Rapid Estimation of the Probability of SRAM Failure due to MOS Threshold Variations

Shweta Srivastava and Jaijeet Roychowdhury
{shwetas,jr}@umn.edu
Department of Electrical and Computer Engineering, University of Minnesota

***Abstract*— Accurate estimation of the effects of threshold variations, in particular yield loss, is crucial during the design of robust SRAM cells and memory arrays in deep submicron technologies. We present an efficient technique to calculate yield loss due to access-time, static noise margin, *etc.*, related failures. Our method does not rely on Monte-Carlo techniques; instead, it finds the boundary in Vt (threshold voltage) parameter space between success and failure regions and uses quick geometrical calculations to find the yield. The V_t boundary curve is found efficiently via an Euler-Newton curve tracing technique, adapted from mixed-signal/RF simulation, that guides detailed SPICE-level simulation with accurate MOS device models. We compare and validate the new method against Monte-Carlo style yield estimation, obtaining superior accuracies and speedups of more than $10\times$.**

I. Introduction

SRAM arrays have been becoming increasingly important in System-on-Chip (SoC) and microprocessor designs, as more and more chip area is dedicated to data and instruction cache. For example, in modern multi-core processors like AMD's Opteron and Intel's Core Duo, more than half the chip's area is dedicated to cache; today, more than 70% of this internal cache consists of SRAM, with an increase to beyond 90% expected in future years. As a result, their manufacturability, reliability and robustness are major concerns for ensuring high yield in SoCs and microprocessors. But even as SRAMs are becoming more and more integral to SoCs and microprocessors, their performance and reliability are being eroded by the ever increasing problem of parametric variations in nanometer technologies. SRAMs are especially susceptible to variations (to a much greater extent than other logic circuitry) because they are made of minimum sized devices.

Parametric variations can be broadly categorized into global (inter-die, wafer) vs local (intra-die or single chip) variations, and further subdivided into deterministic (systematic), correlated, and random variations [1]. Global variations, which stem mainly from wafer manufacturing processes, cause the same device to feature different characteristics across different dies. Local variations, caused by the processes like sub-wavelength lithography, gas diffusion, molecular beam epitaxy, *etc.*, are responsible for variations in device characteristics within a single chip. Furthermore, local variations in a number of parameters (mainly lithography/geometrical ones such as channel length and width, oxide thickness, *etc.*) exhibit significant deterministic/systematic spatial dependence (*i.e.*, they are not totally random in nature) and correlation. Finally, fluctuations in many process related parameters (such as dopant concentration within channels, surface roughness, *etc.*) are largely random and statistically independent. Of these, random dopant and surface roughness effects are particularly important for SRAMs, since they are harder to control and their effects are more pronounced for minimum sized devices, which are used commonly in SRAMs. A key effect of these variations is that MOS threshold voltages (V_ts) are significantly affected by variability, but in a statistically independent manner across devices.

As a result, static random-access memory (SRAM) cell performance is significantly affected by variability in the form of mismatches in V_t amongst MOS devices. V_t variations affect not only the driving strength of `ON` transistors, but also leakage currents in `OFF` devices. Variations in driving strength and leakage affect three primary SRAM performance parameters: 1) read access-time, 2) read static noise margin (SNM) and 3) write SNM [2]–[5]. SRAM failure can result from increases in read access-time or decreases in read/write SNM [4], or both. For robust and reliable design of SoCs and microprocessors, it is of paramount importance to accurately estimate the probability of SRAM failures (or equivalently, the *yield*) caused by V_t variations.

The prevalent method today for computing SRAM yield is Monte-Carlo simulation. Fig. 1(a) shows a typical and widely used 6-T SRAM cell. To assess failure probabilities (due to access-time or read/write SNM failure) via Monte-Carlo, the varying thresholds (V_ts) of transistors are first modeled as random variables having known or assumed PDFs (probability distribution functions)[1]. The distribution of the performance of interest (*e.g.*, read access-time, read-SNM or write-SNM), is computed for many parameter samples by means of repeated simulations – typically highly detailed and expensive SPICE-level transient simulations, using the most accurate device models available. The yield is computed by collecting the distribution of the performance metric into two bins (failed/successful), where failure/success is gauged using a threshold criterion. For example, if read access-time is the performance metric of interest, the threshold criterion is the minimum required voltage difference between the bit-lines (*BL* and *BL_B* in Fig. 1(a))[2].

The main disadvantage of Monte-Carlo is that the number of simulations required to obtain the distribution increases significantly with each additional varying parameter. For example, in a SRAM cell, if roughly n samples are required to reasonably sample the varying threshold voltage of a single transistor, then the total number of transient simulations required to find the distribution of a performance metric varies as $O(n^m)$ if there are m transistors. Moreover, the convergence of Monte-Carlo to a specified yield accuracy improves only as $O(\sqrt{n})$ *i.e.*, relatively slowly, further exacerbating the computational problem.

In this paper, we present an alternative approach for computing SRAM yields that is not based on Monte-Carlo or random sampling. Instead, our method views the parameter space as divided into two regions, corresponding to success or failure of the circuit using any given performance metric as a criterion[3]. The problem of finding yield is equivalent to finding the relative (probabilistically weighted) area/volume of the failure region (see Fig. 4).

The Monte-Carlo method estimates this area/volume by placing many probabilistically chosen samples over the entire parameter space of interest and counting those samples which fall in the failure region, *i.e.*, those parameter choices that lead to circuit failure. As more and more such points are uniformly spread over the parameter space of interest, the ratio of the failed points to the total number provides improving estimates of the relative area/volume.

In contrast to Monte-Carlo, our method finds the area/volume of the failure region much more directly, avoiding random choices of parameter samples. It operates by first finding the *boundary curve* between the failure region and the success region (see Fig. 4) in a computationally efficient manner. Once the boundary is obtained, the area on one side of the boundary curve is calculated rapidly using simple formulæ for the areas of triangles and quadrilaterals.

The main effort involved in our method is the calculation of the boundary curve that separates the success and failure regions. We achieve this in an efficient and elegant manner, using a technique called Euler-Newton curve tracing which we adapt from homotopy/continuation methods for finding DC operating points [6] of circuits reliably. Unlike for DC operating point calculation, though, the Euler-Newton curve tracing technique here drives a few carefully-chosen transient simulations of the SRAM cell, corresponding to parameter choices near the boundary curve. The underlying algorithm (which has strong connections with the shooting algorithm used for mixed-signal/RF simulation, and has also been recently employed for finding setup/hold time tradeoffs in latches [7], [8]) rapidly refines these choices to find points on the curve.

The SPICE-level simulations involved are ideally run with tight tolerances and the best device models available, since SRAM performance depends critically on the nuances of device characteristics, especially at the 90nm and 65nm technology nodes. Because our method only performs the relatively few transient simulations that it needs to find points on the curve, it avoids sampling the entire parameter space, hence is much more efficient than Monte Carlo. Also, by not relying on any form of randomness, the accuracy of our yield algorithm is not subject to the vagaries of pseudo-random number generation algorithms. As detailed further in Section IV, for a

[1]However, obtaining reliable distributions of parameters from foundries is notoriously difficult; as a result, the use of min/max bounds and uniform distributions is typical in practice.

[2]Refer to Section II for a detailed explanation of minimum required bit-differential voltage between *BL* and *BL_B* and its relevance to access-time failure/success.

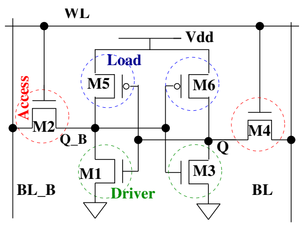
[3]For concreteness, we use read access-time as the performance of interest throughout the paper, but we emphasize that any other criterion may also be employed instead.

prototypical 6T SRAM cell, we obtain speedups of 11× over Monte-Carlo.

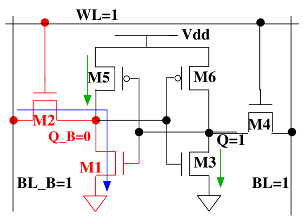
The remainder of the paper is organized as follows. We first review basic notions of SRAM read operations and access-time failure in the presence of variations in Section II. Section III-A presents our formulation for finding the boundary curve (V_t curve) separating the pass-fail boundary and develops our algorithm for calculating the probability of access-time failure. Section III-B provides details of the Euler-Newton curve tracing method we use for finding V_t boundary curve. In Section IV, we validate the proposed technique against Monte-Carlo type simulations and confirm its efficiency and accuracy.

II. SRAM: Read operation and access-time failure

A 6T SRAM cell is shown in Fig. 1(a). It is composed of a pair of cross-connected inverters used for storing the data in the cell, and two pass transistors (also known as *access* transistors) used for accessing or writing the data from/to the SRAM cell. In Fig. 1(a), $M1$, $M5$, $M6$ and $M3$ forms the cross-connected inverters; $M2$ and $M4$ are the access transistors. ($M1$, $M3$) and ($M5$, $M6$) are also commonly referred to as *driver* and *load* transistors respectively. Q and Q_B are the storage nodes of the SRAM cell; BL and BL_B are the bit-lines, which transfers both the stored data and its inverse during the read and write operation. As can be seen from the Fig. 1(a), access to the cell is enabled by the wordline(WL), which controls the access transistors $M2$ and $M4$.



(a) **6T SRAM Cell**.



(b) **Read Operation**.

Fig. 1. (b) Discharge path of BL_B during read operation.

We can assume that $Q = 1$ and $Q\_B = 0$ is stored inside the SRAM cell without loss of generality. During the read operation, both bit-lines BL and BL_B are first precharged to vdd(supply-voltage) and then at the assertion of WL: bit-line BL_B starts to discharge through $M2$ and $M1$ as shown in Fig. 1(b) via a path denoted by the blue arrow; while bit-line BL stays at vdd. A certain minimum difference in voltages of BL and BL_B is required to detect the data stored inside the cell and therefore, an access-time failure (or detection failure) is said to occur if the bit-differential(ΔBL) is less than the minimum required at a certain time for giving a correct detection of stored data value.

Due to the variation in the thresholds of $M1$ and $M2$, driving strengths of these transistors change and affect the required time to generate some predefined voltage difference between BL and BL_B.

Behavior of BL_B for varying thresholds of $M1$ and $M2$ is shown in Fig. 2. If the prespecified minimum bit-differential ΔBL_{min} is required to occur at some certain time t_f for the correct detection of the data stored inside the SRAM cell, then the BL_B2, shown in Fig. 2, denotes the boundary between access-time failure and success as it generates the bit-differential(ΔBL) exactly equal to ΔBL_{min} at time t_f. Certainly, BL_B1 will take more time than t_f to create the minimum required bit-differential(ΔBL_{min}) and therefore, denotes a case of access-time failure; while BL_B3 and BL_B4 are successful candidates as they generate the bit-differential(ΔBL) larger than ΔBL_{min} at time t_f.

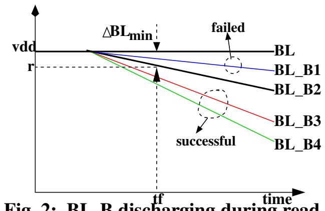
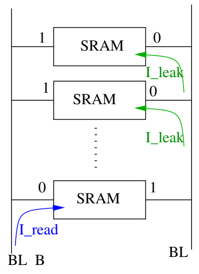
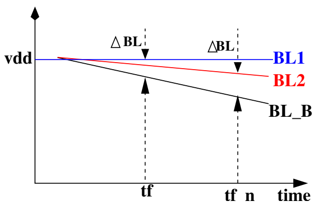


Fig. 2: BL_B discharging during read operation.

There is one more candidate which affects the access-time by a considerable amount and i.e the leakage in bit-line BL; which ideally, should stay at vdd during read operation, but droops due to leakage currents feeding other un-accessed SRAM cells connected to the same column in a memory array and storing the data inverse of the accessed cell (refer to Fig. 3(a) and Fig. 3(b)). In Fig. 3(b), $BL2$ depicts the droop in voltage of BL due to leakage currents and clearly shows the increase in access-time in the generation of desired



(a)



(b)

Fig. 3. (a) BL leaking through un-accessed SRAM cells in a memory column. (b) Increase in access time due to BL leakage.

bit-differential(ΔBL) voltage; which is now happening at time t_{f_n} but could have occurred at time t_f in absence of leakage currents.

Therefore, one should not only monitor the actively discharging bit-line(like BL_B in our case) to make the decision of access-time success or failure, but should also monitor the other bit-line (BL), whose voltage may droop due to leakage currents.

Hence, if the required minimum prespecified voltage difference between BL and BL_B at a certain time t_f is given as ΔBL_{min}, then the probability of access time failure can be given as follows:

$$P_{access-failure} = P(\Delta BL(t_f) < \Delta BL_{min}(t_f)) \tag{1}$$

where, $\Delta BL(t_f)$ is the actual bit-differential at time t_f and $\Delta BL_{min}(t_f)$ is the minimum required bit-differential at time t_f for correct detection of the stored data inside the SRAM cell.

In the next section, we will propose an efficient methodology of finding the probability of access-time failure because of the variation in the thresholds of $M1$ and $M2$ of SRAM cell. Note that the proposed methodology can easily be extended to find the failure probability due to the variations in thresholds of all the transistors of SRAM cell.

III. Problem formulation for finding the probability of access-time failure

In this section, we first develop a formulation that directly solves for those pairs of thresholds of $M1$ and $M2$, which results in the bit-differential of ΔBL_{min} at a given time t_f. The solution will be a curve(V_t curve) in two-dimensional space of V_{t1} and V_{t2}, where V_{t1} and V_{t2} denotes the threshold voltages of $M1$ and $M2$ respectively. The V_t curve will divide the two-dimensional region of V_{t1} and V_{t2} in to two parts: 1) a failure region and 2) a successful region. In the end, probability of failure will be obtained by measuring the properly scaled areas of the successful and the failure regions.

A. Problem formulation to find the V_t curve dividing the successful and the failure regions

Any nonlinear circuit or system can be represented by the following vector differential algebraic equation [9]:

$$\frac{d}{dt}\vec{q}(\vec{x}) + \vec{f}(\vec{x}) + \vec{b}u(t) = 0. \tag{2}$$

(2) is a size n system; $\vec{x} \in \mathbb{R}^n$ is the state vector of internal node voltages and branch currents; $\vec{q} \in \mathbb{R}^n$ and $\vec{f} \in \mathbb{R}^n$ are the charge/flux and the current terms respectively and $\vec{b}u(t) \in \mathbb{R}^n$ represents all the input source voltages and currents. Therefore, a SRAM cell with varying V_{t1} and V_{t2} can be written as follows to explicitly show its dependence on V_{t1} and V_{t2}.

$$\frac{d}{dt}\vec{q}(\vec{x}(t,V_{t1},V_{t2}),V_{t1},V_{t2}) + \vec{f}(\vec{x}(t,V_{t1},V_{t2}),V_{t1},V_{t2}) + \vec{b}u(t) = 0. \tag{3}$$

Let the state transition function of (3) be denoted by $\vec{\phi}(t;\vec{x}_0,t_0 = 0,V_{t1},V_{t2})$, and the initial condition $\vec{x}_0 = \vec{x}(t = t_0)$ be fixed to a given value. The bit-differential voltage at time t_f can be computed as $\vec{c}_{BL}^T\vec{x}(t_f) - \vec{c}_{BL\_B}^T\vec{x}(t_f)$, where $\vec{c}_{BL}^T$ and $\vec{c}_{BL\_B}^T$ are unit vectors which selects the BL and BL_B node voltages respectively. The condition of finding those pairs of V_{t1} and V_{t2}, which result in the bit-differential voltage equal to ΔBL_{min} at time t_f can be written in terms of the state transition function as follows:

$$\begin{aligned}(\vec{c}_{BL}^T - \vec{c}_{BL\_B}^T)\vec{\phi}(t_f;\vec{x}_0,0,V_{t1},V_{t2}) - \Delta BL_{min} &= 0,\\ \text{or } \vec{c}^T\vec{\phi}_{V_t}(V_{t1},V_{t2}) - \Delta BL_{min} &= 0,\\ \text{where } \vec{\phi}_{V_t}(V_{t1},V_{t2}) \equiv \vec{\phi}(t_f;\vec{x}_0,0,V_{t1},V_{t2}), \text{ and } c^T &= \vec{c}_{BL}^T - \vec{c}_{BL\_B}^T. \end{aligned} \tag{4}$$

Hence, the nonlinear equation we need to solve to obtain (V_{t1}, V_{t2}) is

$$\boxed{h(\vec{V}_t) \equiv h(V_{t1},V_{t2}) \equiv \vec{c}^T\vec{\phi}_{V_t}(V_{t1},V_{t2}) - \Delta BL_{min} = 0.} \tag{5}$$

where $\vec{V}_t = [V_{t1}, V_{t2}]$.

Let, the expected range of variations in V_{t1} and V_{t2} be (V_{t1_a}, V_{t1_b}) and (V_{t2_a}, V_{t2_b}) respectively. The solution curve of $h(\vec{V}_t) = 0$ will divide the two-dimensional $V_{t1} - V_{t2}$ space in to two regions as shown in Fig. 4. Access time failure will occur for the region where $h(\vec{V}_t) < 0$, *i.e.* the bit-differential voltage is less than the ΔBL_{min} at time t_f. On the other hand, $h(\vec{V}_t) > 0$ denotes the region, where the bit-differential voltage is larger than the prespecified ΔBL_{min} at time t_f and thus, the access-time is smaller than t_f. Also, it should be noted that the decrease in thresholds of $M1$ and $M2$ makes them stronger in discharging BL_B, therefore, the successful region will be towards lower values of V_t's and the failure region will occur for higher values of V_t's as depicted in Fig. 4 also.

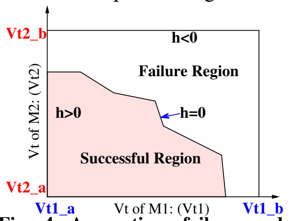


Fig. 4: Access-time failure and successful regions divided by $h(\vec{V}_t) = 0$ curve.

If V_{t1} and V_{t2} are uniformly distributed, then the probability of access-time failure can be calculated as

$$P_{access\_failure} = \frac{Area(\text{successful + failure}) - Area(\text{successful})}{Area(\text{successful + failure})} \tag{6}$$

We also want to emphasize the point that we have assumed the uniform distribution for V_{t1} and V_{t2} for the sake of simplicity in presenting our idea of using V_t curve for the estimation of probability of failure. If any other PDFs of varying V_ts are given, it can easily be incorporated in (6) by prescaling the parameter axes by their respective cumulative distribution functions(CDFs).

But in the end, to find the probability of failure efficiently, we first need to solve $h(\vec{V}_t) = 0$ *i.e.* (5) to get the V_t curve. In the next section, we will give the details of solving (5).

B. *Solving for underdertermined nonlinear equation via Moore-Penrose based Newton-Raphson and V_t curve tracing by Euler-Newton.*

Since (5) is an underdetermined scalar nonlinear equation with two unknowns V_{t1} and V_{t2}, a modified Newton-Raphson, using the Moore-Penrose pseudo-inverse [10] can be applied as an iterative solver for (5).

Intuitively, Moore-Penrose pseudo-inverse Newton-Raphson (MPPI-NR) starts with an initial guess of (V_{t1_0}, V_{t2_0}) and converges to a solution (V_{t1}^c, V_{t2}^c) of (5). A solution curve of (5) is shown in Fig. 5(a), where A denotes the initial guess, while B denotes the point on the solution curve.

In order to solve (5) using MPPI-NR to get one solution of (5), it is necessary to perform three tasks: 1) evaluate $h(V_{t1}, V_{t2})$ given any (V_{t1}, V_{t2}), 2) evaluate $\left[\frac{dh(\vec{V}_t)}{d\vec{V}_t}\right] = \left[\frac{dh(\vec{V}_t)}{dV_{t1}}, \frac{dh(\vec{V}_t)}{dV_{t2}}\right]$, a 1x2 matrix, and 3) compute the Moore-Penrose pseudo inverse of $\left[\frac{dh(\vec{V}_t)}{d\vec{V}_t}\right]$.

$h(V_{t1}, V_{t2})$ is evaluated simply by running a transient simulation with the given (V_{t1}, V_{t2}) and then evaluating (5). To compute $\left[\frac{dh(\vec{V}_t)}{d\vec{V}_t}\right]$, we need to evaluate $\left[\frac{d}{d\vec{V}_t}\vec{\phi}(t_f;\vec{x}_0,0,V_{t1},V_{t2})\right]$.

Next, noting that $\left[\frac{d\vec{\phi}}{d\vec{V}_t}\right] = \left[\frac{d\vec{x}(t,V_{t1},V_{t2})}{dV_{t1}}, \frac{d\vec{x}(t,V_{t1},V_{t2})}{dV_{t2}}\right]$, we first evaluate $\frac{d\vec{x}(t,V_{t1},V_{t2})}{dV_{t1}}$; computation of $\frac{d\vec{x}(t,V_{t1},V_{t2})}{dV_{t2}}$ follows a similar procedure.

To compute $\frac{d\vec{x}(t,V_{t1},V_{t2})}{dV_{t1}}$, we differentiate (3) with respect to V_{t1} and interchange the order of differentiation w.r.t t and V_{t1} in the first term, to obtain:

$$\frac{d}{dt}\left[\frac{d\vec{q}}{d\vec{x}}\frac{d\vec{x}}{dV_{t1}} + \frac{d\vec{q}}{dV_{t1}}\right] + \frac{d\vec{f}}{d\vec{x}}\frac{d\vec{x}}{dV_{t1}} + \frac{d\vec{f}}{dV_{t1}} = 0. \tag{7}$$

Since we want to evaluate $\frac{dh(\vec{V}_t)}{dV_{t1}}$ at any given value of (V_{t1}, V_{t2}) (*e.g.*, at (V_{t1}^*, V_{t2}^*)), we define the following terms for notational convenience:

$$C^\dagger(t) = \frac{d\vec{q}(t,V_{t1}^*,V_{t2}^*)}{d\vec{x}}, G^\dagger(t) = \frac{d\vec{f}(t,V_{t1}^*,V_{t2}^*)}{d\vec{x}}, \vec{n}_1{}^\dagger(t) = \frac{d\vec{q}(t,V_{t1}^*,V_{t2}^*)}{dV_{t1}},$$
$$\vec{p}_1{}^\dagger(t) = \frac{d\vec{f}(t,V_{t1}^*,V_{t2}^*)}{dV_{t1}} \text{ and } \vec{m}_1{}^\dagger(t) = \frac{d\vec{x}(t,V_{t1}^*,V_{t2}^*)}{dV_{t1}}. \tag{8}$$

(7) can be discretized using any integration method (*e.g.* BE, TRAP *etc.*) [9], [11]. The discretization of (7) using BE (for example) will give

$$\vec{m}_1{}_i^\dagger = \left(\frac{C_i^\dagger}{\Delta t} + G_i^\dagger\right)^{-1}\left(\frac{C_{i-1}^\dagger}{\Delta t}\vec{m}_1{}_{(i-1)}^\dagger - \vec{n}_1{}^\dagger(t_i) + \vec{n}_1{}^\dagger(t_{i-1}) - \vec{p}_1{}^\dagger(t_i)\right). \tag{9}$$

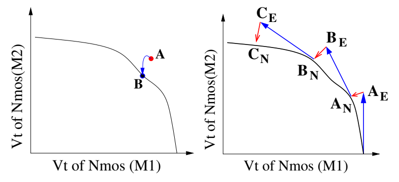
The subscript i denotes that the quantity is being evaluated at $t = t_i$; $\Delta t = t_i - t_{i-1}$ is the time step used in the integration.

To start the integration process, we set $\vec{m}_1{}_0^\dagger$ to $\vec{0}$, because, $\vec{x}_0 = \vec{x}(t = t_0)$ do not change with V_{t1} or V_{t2}. Evaluating (9) from $t = t_1$ to $t = t_f$ (*i.e.* $i \in 1, 2, \ldots, f$), we obtain $\vec{m}_1{}_f^\dagger = \frac{d\vec{\phi}_{V_t}(V_{t1}^*,V_{t2}^*)}{dV_{t1}}$ and then $\frac{dh(V_{t1}^*,V_{t2}^*)}{dV_{t1}} = \vec{c}^T\frac{d\vec{\phi}_{V_t}(V_{t1}^*,V_{t2}^*)}{dV_{t1}}$. Clearly, $\frac{dh(V_{t1}^*,V_{t2}^*)}{dV_{t2}} = \vec{c}^T\frac{d\vec{\phi}_{V_t}(V_{t1}^*,V_{t2}^*)}{dV_{t2}}$ can be obtained in a similar way. Finally, denoting the matrix $\left[\frac{dh(\vec{V}_t)}{d\vec{V}_t}\right]$ by $H(\vec{V}_t)$, its Moore-Penrose pseudo-inverse [10] can be expressed as

$$H(\vec{V}_t)^+\Big|_{(V_{t1}^*,V_{t2}^*)} = H(\vec{V}_t)^t\left(H(\vec{V}_t)H(\vec{V}_t)^t\right)^{-1}\Big|_{(V_{t1}^*,V_{t2}^*)}, \tag{10}$$

where $H(\vec{V}_t)^+$ and $H(\vec{V}_t)^t$ represent the pseudo inverse and transpose of the matrix $H(\vec{V}_t)$, respectively.

To trace the entire solution curve (*i.e.* all the solutions of (5)), we employ Euler-Newton(EN) curve tracing methodology, which follows a standard predictor-corrector methodology [11], using Euler steps as predictors and MPPI-NR steps as correctors. Taking an Euler predictor step involves computing the tangent vector to the solution curve at a previously known point on the curve, and extrapolating to a new point along the tangent. The MPPI-NR is then used as a corrector that uses this new point as its initial guess and converges to a nearby solution point on the curve. The EN curve tracing procedure is depicted graphically in Fig. 5(b), where the blue and red arrows denote the Euler predictor steps and the MPPI-NR corrector steps, respectively.



(a) (b)

Fig. 5. (a)Convergence of a point via Newton-Raphson on the solution curve (V_t curve). (b) Curve tracing using the Euler-Newton method.

We have already given details above for evaluation of MPPI-NR steps. For the Euler step, the key quantity we need to evaluate is a unit tangent vector at any given point $(V_{t1}{}^c, V_{t2}{}^c)$ on the curve. The unit tangent vector at point $\vec{V_t}^c = (V_{t1}{}^c, V_{t2}{}^c)$, denoted by $T\left(H(\vec{V}_t)\right)$ can be computed as follows [10]:

$$T\left(H(\vec{V}_t)\right)\Big|_{\vec{V}_t = \vec{V_t}^c} = \begin{pmatrix} -\frac{dh(\vec{V}_t)}{dV_{t2}} \\ \frac{dh(\vec{V}_t)}{dV_{t1}} \end{pmatrix} \frac{1}{\sqrt{\left(\frac{dh(\vec{V}_t)}{dV_{t1}}\right)^2 + \left(\frac{dh(\vec{V}_t)}{dV_{t2}}\right)^2}}\Bigg|_{\vec{V}_t = \vec{V_t}^c}. \tag{11}$$

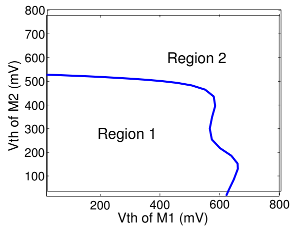
The rectangular matrix in (11) is simply the MPPI-NR Jacobian matrix at the current solution point, hence is already available; since it is of size 2, the computation involved in finding the tangent vector is trivial. An algorithm for Euler-Newton can be found in [8], [10].

IV. Results and Validation

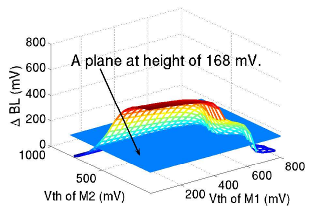
In this section, we validate our new yield calculation method on the SRAM cell shown in Fig. 1(a). Our results, which we compare with the Monte-Carlo simulation technique, confirm that the new direct V_t curve tracing method for yield provides accurate results, with computation that is linear in the number of points on the solution curve desired. We obtain speedups of 11.2× over Monte-Carlo like simulation. Our results are obtained for MPPI-NR tolerances set such that the points obtained on the curve are accurate up to 1mV of precision for bit-differential voltage(ΔBL) (the supply voltage was 900mV). All algorithms are implemented in a MATLAB/C/C++ simulation prototyping environment; simulations were performed on an AMD Athlon64 3000+ based PC, with 512MB RAM, running linux kernel 2.6.12.

A. *Finding the V_t curve separating the failure/success regions*

The nominal threshold voltages of M_1 and M_2 are 500mV. The supply voltage is 900mV for all simulations. We set the failure threshold condition to be $\Delta BL_{min} = 168mV$ at $t_f = 3.508ns$ in (5). The solution curve of $h(\vec{V_t}) = 0$ for the above values of ΔBL_{min} and t_f obtained using our method is shown in Fig. 6(a).



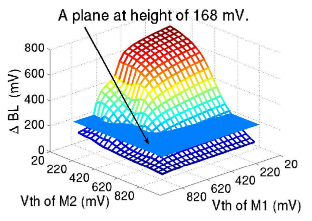
(a) V_t **curve.**



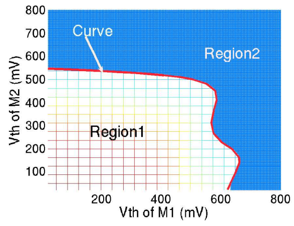
(b) ΔBL **surface**

Fig. 6. (a) V_t curve obtained by Euler-Newton method. This curve partitions the region of Vt's of Driver (M1) and Access (M2) transistors in to the failure and the successful regions. (b) Bit-differential (ΔBL) surface generated using Monte-Carlo style simulations; the plane at the failure threshold 168mV intersects the surface to produce the same V_t curve found by our method.

As explained earlier, Region 2 in Fig. 6(a) corresponds to read access failure, and Region 1 to success. To validate the accuracy of the V_t curve computed via our Euler-Newton curve tracing method, we also generated the V_t curve using brute-force Monte-Carlo style simulations. We analysed the SRAM cell using transient simulations for many sets of independently-varying threshold voltages V_{t1} and V_{t2} for the transistors M_1 and M_2, respectively, measured the bit-differential voltage at the threshold time $t_f = 3.508ns$. The plot of this ΔBL against V_{t1} and V_{t2} is a surface, shown in Fig. 6(b).



(a) **Rotated view.**



(b) **Top View**

Fig. 7. (a) Rotated view of Fig. 6(b). (b) Top view of the extracted contour from ΔBL surface and the superimposition of the V_t curve shown in Fig. 6(a)

To find the curve in the (V_{t1}, V_{t2}) plane corresponding to the failure criterion, we intersected a plane at $\Delta BL = 168$mV with the surface, as shown in Fig. 6(b) and Fig. 7(a). The curve obtained from our Euler-Newton curve tracing method (Fig. 6(a)) is overlaid on the intersection of the plane and the ΔBL surface in Fig. 7(b). It is apparent from Fig. 7(b) that the V_t curve obtained by Euler-Newton exactly matches the V_t contour obtained from the ΔBL surface, verifying the correctness of our method. For 24 points on V_t curve, the brute-force Monte-Carlo style method took 3 hours and 21 minutes of computer time, while our Euler-Newton method took 18 min — representing a speedup of 11.2, over an order of magnitude.

B. *Probability of access-time failure due to V_t variations*

Having efficiently obtained the V_t curve via our Euler-Newton method, , we now proceed to find the area in Region 1 to obtain the yield. An additional advantage of our Euler-Newton method is that it computes "exact" points on V_t curve (instead of the effectively interpolated ones by Monte Carlo) thus enhancing the accuracy of yield estimates.

The area inside Region 1 is divided into a number of trapezoids and their areas calculated via a simple formula and summed up, as follows (the area in Region 1 is denoted by A_1; points on the V_t curve are denoted by (x_i, y_i)):

$$A_1 = \sum_{i=1}^{n} \frac{1}{2}\left((y_i + y_{i+1})(x_{i+1} - x_i)\right) \qquad (12)$$

Using (6), we calculate the access-time failure probability equal to be 0.4325. This yield number, depends, of course, on not only the circuit and the parameter distributions/bounds, but also on the failure criterion for the bit-differential voltage. We emphasize that this procedure can be extended to threshold voltage variations in additional transistors, retaining the same methodology. Also, we have focused on read access-time for the purpose of concreteness and illustration; our method is generally applicable to any other performance metrics, such as read/write SNM.

Acknowledgments

It is a pleasure to acknowledge Sani Nassif (IBM Austin Research Laboratory), who introduced us to the variability problem for SRAMs during a visit to the University of Minnesota [12] and has been a supporter and mentor of this work. Support for this work was provided in part by MARCO/GSRC, the National Science Foundation and the Semiconductor Research Corporation. Computational and infrastructural resources provided by the Digital Technology Center and the Supercomputing Institute of the University of Minnesota have been important enablers of this work. Konina Sircar offered a number of valuable suggestions for improving the manuscript.

References

[1] S. R. Nassif. Modeling and Analysis of Manufacturing Variations. In *Proc. IEEE Custom Integrated Circuits Conference*, pages 223–228, 2001.

[2] Q. Chen, H. Mahmoodi, S. Bhunia, and K. Roy. Efficient Testing of SRAM With Optimized March Sequences and a Novel DFT Technique for Emerging Failures Due to Process Variations. *IEEE Transactions on Very Large Scale Integration Systems*, 13:1286—1295, November 2005.

[3] X. Li, B. Huang, X. Zhang, and J. B. Bernstein. SRAM Circuit-Failure Modeling and Reliability Simulation With SPICE. *IEEE Transactions on Device and Materials Reliability*, 6:235—246, June 2006.

[4] S. Mukhopadhyay, H. Mahmoodi, S. Bhunia, and K. Roy. Modeling of Failure Probability and Statistical Design of SRAM Array for Yield Enhancement in Nanoscaled CMOS. *IEEE Transactions on Computer-Aided Design Of Integrated Circuits and Systems*, 24:1859—1880, December 2005.

[5] Y. Tsukamoto, K. Nii, S. Imaoka, Y. Oda, and S. Ohbayashi. Worst-Case Analysis to Obtain Stable Read/Write DC Margin of High Density 6T-SRAM-Array with Local Vth Variability. In *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pages 398–405, November 2005.

[6] J. Roychowdhury and R. Melville. Delivering Global DC Convergence for Large Mixed-Signal Circuits via Homotopy/Continuation Methods. *IEEE Transactions on Computer-Aided Design*, January 2006.

[7] S. Srivastava and J. Roychowdhury. Rapid and Accurate Latch Characterization via Direct Newton Solution of Setup/Hold Times. In *Proc. IEEE Design, Automation, and Test in Europe Conference*, April 2007.

[8] S. Srivastava and J. Roychowdhury. Interdependent Latch Setup/Hold Time Characterization via Euler-Newton Curve Tracing on State-Transition Equations. In *Proc. IEEE Design Automation Conference*, June 2007.

[9] L. O. Chua and P. M. Lin. *Computer-Aided Analysis of Electronic Circuits: Algorithms and Computation Techniques*. Prentice-Hall, Englewood Cliffs, NJ, 1975.

[10] E. L. Allgower and K. Georg. *Numerical Continuation Methods*. Springer-Verlag, New York, 1990.

[11] C. W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice-Hall series in automatic computation. Prentice-Hall, Englewood Cliffs, N.J., 1971.

[12] S. Nassif. SRAM Memory Cell Modeling, February 2007. Digital Technology Center, University of Minnesota.