An Efficient, Fully Nonlinear, Variability-Aware non-Monte-Carlo Yield Estimation Procedure with Applications to SRAM cells and Ring Oscillators

Chenjie Gu
ECE Department
University of Minnesota
Minneapolis, MN, 55455
e-mail: gcj@umn.edu

Jaijeet Roychowdhury
ECE Department
University of Minnesota
Minneapolis, MN, 55455
e-mail: jr@umn.edu

Abstract— Failures and yield problems due to parameter variations have become a significant issue for sub-90-nm technologies. As a result, CAD algorithms and tools that provide designers the ability to estimate the effects of variability quickly and accurately are being urgently sought. The need for such tools is particularly acute for static RAM (SRAM) cells and integrated oscillators, for such circuits require expensive and high-accuracy simulation during design. We present a novel technique for fast computation of parametric yield. The technique is based on efficient, adaptive geometric calculation of probabilistic hypervolumes subtended by the boundary separating pass/fail regions in parameter space. A key feature of the method is that it is far more efficient than Monte-Carlo, while at the same time achieving better accuracy in typical applications. The method works equally well with parameters specified as corners, or with full statistical distributions; importantly, it scales well when many parameters are varied. We apply the method to an SRAM cell and a ring oscillator and provide extensive comparisons against full Monte-Carlo, demonstrating speedups of 100-1000×.

I. Introduction

Parameter variations are inevitable in any IC process. Process steps such as oxidation, doping, molecular beam epitaxy, *etc.*, are all fundamentally statistical in nature. Design of functioning circuits and systems has traditionally relied heavily on the presumption that the law of large numbers applies and that statistical averaging predominates over random variations – more precisely, that the statistical distributions of important process, geometrical, environmental and electrical parameters cluster closely about their means. Unfortunately, with feature sizes having shrunk from 90*nm* to 65*nm* recently (with further scaling down to 45*nm* and 32*nm* predicted by the SIA roadmap [1]), this assumption is no longer valid – in spite of efforts to control them, large variations in process parameters are the norm today. With transistors having become extremely small (*e.g.*: gates are only 10 molecules thick; minority dopants in the channel number in the 10s of atoms), small absolute variations in previous processes have become large relative ones. Lithography-related variability at nanoscales [2], which affect geometrical parameters such as effective length and width, further compound parameter variation problems.

Due to such variations in physical, process or environmental parameters, electrical characteristics of circuit devices (*e.g.*, MOS threshold voltages) change; this leads to undesired changes in circuit performances. For example, in static RAM (SRAM) circuits, like the 6T cell in Fig. 5, the threshold voltages of M2 and M1 determine how well they turn on and hence determine the charge/discharge rate in read/write operations. This rate is closely related to SRAM read/write access time, stability and read/write frequency. As another example: in ring oscillators, threshold voltage and load capacitance variations affect inverter delays significantly, resulting in changes in oscillation frequency. It is crucial to predict what fraction of circuits is likely to fail because of variability, so that measures may be taken to increase the yield of correctly functioning circuits, to the extent possible via design and process changes. Reasonably accurate prediction of yield (*e.g.*, within a percent or better) is important: over-estimation leads to fewer working components than expected, while under-estimation can lead to lowered performance as a result of design efforts to increase yield.

The simplest way to estimate yield is to apply the well known Monte-Carlo method [3]. Monte-Carlo samples parameters according to their probability distributions and simulates the circuit for each parameter set to find the circuit performance of interest. The number of samples/simulations which result in acceptable performance is counted; dividing this number by the total number of samples is the yield. For example, Fig. 10(b) shows results from Monte-Carlo simulation on a three-stage ring oscillator. The green points represent those MOS threshold voltages that yield an acceptable oscillator frequency variation (within 2.5% of nominal). While Monte-Carlo has the advantages of simplicity and extremely general applicability, its main limitation is that it can require very large numbers of expensive simulations for accurate yield estimation.

The computation required for Monte-Carlo type yield estimation methods is significantly exacerbated if each individual simulation is expensive. For example, transient analyses with many timepoints, or RF analyses such as harmonic balance [4], are significantly more expensive than DC and AC analyses. Whether such analyses are required depends on the circuit and performance of interest. While, for example, op-amp performances can usually be determined via DC/AC analyses, transient/RF analyses are essential for circuits such as SRAM cells and oscillators. Transient simulations are required for finding SRAM read/write access times, while harmonic balance, shooting, or extremely expensive transient simulations [5]–[7] are typically needed for finding oscillator frequencies. As a result, alternatives to Monte-Carlo for calculating parametric yield are particularly important for such circuits.

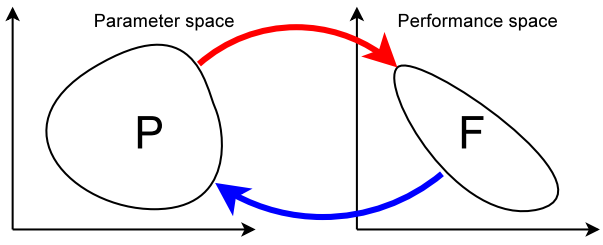


Fig. 1. Boundaries in parameter space and performance space.

In order to overcome the disadvantages of brute-force Monte-Carlo, various techniques have been proposed to im-

978-1-4244-1922-7/08/$25.00 ©2008 IEEE

prove its efficiency. Techniques like importance sampling, stratified sampling (or its extension Latin hypercube sampling [8]) and control variates [9], [10] try to reduce the variance of sample points, so that fewer samples are required to achieve the same accuracy. However, such techniques for smarter sampling have their limitations; *e.g.*, importance sampling is most useful only when few parameters are considered and the yield is very high or low [9], while stratified sampling often does not achieve very significant reduction in the number of samples.

Modifications to Monte-Carlo that use boundary approximations for speeding up yield calculation are also available. In [11], [12], the boundary surface in parameter space separating success/failure performance regions is approximated by a series of linear constraints. The utility of the approximate linear constraints is that testing whether a point is in the success region is sped up significantly for most (but not all) Monte-Carlo samples. For those that fail this fast check, a more computationally expensive check, involving a linear search between the approximate linear constraints and the real boundary is performed. However, this method is based on the assumption that the boundary is convex, which is not always true in reality (indeed, the boundary for SRAM read access failure, in Fig. 6(b), is not convex). Furthermore, since approximations are generated using randomly selected points on the boundary, it is difficult to guarantee that the linear constraints will approximate the success region well – *i.e.*, the shape of the approximated boundary can be quite different from that of the real boundary.

The method in [11], [12], like the present work, is concerned with boundaries in the parameter space. Given a region of desired (*e.g.*, worst-case) performances of the circuit, the corresponding boundary in the parameter space is determined, as depicted by the blue (lower) arrow in Fig. 1. Techniques that solve the reverse problem, as depicted by the red (upper) arrow in Fig. 1, are important in performance optimization and tradeoff analysis. In other words, given a region in the parameter space (such as a hypercube), the problem solved is to find the corresponding boundary in the performance space. While this is a different class of problem from the one being considered here, we note that [13] uses a boundary approximation technique in the performance space that is similar to a component of our method (Section III-D, Fig. 3) for boundaries in the parameter space. [13] uses a search scheme (termed *Normal Boundary Intersection* (NBI)) to find equally spaced points on the boundary in the performance space. However, there are key differences between our method and NBI, as detailed in Section III-D. Our method, unlike NBI, adapts the points it chooses to the shape of the boundary. Further, since we are interested in boundaries in parameter space, our method uses a line-based (or least-squares-based) Newton algorithm for finding boundary points, while NBI uses a sequential quadratic programming algorithm to solve a min/max problem in performance space.

In this paper, we propose a new method, termed **Y**ield **E**stimation via **N**onlinear **S**urface **S**ampling (**YENSS**), for finding parametric yield given tolerance limits for success/failure with respect to any circuit performance of interest. The method does not rely on Monte-Carlo sampling; as such, it is particularly well suited for circuits that require expensive computations (such as transient and harmonic balance) for finding performances. **YENSS** works by first locating the boundary separating regions in parameter space that correspond to success/failure of the circuit. If p parameters are considered, this boundary is a $(p-1)$-dimensional hypersurface or manifold. Once this boundary is found, **YENSS** rapidly and incrementally calculates the probabilistic volume on one side of the boundary (*i.e.*, the yield) in a geometrical manner, by employing analytical formulae for volumes of "tetrahedra" (or simplices) in p dimensions (rather than Monte-Carlo style integration, as in prior work such as [14]). This key difference provides not only computational advantages over Monte-Carlo integration, but also enables a better error control scheme, as discussed in Section III-F.

By focusing only on the boundary, **YENSS** avoids the expense Monte-Carlo incurs of sampling the entire parameter space. However, if there are many parameters (*i.e.*, p is large), fully characterizing a $(p-1)$ boundary manifold is also very computationally expensive. **YENSS** uses an adaptive technique to calculate only as many points on the boundary as needed to provide an estimate of yield to a desired accuracy. By doing so, **YENSS** avoids redundant/unnecessary simulations in the yield estimation procedure. This is especially useful when many parameters are considered simultaneously: the accuracy is controlled by the "yield volume" increment at each iteration, compared to $\frac{1}{\sqrt{N}}$ accuracy improvement in Monte-Carlo simulation.

A crucial feature of **YENSS**, that is central to its efficiency for large p, is its automatic exploitation of *linearity* of the boundary. If the boundary is perfectly linear (*e.g.*, a plane embedded in 3-dimensional parameter space), the computational expense of **YENSS** increases only linearly with p. Because it adaptively estimates the probabilistic volume, the computation needed by **YENSS** increases gracefully to handle boundaries that are not linear. Since failure/success boundaries in parameter space tend to be close to linear for many practical problems, **YENSS** can achieve great speedups over Monte-Carlo.

For the underlying computations involved in finding points on the boundary, **YENSS** uses full SPICE-level transient or harmonic balance (HB) simulations, with no compromises in accuracy – a feature particularly important for reliable yield numbers in SRAM cells and oscillators. Another strength of **YENSS** is that it is equally easily able to handle parameters specified using corners or with probability density functions (PDFs). We present two different versions of the algorithm used by **YENSS** to locate boundary points efficiently, based on Newton-Raphson and line (bisection) search, respectively – the latter can be useful in situations where parameter derivative/sensitivity information, required by Newton-Raphson, is difficult to obtain.

We demonstrate and validate **YENSS** on two circuits – a 6T SRAM cell and a three stage inverter-based ring oscillator – using tolerances on SRAM read access time and oscillator center frequency, respectively, as performance criteria. We use the threshold voltages of the MOS devices as the varying parameters of interest. We provide detailed comparisons against Monte-Carlo of both accuracy and performance, obtaining speedups of 100-1000$\times$ while at the same time calculating yield with higher accuracies.

The remainder of the paper is organized as follows. We formulate the yield estimation problem in Section II and provide an overview of the flow of our method. In Section III, we show how to find points on the boundary surface between success/failure regions efficiently. In Section III-E.1 and Section III-E.2, we derive and explain our procedure for obtaining sensitivity information from transient and harmonic balance simulation. In Section III-F, we describe our procedure for calculating the "yield volume" and our adaptive error control scheme. Finally, in Section IV, we validate **YENSS** on an SRAM cell and a ring oscillator.

II. Overview of our method

A. Yield via hypervolume computation

For purposes of illustration, we first consider two uncorrelated parameters p_1 and p_2, with nominal values p_{1nom}, p_{2nom}, uniformly distributed over $[p_{1min}, p_{1max}]$ and $[p_{2min}, p_{2max}]$ respectively, as shown in Fig. 2. We assume that the circuit performs correctly at its nominal point, *i.e.*, the circuit performance f is f_{nom} when parameters are (p_{1nom}, p_{2nom}). As the parameters move away from the design point, the circuit's performance also changes away from its nominal value. Therefore, in the parameter space, there exists a region around the nominal design point where the performance remains acceptable – this region is depicted by the interior of the closed curve in Fig. 2, with the portion within the min/max bounds shown shaded (green). Finding the ratio of this area to that of the box $[p_{1min}, p_{1max}] \times [p_{2min}, p_{2max}]$ provides the yield.

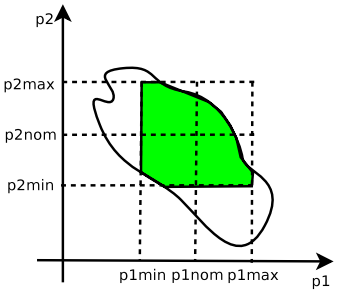


Fig. 2. Evaluating yield is equivalent to evaluating the area of the shaded (green) part in the figure.

This notion is easily generalized to the case of multiple parameters. For the case of 3 parameters, the volume of the 3-dimensional region that corresponds to acceptable performance in parameter space needs to be calculated. For N parameters, the hypervolume of an N-dimensional region in parameter space is calculated.

For non-uniform parameter distributions (*e.g.*, Gaussian and truncated Gaussian distributions are popular [10], [15]), the parameter axes are pre-scaled by the cumulative density function, such that the scaled parameters are uniformly distributed (*i.e.*, the axes p_1 and p_2 in Fig. 2 are replaced by $\hat{p}_1 \equiv C_1(p_1)$ and $\hat{p}_2 \equiv C_2(p_2)$, respectively, where C_1 and C_2 are the cumulative distribution functions of the two parameters). The boundary in the space of original parameters p effectively becomes mapped to a different boundary in the "probabilistically deformed parameters" $\hat{p}$. **YENSS** then operates in the $\hat{p}$ space to find the yield.

For the case of correlated parameters, a principal component analysis [16] (PCA) is applied first. PCA de-correlates correlated parameters and, potentially, can identify a reduced number of important parameters. Once PCA is performed, **YENSS** can be applied on the de-correlated parameters.

Hence, non-uniformly-distributed and/or correlated parameters can always be mapped into uniformly-distributed, uncorrelated parameters. The hypervolume inside the boundary in the new parameter space represents the parametric yield. In this paper, we focus on the core problem, *i.e.*, that involving only uniformly-distributed, uncorrelated parameters.

B. Basic flow of our method

In order to calculate the yield volume of the multi-dimensional region defining the success region, we find several points on the hypersurface which separates the success and failure region – we refer to this as the "boundary surface" in the following. We use these points to estimate the hypervolume of the region enclosed by the boundary surface via a hyper-polygonal approximation which we refine iteratively.

We first outline the overall algorithm flow of **YENSS**, and explain / develop each step in detail in the following subsections.

The flow of **YENSS** is:

1) Formulate the boundary surface as a nonlinear scalar equation (or constraint) in the parameter space:
$$h(\vec{p}; f_m) = 0 \tag{1}$$
(1) expresses the relationship between m parameters $\vec{p}$ and the circuit performance of interest f_m in a general, implicit form. As we show later, this form is very general, capturing transient simulations, harmonic balance, *etc.*.
2) Find the intersection of the boundary surface with each parameter axis.
   a) by augmenting equation (1) with parametric equations (6) of each parameter axis.
   b) then solving the system of equations (1) and (6); the solution is the intersection of the parameter axis and the boundary surface.
3) Use an analytical volume calculation formula to calculate the initial approximation to the yield – the hypervolume of the simplex defined by those intersections.
4) Volume refinement using additional boundary points.
   a) Choosing and computing additional points on the boundary surface.
      i) Construct the manifold by connecting the points already found in previous refinements, and choose the centroid of this manifold to be the initial guess P_{guess} (see Fig. 3 for example) for the point to be found on the boundary.
      ii) Either use MPNR method to solve the intersection point;
      iii) Or use line search method to solve the intersection point.
         A) Construct the line which goes through point P_{guess} and is perpendicular to the manifold constructed in 4(a)i.
         B) Apply line search method to find the intersection of the boundary and this line, using Newton-Raphson method, or secant/bisection method.
         C) If Newton-Raphson method is used, the derivatives of function $h(\cdot)$ *wrt.* parameters $\vec{p}$ need to be evaluated, which are developed in Section E.1 and Section E.2 for transient simulation and harmonic balance simulation, respectively.
         D) If calculating the derivatives are computationally expensive or practically difficult, bisection or secant mehod is applied to find the intersection point.
   b) Update the yield estimation using newly found boundary points.
      i) The yield volume increment is evaluated by calculating the simplices defined by the newly found point and previous boundary points, using analytical formula , as shown in Section F.
      ii) Based on the volume increment, the error is estimated for the current yield estimation.
      iii) If the error estimation is small enough, finish the algorithm; else, repeat the volume refinement procedure until small error is obtained.

Details of the above steps are presented in the following sections.

III. Details of the algorithm

In this section, we provide details of each of the steps of the algorithm outlined in Section II.

A. Implicit formulation for the boundary surface (step 1)

We start by expressing the relationship between the performance metric and the varying parameters as an implicit *scalar equation* (1) which encapsulates all the complex nonlinear dynamics in the circuit. Suppose $\vec{p} \in \mathbb{R}^m$, then this scalar equation defines a $(m-1)$-dimensional hypersurface in the parameter space. Our goal is to find points on the surface efficiently and accurately, as described later.

Re-write equation (1) in (2), we can see that the boundary equation defines the surface on which the circuit performance is the worst-case performance, *i.e.*, the surface separates the failure/success region.

$$h(\vec{p}; f_m) \equiv f_m(\vec{p}) - f_{worst} = 0 \tag{2}$$

Depending on the problem or circuit being considered, equation (1) will encapsulate different kinds of analyses. We next provide two examples of such systems, for transient simulation (applied to SRAM cells) and oscillator harmonic balance (applied to ring oscillators).

1) SRAM read access (transient simulation up to a fixed time): An example of formulating the boundary equation based on transient analysis is the SRAM read access time constraint (shown in equation (3)). Given a read access time t_f, the voltage difference between node BL and BLB ΔBL at time $t = t_f$ can be calculated through performing a transient analysis on the SRAM circuit. If this voltage difference is smaller than the minimum voltage difference ΔBL_{min} which can be sensed by the sense amplifier, the read failure will occur. Therefore, expressing the above description in the form of (2), we obtain the equation defining the boundary surface (3)

$$h(\vec{p}; f_m) \equiv h(\vec{p}; \Delta BL) \equiv \Delta BL(\vec{p})_{t=t_f} - \Delta BL_{min} = 0 \tag{3}$$

2) Oscillator center frequencies (autonomous harmonic balance): An example of formulating the boundary equations based on harmonic balance simulation is the oscillator frequency constraint. As is shown in equation (4), f_{nom} is the nominal frequency of the oscillator at the design point, and Δf is the maximum allowed frequency deviation. The free-running frequency of the oscillator $f_o(\vec{p})$ is calculated via running a harmonic balance analysis, on the circuit with parameters $\vec{p}$. Whether this frequency is out of or in the acceptable frequency region $[f_{nom} - \Delta f, f_{nom} + \Delta f]$ determines whether the oscillator circuit fails or not. So equation (4) defines the surface on which the points correspond to the circuit whose output frequency is the worst case frequency $f_{nom} \pm \Delta f_m$, *i.e.*, the boundary surface.

$$h(\vec{p}; f_m) \equiv h(\vec{p}; f_o) \equiv f_o(\vec{p}) - (f_{nom} \pm \Delta f) = 0 \tag{4}$$

B. Find the intersections of the boundary surface with each parameter axis (step 2)

After the scalar equation for the boundary curve is formulated, the first step is to find the intersections of the boundary surface with each parameter axis. We assume that the boundary surface will intersect the axes; this is a reasonable assumption for realistic problems, motivated by the fact that if only one parameter is allowed to vary to change the circuit performance, there will always be a parameter value which will make the circuit fail.

To solve the intersections of the boundary surface (defined by equation (1)) with a specific parameter axis (say p_i), we fix the other parameters at their nominal parameter values $p_j = p_{jnom}, j \neq i$. So the only unknown in the equation is now a scalar p_i, rather than a vector $\vec{p}$, and we can find the intersections using Newton-Raphson method. To use Newton-Raphson method, $\frac{dh}{dp}$ need to be calculated – how to calculate this quantity will be discussed in Section E.1 and Section E.2.

Newton-Raphson has faster convergence than other methods, but it requires the Jacobian matrix of the equations. Since this step of our method requires solving for only the *scalar* unknown p_i, it is also possible to use robust methods that do not require Jacobian calculation (which can, in many situations, be inconvenient to implement, since parameter sensitivity code is required), such as bisection or secant methods [17]. This option can be very valuable for quick practical implementation; however, if parameter sensitivities and the Jacobian matrix are available, we prefer Newton-Raphson, on account of its quicker convergence and higher accuracy.

C. Initial estimation to the yield volume (step 3)

Based on the intersections of the boundary surface with each parameter axis, the initial approximation of the "yield volume" can be calculated using an analytical formula, as shown in equation (5), which shows that the hypervolume of a parallelotope and a simplex defined by $N+1$ points $x_i, i = (1 \ldots N+1)$ in N-D space can be evaluated by calculating a determinant composed of coordinates of all the points.

$$C_{parallelotope} = \begin{vmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,N} & 1 \\ x_{2,1} & x_{2,2} & \cdots & x_{2,N} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{N+1,1} & x_{N+1,2} & \cdots & x_{N+1,N} & 1 \end{vmatrix}$$
$$C_{simplex} = \frac{1}{N!} C_{parallelotope} \tag{5}$$

D. Centroid computation and line intersection with surface (step 4a)

The procedure in B to find the intersection points can also be formulated as a line search method. Actually, it is a special case of finding the intersections of the boundary surface with any straight line, which will be needed in the refinement step, as shown in the algorithm flow.

Since the scalar equation $h(\vec{p}, f_m) = 0$ has m unknowns, to force the solution to be the intersection of the boundary surface and a straight line, we must augment this equation with another $m-1$ linear equations that define the straight line in m-D space. Because a line in m-D space can always be expressed in the form of parametric equations, equivalent to $m-1$ linear constraints, we augment the scalar equation with m parametric equations (6) of the line, with one more unknown s introduced. In equation (6), $\vec{p}_0$ is the coordinate of a point the constraint line (parameter axis in this case) goes through, $\vec{u}$ is the unit vector denoting the direction of the line.

$$\vec{p} = \vec{p}_0 + s\vec{u} \tag{6}$$

For example, the parametric equation of p_1 axis can be written as:

$$\begin{aligned} p_1 &= s \\ p_i &= 0, (2 \leq i \leq m) \end{aligned} \tag{7}$$

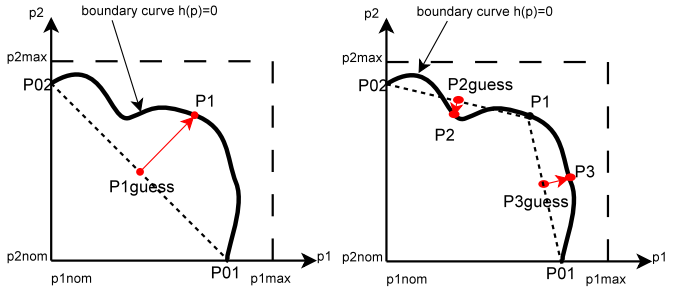
So the augmented equations are no longer under-determined, and the solution will be constrained on this straight line. Therefore, using Newton-Raphson method we can search the line, and find the intersection. Or we can apply bisection or secant method to do the line search if Newton-Raphson has some convergence problem.

After the intersections with each parameter axis are found, we start our procedure iteratively to find additional points on the boundary and to update the yield volume. Based on the m intersection points, we use an $(m-1)$-D manifold connecting those points to approximate the boundary.

The key problem here is to ensure the points to be found are a good representation of the boundary surface, *e.g.*. the points on the boundary are equally spaced. To align all the points evenly on the boundary surface, we again apply the line constraint on the solution using equation (6) such that the line goes through the centroid of the points got from previous iterations, and is perpendicular to the manifold we have already constituted. Then line search with Newton-Raphson method can be applied to solve the augmented system.

Another option is to use Moore-Penrose pseudo-inverse Newton Raphson (MPNR) method [18] to solve the scalar equation (2) directly, without augmenting additional linear equations.

For illustration, two iterations of our procedure in the first quadrant on 2-D parameter plane is shown in Fig. 3(a) and Fig. 3(b), where the bold curve represents the boundary curve defined by equation (2). In Fig. 3(a), P_{01}, P_{02} are two intersections with parameter axes, calculated in the first step. Then P_{1guess} is chosen to be the centroid of $P_{01}P_{02}$, and a line (the red arrow) perpendicular to $P_{01}P_{02}$ is applied to constrain the solution to be the intersection of the line and the boundary curve. Using P_{1guess} to be the initial guess for Newton-Raphson solver, it finally converges to P_1, as denoted by the red arrow. In the second iteration, (shown in Fig. 3(b)) P_{2guess} and P_{3guess} are selected to be the centroid of $P_{02}P_1$ and $P_{01}P_1$, and Newton-Raphson will follow the red arrow, converging to P_2 and P_3 respectively. Repeating this procedure, we will finally get enough points on the boundary curve needed for yield estimation. Using similar procedure, this curve-finding scheme can be generally applied to higher-dimensional problems.



(a) first iteration (b) second iteration

Fig. 3. Finding points on the boundary curve using line search (2 iterations).

E. Sensitivity evaluation based on implicit boundary equation

In order to obtain the quadratic convergence of Newton-Raphson method, the Jacobian matrix of the system must be calculated. As shown in (2), the derivative of function $h(\cdot)$ *wrt.* $\vec{p}$ is just the sensitivity of the circuit performance *wrt.* $\vec{p}$. So no matter how $h(\cdot)$ is defined, we can always do differentiation of $h(\cdot)$ *wrt.* $\vec{p}$, as shown in (8), in which the unknown is $\frac{\partial f_m}{\partial \vec{p}}$.

$$\frac{dh}{d\vec{p}} = \frac{\partial h}{\partial \vec{p}} + \frac{\partial h}{\partial f_m}\frac{\partial f_m}{\partial \vec{p}} = 0 \tag{8}$$

Using this notion, Section E.1 and Section E.2 gives the derivations of the sensitivities of $h(\vec{p})$ *wrt.* $\vec{p}$, based on transient simulation and harmonic balance simulation, respectively.

1) Transient simulation based sensitivity evaluation: In equation (3), the evaluation of the equation is through doing a transient analysis on the circuit, *i.e.*, doing a numerical integration on the differential algebraic equations of the circuit (9)

$$\frac{d}{dt}\vec{q}(\vec{x}) + \vec{f}(\vec{x}) + \vec{b}(t) = 0 \tag{9}$$

where $\vec{x}$ is the unknowns (node voltages and branch currents).

For convenience, we re-write (3) in a clearer way in equation (10)

$$h(\vec{p}) \equiv \Delta BL - \Delta BL_{min} = \vec{c}^T\vec{x}(t_f;\vec{p}) - \Delta BL_{min} = 0 \tag{10}$$

where $\vec{c}^T$ is a vector which picks up the voltage difference ΔBL from nodal voltages $\vec{x}$.

In order to apply Newton-Raphson method to solve the equation, the derivatives of $f_m(\vec{p})$ *wrt.* $\vec{p}$ must be calculated. That is

$$\frac{\partial f_m(\vec{p})}{\partial \vec{p}} = \vec{c}^T\frac{\partial \vec{x}}{\partial \vec{p}} \tag{11}$$

So in order to calculate $\frac{\partial \vec{x}}{\partial \vec{p}}$, we do differentiation to equation (9) *wrt.* parameters $\vec{p}$. We get

$$\frac{d}{dt}\left[\frac{\partial \vec{q}}{\partial \vec{x}}\frac{\partial \vec{x}}{\partial \vec{p}} + \frac{\partial \vec{q}}{\partial \vec{p}}\right] + \frac{\partial \vec{f}}{\partial \vec{x}}\frac{\partial \vec{x}}{\partial \vec{p}} + \frac{\partial \vec{f}}{\partial \vec{p}} + \frac{d\vec{b}}{d\vec{p}} = 0 \tag{12}$$

Equations (12) is a set of differential equations with unknowns $\frac{\partial \vec{x}}{\partial \vec{p}}$, and hence can be solved using any numerical integration method, such as Backward-Euler method or Trapezoid method [19], [20].

Thus, the sensitivities of the nodal voltages and branch currents *wrt.* varying parameters, which are required in Newton-Raphson method, are calculated.

2) Harmonic balance simulation based sensitivity evaluation: As mentioned in Section I, transient analysis on oscillator circuits can be quite inaccurate and inefficient. Instead, harmonic balance simulation is always performed on this kind of circuits in practice [4]. Thus calculating the sensitivities based on harmonic balance method [21] is also desirable when harmonic balance simulation is used to extract the circuit performance, such as output frequency of oscillators.

Equations for harmonic balance method [4], with dependency on parameters explicitly shown, can be written as

$$\vec{H}(\vec{X}, f_o; p) = \Omega D\vec{Q}(\cdot,\vec{p})D^{-1}\vec{X} + D\vec{F}(\cdot,\vec{p})D^{-1}\vec{X} + \vec{B} = 0 \tag{13}$$

where $\vec{X}$ is the Fourier coefficients of the unknowns $\vec{x}$ in circuit differential equations (9), D and D^{-1} are stacked DFT and IDFT matrices, $\vec{Q}(\cdot,\vec{p})$ and $\vec{F}(\cdot,\vec{p})$ are the stacked form of $\vec{q}(\cdot)$ and $\vec{f}(\cdot)$ terms in equation (9).

To solve equation (4) using Newton-Raphson method, we need to know the derivatives of the output frequency *wrt.* varying parameters. To get this sensitivity, we do differentiation *wrt.* parameters $\vec{p}$ to equation (13).

$$\frac{\partial \vec{H}}{\partial \vec{X}}\frac{\partial \vec{X}}{\partial \vec{p}} + \frac{\partial \vec{H}}{\partial f_o}\frac{\partial f_o}{\partial \vec{p}} + \frac{\partial \vec{H}}{\partial \vec{p}} = 0 \tag{14}$$

So the only unknowns in equations (14) are $\frac{\partial \vec{X}}{\partial \vec{p}}$ and $\frac{\partial f_o}{\partial \vec{p}}$, and hence, the sensitivity of the output frequency *wrt.* parameters

can be calculated by solving the equations

$$\left[\frac{\partial \vec{H}}{\partial \vec{X}}, \frac{\partial \vec{H}}{\partial f_o}\right] \begin{bmatrix} \frac{\partial \vec{X}}{\partial \vec{p}} \\ \frac{\partial f_o}{\partial \vec{p}} \end{bmatrix} = -\frac{\partial \vec{H}}{\partial \vec{p}}. \quad (15)$$

F. Yield volume calculation and error control (step 4b)

When the new points on the boundary curve/surface are found, we update the hypervolume estimate by adding/subtracting the hypervolume of the new simplexes we find. As an example, two iterations of our refinement procedure is shown in Fig. 4. S_0 (area in green) is the original area estimate after the intersections have been found. After one iteration, P_1 is found, and S_1 (area of triangle $P_{01}P_{02}P_1$, marked in yellow) is added to S_0 to refine the area estimate. After two iterations, P_2 and P_3 are found, and S_2 (area of triangle $P_{02}P_1P_2$, marked in pink) is subtracted and S_3 (area of triangle $P_{01}P_1P_3$, marked in blue) is added. So, after two level of refinements, the yield volume is $S_0+S_1-S_2+S_3$. It is clear that after several iterations, we can get a fairly well estimate of the total area under the boundary curve.

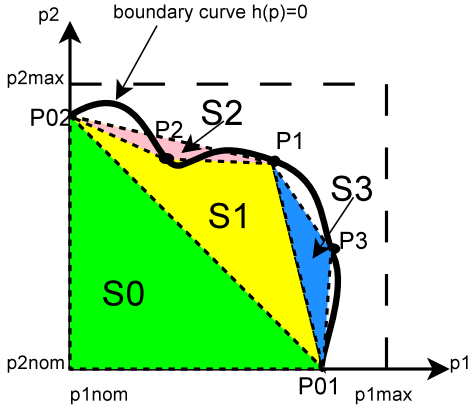


Fig. 4. Illustration of yield volume update scheme.

According to how much area/volume increment we get in a new iteration, we have an estimate of the error of the current estimate. Based on the this volume increment, **YENSS** adaptively decides whether to continue further iterations, depending on the accuracy wanted. For example, in Fig. 4 if $S2$ is small enough (smaller than accuracy wanted), there is no need to find more points on curve $P_{02}-P_1$; but if $S3$ is still large, we can continue our curve-finding scheme to find more points on curve P_1-P_{01}, until the designated accuracy is reached.

This automatic error control scheme helps to avoid redundant simulations that give unnecessary high accuracy. If the curve is a line, the error calculated by this scheme after one refinement is 0, and the algorithm terminates, with the highest accuracy. This explains how this error estimation works and why **YENSS** is extremely efficient if the performance function varies linearly with parameter variations.

IV. Validation

In this section, we apply **YENSS** to a 6T SRAM cell (varying 2 parameters) and a three-stage ring oscillator (varying 3 and 6 parameters) and validate its applicability. We provide comparisons against extensive Monte Carlo simulations to validate accuracies and to gauge computational advantage. We vary the all-important threshold voltages of the MOS transistors in the circuit as parameters; the method applies to any other parameters (*e.g.*gate length, width, load capacitor, *etc.*), of course. All simulations were performed using a MATLAB/C/C++-based circuit/system simulation environment, on an 2.4GHz Athlon XP-based PC running Linux.

A. SRAM yield with respect to access failure

Fig. 5 shows a typical SRAM 6T cell, which is composed of two driver MOSFETs (M1 and M3), two load MOSFETs (M5 and M6) and two access MOSFETs (M2 and M4). As colored by red, M1 and M2 (and equivalently M3 and M4) are critical devices that determines the access time of the SRAM.

While reading an SRAM cell, the access time is defined as the time required to produce a voltage difference between BL and BLB that is large enough for the sense amplifier to sense. (normally defined by $\Delta BL_{min} \simeq 0.1Vdd$) So if the access time is longer than the prescribed maximum allowed value T_{max}, then an access time failure is said to have happened. In other words, if at time $t = T_{max}$, the voltage difference between node BL and BLB is still smaller than ΔBL_{min}, the access failure is happened. Thus we can formulate the success/failure boundary curve for SRAM access failure by equation (16), where $\Phi(\cdot)$ is the state-transition function defining the node voltages and branch currents in the circuit, which is calculated by performing a transient simulation; $(\vec{c}_{BL}^T - \vec{c}_{BLB}^T)$ is the vector which select the voltage difference between node BL and BLB. In (16), we only consider threshold voltage variations of MOSFET M1 and M2, and variations of other MOSFETs can be included in a similar manner.

$$h(V_{th1}, V_{th2}) \equiv (\vec{c}_{BL}^T - \vec{c}_{BLB}^T)\vec{\Phi}(T_{max}; V_{th1}, V_{th2}) - \Delta BL_{min} = 0. \quad (16)$$

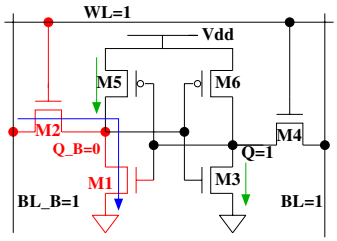
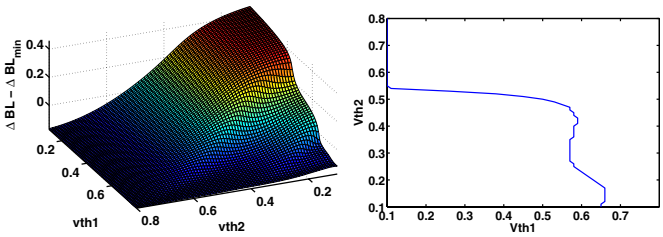


Fig. 5. SRAM 6T cell.

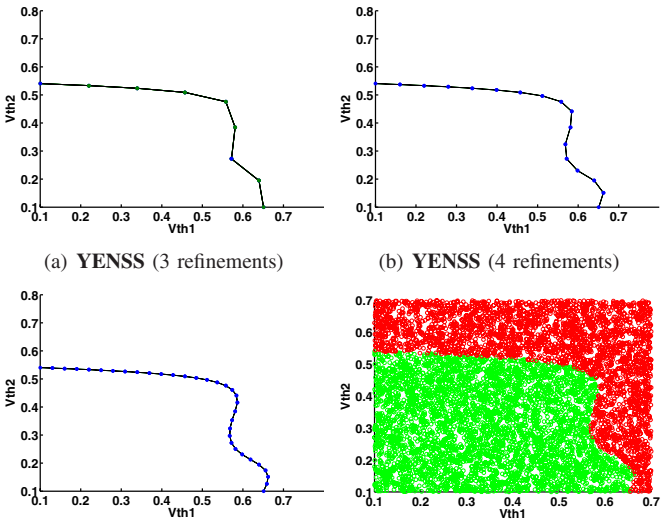
Assuming that $\Delta BL_{min} = 168mV$ at $T_{max} = 3.508ns$, we first vary the threshold voltage of M1 (V_{th1}) and M2 (V_{th2}), and run simulations with different (V_{th1}, V_{th2}) pairs. Fig. 6(a) depicts $h(V_{th1}, V_{th2})$ (*i.e.*, $\Delta BL - \Delta BL_{min}$) surface over $((V_{th1}, V_{th2})$ parameter space. From this surface generated by bruteforce simulations, the boundary curve can be identified, as shown in Fig. 6(b). There are two problems for the bruteforce method: (1) the accuracy is limited by the step size on each parameter axis - each point generated in this method does not lie exactly on the boundary curve, and this is obviously observed in Fig. 6(b); (2) the number of simulations can be too large to be affordable. Although some heuristics can be applied, (such as skip simulate the "safe" area where V_{th1} and V_{th2} are both small), the computational cost is still expensive.

We then apply our method to find the boundary curve directly. Fig. 7 shows the results of **YENSS** using different level of refinements, and Monte Carlo simulation, supposing the min/max bound for threshold voltage is 0.1V-0.7V. In Fig. 7(a), only 3 levels of refinements are performed – 7 points (marked on the curve) on the boundary curve is obtained. So the boundary curve we got is somewhat not smooth compared to the real boundary curve. Fig. 7(b) and Fig. 7(c) shows the results when 4 and 5 levels of refinements are done, and an almost perfect boundary curve is obtained, as compared to that found by Monte Carlo simulation in Fig. 7(d).



(a) (b)

Fig. 6. Brute force simulation to get the boundary curve.



(a) **YENSS** (3 refinements) (b) **YENSS** (4 refinements)

(c) **YENSS** (5 refinements) (d) Monte Carlo

Fig. 7. Simulation results of **YENSS** and Monte Carlo on SRAM access time constraint.

Detailed comparisons between **YENSS** and Monte Carlo method is shown in Table I. To achieve an accuracy of 1%, Monte Carlo method needs over 6000 simulations to be 95% certain about this accuracy. However, only 3 levels of refinements using **YENSS**, though the curve is not smooth in Fig. 7(a), 1% accuracy is obtained, with 255× speedup over Monte Carlo method.

TABLE I
COMPARISON OF **YENSS** VS MONTE CARLO FOR SRAM YIELD

method	accuracy	yield	number of transient simulations	speedup
Monte Carlo	10%	0.6173	80	
Monte Carlo	1%	0.5730	6640	1×
YENSS				
3 levels	1%	0.5756	26	255×
4 levels	0.1%	0.5789	41	162×
5 levels	<0.1%	0.5797	66	101×

[22] provides an alternative method using Euler-Newton curve tracking technique to find this SRAM read access failure curve, and obtains similar speedups to **YENSS**. While efficient in cases when only two parameters are taked into consideration, it has the difficulty to include more parameter variations. However, **YENSS** features the natural extension to the case of multiple parameters, which is shown in the next subsection.

B. Oscillator yield estimation considering output frequency

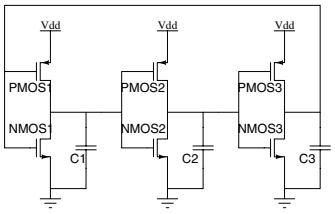


Fig. 8. Three stage ring oscillator.

The circuit diagram of a three-stage ring oscillator is shown in Fig. 8. For the nominal threshold voltage (0.3V for all 6 MOSFETS in Fig. 8), the nominal frequency of the oscillator (calculated via harmonic balance simulation) is 0.9058GHz. In this case, our objective is to determine the yield of the circuit if a $\pm 2.5\%$ frequency variation is considered acceptable, given that the threshold voltage of each MOS device can vary between min/max bounds from 0.1V to 0.5V.

As a first step, we just vary the threshold voltages of the MOS devices of the first stage, V_{tn1} and V_{tp1}. We show the result of bruteforce simulations (via harmonic balance analysis) *wrt.* different (V_{tn1}, V_{tp1}) pairs in Fig. 9 . From Fig. 9, it is noticed that the output frequency varies almost linearly with the threshold voltage variations. This implies that **YENSS** will rapidly converge with very few levels of refinement, as noted before.

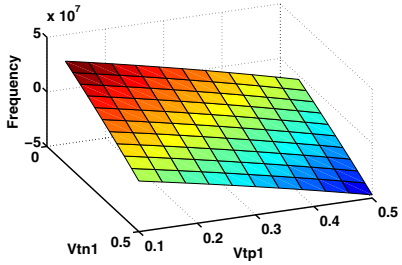
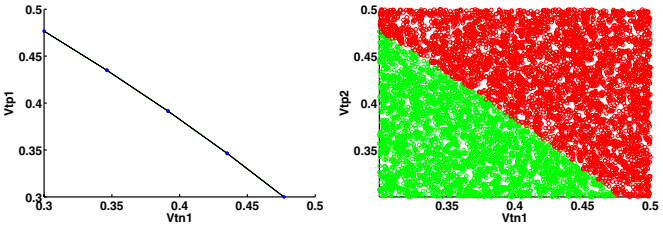


Fig. 9. frequency variation vs (V_{tn1}, V_{tp1}) surface, generated by bruteforce simulations.

The boundary between acceptable performance and failure in the first quadrant of the (V_{tn1}, V_{tp1}) plane is shown in Fig. 10(a). This boundary was obtained by allowing **YENSS** to go through only 2 refinement levels, thereby obtaining 3 points on the curve, while still maintaining a good accuracy for the yield estimation. The ratio of the area to the left of the curve to the area of the min/max parameter bound box is the "partial yield" of the chip, corresponding to the first quadrant.



(a) (b)

Fig. 10. Simulation results of **YENSS** and Monte Carlo on oscillator frequency constraint (2 parameters).

We further validate **YENSS** against Monte Carlo (with accuracy 1%) by color-coding the samples that lead to frequencies remaining in the acceptable range, and being in the failure range, as shown in Fig. 10(b). The interface curve

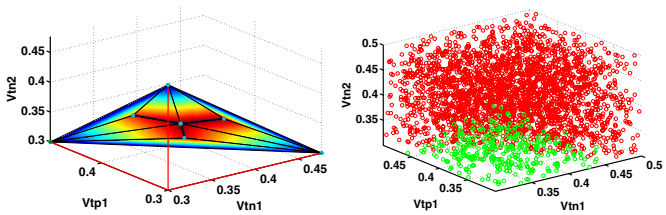
extracted from Fig. 10(b) is identical to the result from **YENSS** in Fig. 10(a).

Table II shows partial yields obtained using **YENSS** for 2 - 5 levels of refinement and compares accuracy and speed against Monte-Carlo. As can be seen, even for 2 levels of refinement (corresponding to 2-3 orders of magnitude speedup over Monte Carlo), **YENSS** is more accurate than Monte Carlo. At 5 levels, **YENSS** converges to accuracies of 0.01% and still is faster by a factor of about 100×, compared to 1% accuracy of Monte Carlo with 95% confidence interval.

TABLE II
COMPARISON OF **YENSS** VS MONTE CARLO FOR OSCILLATOR YIELD

method	accuracy	partial yield	number of HB simulations	speedup
Monte Carlo	10%	0.4198	90	
Monte Carlo	1%	0.4057	6540	1×
YENSS				
2 levels	1%	0.4075	16	409×
3 levels	0.1%	0.4083	24	273×
4 levels	0.02%	0.4085	41	160×
5 levels	<0.02%	0.4086	76	86×

We performed the same simulation considering three threshold voltage parameters. Fig. 11(a) shows the result of Monte-Carlo simulation (1% accuracy with 95% confidence interval). Results from **YENSS** are shown in Table III. Similar speedups and accuracy trends are observed in this case.



(a) (b)

Fig. 11. Simulation results of **YENSS** and Monte Carlo on oscillator frequency constraint (3 parameters).

TABLE III
COMPARISON OF **YENSS** VS MONTE CARLO FOR OSCILLATOR YIELD

method	accuracy	partial yield	number of HB simulations	speedup
Monte Carlo	10%	0.0976	40	
Monte Carlo	1%	0.1190	2841	1×
YENSS				
2 levels	1%	0.1235	24	118×
3 levels	0.2%	0.1252	45	63×
4 levels	0.1%	0.1261	106	27×
5 levels	<0.1%	0.1266	323	9×

Finally, we consider the variability of the threshold voltages of all six MOS transistors in the ring oscillator, and calculate the partial yield, In this example, we sample 10^5 points using Monte-Carlo method, but only 103 points lie in the acceptable region, which makes it extremely inaccurate, and meanwhile takes too much time. In contrast, 3 iterations of **YENSS** takes 362 seconds, and get the partial yield 0.0759%, which has a much better accuracy than Monte-Carlo.

V. CONCLUSION

In this paper, we have presented a new method termed **Y**ield **E**stimation via **N**onlinear **S**urface **S**ampling that estimates yield rapidly and with SPICE-level accuracy. We have demonstrated **YENSS** on SRAM and oscillator circuits with varying threshold voltages. Using **YENSS** results in speedups of two to three orders of magnitude faster than Monte-Carlo simulation, while achieving higher accuracies.

ACKNOWLEDGEMENTS

We are indebted to Amith Singhee (CMU) for illuminating discussions and for bringing [13] to our attention. The 6T SRAM cell in this paper was designed by Shweta Srivastava (Synopsys). Partial support for this work has been provided by the Semiconductor Research Corporation and the National Science Foundation (award 0541396). Computational and infrastructural resources from the Digital Technology Center and the Supercomputing Institute of the University of Minnesota are gratefully acknowledged.

REFERENCES

[1] The International Technology Roadmap for Semiconductors. 2006.
[2] P. Gupta and A.B. Kahng. Manufacturing-Aware Physical Design. In *Computer Aided Design, 2003. ICCAD-2003. International Conference on*, pages 681–687, 9-13 Nov. 2003.
[3] J.F. Swidzinski and Kai Chang. Nonlinear Statistical Modeling and Yield Estimation Technique for Use in Monte Carlo Simulations [Microwave Devices and ICs]. *Microwave Theory and Techniques, IEEE Transactions on*, 48(12):2316–2324, Dec. 2000.
[4] K. Kundert, J. White, and A. Sangiovanni-Vincentelli. *Steady-State Methods for Simulating Analog and Microwave Circuits*. Kluwer Academic Publishers, 1990.
[5] Xiaolue Lai and J. Roychowdhury. TP-PPV: Piecewise Nonlinear, Time-Shifted Oscillator Macromodel Extraction For Fast, Accurate PLL Simulation. In *Computer-Aided Design, 2006. ICCAD '06. IEEE/ACM International Conference on*, pages 269–274, Nov. 2006.
[6] J. Roychowdhury. Exact Analytical Equations for Predicting Nonlinear Phase Errors and Jitter in Ring Oscillators. In *VLSI Design, 2005. 18th International Conference on*, pages 516–521, 3-7 Jan. 2005.
[7] A. Demir, A. Mehrotra, and J. Roychowdhury. Phase Noise in Oscillators: a Unifying Theory and Numerical Methods for Characterisation. In *Design Automation Conference, 1998. Proceedings*, pages 26–31, 15-19 Jun 1998.
[8] Mansour Keramat and Richard Kielbasa. Worst Case Efficiency of Latin Hypercube Sampling Monte Carlo (LHSMC) Yield Estimator of Electrical Circuits. *Circuits and Systems, IEEE International Symposium on*, June 1997.
[9] Michael Lightner Dale Hocevar and Timothy Trick. A Study of Variance Reduction Techniques for Estimating Circuit Yields. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 1983.
[10] Peng Li. Statistical Sampling-Based Parametric Analysis of Power Grids. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25(12):2852–2867, Dec. 2006.
[11] S. Director and G. Hachtel. The Simplicial Approximation Approach to Design Centering. *Circuits and Systems, IEEE Transactions on*, 24(7):363–372, Jul 1977.
[12] S. Director, G. Hachtel, and L. Vidigal. Computationally Efficient Yield Estimation Procedures Based on Simplicial Approximation. *Circuits and Systems, IEEE Transactions on*, 25(3):121–130, Mar 1978.
[13] G. Stehr, H. Graeb, and K. Antreich. Performance Trade-off Analysis of Analog Circuits by Normal-Boundary Intersection. In *Design Automation Conference, 2003. Proceedings*, pages 958–963, 2-6 June 2003.
[14] R. Brayton, S. Director, and G. Hachtel. Yield Maximization and Worst-Case Design with Arbitrary Statistical Distributions. *Circuits and Systems, IEEE Transactions on*, 27(9):756–764, Sep 1980.
[15] H. Chang, V. Zolotov, S. Narayan, and C. Visweswariah. Parameterized Block-based Statistical Timing Analysis with Non-Gaussian Parameters, Nonlinear Delay Functions. In *Design Automation Conference, 2005. Proceedings. 42nd*, pages 71–76, 13-17 June 2005.
[16] Christopher. Michael and Mohammed I. Ismail. *Statistical Modeling for Computer-Aided Design of Mos VLSI Circuits*. Springer, 1993.
[17] Michael T. Heath. *Scientific Computing: An Introductory Survey (second edition)*. McGraw Hill, 2002.
[18] Eugene L. Allgower and Kurt Georg. *Numerical Continuation Methods*. Springer-Verlag, 1990.
[19] L.O. Chua and P-M. Lin. *Computer-aided Analysis of Electronic Circuits : Algorithms and Computational Techniques*. Prentice-Hall, Englewood Cliffs, N.J., 1975.
[20] C. W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice-Hall series in automatic computation. Prentice-Hall, Englewood Cliffs, N.J., 1971.
[21] R. Stuffle and Pen-Min Lin. New Approaches to Computer-Aided Determination of Oscillator Frequency Sensitivities. *Circuits and Systems, IEEE Transactions on*, 27(10):882–891, Oct 1980.
[22] Shweta Srivastava and Jaijeet Roychowdhury. Rapid Estimation of the Probability of SRAM Failure due to MOS Threshold Variations. In *Custom Integrated Circuits Conference, 2007., Proceedings of the IEEE 2007*, 2007.