}{\beta}\right)^{1/\xi}, & \xi \neq 0; \quad z \in D(\xi,\beta) \\ 1 - \mathrm{e}^{-z/\beta}, & \xi = 0; \quad z \geq 0 \end{cases} \tag{18}$$

where

$$D(\xi,\beta) = \begin{cases} [0,\infty), & \xi \leq 0 \\ [0,\beta/\xi], & \xi > 0. \end{cases}$$

It is defined by two parameters: ξ and β.

We exploit the following seminal result in this work.

Theorem 1 (Balkema and de Haan [20] and Pickands [21]): For every $\xi \in \mathbb{R}$, $F \in MDA(H_\xi)$ if and only if

$$\lim_{t\to\infty} \sup_{z\geq 0} \left|F_t(z) - G_{\xi,\beta(t)}(z)\right| = 0 \tag{19}$$

for some positive function $\beta(t)$, if and only if F is in the *maximum domain of attraction* (MDA) of the *generalized extreme value* (GEV) *distribution*: $F \in MDA(H_\xi)$. Here, $G_{\xi,\beta}(z)$ is the GPD.

In other words, if any distribution F satisfies the given condition ($F \in MDA(H_\xi)$, its conditional tail distribution F_t converges to a GPD as we move further out in the tail. Examples of GPDs are shown in Fig. 2. We see that, for $\beta = 1$, the values of $\xi \leq 1/2$ produce distributions that resemble to commonly seen distribution tails and that the GPD can also match finite support tails, as for $\xi \geq 1/2$ in Fig. 2(a). Let us look at the condition in Theorem 1 in more detail.

The GEV is a one-parameter distribution, and its CDF is as follows:

$$H_\xi(y) = \begin{cases} \mathrm{e}^{-(1-\xi y)^{1/\xi}}, & \xi \neq 0 \\ \mathrm{e}^{-\mathrm{e}^{-y}}, & \xi = 0, \end{cases} \quad \text{where } 1 - \xi y > 0. \tag{20}$$

It combines three simpler distributions into one unified form. These distributions are

$$\Phi_\alpha(y) = \begin{cases} 0, & y \leq 0 \\ e^{-y^{-\alpha}}, & y > 0, \end{cases} \quad \alpha > 0 \text{ (Fréchet)} \tag{21}$$

$$\Psi_\alpha(y) = \begin{cases} e^{-(-y)^{\alpha}}, & y \leq 0 \\ 1, & y > 0, \end{cases} \quad \alpha > 0 \text{ (Weibull)} \tag{22}$$

$$\Lambda(y) = e^{-e^{-y}}, \qquad y \in \mathbb{R} \text{ (Gumbel)}. \tag{23}$$

They are obtained from the GEV as follows.

1) $\xi = -\alpha^{-1} < 0$ gives the Fréchet CDF Φ_α.
2) $\xi = \alpha^{-1} > 0$ gives the Weibull CDF Ψ_α.
3) $\xi = 0$ gives the Gumbell CDF Λ_α.

Let us now look at what the "MDA" means. Suppose that $Y_1, Y_2, \ldots$ is a sequence of independent identically distributed random variables from the CDF F. For any sample $\{Y_1, Y_2, \ldots, Y_N\}$ of size N, define the *sample maximum* as

$$M_N = \max(Y_1, Y_2, \ldots Y_N), N \geq 2. \tag{24}$$

From (4), we know that

$$P(M_N \leq y) = F^N(y). \tag{25}$$

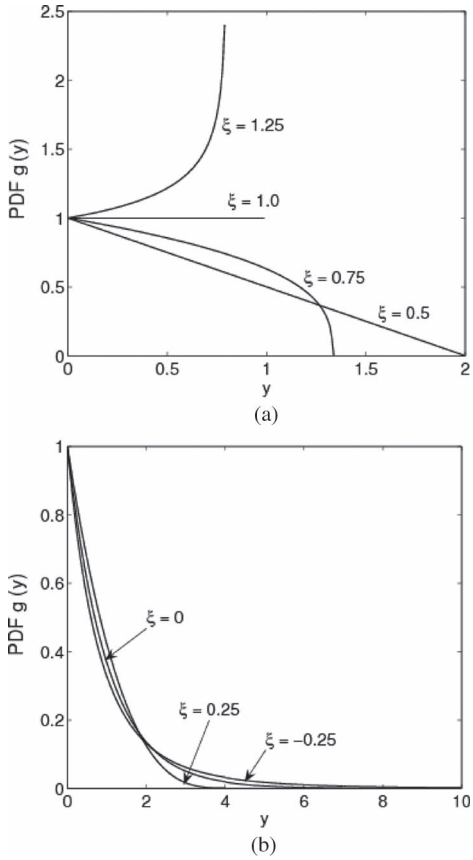


Fig. 2. Probability density function for a GPD with $\beta = 1$. We get long unbounded tails for $\xi \leq 0$. (a) $\xi \geq 1/2$. (b) $\xi < 1/2$.

Suppose that there exist normalizing constants a_N, b_N, and some nondegenerate CDF H, such that

$$P\left(\frac{M_N - b_N}{a_N} \leq y\right) = F^N(a_N y + b_N) \to H(y) \text{ as } N \to \infty, y \in \mathbb{R}. \tag{26}$$

Then, we say that F lies in the MDA of H or $F \in MDA(H)$. In other words, the maxima of N i.i.d. random variables with CDF F, when properly normalized, converge in distribution to a random variable with the distribution H.

Fisher and Tippett [22] showed the following.

Theorem 2 (Fisher–Tippett [22]):

$$F \in MDA(H) \Rightarrow H \text{ is of type } H_\xi$$

where H_ξ is the GEV defined in (20).

Hence, if any distribution F lies in $MDA(H)$, then H must be the GEV H_ξ, for some ξ. The conditions for which $F \in MDA(H)$ for some nondegenerate H are quite general and known well: see [17]. For example, we list some common distributions belonging to $MDA(H_\xi)$, in Table I. Hence, for a very large class of distributions, Theorem 1 holds true.

This is an extremely useful result: It implies that, if we can generate enough points in the tail of a distribution ($y \geq t$), in most practical cases, we can fit the simple analytical GPD to the data and make predictions further out in the tail. This approach would be independent of the circuit or the performance metric

TABLE I
SOME COMMON DISTRIBUTIONS LYING IN MDA (H_ξ). FOR A LONGER LIST, SEE [17]

H_ξ	Distributions in MDA(H_ξ)
$\Phi_{-1/\xi}$	Cauchy Pareto Loggamma
$\Psi_{1/\xi}$	Uniform Beta
Λ	Normal Lognormal Gamma Exponential

being considered. This result also suggests that most prior *ad hoc* fitting strategies are at best suboptimal and, at worst, simply wrong. Of course, two important questions remain as follows: 1) How do we efficiently generate a large number of points in the tail ($y \geq t$)? and 2) how do we fit the GPD to the generated tail points?

We address the latter question first, in the next section.

C. Estimating the Tail: Fitting the GPD to Data

For now, let us suppose that we can generate a reasonably large number of points in the tail of our performance distribution. For this, we might theoretically use standard Monte Carlo simulation with an extremely large sample size or, more practically, the statistical blockade sampling method proposed in Section IV. Let this data be $\mathbf{Z} = (Z_1, \ldots, Z_n)$, where each Z_i is the *exceedance* over the tail threshold t ($Z_i > 0 \quad \forall i$). All Z_i are i.i.d. random variables with common $CDF\ F_t$. Then, we have the problem of estimating the optimal GPD parameters ξ and β from this tail data, so as to best fit the conditional tail $CDF\ F_t$. There are several options, e.g., as follows:

1) probability-weighted moment (PWM) matching;
2) maximum likelihood estimation (MLE);
3) moment matching.

For a comparison of these methods, see [23]. All three methods can be completely automated. Manual methods based on graphical exploration of the data are also possible but are not of interest to us here: See [17] for a review.

PWMs are linear combinations of and, hence, equivalent to L-moments [24]. L-moments are derived from the order statistics of the data sample and have been seen to give more accurate estimates than standard moments in many cases. For many distributions where the standard moments are nonlinear functions of the distribution parameters, the estimated standard moments may show large errors. L-moments/PWMs being linear functions of the order statistics tend to behave much better in these cases. One example where this has been seen to be true is the Wakeby distribution of which the generalized Pareto is a special case. The L-moments are also more robust to the presence of outliers and less subject to bias in the approximation. For detailed expositions on the L-moments and PWMs, see, for example, [24], [25], and [26]. Based on an extensive simulation study, Hosking and Wallis [23] suggest that estimates from the PWM method often have lower bias than those from moment matching and MLE for sample sizes up to 500. Although MLE performs the desired task of finding the most likely solution, the MLE search is shown to suffer from some convergence problems when ξ is estimated close to 1/2. Because of these reasons, we choose PWM matching for the purpose of this paper.

The PWMs [27] of a continuous random variable Y with CDF K are generalizations of the standard probability moments [23] and are defined as

$$M_{p,r,s} = E\left[Y^p K^r(Y)\left(1 - K(Y)\right)^s\right]. \tag{27}$$

The standard pth moment is given by $M_{p,0,0}$. For the GPD ($K = G_{\xi,\beta}$), we have a convenient relationship between $M_{1,0,s}$ and (ξ, β), given by

$$m_s = M_{1,0,s} = \frac{\beta}{(1+s)(1+s+\xi)}, \qquad \xi > 0. \tag{28}$$

Then, we can write

$$\beta = \frac{2m_0 m_1}{m_0 - 2m_1} \qquad \xi = \frac{m_0}{m_0 - 2m_1} - 2. \tag{29}$$

We estimate these PWMs and, hence, the GPD parameters from the data sample, as

$$\hat{m}_s = \frac{1}{n}\sum_{i=1}^{N}(1-q_i)^s Y_{i,n} \tag{30}$$

where

$$Y_{1,n} \leq Y_{2,n} \leq \cdots \leq Y_{n,n} \tag{31}$$

is the *ordered sample*, and

$$q_i = \frac{i+\gamma}{n+\delta} \tag{32}$$

with $\gamma = -0.35$ and $\delta = 0$, as suggested in [23]. The estimates $(\hat{\xi}, \hat{\beta})$ converge to the exact values as $n \to \infty$ and are asymptotically normally distributed with covariance given by [23], (33) see at the bottom of the page.

Although this is only an asymptotic relation, and not what we would observe with realistic sample sizes, it does capture the general error behavior. As would be expected, the estimation error decreases with the tail sample size (n). The accuracy can be arbitrarily improved by increasing n. This gives us a convenient knob for tuning the accuracy by employing more or less computational resource.

We note here that traditional methods like Gaussian fitting or asymptotic probability extraction [28] try to match the moments

$$\Sigma_{\hat{\xi},\hat{\beta}} \to \frac{n^{-1}}{(1+2\xi)(3+2\xi)} \times \begin{bmatrix} (1+\xi)(2+\xi)^2(1+\xi+2\xi^2) & \beta(2+\xi)(2+6\xi+7\xi^2+2\xi^3) \\ \beta(2+\xi)(2+6\xi+7\xi^2+2\xi^3) & \beta^2(7+18\xi+11\xi^2+2\xi^3) \end{bmatrix} \tag{33}$$

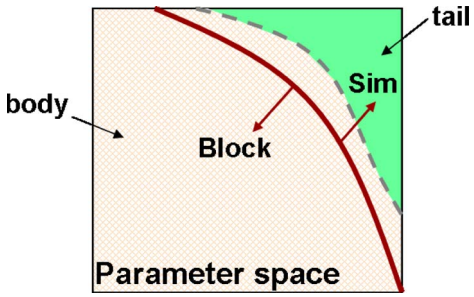


Fig. 3. Tail and body regions in the statistical parameter space. The dashed line is the exact tail region boundary for tail threshold t. The solid line is the relaxed boundary modeled by the classifier for a classification threshold $t_c < t$.

of the overall CDF F. However, here, we are surgically looking at the tail of F and matching the (weighted) moments of the tail CDF F_t. The EVT results of Section III-B allow us to maximally remove the effect of the body of F in estimating the tail. Previous methods are usually unable to achieve this surgical separation of the body and tail in the fitting process.

Once we have estimated a GPD model of the conditional CDF above a threshold t, we can estimate the failure probability for any value y_f by substituting the GPD in (17) as

$$P(Y > y_f) \approx (1 - F(t))\,(1 - G_{\xi,\beta}(y_f - t)). \tag{34}$$

The next section addresses the important remaining question: How do we efficiently generate a large number of points in the tail $(y \geq t)$?

IV. Statistical Blockade

Let any circuit performance metric or, simply, output y be computed as

$$y = f_{\text{sim}}(\mathbf{x}). \tag{35}$$

Here, $\mathbf{x}$ is a point in the statistical parameter (e.g., V_t and t_{ox}) space or, simply, the input space, and f_{sim} includes expensive SPICE simulation. We assume that y has some probability distribution F, with an extended tail. Suppose we define a large tail threshold t for y, and then, from the developments in Section III-B, we know that we can approximate the conditional tail CDF F_t by a GPD $G_{\xi,\beta}$. Section III-C shows how we can estimate the GPD parameters (ξ, β) using data drawn from the tail distribution. We now introduce our efficient tail sampling strategy that will generate the tail points for fitting this GPD.

Corresponding to the tail of output distribution F, we expect a "tail region" in the input space: Any statistical parameter values drawn from this tail region will give an output value $y > t$. Fig. 3 shows an example of such a tail region for two inputs. The rest of the input space is called the "body" region, corresponding to the body of the output distribution F. In Fig. 3, these two regions are separated by a dashed line. The key idea behind the proposed sampling technique is to identify the tail region and simulate only those Monte Carlo points that are likely to lie in this tail region. Here, we exploit the common fact that *generating* the random values for a Monte Carlo sample point is very cheap compared to actually *simulating* the point as in (35). Hence, if we generate points as in standard Monte Carlo, but *block*—not simulate—those points that are unlikely to fall in the tail region, we can drastically cut down the total time spent. This reduction in time spent is drastic because we are trying to simulate only the rare events, which, by definition, constitute a very small percentage of the total Monte Carlo sample size. The algorithm derives its name from this blocking activity.

We use a *classifier* to distinguish between the tail and body regions and to block out the body points. A classifier [11] is an indicator function that takes as input any point in the input space (the statistical parameter space) and predicts the membership of this point in one of multiple classes (the "body" or "tail" classes). In the context of Fig. 3, it essentially builds a model of the boundary between the tail and body regions. Using this model of the boundary, it can label points from the Monte Carlo sample set as either "tail" or "body." Only the "tail" points are then simulated. We also refer to this classifier as the *blockade filter* and its blocking activity as *blockade filtering*.

To build this model of the tail region boundary, the classifier can be trained with a small (e.g., 1000 points) training set of simulated Monte Carlo sample points. The specific type of classifier that we have used for our implementation of statistical blockade is called *support vector machine* (SVM) [11], [29]. The time taken for classifier training and classification is negligible compared to the total simulation time. Apart from this practical consideration, there is no restriction on the type of classifier that can be used. Classification is a rich and active field of research in the data mining community, and there are many options for choosing a classifier [11]. SVMs are a particularly popular well-researched classification strategy, and optimized software implementations are readily available, for example, SVM$^{\text{light}}$ [30] and WEKA [31].

It is difficult, if not impossible, to build an *exact* model of the boundary in general. Misclassifications, at least on points unseen during training, are unavoidable. Hence, we relax the accuracy requirement to allow for classification error. This is done by building the classification boundary at a *classification threshold* t_c that is less than the tail threshold t. Since we have assumed that only the upper (or right) tail is relevant, the tail region corresponding to t will be a subset of the tail region corresponding to t_c, if $t_c < t$. This will help to ensure that, even if the classifier is imperfect, it is unlikely that it will misclassify points in the true tail region (defined by t). The relaxed boundary corresponding to such a t_c is shown as solid line in Fig. 3.

A. Statistical Blockade Algorithm

The statistical blockade algorithm is shown as Algorithm 1. The thresholds $t = p_t$th percentile and $t_c = p_c$th percentile are estimated from the small initial Monte Carlo run, which also gives the n_0 training points for the classifier. Typical values for these constants are shown in Algorithm 1.[2] The function `MonteCarlo`(n) generates n points in the statistical parameter space, which are stored in the $n \times s$ matrix $\mathbf{X}$, where s is the

[2]These typical values have been determined empirically to provide a good tradeoff between classifier accuracy, simulation time, and tail model fit. Different implementations of the statistical blockade framework (different sample sizes and different classification schemes) may require an adjustment to these values.

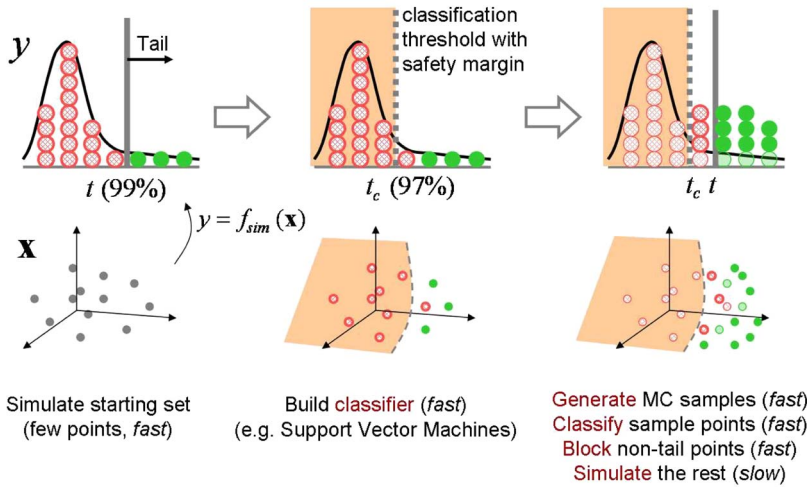


Fig. 4. Efficient tail (rare event) sampling method of statistical blockade.

input dimensionality. Each row of $\mathbf{X}$ is a point in s dimensions. $\mathbf{y}$ is a vector of output values computed from simulations. The function $\texttt{BuildClassifier}(\mathbf{X}, \mathbf{y}, t_c)$ trains and returns a classifier using the training set $(\mathbf{X}, \mathbf{y})$ and classification threshold t_c. The function $\texttt{Filter}(C, \mathbf{X})$ blocks the points in $\mathbf{X}$ classified as "body" by the classifier C and returns only the points classified as "tail." $\texttt{FitGPD}(\mathbf{y}_{\texttt{tail}} - t)$ computes the parameters (ξ, β) for the best GPD approximation $G_{\xi,\beta}$ to the conditional CDF of the exceedances of the tail points in $\mathbf{y}_{\text{tail}}$ over t. We can then use this GPD model to compute statistical metrics for rare events, for example, the failure probability for some threshold y_f, as in (34). This sampling procedure is also shown in Fig. 4.

Algorithm 1 The *statistical blockade* algorithm for efficiently sampling rare events and estimating their probability distribution.

Require: training sample size n_0 (e.g., 1 000); total sample size n; percentages p_t (e.g., 99%), p_c (e.g., 97%)

1: $\mathbf{X} = \texttt{MonteCarlo}(n_0)$
2: $\mathbf{y} = f_{\text{sim}}(\mathbf{X})$
3: $t = \text{Percentile}(\mathbf{y}, p_t)$
4: $t_c = \text{Percentile}(\mathbf{y}, p_c)$
5: $C = \texttt{BuildClassifier}(\mathbf{X}, \mathbf{y}, t_c)$ // C is a classifier
6: $\mathbf{y} = f_{\text{sim}}$ (Filter(C, $\texttt{MonteCarlo}(n)$))
7: $\mathbf{y}_{\texttt{tail}} = \{y_i \in \mathbf{y} : y_i > t\}$
8: $(\xi, \beta) = \texttt{FitGPD}(\mathbf{y}_{\texttt{tail}} - t)$

Two technical details of note are the issues of *choosing* and *unbiasing* the classifier. The algorithm places no restrictions on the choice of classifier. However, we make some practical observations here that are relevant for the choice of classifier. HRCs naturally tend to be small relatively simple circuits. It is highly unlikely that a complex large circuit will be replicated thousands to millions of times on the same chip. This level of replication often naturally coincides with simple functionality. As a result, we often do not expect to see drastically nonlinear boundaries for the tail regions of these circuits nor do we expect to see very complex topologies of the tail regions. These considerations, along with the safety margin awarded by a classification threshold t_c less than t, led us to use linear SVMs. Indeed, linear SVMs suffer minimally from the undesirable overfitting issues and the complex parameter selection problems of nonlinear kernel-based SVMs. As we shall demonstrate with experiments in Section V, this choice does result in an effective implementation of statistical blockade. For cases where a strongly nonlinear boundary exists, a linear classifier may not suffice, and more sophisticated classification techniques may be required [11]. The statistical blockade framework, however, should not need any fundamental change.

An important technical point to note about the classifier construction is as follows. The training set will typically have many more body points than tail points. Hence, even if all or most of the tail points are misclassified, the training error will be low as long as most of the body points are correctly classified. This will result in a classifier that is biased to allow more misclassifications of points in the tail region. However, we need to minimize misclassification of tail points to avoid distorting the statistics of the simulated tail points. Hence, we need to reverse bias the classification error. Using the technique proposed in [32], we penalize the misclassifications of tail points more than the misclassifications of body points. Let γ_t and γ_b be possibly different penalty factors for the "tail" and "body" classes: Misclassifications of any training points in the tail (body) region are penalized by a factor of γ_t (γ_b). If, as in [32], we choose

$$\frac{\gamma_t}{\gamma_b} = \frac{\text{Number of "body" points}}{\text{Number of "tail" points}}$$

we can obtain an unbiased classifier.

V. Experimental Results

We now apply the statistical blockade method to the following three test cases:

1) a 6T SRAM cell;
2) a complete 64-b SRAM column with write driver;
3) a master–slave flip-flop (MSFF) with the scan chain component.

The initial training sample used to construct each blockade filter is from a standard Monte Carlo run of $n_0 = 1000$ points. The filter is an SVM classifier built using the ninety-seventh percentile of each relevant performance metric as the classification threshold t_c. The tail threshold is defined as the ninety-ninth percentile.

In all cases, the rare event statistic we compute is the failure probability $\bar{F}(y_f)$ for any failure threshold y_f, using the GPD fit to the tail defined by the tail threshold t. We represent this failure probability as the equivalent quantile y_σ on the standard normal distribution

$$y_\sigma = \Phi^{-1}\left(1 - \bar{F}(y_f)\right) = \Phi^{-1}\left(F(y_f)\right) \tag{36}$$

where Φ is the standard normal CDF. For example, a failure probability of $\bar{F} = 0.00135$ implies a cumulative probability of $F = 1 - \bar{F} = 0.99865$. The equivalent point on a standard normal, having the same cumulative probability, is $y_\sigma = 3$. In other words, any y_f with a failure probability of 0.00135 is a "3σ" point.

We can compute $\bar{F}(y_f)$, and hence y_σ, in three different ways as follows.

1) *Empirically*. Run a large Monte Carlo simulation where all points are fully simulated, i.e., with no use of blockade filtering or EVT. For example, we use a sample size of n_{MC} (e.g., one million), giving us n_{MC} values $y_i, i = 1, \ldots, n_{\mathrm{MC}}$. Then, we can empirically compute $\bar{F}(y_f)$ as

$$\bar{F}(y_f) \approx \frac{|\{y_i : y_i > y_f\}|}{n_{\mathrm{MC}}}. \tag{37}$$

   Of course, for any $y_f > \max(\{y_1, \ldots, y_{n_{\mathrm{MC}}}\})$, we will get the same estimate of zero failure probability, and $y_\sigma = \infty$, since there are no points beyond this y_f to give us any information about such rare events. Hence, the prediction power of the empirical method is limited by the Monte Carlo sample size.

2) *Using the GPD model, with no blockade filtering*. We can run a full Monte Carlo simulation with no filtering, as in the empirical estimation case, but then fit a GPD to the points in the tail defined by the tail threshold t. These are the points $\{y_i : y_i > t\}$. Using this GPD $G_{\xi,\beta}$ in (34), we can compute the failure probability. $F(t)$ can be estimated empirically with good accuracy. The GPD model extends the prediction power beyond the maximum y_i value in the Monte Carlo sample set.

3) *Using statistical blockade*. Here, we use the complete statistical blockade flow, where only candidate tail points identified by the blockade filter are simulated, and a GPD tail model is estimated from the actual tail points $y > t$. Here, too, we use (34), but the points used to estimate (ξ, β) are obtained from blockade filtering. Furthermore, we use a Monte Carlo sample size that is much smaller than for method 2), to test statistical blockade in a practical setting, where we want to use as small a sample size as possible.

For all the test cases, we compare the predictions of y_σ from these three methods. Method 2) gives the most accurate estimates, since it uses a large number of points and no filtering. In some cases, we also show estimates computed using a Gaussian distribution fit to highlight the error in such an approach. Of course, a Gaussian may not be the choice if the distribution form is known, but here, we assume that we do not know the type of the distribution F. The Gaussian is chosen as a popular representative for a variety of distributions that may be chosen *ad hoc* in such cases. Let us now look at the test circuits in more detail, along with the results we obtain.

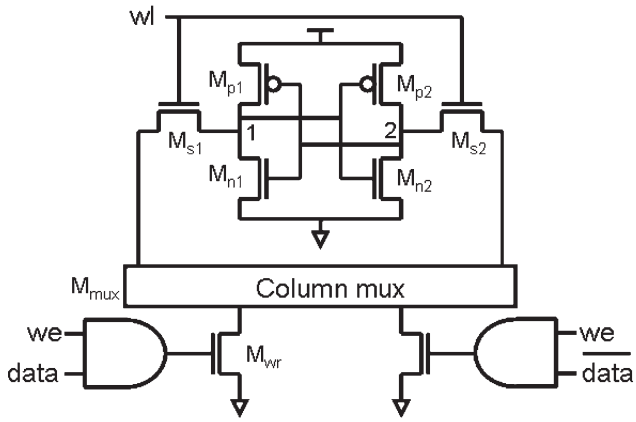


Fig. 5. A six-transistor SRAM cell with write driver and column mux.

A. 6T SRAM Cell

The first test case is a standard 6T SRAM cell with bit lines connected to a column multiplexor and a nonrestoring write driver, shown in Fig. 5. We use the Cadence 90-nm Generic PDK library, with independent normally distributed threshold voltage (V_t) variation per transistor and a global gate oxide thickness (t_{ox}) variation, which is also normally distributed. The V_t standard deviation is (about 18% of nominal V_t)

$$\sigma_{V_t} = \frac{5\,\mathrm{mV}}{\sqrt{WL}} \tag{38}$$

where W and L are the transistor width and length in micrometers. The standard deviation for t_{ox} is taken as 2% of the nominal value. This gives us a total of nine statistical parameters. The metric being measured is the write time τ_w: the time between the word line going high and the nondriven cell node (node 2) transitioning. Here, "going high" and "transitioning" imply crossing 50% of the full voltage change. For methods 1) and 2), we use $n_{\mathrm{MC}} = 1$ million Monte Carlo points. For statistical blockade [method 3)], 100 000 Monte Carlo points are filtered through the classifier, generating 4379 tail candidates. On simulating these 4379 points, 978 true tail points $(\tau_w > t)$ were obtained, which were then used to compute a GPD model for the tail conditional CDF. Table II shows a comparison of the y_σ values estimated by the three different methods. We can see a close match between the predictions by the accurate method 2) and statistical blockade, method 3). Note that the GPD model for each method does not change across the rows of the columns. Fig. 6 shows the conditional tail CDFs computed from the empirical method and from statistical blockade, showing a good match.

TABLE II
PREDICTION OF FAILURE PROBABILITY AS y_σ BY METHODS 1), 2), AND 3), FOR A 6T SRAM CELL. THE NUMBER OF SIMULATIONS FOR STATISTICAL BLOCKADE INCLUDES THE 1000 TRAINING SAMPLES. THE WRITE TIME VALUES ARE SHOWN IN "FAN-OUT OF 4" UNITS

τ_w (y_f) (FO4)	(I) Standard Monte Carlo	(II) GPD *no* blockade filter	(III) Statistical blockade
2.4	3.404	3.408	3.379
2.5	3.886	3.886	3.868
2.6	4.526	4.354	4.352
2.7	∞	4.821	4.845
2.8	∞	5.297	5.356
2.9	∞	5.789	5.899
3.0	∞	6.310	6.493
Num. Sims.	1,000,000	1,000,000	5,379

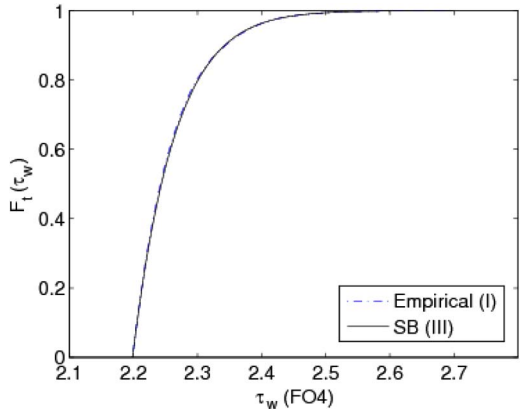


Fig. 6. Comparison of GPD tail model from statistical blockade (5379 simulations) and the empirical tail CDF (one million simulations) for the write time of the 6T SRAM cell.

Some observations highlighting the efficiency of statistical blockade can be made immediately as follows.

1) The empirical method fails beyond 2.6 FO4, corresponding to about 1 ppm circuit failure probability, because there are no points generated by the Monte Carlo run so far out in the tail.
2) Fitting a GPD model to the tail points [method 2)] allows us to make predictions far out in the tail, even though we have no points that far out.
3) Using blockade filtering, coupled with the GPD tail model, we can drastically reduce the number of simulations (from one million to 5379) with very small change to the tail model.

B. 64-b SRAM Column

The next test case is a 64-b SRAM column, with a nonrestoring write driver and a column multiplexor, shown in Fig. 7. Only one cell is being accessed, while all the other word lines are turned off. Random threshold variations on all 402 transistors (including the write driver and column mux) are considered, along with a global gate oxide variation. The device and variation models are the same 90-nm technology as for the 6T SRAM cell. In scaled technologies, leakage current is no longer negligible [33]. Hence, process variations on transistors that are meant to be inaccessible (or off) can also impact the overall behavior of a circuit. This test case allows us to see the

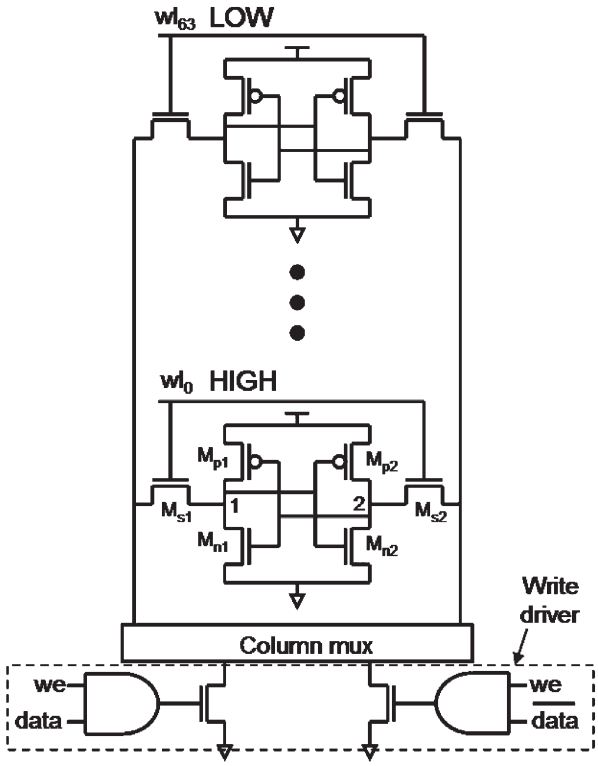


Fig. 7. A 64-bit SRAM column with write driver and column multiplexor.

impact of variations in the leakage current passing through the 63 off cells, along with variations in the write driver. Since the BSIM3v3 models [34] are used, the gate leakage is not well modeled, but the drain leakage is.

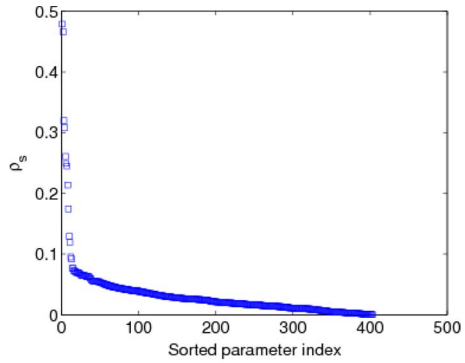
Once again, we measure the write time, in this case, from the word line wl_0 to node 2, for falling node 2. The number of statistical parameters is 403. Building a reliable classifier with only $n_0 = 1000$ points in 403 dimensional space is nearly impossible. However, we can reduce the dimensionality by choosing only those dimensions (statistical parameters) that have a significant impact on the write time. We use Spearman's rank correlation coefficient ρ_S (39) between each statistical parameter and the circuit performance metric to quantitatively estimate the strength of their relationship. Spearman's rank correlation [35] between two variables x and y, given the sample set $\{x_j, y_j\}_{j=1}^n$, is given by

$$\rho_S(x,y) = \frac{\sum_{j=1}^n (P_j - \bar{P})(Q_j - \bar{Q})}{\sqrt{\sum_{j=1}^n (P_j - \bar{P})^2}\sqrt{\sum_{j=1}^n (Q_j - \bar{Q})^2}} \quad (39)$$

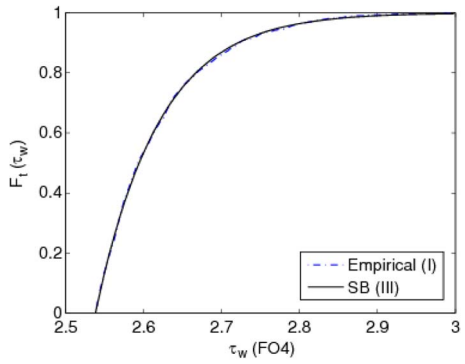
where P_j and Q_j are the *ranks* of x_j and y_j in the sample set, as shown by the example in Table III. To compute the rank of, for example, x_j, we sort all the x values in increasing order and take the position of x_j in this sorted list as its rank. $\bar{P}$ and $\bar{Q}$ denote the means of the ranks. Hence, ρ_S is just Pearson's linear correlation on the ranks. However, this measure of correlation does not assume linearity like the latter and hence gives more relevant estimates of the sensitivities. For classification, only parameters with $|\rho_S| > 0.1$ are used, reducing the dimensionality to only 11. Note that we have used the same 1000-point

TABLE III
EXAMPLE ILLUSTRATING THE CONCEPT OF RANKS FOR SPEARMAN'S RANK CORRELATION. THE RANK OF A VALUE IS ITS POSITION IN A SORTED LIST OF ITS CLASS, FOR EXAMPLE, 0.76 IS THIRD IN THE LIST OF x VALUES SORTED IN INCREASING ORDER

x	P	y	Q
0.1	2	101	2
-0.1	1	89	1
0.89	4	130	3
0.76	3	132	4



(a)



(b)

Fig. 8. Results for the SRAM column test circuit. (a) Magnitudes of rank correlation between the statistical parameters and the write time of the SRAM column. Only a few parameters have a strong relationship with the write time. (b) GPD model of SRAM column write time from statistical blockade (6314 simulations) compared with the empirical conditional CDF (100 000 simulations).

sample for both ranking and classifier training. Fig. 8(a) shows the sorted magnitudes of the 403 rank correlation values: We can see that only a handful of the statistical parameters have significant correlation with the write time. The transistors (the threshold voltages) chosen by this method are as follows:

1) the pull-down and output transistors in the active write-driver AND gate;
2) the bit line pull-down transistors;
3) all transistors in the active 6T cell, except for M_{p2} (since node 2 is being pulled down in this case).

This selection coincides with a designer's intuition of the devices that would have the most impact on the write time.

y_σ is computed for increasing failure thresholds, using all three methods. We use $n_{\mathrm{MC}} = 100\,000$ simulated Monte Carlo points for methods 1) and 2). For statistical blockade, method 3), we filter these 100 000 points through the classifier in reduced dimensions, giving 5314 candidate tail points. On

TABLE IV
PREDICTION OF FAILURE PROBABILITY AS y_σ BY METHODS 1), 2), AND 3) AND BY GAUSSIAN APPROXIMATION, FOR THE SRAM COLUMN. THE NUMBER OF SIMULATIONS FOR STATISTICAL BLOCKADE INCLUDES THE 1000 TRAINING SAMPLES. THE WRITE TIME VALUES ARE SHOWN IN "FAN-OUT OF 4" UNITS

τ_w (y_f) (FO4)	(I) Standard Monte Carlo	(II) GPD *no* filter	(III) Statistical blockade (100K)	(III) Statistical blockade (20K)	Gaussian approximation
2.7	2.966	2.986	3.010	2.990	3.364
2.8	3.367	3.373	3.390	3.425	3.898
2.9	3.808	3.743	3.747	3.900	4.432
3.0	∞	4.101	4.088	4.448	4.966
3.1	∞	4.452	4.416	5.138	5.499
3.2	∞	4.799	4.736	6.180	6.033
3.3	∞	5.147	5.049	–	6.567
3.4	∞	5.496	5.357	–	7.100
Num. Sims.	100,000	100,000	6,314	2,046	20,000

simulation, we finally obtain 1077 true tail points. Table IV compares the predictions by these three methods. We can see the close match between the accurate method 2) and statistical blockade, even though the total number of simulations is reduced from 100 000 to 6314. The empirical method, once again, falls short of our needs, running out of data beyond $\tau_w = 2.9$ FO4. Fig. 8(b) graphically shows the agreement between the conditional tail models extracted empirically and using statistical blockade.

We further reduce the Monte Carlo sample size for statistical blockade, to see if the simulation cost can be further reduced while maintaining accuracy. We use statistical blockade on only 20 000 Monte Carlo points, giving 1046 filtered candidate tail points and 218 true tail points. However, the predictions (column 5) show large errors compared to our reference, method 2). This suggests that a tail sample of only 218 is not sufficient to obtain a reliable model. We also use a Gaussian fit to 20 000 simulated Monte Carlo points for estimating y_σ. It is clear from the table that, in this case, a Gaussian fit underestimates the failure probability, with the error increasing as we move to rarer events.

Comparing the statistics for the SRAM column in Table IV with the statistics for the SRAM cell in Table II, we can see that the distribution of write time has a larger spread for the SRAM column than for the SRAM cell. For example, the 4.8 σ point for the SRAM cell is 2.7 FO4, while for the SRAM column, it is 3.2 FO4. The reason for this increased spread is that the variations in the leakage current of the entire column contribute significantly to the variation of the performance of any single cell. This shows that, in general, simulating variations in a single circuit, without modeling variations in its environment circuitry, can lead to large errors in the estimated statistics.

C. MSFF With Scan Chain

The last test case is a commonly seen MSFF with scan chain shown in Fig. 9. Flip-flops are ubiquitous in digital circuits and can be highly replicated in large chips. The circuit has been implemented using the 45-nm CMOS predictive technology models of [36]. The variations considered are RDF for all transistors and one global gate oxide thickness (t_{ox}) variation. The RDF is modeled as normally distributed independent threshold voltage (V_t) variation, with

$$\sigma_{V_t} = \frac{13.5 V_{t0}}{\sqrt{WL}} \tag{40}$$

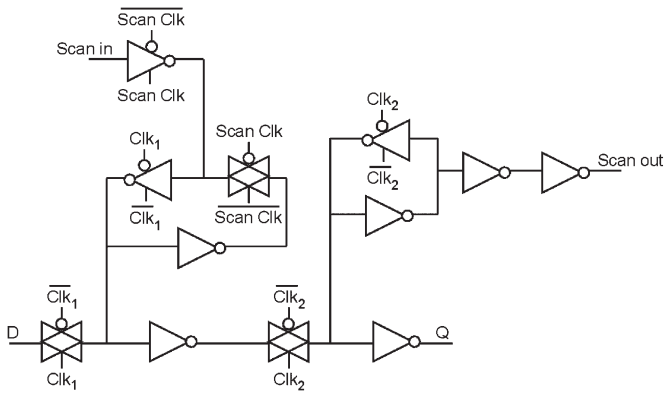


Fig. 9. Master–slave flip-flop with the scan chain component.

where W and L are the transistor width and length in nanometers, and V_{t0} is the nominal threshold voltage. This results in about 30% standard deviation for a minimum-sized transistor. This is large for current CMOS technologies, but we want to make sure that statistical blockade is powerful enough for future technologies too, where larger variations will be inevitable. The standard deviation for t_{ox} is taken as 2% of the nominal value, since t_{ox} is typically better controlled than RDF. The number of statistical parameters is 31, and the metric we are measuring is the clock-output delay τ_{cq}. The flip-flop has a peculiarity in its rare event behavior. For large deviations in the statistical parameters, the flip-flop reaches metastable behavior, and using standard circuit simulators, we fail to see the flip-flop output converges to a stable low/high value. This leads to an undefined clock-output delay for some Monte Carlo points. We reject any such points without replacement in this experiment. Although these rejected points are also rare events, they distort the smoothness of the tail that is required to apply the EVT limit theorem, if not rejected. This still allows us to test the speed and tail modeling efficiency of statistical blockade, since we use the same rejection method across all estimation methods. In practice, the fraction of such undefined delay events can be estimated from the simulated points in statistical blockade and added to the failure probability estimated using the GPD model, to give the overall failure probability.

For methods 1) and 2), we simulate $n_{\mathrm{MC}} = 500\,000$ Monte Carlo points. For statistical blockade [method 3)], we filter 100 000 Monte Carlo points to obtain 7785 candidate tail points which, on simulating, yield 692 tail points. Note that, here, we have ignored any tail points for which the flip-flop output did not converge to a stable low/high value. Fig. 10(b) shows the conditional tail CDFs from the empirical and statistical blockade models. We also compare predictions from these methods with those from a Gaussian fit to 20 000 simulated Monte Carlo points. Fig. 10(a) shows a histogram of the delay values obtained from the 500 000-point Monte Carlo run. The extreme skewness and the heavy tail of the histogram suggest that a Gaussian fit would be grossly inaccurate.

Table V shows the estimates of y_σ computed by these four methods: We can clearly see the gross errors in the Gaussian estimates. The GPD fits do, however, capture the heavy tail of the distribution. To see this, compare Table V with the results for the SRAM cell in Table II. A 20% increase in the SRAM

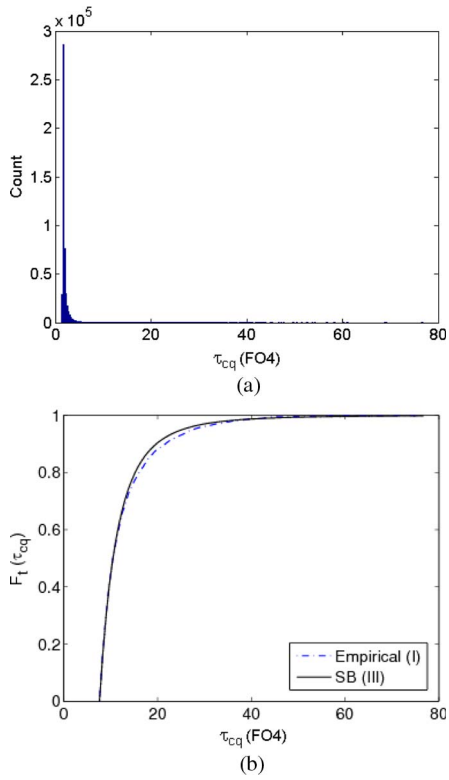


Fig. 10. Results for the MSFF test circuit. (a) Histogram of clock-output delay of the MSFF, showing a long heavy tail and high skewness. (b) GPD model of MSFF delay from statistical blockade (8785 simulations) compared with the empirical conditional CDF (500 000 simulations).

write time, from 2.5 FO4 to 3 FO4, results in an increase of 2.424 in y_σ, while a similar percentage increase in the MSFF delay, from 50 FO4 to 60 FO4, increases y_σ by only 0.136, even though the increases are from similar probability levels (3.886 σ for the SRAM cell and 3.854 σ for the MSFF).

In summary of these results, we see that statistical blockade provides an efficient way of sampling rare events and modeling their statistics. We see large improvements in fitting accuracy over simple Gaussian fits, on using our GPD model. Furthermore, we also see large speedups over simple Monte Carlo, ranging from roughly one to two orders of magnitude.

VI. Convergence Behavior of the Tail Model

The GPD tail model can be used to make predictions regarding rare events that are farther out in the tail than any of the data we used to compute the GPD model. Indeed, this is the compelling reason for adopting the GPD model. However, from (33), we expect the statistical confidence in the estimates to decrease as we predict farther out in the tail. We can estimate the confidence interval in the following way. Replace the unknown exact values of (ξ, β) in (33) with the estimated $(\hat{\xi}, \hat{\beta})$ to obtain an estimate of the covariance matrix $\Sigma_{\hat{\xi},\hat{\beta}}$. Then, sample the GPD parameters from the joint normal with this covariance matrix and the mean vector $(\hat{\xi}, \hat{\beta})$, to compute different estimates of some quantile (e.g., the 5 σ point). Compute a confidence interval from this set of quantile estimates.

TABLE V
PREDICTION OF FAILURE PROBABILITY AS y_σ BY METHODS 1), 2), AND 3) AND BY GAUSSIAN APPROXIMATION, FOR THE MSFF. THE NUMBER OF SIMULATIONS FOR STATISTICAL BLOCKADE INCLUDES THE 1000 TRAINING SAMPLES. THE DELAY VALUES ARE SHOWN IN "FAN-OUT OF 4" UNITS

τ_{cq} (y_f) (FO4)	(I) Standard Monte Carlo	(II) GPD *no* blockade filter	(III) Statistical blockade	Gaussian approximation
30	3.424	3.466	3.431	22.127
40	3.724	3.686	3.661	30.050
50	4.008	3.854	3.837	37.974
60	4.219	3.990	3.978	45.898
70	4.607	4.102	4.095	53.821
80	∞	4.199	4.195	61.745
90	∞	4.283	4.282	69.669
Num. Sims.	500,000	500,000	8,785	20,000

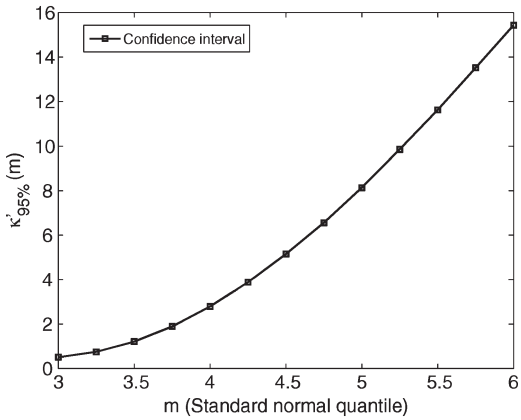


Fig. 11. Ninety-five percent confidence intervals as a percentage of the estimate.

Let $y_i(m)$ be the ith estimate of the $m\sigma$ point, and let $y_{97.5\%}(m)$ and $y_{2.5\%}(m)$ be the 97.5% and 2.5% percentile points, respectively. A 95% confidence interval is then

$$\kappa_{95\%}(m) = y_{97.5\%}(m) - y_{2.5\%}(m). \tag{41}$$

We express this interval as a percentage of the $y(m)$ estimated using $(\hat{\xi}, \hat{\beta})$

$$\kappa'_{95\%}(m) = \frac{\kappa_{95\%}(m)}{y(m)} \times 100. \tag{42}$$

Fig. 11 shows the $\kappa'_{95\%}(m)$ against m for the case of the SRAM cell write time ($y = \tau_w$). Ten thousand GPD parameter pairs were sampled for computing these interval estimates. The tail model was estimated using 1000 points, and the tail threshold t is at 2.326 σ (ninety-ninth percentile). We see that, to keep the error within $\pm 5\%$ with a confidence of 95%, we should not be predicting farther than 5.3 σ. At 6 σ, the error is $\pm 8\%$. Hence, the predictions from the tail model can be trusted only up to some distance from the tail threshold. This distance is determined by the number of points used to fit the GDP model. Hence, if we use a higher tail threshold t to fit more extreme tail regions, we would also need to ensure that the number of tail points is not reduced.

A practical implementation of statistical blockade may require a criterion for stopping further sample generation. Although an exact theory of the exact dependence of accuracy on the sample size remains elusive, (33) and (42) provide reasonable approximations. One possible stopping criterion can then be a maximum threshold on the estimated confidence interval (e.g., $\kappa'_{95\%} < 5\%$): Keep generating more samples until this threshold is reached. Similar estimators for the confidence intervals of the GPD parameters are available in the existing literature for other fitting methods like MLE and moment matching [23].

VII. CONCLUSION

Statistical blockade is a novel, efficient, and flexible framework for the following: 1) generating samples in the tails of distributions of circuit performance metrics and 2) deriving sound statistical models of these tails. This enables us to make predictions of failure probabilities given thresholds far out in the tails. This capability has become critical for reliable and efficient design of HRCs, such as SRAMs, as transistor sizes move deeply into the nanometer regime. Our methods offer both significantly higher accuracy than standard Monte Carlo and speedups of one to two orders of magnitude across a range of realistic circuit test cases and variations. Our future work will focus on extending the range of the tail estimates to extremely rare events with good confidence.

REFERENCES

[1] M. Hane, T. Ikezawa, and T. Ezaki, "Atomistic 3D process/device simulation considering gate line-edge roughness and poly-Si random crystal orientation effects," in *IEDM Tech. Dig.*, 2003, pp. 9.5.1–9.5.4.
[2] M. J. M. Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers, "Matching properties of MOS transistors," *IEEE J. Solid-State Circuits*, vol. 24, no. 5, pp. 1433–1440, Oct. 1989.
[3] A. J. Bhavnagarwala, X. Tang, and J. D. Meindl, "The impact of intrinsic device fluctuations on CMOS SRAM cell stability," *IEEE J. Solid-State Circuits*, vol. 36, no. 4, pp. 658–665, Apr. 2001.
[4] S. Mukhopadhyay, H. Mahmoodi, and K. Roy, "Statistical design and optimization of SRAM cell for yield enhancement," in *Proc. IEEE/ACM Int. Conf. CAD*, 2004, pp. 10–13.
[5] B. H. Calhoun and A. Chandrakasan, "Analyzing static noise margin for sub-threshold SRAM in 65 nm CMOS," in *Proc. Eur. Solid State Circuits Conf.*, 2005, pp. 363–366.
[6] H. Mahmoodi, S. Mukhopadhyay, and K. Roy, "Estimation of delay variations due to random-dopant fluctuations in nanoscale CMOS circuits," *IEEE J. Solid-State Circuits*, vol. 40, no. 9, pp. 1787–1796, Sep. 2005.
[7] X. Du and W. Chen, "A most probable point based method for efficient uncertainty analysis," *J. Design Manuf. Autom.*, vol. 4, no. 1, pp. 47–66, Oct. 2001.
[8] D. Hocevar, M. Lightner, and T. Trick, "A study of variance reduction techniques for estimating circuit yields," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. CAD-2, no. 3, pp. 180–192, Jul. 1983.
[9] R. Kanj, R. Joshi, and S. Nassif, "Mixture importance sampling and its application to the analysis of SRAM designs in the presence of rare event failures," in *Proc. IEEE/ACM Des. Autom. Conf.*, 2006, pp. 69–72.
[10] T. C. Hesterberg, "Advances in importance sampling," Ph.D. dissertation, Dept. Statist., Stanford Univ., Stanford, CA, 1988.

[11] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer-Verlag, 2001.
[12] S. I. Resnick, *Extreme Values, Regular Variation and Point Processes*. New York: Springer-Verlag, 1987.
[13] 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. Des. Autom. Test Eur.*, 2007, pp. 1–6.
[14] *Embedded Memories in Nanoscale VLSIs*, K. Zhang, Ed. New York: Springer-Verlag, 2009.
[15] H. Liu, A. Singhee, R. A. Rutenbar, and L. R. Carley, "Remembrance of circuits past: Macromodeling by data mining in large analog design spaces," in *Proc. Des. Autom. Conf.*, 2002, pp. 437–442.
[16] L. de Haan, "Fighting the arch-enemy with mathematics," *Stat. Neerl.*, vol. 44, no. 2, pp. 45–68, Jun. 1990.
[17] P. Embrechts, C. Klüppelberg, and T. Mikosch, *Modelling Extremal Events for Insurance and Finance*, 4th ed. Berlin, Germany: Springer-Verlag, 2003.
[18] K. Agarwal, F. Liu, C. McDowell, S. Nassif, K. Nowka, M. Palmer, D. Acharyya, and J. Plusquellic, "A test structure for characterizing local device mismatches," in *Symp. VLSI Circuits Dig. Tech. Papers*, 2006, pp. 67–68.
[19] B. Joshi, R. K. Anand, C. Berg, J. Cruz-Rios, A. Krishnamurthi, N. Nettleton, S. Nguyen, J. Reaves, J. Reed, A. Rogers, S. Rusu, C. Tucker, C. Wang, M. Wong, D. Yee, and J.-H. Chang, "A BiCMOS 50 MHz cache controller for a superscalar microprocessor," in *Int. Solid State Circuits Conf.*, 1992, pp. 110–111.
[20] A. A. Balkema and L. de Haan, "Residual life time at great age," *Ann. Probab.*, vol. 2, no. 5, pp. 792–804, 1974.
[21] J. Pickands, III, "Statistical inference using extreme order statistics," *Ann. Stat.*, vol. 3, no. 1, pp. 119–131, 1975.
[22] R. A. Fisher and L. H. C. Tippett, "Limiting forms of the frequency distribution of the largest or smallest member of a sample," *Math. Proc. Camb. Philos. Soc.*, vol. 24, no. 2, pp. 180–190, Apr. 1928.
[23] J. R. M. Hosking and J. R. Wallis, "Parameter and quantile estimation for the generalized Pareto distribution," *Technometrics*, vol. 29, no. 3, pp. 339–349, Aug. 1987.
[24] J. R. M. Hosking, "L-moments: Analysis and estimation of distributions using linear combinations of order statistics," *J. R. Stat. Soc. Ser. B (Methodological)*, vol. 52, pp. 105–124, 1990.
[25] J. M. Landwehr and N. C. Matalas, "Estimation of parameters and quantiles of Wakeby distributions 1," *Water Resour. Res.*, vol. 15, no. 6, pp. 1361–1372, Dec. 1979.
[26] J. M. Landwehr and N. C. Matalas, "Estimation of parameters and quantiles of Wakeby distributions 2," *Water Resour. Res.*, vol. 15, no. 6, pp. 1373–1379, Dec. 1979.
[27] J. R. M. Hosking, "The theory of probability weighted moments," IBM Res. Division, Yorktown Heights, NY, Research Report RC12210, 1986.
[28] X. Li, J. Le, P. Gopalakrishnan, and L. T. Pileggi, "Asymptotic probability extraction for non-normal distributions of circuit performance," in *IEEE Int. Conf. Comput. Aided Des.*, 2004, pp. 2–9.
[29] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining Knowl. Discovery*, vol. 2, no. 2, pp. 121–167, Jun. 1998.
[30] T. Joachims, "Making large-scale SVM learning practical," in *Advances in Kernel Methods—Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds. Cambridge, MA: MIT Press, 1999.
[31] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Francisco, CA: Morgan Kaufmann, 2005.
[32] K. Morik, P. Brockhausen, and T. Joachims, "Combining statistical learning with a knowledge-based approach—A case study in intensive care monitoring," in *Proc. 16th Int. Conf. Mach. Learn.*, 1999, pp. 268–277.
[33] R. Rao, A. Srivastava, D. Blaauw, and D. Sylvester, "Statistical analysis of subthreshold leakage current for VLSI circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 2, pp. 131–139, Feb. 2004.
[34] W. Liu, X. Jin, J. Chen, M.-C. Jeng, Z. Liu, Y. Cheng, K. Chen, M. Chan, K. Hui, J. Huang, R. Tu, P. Ko, and C. Hu, "Bsim 3v3.2 MOSFET model users' manual," Univ. California, Berkeley, CA, Tech. Rep. No. UCB/ERL M98/51, 1988.
[35] W. H. Press, B. P. Flannery, A. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 1992.
[36] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45 nm early design exploration," *IEEE Trans. Electron Devices*, vol. 53, no. 11, pp. 2816–2823, Nov. 2006.

Amith Singhee (S'06–M'09) received the B.Tech. (with honors) in electrical engineering from the Indian Institute of Technology (IIT), Kharagpur, India, in 2000 and the M.S. and Ph.D. degrees in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 2002 and 2007, respectively.

He is a Research Staff Member with IBM T. J. Watson Research Center, Yorktown Heights, NY, with current interests in statistical simulation, process variation modeling, and general design for manufacturability. He was with Neolinear and, subsequently, Cadence Design Systems, from 2002 to 2004.

Dr. Singhee is a recipient of several awards, including the European Design Automation Association Outstanding Dissertation Award; the A. G. Milnes Award for his Ph.D. dissertation; the Silver Medal at IIT; the best paper award at the Design Automation Conference in 2002 and the Design, Automation and Test in Europe (DATE) in 2007; and the best student paper award at the International Conference on Very Large Scale Integration Design in 2008. His paper was published in the book "The Most Influential Papers of 10 Years DATE," and he received the Inventor Recognition Award from the Global Research Consortium in 2008.

Rob A. Rutenbar (S'77–M'84–SM'90–F'98) received the Ph.D. degree from the University of Michigan, Ann Arbor, in 1984.

He is with Carnegie Mellon University, Pittsburgh, PA, where he currently holds the Stephen Jatras (E'47) Chair in the Department of Electrical and Computer Engineering. He has worked on tools for custom circuit synthesis and optimization for over 20 years. In 1998, while he was on leave from Carnegie Mellon University, he cofounded Neolinear Inc. to commercialize the first practical synthesis tools for analog designs. He served as Neolinear's Chief Scientist until its acquisition by Cadence in 2004. He is the Founding Director of the U.S. National Focus Center Research Program Focus Research Center for Circuit and System Solutions, which is a consortium of roughly 20 U.S. universities and over 50 faculty funded by the U.S. semiconductor industry and U.S. government to address future circuit challenges. He has published over 150 papers in his career, and his work has been featured in venues ranging from EETimes to the Economist magazine.

Dr. Rutenbar has won many awards over his career. He received a Presidential Young Investigator Award from the National Science Foundation in 1987. He has won several Best Paper Awards, e.g., Design Automation Conference (DAC)'87, DAC'02, Design, Automation and Test in Europe 2007, and Very Large Scale Integration 2008. He was the 2001 winner of the Semiconductor Research Corporation Aristotle Award for excellence in education and the 2007 winner of the IEEE Circuits and Systems Industrial Pioneer Award for his work in making analog synthesis as a commercial technology. He received the 2002 University of Michigan Alumni Merit Award for Electrical Engineering. He is a member of the Association for Computing Machinery and Eta Kappa Nu.