Statistical Soft Error Rate (SSER) Analysis for Scaled CMOS Designs

HUAN-KAI PENG, HSUAN-MING HUANG, YU-HSIN KUO, and CHARLES H.-P. WEN,
National Chiao Tung University, Taiwan

This article re-examines the soft error effect caused by radiation-induced particles beyond the deep submicron regime. Considering the impact of process variations, voltage pulse widths of transient faults are found no longer monotonically diminishing after propagation, as they were formerly. As a result, the soft error rates in scaled electronic designs escape traditional static analysis and are seriously underestimated. In this article we formulate the statistical soft error rate (SSER) problem and present two frameworks to cope with the aforementioned sophisticated issues. The *table-lookup* framework captures the change of transient-fault distributions implicitly by using a Monte-Carlo approach, whereas the *SVR-learning* framework does the task explicitly by using statistical learning theory. Experimental results show that both frameworks can more accurately estimate SERs than static approaches do. Meanwhile, the SVR-learning framework outperforms the table-lookup framework in both SER accuracy and runtime.

Categories and Subject Descriptors: J.6 [**Computer Applications**]: Computer-Aided Engineering—*Computer-aided design (CAD)*

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Soft error, transient fault, statistical learning, Monte Carlo method, support vector machine

ACM Reference Format:
Peng, H.-K., Huang, H.-M., Kuo, Y.-H., and Wen, C. H.-P. 2012. Statistical soft error rate (SSER) analysis for scaled CMOS designs. ACM Trans. Des. Autom. Electron. Syst. 17, 1, Article 9 (January 2012), 24 pages.
DOI = 10.1145/2071356.2071365 http://doi.acm.org/10.1145/2071356.2071365

1. INTRODUCTION

Soft errors have emerged to be the dominant failure mechanism for reliability in modern CMOS technologies. Soft errors result from radiation-induced transient faults latched by memory elements were of concern for memory units only, but are now commonplace for logic units beyond deep submicron technologies. As predicted in [Amusan et al. 2007; Dodd and Massengill 2003; Shivakumar et al. 2002], the soft error rate in combinational logic will be comparable to that of unprotected memory cells in 2011. Therefore, numerous studies have been dedicated to modeling transient faults [Cha and Patel 1993; Garg et al. 2008; Omana et al. 2003; Tosaka et al. 1999], propagation and simulation/estimation of soft error rates [Rajaraman et al. 2006; Rao et al. 2006; Zhang and Shanbhag 2004; Zhang et al. 2006] and circuit hardening techniques including detection and protection [Mukherjee et al. 2002; Bartlett and Spainhower 2004; Mitra et al. 2005; Zhang et al. 2007].

This work was supported in part by National Science Council of Taiwan, NSC-99-2220-E-009-011.
Authors' address: H.-K. Peng, H.-M. Huang, Y.-H. Kuo, and C. H.-P. Wen, Department of Electrical Engineering, National Chiao Tung University, Taiwan; email: opwen@g2.nctu.edu.tw.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from the Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701, USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2012 ACM 1084-4309/2012/01-ART9 $10.00
DOI 10.1145/2071356.2071365 http://doi.acm.org/10.1145/2071356.2071365

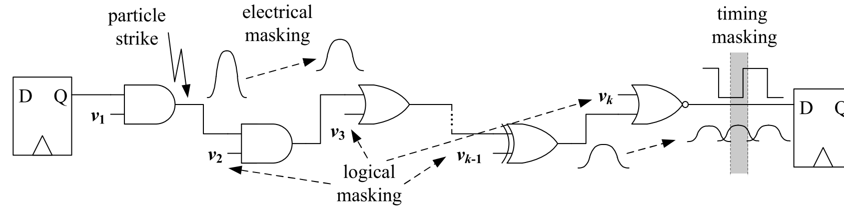

Fig. 1. Three masking mechanisms for soft errors.

Three masking mechanisms shown in Figure 1 are indicated by Shivakumar et al. [2002] as the key factors to determine if one transient fault can be latched by the memory elements to become a soft error. *Logical masking* occurs when the input value of one gate blocks the propagation of the transient fault under a specific input pattern. One transient fault attenuated by *electrical masking* may disappear due to the electrical properties of the gates. *Timing masking* represents the situation that the transient fault propagates to the input of one memory element outside the window of its clock transition.

Much previous work such as that of Omana et al. [2003] and Mohanram [2005] propagate transient faults through one gate according to the logic function, and in the meantime use analytical models to evaluate the electrical change of transient faults. A refined model is presented in Garg et al. [2008] to incorporate nonlinear transistor current, which is further applied to different gates with different charges deposited. A static analysis is also proposed in Krishnaswamy et al. [2008] for timing masking by computing backwards the propagation of the error-latching windows efficiently.

Moreover, in recent years, circuit reliability in terms of soft error rate (SER) has been extensively investigated. SERA [Zhang and Shanbhag 2004] computes SER by means of a waveform model to consider the electrical attenuation effect and error-latching probability while ignoring logical masking. Whereas FASER [Zhang et al. 2006] and MARS-C [Miskov-Zivanov and Marculescu 2006] apply symbolic techniques to logical and electrical maskings and scale the error probability according to the specified clock period, AnSER [Krishnaswamy et al. 2008] applies signature observability and latching-window computation for logical and timing maskings to approximate SER for circuit hardening. SEAT-LA [Rajaraman et al. 2006] and the algorithm in Rao et al. [2006] simultaneously characterize cells, flip-flops, and propagation of transient faults by waveform models and result in good SER estimates when compared to SPICE simulation. However, all of these techniques are deterministic and may not be capable of explaining more sophisticated circuit behaviors due to the growing process variations beyond the deep submicron era.

Process variations, including various manufacturing defects, have grown to be one of the major challenges to scaled CMOS designs [Borkar et al. 2003; Bowman et al. 2002]. From Natarajan et al. [1998] and Borkar et al. [2003], 25%-30% variation on chip frequency are observed. For design reliability, 15%-40% SER variations are reported in Ramakrishnan et al. [2007] under the 70nm technology. Also, Miskov-Zivanov et al. [2008] proposed a symbolic approach to propagate transient faults considering process variations.

Using the 45nm Predictive Technology Model (PTM) [Nanoscale Integration and Modeling Group 2008], the impact of process variations on circuit reliability is illustrated in Figure 2, where SERs are computed by SPICE simulation on a sample circuit $c17$ from ISCAS 85 under different values (σ_{proc}s) of process variation applied to perturbing separately the *gate width* and *channel length* of each transistor in each cell's geometry. The X-axis and Y-axis denote σ_{proc} and SER, respectively, where FIT

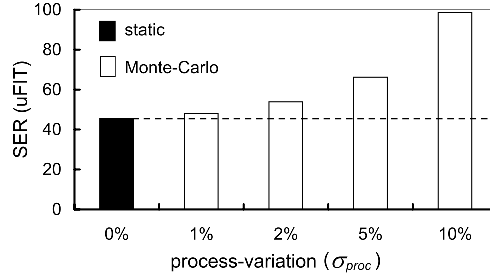

Fig. 2. SER discrepancies between static and Monte-Carlo SPICE simulation w.r.t. process-variation.

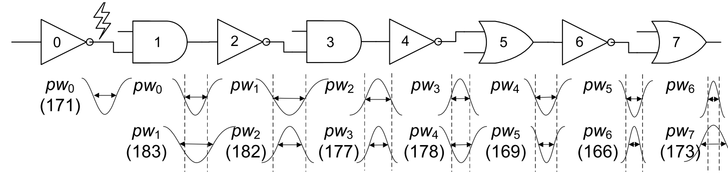

Fig. 3. Static SPICE simulation of a path in the 45nm technology.

(Failure-In-Time) is defined by the number of failures per 10^9 hours. Nominal settings without variation are used in static SPICE simulation, whereas Monte-Carlo SPICE simulations are used to approximate process-variation impacts under different σ_{proc}'s.

As a result, SER from static SPICE simulation is *underestimated*. Considering different σ_{proc}s in Monte-Carlo SPICE simulation, all SERs are higher than that from static SPICE simulation. As process variations deteriorate, the discrepancy between Monte-Carlo and static SERs further enlarges. In Figure 2, $(SER_{monte} - SER_{static})/SER_{static}$ under σ_{proc} = 1%, 2%, 5%, and 10% are 6%, 19%, 46%, and 117%, respectively. Such a result suggests that the impact of process variations to SER analysis may no longer be ignored in scaled CMOS designs.

The rest of the article is organized as follows. A study on the statistical natures of transient faults is first explored in Section 2, and these phenomena help formulate the statistical soft error rate (SSER) problem in Section 3. Sections 4 and 5, respectively, propose a table-lookup framework and a SVR-learning framework as the treatments for the SSER problem. Experimental results from two proposed frameworks are presented and compared to SPICE simulation results in Section 6. Conclusions and future work are discussed in Section 7.

2. TRANSIENT-FAULT BEHAVIOR IN VERY DEEP SUBMICRON ERA

Transient faults exhibit two characteristics in the very deep submicron era. One makes the faults more unpredictable, whereas the other causes the discrepancy in Figure 2. In this section, they are associated with the electrical and timing masking mechanisms, respectively.

2.1 To Be Electrically Better Or Worse?

The first observation is conducted by running static SPICE simulation on a path consisting of various gates (including 2 AND, 2 OR, and 4 NOT gates) in the 45nm PTM technology. As shown in Figure 3, the radiation particle first strikes the output of the first NOT gate with a collection charge of $32fC$, and then propagates the

transient fault along other gates with all side-inputs being set properly. The pulse widths (pw_is) in voltage of the transient fault starting at the struck node and after passing gates along the path in order are 171ps, 183ps, 182ps, 177ps, 178ps, 169ps, 166ps, and 173ps, respectively. Each pw_i and pw_{i+1} can be compared to show the changes of voltage pulse widths during propagation in Figure 3.

As we can see, the voltage pulse widths of such transient faults grow larger through gates #1, #4, and #7, while gates #2, #3, #5, and #6 attenuate such transient faults. Furthermore, gates of the same type behave differently when receiving different voltage pulses. To take AND-type gates for example: the output pw_1 is larger than the input pw_0 on gate #1, while the contrary situation ($pw_3 < pw_2$) occurs on gate #3. This result suggests that the voltage pulse width of a transient fault is not always diminishing, which contradicts some assumptions made in traditional static analysis [Rajaraman et al. 2006]. A similar phenomenon called Propagation Induced Pulse Broadening (PIPB) is discovered by Ferlet-Cavrois et al. [2007], and states that the voltage pulse width of a transient fault widens as it propagates along the long inverter chain.

2.2 When Error-Latching Probability Meets Process Variations

The second observation is dedicated to the timing masking effect under process variations. In Miskov-Zivanov and Marculescu [2006] and Zhang et al. [2006], the error-latching probability (PL) for one flip-flop is defined as

$$PL = \frac{pw - w}{t_{clk}}. \tag{1}$$

where pw, w and t_{clk} denote the pulse width of the arrival transient fault, the latching window of the flip-flop, and the clock period, respectively. However, process variations make pw and w become *random variables*. Therefore, we need to redefine Eq. (1) as follow.

Definition 1 *($P_{err-latch}$, error-latching probability)*. Assume that the pulse width of one arrival transient fault and the latching window ($t_{setup} + t_{hold}$) of the flip-flop are random variables and denoted as pw and w, respectively. Let $x = pw - w$ be another random variable and μ_x and σ_x be its mean and variance. The latch probability is defined as

$$\mathrm{P}_{err-latch}(pw, w) = \frac{1}{t_{clk}} \int_0^{\mu_x + 3\sigma_x} x \times \mathrm{P}(x > 0) \times dx. \tag{2}$$

With the above definition, we further illustrate the impact of process variations on SER analysis. Figure 4(a) shows three transient-fault distributions with the same pulse-width mean (95ps) under different σ_{proc}s: 1%, 5%, and 10%. A fixed latching window w = 100ps is assumed as indicated by the solid lines. According to Eq. (1), static analysis result in zero SER under all σ_{proc}s because $95 - 100 < 0$.

From a statistical perspective, however, these transient faults all yield positive and different SERs. It is illustrated using two terms: $\mathrm{P}(x > 0)$ and x in Eq. (2). First, in Figure 4(a), the *cumulative probabilities* for $pw > w$ under three different σ_{proc}s are 17%, 40%, and 49%, respectively. The largest σ_{proc} corresponds to the largest $\mathrm{P}(x > 0)$ term. Second, in Figure 4(b), we compute the pulse-width averages for the portion $x = pw - w > 0$, and they are 1, 13, and 26, respectively. Again, the largest σ_{proc} corresponds to the largest x term.

These two effects jointly suggest that larger σ_{proc} leads to larger $\mathrm{P}_{err-latch}$, which has been neglected in traditional static analysis, and also explain the increasing

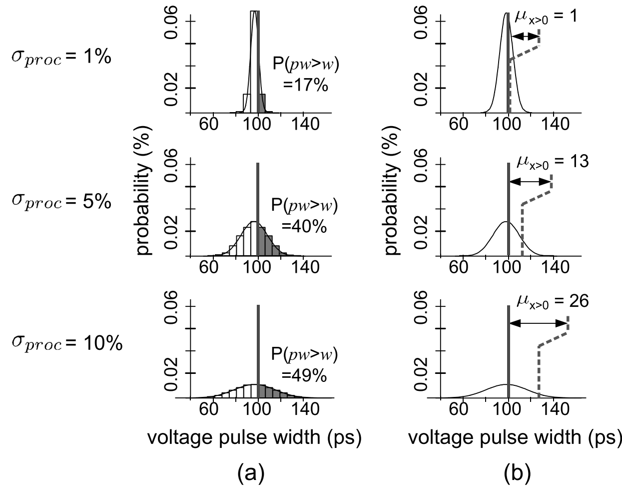

Fig. 4. Process-variation vs. error-latching probabilities.

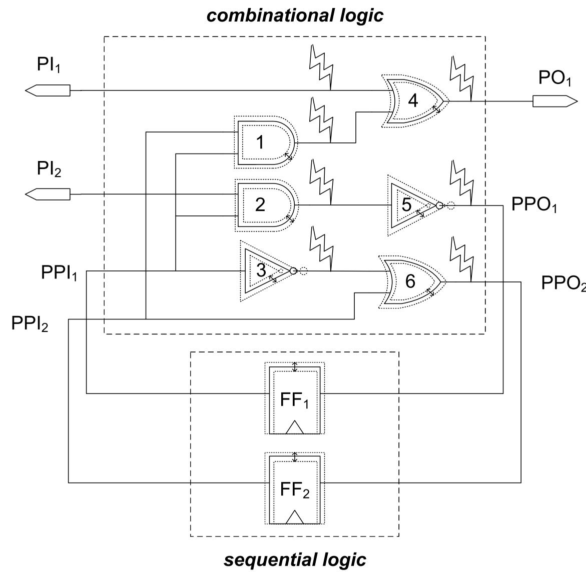

Fig. 5. An example illustrating the SSER problem.

discrepancy shown in Figure 2. In summary, process variations make traditional static analysis no longer effective, and should be considered in accurate SER estimation for scaled CMOS designs.

3. PROBLEM FORMULATION OF STATISTICAL SOFT ERROR RATE (SSER)

In this section, we formulate the statistical soft error rate (SSER) problem for general cell-based circuit designs. Figure 5 illustrates a sample circuit subject to process variations, where the geometries of each cell vary [Natarajan et al. 1998]. Once high-energy particles strike the diffusion regions of these variable-size cells, according to Figures 2, 3, and 4, the electrical performance of the resulting transient faults also vary a lot. Accordingly, to accurately analyze the soft error rate (SER) of a circuit, we

need to integrate both process-variation impacts and three masking effects discussed in Section 1 simultaneously, which brings up the statistical soft error rate (SSER) problem.

The SSER problem is composed of three elements: (1) electrical probability computation; (2) propagation probability computation; and (3) overall SER estimation. A bottom-up mathematical explanation of the SSER problem will start in reverse from overall SER estimation to electrical probability computation.

3.1 Overall SER Estimation

The overall SER for the circuit under test (CUT) can be computed by summing up the SERs of each individual node in the circuit. That is,

$$\mathrm{SER}_{CUT} = \sum_{i=0}^{N_{node}} \mathrm{SER}_i \tag{3}$$

where N_{node} denotes the total number of possible nodes to be struck by radiation particles in the CUT and SER_i denotes the SER results from node i, respectively.

Each SER_i can be further formulated by integrating over the range $q = 0$ to Q_{max} (the maximum collection charge from the environment) the products of *particle-hit rate* and the total number of soft errors that q can induce at node i. Therefore,

$$\mathrm{SER}_i = \int_{q=0}^{Q_{max}} (\mathrm{R}_i(q) \times \mathrm{F}_{soft-err}(i, q))\mathrm{d}q. \tag{4}$$

In a circuit, $\mathrm{F}_{soft-err}(i, q)$ represents the total number of *expected* soft errors from each flip-flop that a transient fault from node i can propagate to. $\mathrm{R}_i(q)$ represents the effective frequency for a particle hit of charge q at node i in unit time according to Shivakumar et al. [2002] and Zhang and Shanbhag [2004]. That is,

$$\mathrm{R}_i(q) = F \times K \times A_i \times \frac{1}{Q_s} e^{\frac{-q}{Q_s}} \tag{5}$$

where F, K, A_i, and Q_s denote respectively, the neutron flux ($>$ 10MeV); a technology-independent fitting parameter; the susceptible area of node i in cm^2; and the charge collection slope.

3.2 Logical Probability Computation

$\mathrm{F}_{soft-err}(i, q)$ depends on all three masking effects and can be decomposed into

$$\mathrm{F}_{soft-err}(i, q) = \sum_{j=0}^{N_{ff}} \mathrm{P}_{logic}(i, j) \times \mathrm{P}_{elec}(i, j, q) \tag{6}$$

where N_{ff} denotes the total number of flip-flops in the circuit under test. $\mathrm{P}_{logic}(i, j)$ denotes the overall logical probability of successfully generating a transient fault and propagating it through all gates along the path from node i to flip-flop j. It can be computed by multiplying the signal probabilities for specific values on target gates, as follows:

$$\mathrm{P}_{logic}(i, j) = \mathrm{P}_{sig}(i = 0) \times \prod_{k \in i \rightsquigarrow j} \mathrm{P}_{side}(k) \tag{7}$$

where k denotes one gate along the target path ($i \rightsquigarrow j$) starting from node i and ending at flip-flop j; P_{sig} denotes the signal probability for the designated logic value; and P_{side}

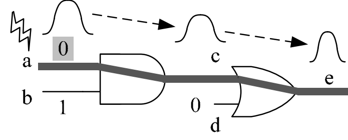

Fig. 6. Logical probability computation for one sample path.

denotes the signal probability for the noncontrolling values (i.e., 1 for AND gates and 0 for OR gates) on all side inputs along the target path.

Figure 6 illustrates an example where a particle striking net a results in a transient fault that propagates through net c and net e. Suppose that the signal probability of being 1 and 0 on one arbitrary net i is P_i and (1-P_i), respectively. In order to propagate the transient fault from a towards e successfully, net a needs to be 0 while net b, the side input of a, and net d, the side input of c, need to be noncontrolling, simultaneously. Therefore, according to Eq. (7),

$$\begin{aligned} P_{logic}(a, e) &= P_{sig}(a = 0) \times P_{side}(a) \times P_{side}(c) \\ &= P_{sig}(a = 0) \times P_{sig}(b = 1) \times P_{sig}(d = 0) \\ &= (1 - P_a) \times P_b \times (1 - P_d). \end{aligned}$$

3.3 Electrical Probability Computation

Electrical probability $P_{elec}(i, j, q)$ comprises the electrical and timing masking effects, and can be further defined as

$$\begin{aligned} P_{elec}(i, j, q) &= P_{err-latch}(pw_j, w_j) \\ &= P_{err-latch}(\lambda_{elec-mask}(i, j, q), w_j). \end{aligned} \tag{8}$$

While $P_{err-latch}$ accounts for the timing masking effect as defined in Equation (2), $\lambda_{elec-mask}$ accounts for the electrical masking effect with the following definition.

Definition 2 ($\lambda_{elec-mask}$, *electrical masking function*). Given the node i where the particle strikes to cause a transient fault and flip-flop j is the destination that the transient fault finally ends at, assume that the transient fault propagates along one path ($i \rightsquigarrow j$) through $v_0, v_1, ..., v_m, v_{m+1}$ where v_0 and v_{m+1} denote node i and flip-flop j, respectively. Then the electrical masking function is defined as

$$\lambda_{elec-mask}(i, j, q) = \underbrace{\delta_{prop}(\cdots(\delta_{prop}(\delta_{prop}}_{m \text{ times}}(pw_0, 1), 2), \cdots), m) \tag{9}$$

where $pw_0 = \delta_{strike}(q, i)$ and $pw_k = \delta_{prop}(pw_{k-1}, k)\ \forall k \in [1, m]$.

In the above definition, two undefined functions, δ_{strike} and δ_{prop}, respectively, represent the *first-strike* function and the electrical *propagation* function of transient-fault distributions. $\delta_{strike}(q, i)$ is invoked once and maps the collection charge q at node i into a voltage pulse width pw_0. $\delta_{prop}(pw_{k-1}, k)$ is invoked m times and *iteratively* computes the pulse width pw_k after the input pulse width pw_{k-1} propagates through the kth cell from node i. These two types of functions are also the most critical components to the success of a statistical SER analysis framework due to the difficulty of integrating process-variation impacts.

The theoretical SSER in Eq. (7) and Eq. (9) is analyzed from a path perspective. However, in reality, since both the signal probabilities and transient-pulse changes through a cell are independent of each other, the computation of SSER only needs to

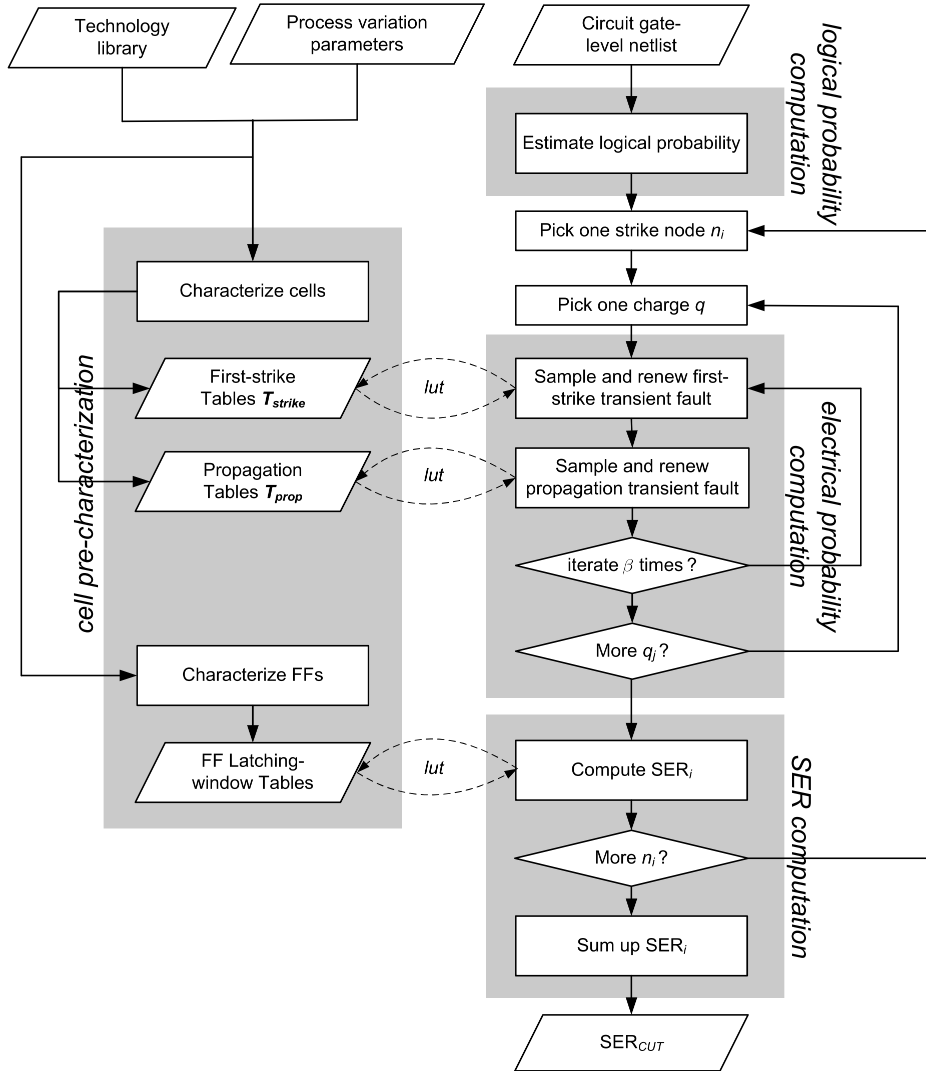

Fig. 7. Proposed table-lookup framework.

proceed stage by stage, and thus can be implemented in a block-based fashion. Sections 4 and 5 will present two different block-based SSER frameworks, a table-lookup framework, and SVR learning framework, respectively. Both frameworks consider process variations but differ from the way they compute δ_{strike} and δ_{prop}: the former does it implicitly; the later, explicitly.

4. A BASELINE TABLE-LOOKUP MONTE-CARLO (MC) FRAMEWORK

The first framework combines the current static approaches with the Monte-Carlo (MC) method, a computational algorithm using repeated random samplings to portray complex statistical behaviors of physical or mathematical systems. As depicted in Figure 7, this framework maps to the formulation in Section 3, using three loops: the outmost loop considers various levels of collection charge q_i, which forms the discrete

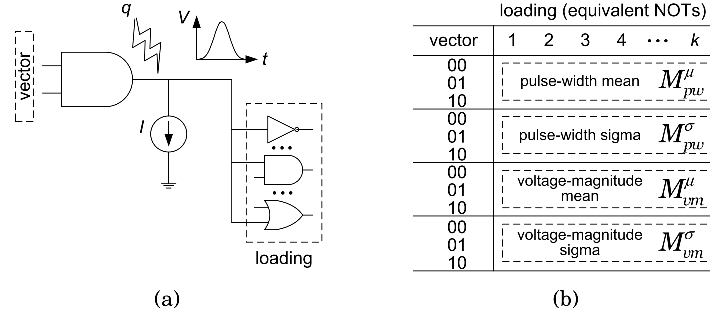

Fig. 8. Precharacterization of particle-strike table T_{strike} for an AND gate.

approximation of Eq. (4); the second loop accounts for all vulnerable nodes within a circuit, which corresponds to Equation (6); the innermost loop maps to Eq. (9) and computes δ_{strike} and δ_{prop} implicitly. As the key component of the framework, the last loop can be further decomposed into two parts: (1) cell precharacterization and (2) sampling and renewal of transient faults.

4.1 Cell Precharacterization

To reflect the electrical masking effect of transient faults on one cell intertwined with process variations, an approach similar to Edamatsu et al. [1998] is employed to extract precharacterized tables. The objective of such precharacterized tables is to model the pulse width and voltage magnitude for each cell as random variables that can be sampled during the particle-strike process and transient-fault propagation of one cell.

Table contents are derived on the basis of data from Monte-Carlo SPICE simulation with targeted process-variation parameters (or direct silicon measurement on test structures if applicable). Considering the mapping relationship, two types of tables are built for each cell separately: one for the particle-strike process, T_{strike}, and the other for transient-fault propagation, T_{prop}.

4.1.1 Particle-Strike Table T_{strike}. T_{strike} maps the collection charge q incurred by the particle strike to electrical properties of cells. Figure 8 illustrates the example to precharacterize one AND gate by properly setting up a SPICE simulation environment. Figure 8(a) is the circuit netlist where a charge q is injected at the output of the AND gate as an independent current source, according to Garg et al. [2008]:

$$I(q,t) = \frac{q}{\tau_\alpha - \tau_\beta} \times (e^{-\frac{t}{\tau_\alpha}} - e^{-\frac{t}{\tau_\beta}}). \tag{10}$$

An arbitrary number of cells are also generated and connected as the output loading for the AND gate. Capacitance of each cell will be normalized in terms of the unit-size inverter (NOT). The final output loading is obtained from summing up each output cell and represented by a total number of equivalent NOTs.

Given a fixed q, a number of MC runs with different SPICE settings are repeated in Figure 7 to compute the means and variances of pulse width and voltage magnitude, respectively, for the resulting transient fault. Figure 8(b) shows the table for the AND gate including four matrices: pulse-width mean matrix (M^{μ}_{pw}); pulse-width variance matrix (M^{σ}_{pw}); voltage-magnitude mean matrix (M^{μ}_{vm}); and voltage-magnitude variance matrix (M^{σ}_{vm}) to store mean and sigma values for pulse widths and voltage magnitudes of transient-fault propagation. Note that since first-strike transient faults are sensitive to input vectors, the input vector also serves as an index in T_{strike}.

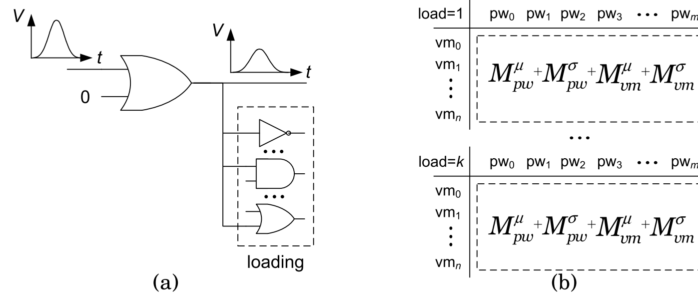

Fig. 9. Precharacterization of transient-fault-propagation table T_{prop} for an OR gate.

4.1.2 Transient-Fault Propagation Table T_{prop}. The transient-fault propagation table T_{prop}, on the other hand, reflects the changes of electrical properties when propagating the transient fault through one cell. Figure 9(a) shows the sample SPICE simulation environment to precharacterize the transient-fault propagation through one OR gate. The output loading is set up similarly to the pre-characterization of T_{strike} for the AND gate mentioned above. Both input and output of the OR gate are described as glitches and pulse widths and voltage magnitudes are measured accordingly.

After performing statistical calculation, four matrices, pulse-width mean (M^{μ}_{pw}), pulse-width sigma (M^{σ}_{pw}), voltage-magnitude mean (M^{μ}_{vm}) and voltage-magnitude sigma (M^{σ}_{vm}) can be obtained for *one* output loading in the tables. Furthermore, each T_{prop} has three dimensions where the first one is the output loading ($load$ = $1..k$), the second one is the input pulse width ($pw_0 ... pw_m$), and the third one is the input voltage magnitude ($vm_0 ... vm_n$). Therefore, the above process iterates k times to derive one T_{prop} of size $4 \times k \times m \times n$.

4.2 Sampling and Renewal of Transient Faults

Each Monte-Carlo (MC) run consists of two types of actions: *sampling* and *renewal*. A two-tuple transient fault f=(pw,vm) is first generated by *randomly choosing pw* and vm from pulse-width and voltage-magnitude distributions in T_{strike} according to probability theory. Later, electrical properties of f after propagating through the next cell are *renewed* and new pulse-width and voltage-magnitude distributions can be looked up from T_{prop} of such a cell. Then, a sampling step repeats to pick the next f'=(pw',vm'), followed by looking up the next (μ'_{pw},σ'_{ps}) and (μ'_{vm},σ'_{vm}) in the renewal step. The sampling and renewal steps alternate until the transient fault either reaches a flip-flop or disappears during propagation.

Transient-fault probability P(f) denotes the updated probability after f propagates through one gate and is also incorporated in the proposed table-lookup framework. Initially, all inputs are assumed to be independent variables with an equal probability of being 1 or 0. Probabilities for each node can be derived statically according to its input probabilities. Later, during computing the change of f on each cell, P(f) is updated simultaneously to reflect the logic masking effect mentioned in Section 3. Two different cases are discussed in detail, as follows.

4.2.1 First-Strike Cases. For the first-strike cases, the struck node is required to remain 0 for a positive transient fault and 1 for a negative transient fault. Let's take one AND gate shown in Figure 10(a), for example. Given the collection charge q, transient fault $f_z = (pw_z, vm_z)$ can be looked up in T_{strike} and denoted as $f_z = lut_{P.S.}(q, z)$. Assume that the probabilities of being 1 for inputs x and y are denoted by P_x and P_y. For the particle strikes, the output z to induce a positive transient fault (f^+_z), z is required to

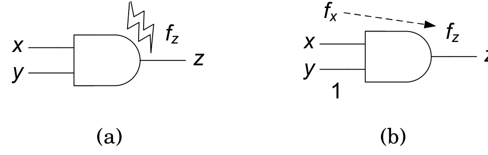

Fig. 10. Logical probability update for an AND gate.

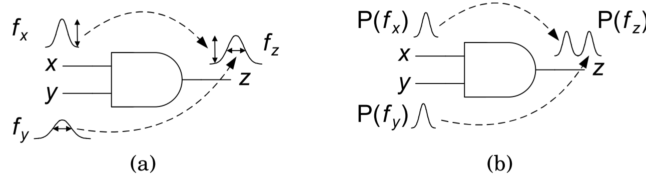

Fig. 11. Reconvergent transient faults on an AND gate.

be 0, and thus $P(f_z^+) = (1 - P_x) \times P_y$. Similarly, for a negative transient fault (f_z^-) striking output z, $P(f_z^-) = P_x \times P_y$.

4.2.2 Propagation Cases. For the propagation cases, in order to propagate the positive (or negative) transient faults through one gate, all other side-inputs are required to be noncontrolling values (n.c.v.). Besides, nonconvergent and convergent conditions need to be considered separately. Figure 10(b) illustrates a nonconvergent example of the AND gate. Similarly, given f_x, f_z can be looked up in T_{prop} by $f_z = lut_{prop}(f_x, z) = (pw_z, vm_z)$. As to transient-fault probability, since the nonconvergent condition assumes that only one positive (or negative) transient fault arrives, one input of the AND gate, say x in this example, only y is required to be 1 (n.c.v. for the AND gate). Therefore, $P(f_z) = P(f_x) \times P_y$.

A transient fault going through multiple propagation paths may reconverge to one node, which is very expensive to handle by using enumeration. Currently, the worst-case approximation is used, since it is reported to have only minor estimation error [Zhang et al. 2006, 2007]. From the electrical perspective, the worst case denotes a reconvergent transient fault that has the maximum pulse-width and voltage-magnitude among updated values from each input transient fault. An example for the AND gate is shown in Figure 11(a). Following this notion, a MAX operation is defined to facilitate such computation:

$$\begin{aligned} f_z &= \text{MAX}(lut_{prop}(f_x), lut_{prop}(f_y)) \\ &= \text{MAX}(lut_{prop}((pw_x, vm_x)), lut_{prop}((pw_y, vm_y))) \\ &= \text{MAX}((pw'_x, vm'_x), (pw'_y, vm'_y)) \\ &= (\text{MAX}(pw'_x, pw'_y), \text{MAX}(vm'_x, vm'_y)) \\ &= (pw_z, vm_z) \end{aligned} \tag{11}$$

where lut() looks up values from T_{prop} in the renewal step.

From the logical perspective, the worst case happens when the arrival windows of two transient pulses are not overlapped; Figure 11(b) illustrates this concept. The

corresponding transient-fault probability $P(f_z)$ can be computed by the summation of two transient-fault probabilities from inputs x and y. That is,

$$P(f_z) = P(f_x) + P(f_y). \tag{12}$$

5. A SUPPORT-VECTOR-REGRESSION (SVR) LEARNING FRAMEWORK

The table-lookup Monte-Carlo framework is inherently limited in execution efficiency because it computes δ_{strike} and δ_{prop} *indirectly* using extensive samplings of Monte-Carlo runs. In this section, we propose another *learning-based* framework to do the task *directly* with the support of vector regression (SVR), and is found to be both more efficient and more accurate. Note that our SVR-learning framework can be represented in the same flowchart as Figure 7, with the replacement of first-strike tables (T_{strike}) and propagation tables (T_{prop}) with the respective learning models (δ_{strike} and δ_{prop}).

By definition, δ_{strike} and δ_{prop} are functions of pw, which is a *random variable*. From Figures 3 and 4, we assume pw follows the normal distribution, which can be written as

$$pw \sim N(\mu_{pw}, \sigma_{pw}). \tag{13}$$

Therefore, we can decompose δ_{strike} and δ_{prop} into four models: δ^{μ}_{strike}, δ^{σ}_{strike}, δ^{μ}_{prop}, and δ^{σ}_{prop} where each can be defined as

$$\delta : \vec{x} \mapsto y \tag{14}$$

where $\vec{x}$ denotes a vector of *input variables* and y is called the model's *label* or *target value*.

To integrate the impact of process variations, four models are traditionally built using lookup tables. However, lookup tables have two limitations on applicability: (1) inaccurate interpolation and (2) coarse model-size control. First, lookup tables can take only finite table indices and must use interpolation. However, interpolation functions are often not accurate enough or difficult to obtain, especially as the table dimensionality grows. Second, a lookup table stores data samples in a grid-like fashion, where the table will grow prohibitively large for fine resolution. Meanwhile, the *information richness* often differs across different parts of a table. For example, we observe that pulse widths generated by strong charges behave much more simply than weaker charges do. Naturally, simple behaviors can be encoded with fewer data points in the model, whereas complicated behaviors need to be encoded with more.

In statistical learning theory, such models are built using regression, which can be roughly divided into linear [Weisberg 2005] and nonlinear [Bates and Watts 1988] methods. Among them, Support Vector Regression (SVR) [Smola et al. 2003; Vapnik 1995] combines linear methods' efficiency and nonlinear methods' descriptive power. SVR has two advantages over lookup tables: (1) it gives an explicit function and needs no interpolation; (2) it filters out unnecessary points and yields compact models.

In the following, we propose a methodology to adapt the framework in Section 4 to a learning-based one based on SVR models, which comprises training sample preparation, SVR model training, and parameter selection. Also, the modification of the MAX operation in Eq. (11) is addressed.

5.1 Training Sample Preparation

SVR models differ from lookup tables in the way we prepare training samples for them. For lookup tables, we start by selecting a finite set of points along each table dimension. On one hand, they should be chosen economically; on the other hand, it is difficult to cover all corner cases with only a limited number of points. For SVR models, we do not

need to select these points. Instead, we provide large sets of *training samples*, and let the SVR algorithm do the selection task.

A training sample set S of m samples is defined as

$$S \subset (\vec{X} \times Y)^m = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_m, y_m)\} \tag{15}$$

where m pairs of input variables $\vec{x}_i$s and target values y_is are obtained from massive Monte-Carlo SPICE simulation. For δ^{μ}_{strike}, δ^{σ}_{strike}, we use input variables including charge strength, driving gate, input pattern, and output loading; for δ^{μ}_{prop}, δ^{σ}_{prop}, we use input variables including input pattern, pin index, driving gate, input pulse-width distribution (μ^{i-1}_{pw} and σ^{i-1}_{pw}), propagation depth, and output loading.

In our training samples, we implement output loading using combinations of arbitrary cell input pins. Doing so preserves additional information for the output loading status and saves the labor (and risk) of characterizing the capacity of each cell's input pin. Although the number of such combinations can easily explode, there are usually only a limited number of representatives, which are automatically identified by SVR. Furthermore, from a learning perspective, since both peak voltage and pulse width are the responses of charge injection current formulated in Eq. (10), they are highly correlated. Empirically, using pulse-width information alone can yield satisfactory SSERs, and thus in our framework, we do not need to incorporate models for peak voltage.

5.2 Support Vector Machine and Its Extension to Regression

Support vector machine (SVM) is one of the most widely used algorithms for learning problems [Vapnik 1995], and can be summarized with the following characteristics.

- SVM is an efficient algorithm and finds a global minimum (or maximum) for a convex optimization problem formulated from the learning problem.
- SVM avoids the curse of dimensionality by capacity control and works well with high-dimensional data.
- SVM automatically finds the decision boundary for a collection of samples using a small subset where each sample is called a *support vector*.

The basic idea behind SVM is to find a function as the decision boundary with minimal errors and a maximal margin to separate data in multidimensional space. Given a training set $\mathbf{S}$, with $\vec{x}_i \in \mathrm{R}^n$, $y_i \in \mathrm{R}$, the SVM learning problem is to find a function f (first assume $y = f(\vec{x}) = \langle \vec{w} \cdot \vec{x} \rangle + b$) that models $\mathbf{S}$ properly. Accordingly, the learning task is formulated into a constrained optimization problem, as follows:

$$\begin{aligned} &\text{minimize} \quad \|\vec{w}\|^2 + C(\textstyle\sum_{i=1}^{m} \xi_i)^k \\ &\text{subject to} \quad \begin{cases} y_i(\langle \vec{w} \cdot \vec{x}_i \rangle + b) \geq 1 - \xi_i,\ i = 1, \ldots, m, \\ \xi_i \geq 0,\ i = 1, \ldots, m, \end{cases} \end{aligned} \tag{16}$$

ξ_i is a slack variable providing an estimate of the error induced by the current decision boundary; C and k are user-specified parameters indicating the penalty of function errors in control. Later, the *Lagrange multiplier* method can efficiently solve such a constrained optimization problem [Vapnik 1995] and finds $\vec{w}$ and b for $f(\vec{x}) = \langle \vec{w} \cdot \vec{x} \rangle + b$ with a maximal margin $2/|\vec{w}|$ between $\langle \vec{w} \cdot \vec{x} \rangle + b = +1$ and $\langle \vec{w} \cdot \vec{x} \rangle + b = -1$. Figure 12 shows an example for a two-dimensional data set containing samples of two different classes. Figure 12(a) illustrates many possible decision boundaries to separate the data set, whereas Figure 12(b) shows the one with the maximal margin and the minimal errors that the user can tolerate among all boundaries.

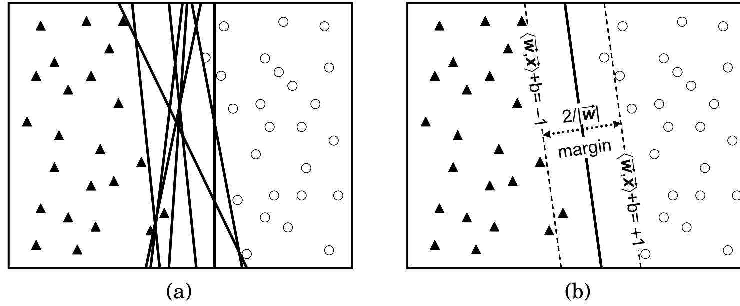

Fig. 12. Linear decision boundaries for a two-class data set.

One SVM algorithm can be applied to regression problems via three steps: (1) *primal form* optimization; (2) *dual form* expansion; and (3) *kernel function* substitution. The primal form presents the nature of the regression, whereas the dual form provides the key to the later nonlinear extension using kernel functions. In our framework, ϵ-*SVR* [Vapnik 1995] is implemented to realize a family of highly nonlinear regression models $f(\vec{x}) : \vec{x} \mapsto y$ for δ^{μ}_{strike}, δ^{σ}_{strike}, δ^{μ}_{prop}, and δ^{σ}_{prop} for pulse-width mean and sigma of first-strike functions and pulse-width mean and sigma of propagation functions, respectively.

5.2.1 Primal Form Optimization. The regression's goal is to derive a function that minimizes slacks and meanwhile to make f as smooth as possible. The corresponding constrained optimization problem for ϵ-*SVR* is modified as follows:

$$\begin{aligned} &\text{minimize} \quad \|\vec{w}\|^2 + C\textstyle\sum_{i=1}^{m}(\xi_i^2 + \hat{\xi}_i^{\,2}) \\ &\text{subject to} \quad \begin{cases} (\langle \vec{w} \cdot \vec{x}_i \rangle + b) - y_i \le \epsilon + \xi_i,\ i = 1, \ldots, m, \\ y_i - (\langle \vec{w} \cdot \vec{x}_i \rangle + b) \le \epsilon + \hat{\xi}_i,\ i = 1, \ldots, m, \\ \xi_i, \hat{\xi}_i \ge 0,\ i = 1, \ldots, m, \end{cases} \end{aligned} \tag{17}$$

where the two slack variables ξ_i and $\hat{\xi}_i$ represent, respectively, variations of the error exceeding and being below the target value by more than ϵ. The parameter C determines the tradeoff between the smoothness of $f(\vec{x}_i)$ and the variation in the number of errors (ξ_i and $\hat{\xi}_i$) to be tolerated. Equation (17) is termed the regression's *primal form*.

5.2.2 Dual Form Expansion. Instead of finding $\vec{w}$ directly, the *Lagrange multiplier* method transforms the optimization problem from the primal form to its dual form and derives f as

$$f(\vec{x}) = \sum_{i=1}^{m}(\alpha_i - \alpha_i^*)\langle \vec{x} \cdot \vec{x}_i \rangle + b\,, \tag{18}$$

where α_i, α_i^* are Lagrange multipliers and b is a function of ϵ, C, αs and α^*s [Smola et al. 2003].

Several findings can be inferred from Eq. (18). First, the only inner product $\langle \vec{x} \cdot \vec{x}_i \rangle$ implies that only an *unseen* sample $\vec{x}$ and a *training* sample $\vec{x}_i$ are sufficient to predict a new unseen target value y. Second, only training samples $\vec{x}_i$s that correspond to nonzero $(\alpha_i - \alpha_i^*)$s contribute to the prediction outcome. All other samples are unnecessary for the model and are filtered out during the training process. Third, the inner product operation is a form of linear combination. As a result, the predicted target

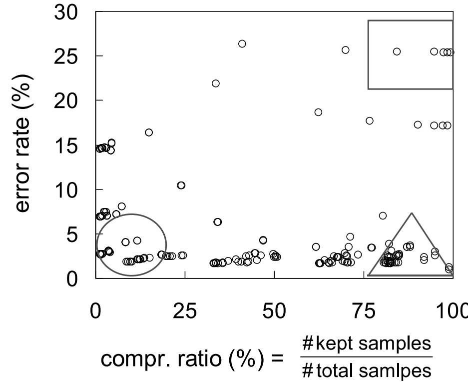

Fig. 13. Quality comparison of 200 models using different parameter combinations.

values of such a model are all linear combinations of training samples, and thus f is a *linear* model. In practice, SVR often keeps only few samples (i.e., $\vec{x}_i$s with nonzero coefficients) in its models, and thus benefits from both smaller model size and faster prediction efficiency.

5.2.3 Kernel Function Substitution. According to the statistical learning theory [Vapnik 1995], SVM remains valid if the inner product operation $\langle \vec{u} \cdot \vec{v} \rangle$ in Eq. (18) is substituted by a *kernel* function $K(\vec{u}, \vec{v})$ [Cristianini and Shawe-Taylor 2002]. That is,

$$f(\vec{x}) = \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) K(\vec{x}, \vec{x}_i) + b\,. \tag{19}$$

Radial Basis Function (RBF) is one kernel function used in our framework, and can be formulated as $K(\vec{u}, \vec{v}) = e^{-\gamma \cdot \|\vec{u} - \vec{v}\|^2}$ where γ is a controlling parameter. Unlike the inner product operation, the RBF kernel is highly nonlinear. This enables the SVM algorithm to produce families of nonlinear models that are suitable to capture complicated behaviors, like that of generation and propagation of pulse-width distributions of transient faults.

5.3 Parameter Search

Now we return to the issue of selecting parameters (ϵ, C, γ) that have an unbounded number of combinations and is critical for achieving fine model quality. Figure 13 illustrates 200 models built from the same training sample set; each point represents one model using a distinct parameter combination. Their quality is measured along two coordinates: the Y-axis denotes the error rate for prediction; the X-axis denotes the sample compression ratio—the ratio between the number of samples kept by the model and the original size of S. Figure 13 shows that while it is possible to obtain an ideal model that is small and accurate (indicated by the circle), it is also possible to obtain a large and inaccurate model (indicated by the square). The differences are 20X in both axes, and there is so far no deterministic method to find the best combination.

Since exhaustive search is clearly impractical, we need an efficient searching process with an effective cost function, which is written as

$$(\hat{\epsilon}, \hat{C}, \hat{\gamma}) = \underset{(\epsilon, C, \gamma)}{\arg} \min(E^k R). \tag{20}$$

In Eq. (20), $E^k R$ denotes the cost function, where E and R, respectively, denote error rate and compression ratio, and k is a parameter controlling the tradeoff between E

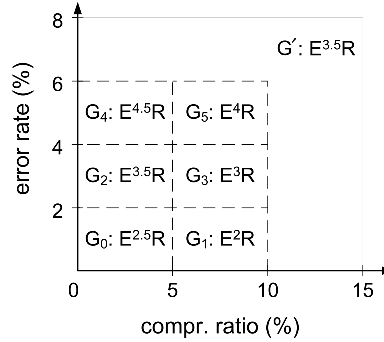

Fig. 14. Prioritized scheme for parameter search.

and R. A larger k makes the cost function more sensitive to the error rate, and vice versa. Note that if a single k is used, the cost function may wrongly select a combination with one matrix being extremely low and the other being undesirably high (e.g,. $E = 0.01\%$ and $R = 0.99\%$, as indicated by the triangle in Figure 13).

Therefore we applied a prioritized scheme according to predefined goals on both matrices, as illustrated in Figure 14. Assuming the goal ($E < 6\%, R < 10\%$), we draw finite grids near these goals, and prioritize them accordingly. For example, G_0 is preferred over G_1, G_1 is preferred over G_2, and $G_0 \sim G_5$ are preferred over G'. The main idea is that in grids where E is small or R is large, the cost function is adjusted to be more insensitive to error. Therefore, k is assigned smaller in grids with a lower error rate or larger compression ratio, as illustrated in Figure 14. In G', $k = 3.5$ generally works well.

After determining the cost function, exhaustive search may still take months. To speed up the searching process, we observe two helpful properties from samples of all our four types of models. First, a sufficiently large (> 500) sample subset shares similar behaviors as the complete sample set. Second, points forming a cluster in Figure 13 have similar parameter combinations. For example, the combination (ϵ, C, γ) of points within the circle has a range of $[2^{-4}, 2^{-6}]$ on ϵ, $[2^4, 2^8]$ on C, and $[2^{-2}, 2^{-6}]$ on γ. The first property enables the use of subset search; the second property allows for incremental search with granularity.

Parameter search is critical for building SVR models. Using the prioritized cost function, we can systematically find a good parameter combination. Further, using subset search and incremental search with granularity, the time consumed by parameter search is reduced to about half an hour.

6. EXPERIMENTAL RESULTS

In this section, all experiments are divided into two parts. The first part is to examine the accuracy of the precharacterized lookup tables and learned models. In the second part, they are then integrated into their respective statistical SER analysis frameworks and are compared in their SSER analysis accuracy and runtime.

6.1 Table and Model Accuracy

We use a unified framework to generate test samples for the precharacterized tables and learning models, considering process-variation impacts. For simplicity, we only consider the with-in die geometric process variation; other types of variation are not included in our work. We perturb *gate width* and *channel length* of each transistor in geometry, used to model with-in die variation, since they are the dominant factors for

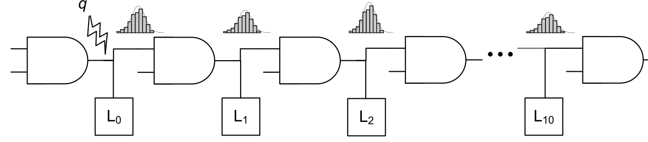

Fig. 15. The framework for test sample generation.

Table I. Summary of Table Error Rates

(a) V_m table error rate

Cell	error rate (%) T^{μ}_{strike}	T^{σ}_{strike}	T^{μ}_{prop}	T^{σ}_{prop}
NOT	3.0	2.2	10.7	22.8
AND	2.8	1.6	11.2	22.6
OR	2.9	1.6	11.6	23.7

(b) pw table error rate

Cell	error rate (%) T^{μ}_{strike}	T^{σ}_{strike}	T^{μ}_{prop}	T^{σ}_{prop}
NOT	0.4	0.2	4.4	5.3
AND	0.3	0.2	4.2	6.2
OR	0.4	0.2	4.3	7.4

gate delay [Choi et al. 2004; Salzmann et al. 2007; Weste and Harris 2005]. Note that, the other important random variation, threshold-voltage (V_{th}) fluctuation, can also be reflected indirectly by considering the process variation. However, more variation sources can be considered as long as their impacts can be reflected onto SPICE simulation results.

As illustrated in Figure 15, the framework first generates a path consisting of a random number of cells, which are connected to additional random cells as loadings. Using Monte-Carlo spice simulation, the transient-fault distributions are recorded along the path, which are later collected as test samples. The training samples for the learning models are also generated in the same manner.

6.1.1 Precharacterized Tables. We build a series of precharacterized tables for pulse width pw and voltage magnitude V_m with a total size of 9.5MB, using about 1 month for data preparation and $<$ 1 second for table construction, according to Section 4.1. Used with samplings and renewals, the tables are later verified with 10K test samples, where the results are presented and categorized according to cell and table types in Table I. For pw tables, the results are better: the error rates for T^{μ}_{strike} and T^{σ}_{strike} are within 0.5%, whereas the error rates for T^{μ}_{prop} and T^{σ}_{prop} are up to 7.4%. For V_m tables, however, the error rates for T^{μ}_{strike} and T^{σ}_{strike} are around 3.0%, whereas the error rates for T^{μ}_{prop} and T^{σ}_{prop} are up to 23.7%. Note that in the table-lookup framework, V_m is also one of the indexing variables of pw tables. Therefore, when used in this framework, the V_m tables' higher error rates will also affect the lookup of pw tables.

6.1.2 SVR Models. We also build the SVR models for three cells with four charge-strength levels. Assuming a 5% process-variation, each model is trained with 10K training samples. Then, we examine these models' accuracy and compression ratios using another 10K test sample.

The mean error rates and compression ratios are first categorized according to model and cell types in Table II. Three messages are observed. (1) All mean error rates and compression ratios of δ^{μ}_{strike}, δ^{μ}_{prop}, and δ^{σ}_{prop} models are below 4% and 4.5%, respectively. Hence, we found these models accurate and compact. (2) δ^{σ}_{strike} models have error rates and compression ratios around 13% and 0.4%, respectively. This type of model is less accurate and smaller, which means the behavior of δ^{σ}_{strike} may not be fully explained by its current input variables. (3) Among different cells, `NOT` has the largest mean compression ratio, whereas `OR` has the smallest. It means that `NOT` models generally have a more complex behavior than `OR` models.

Table II. Model Quality w.r.t. Model Type

error rate (%)				
Cell	δ^{μ}_{strike}	δ^{σ}_{strike}	δ^{μ}_{prop}	δ^{σ}_{prop}
NOT	2.0	12.9	3.7	3.8
AND	2.8	12.0	2.4	3.9
OR	2.6	11.9	3.3	3.7

compression ratio (%)				
Cell	δ^{μ}_{strike}	δ^{σ}_{strike}	δ^{μ}_{prop}	δ^{σ}_{prop}
NOT	2.7	0.4	4.4	1.2
AND	2.4	0.3	1.1	0.9
OR	1.4	0.3	0.4	1.2

training time (sec.)				
Cell	δ^{μ}_{strike}	δ^{σ}_{strike}	δ^{μ}_{prop}	δ^{σ}_{prop}
NOT	273.0	1293.3	706.5	310.5
AND	1329.7	1373.3	1057.4	314.3
OR	1478.5	1341.3	1149.6	286.3

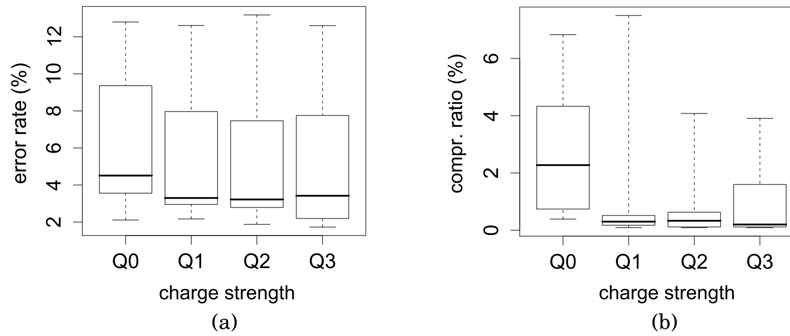

Fig. 16. Model quality w.r.t. charge strength.

In Figure 16, we further categorize the models' error rates and compression ratios according to charge strength. This time, we do not use mean values, as they are known to be easily biased toward extreme values. Instead, we use box-plots that mark the data's minimum, first quartile, median, third quartile, and maximum. For the error rate (shown on the left), while there are extreme values in each category, models of Q_0 in general have a slightly greater error rate (median = 4.5%) compared to Q_1 (median = 3.3%), Q_2 (median = 3.2%), and Q_3 (median = 3.3%). For the compression ratio (displayed on the right), maximum values likewise, exist in all categories. However, models of Q_0 demonstrate 7X$\sim$10X overall compression ratio (median = 2.2%) compared to Q_1 (median = 0.3%), Q_2 (median = 0.3%), and Q_3 (median = 0.2%). Equivalently, a model in the Q_0 category generally needs 220 out of 10k samples to encode its behavior, while 30, 20, and 30 samples, respectively, are needed in the Q_1, Q_2, and Q_3 categories.

These observations show that the pulse-width distribution caused by *weak charges* exhibits more complex behavior than the distribution caused by *strong charges*. This can be explained by the fact that the transient fault caused by weak charges does not reach V_{dd}, and thus is more unpredictable than that caused by strong charges. However, since weak charges strongly dominate the majority of particle hits due to

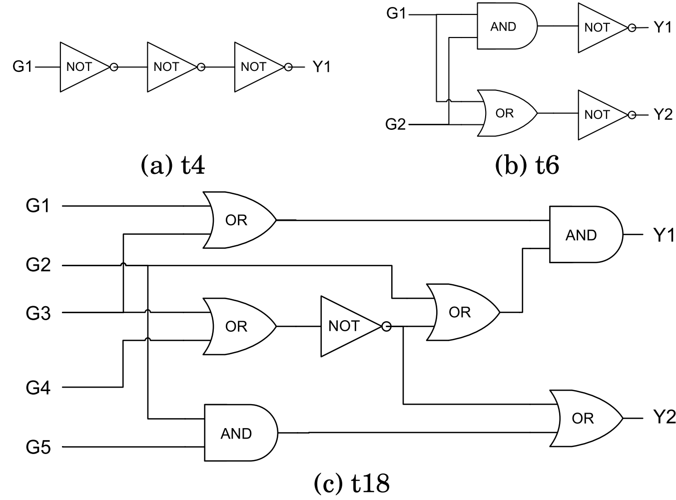

Fig. 17. Three small circuits in our experiment.

their exponential distribution in Eq. (5), accurately predicting their behavior becomes the key to the success of accurate SER estimation.

6.2 SER Computation

The table-lookup Monte-Carlo and the SVR-learning frameworks are built and exercised on a Linux machine with a Pentium Core Duo (2.4GHz) processor and 4GB RAM. We use the the 45nm Predictive Technology Model (PTM) [Nanoscale Integration and Modeling Group 2008] and set the corresponding charge collection slope Q_s as $10.84fC$ according to Semiconductor Roadmap Committee of Japan [2003]. The neuron flux rate is set to $F = 56.5m^{-2}s^{-1}$ at sea-level [Dodd and Massengill 2003]. In Eq. (2), μ_w and σ_w are set to $100ps$ and $10ps$, respectively [Nangate Inc. 2008]. For all circuits, each node under every input pattern combination is injected with four levels of electrical charges: $Q_0 = 34fC$, $Q_1 = 66fC$, $Q_2 = 99fC$ and $Q_3 = 132fC$, where $32fC$ is observed to be the weakest charge capable of generating a transient fault with positive pulse width under our settings. Note that for simplicity, as in Zhang et al. [2006] and Rao et al. [2006], four levels of charges and only one kind of soft error, the 0-to-1 error, are considered for computing SSERs. More levels of charges and a 1-to-0 error can be injected in both of our frameworks to better approximate the total SER in Eq. (4) if time permits.

Both static and Monte-Carlo SPICE simulation are used for SER accuracy evaluation over four benchmark circuits (t4, t6, t18, as shown in Figure 17, and c17 from ISCAS 85). In the static setting, we use static SPICE simulation and calculate the error-latching probability according to Eq. (1). In the statistical setting, we use 100 Monte-Carlo SPICE simulation runs and calculate the error-latching probability according to Eq. (2). The pulse width of the arrival transient fault is then measured at all primary outputs under all input pattern combinations.

Because of the extremely long runtime of Monte-Carlo SPICE simulation, we are only able to perform tests on small circuits, with the largest containing 7 gates, 12 vulnerable nodes, and 5 inputs. The runtime of the Monte-Carlo SPICE simulation ranges from 8 hours to slightly more than one day. The runtime of the table-lookup framework ranges from 0.19 to 0.71 seconds with the average 10^5X speedup. The

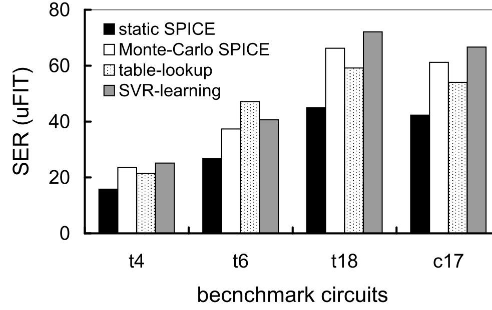

Fig. 18. Soft error rate comparison between static SPICE simulation, Monte-Carlo SPICE simulation, and the proposed frameworks.

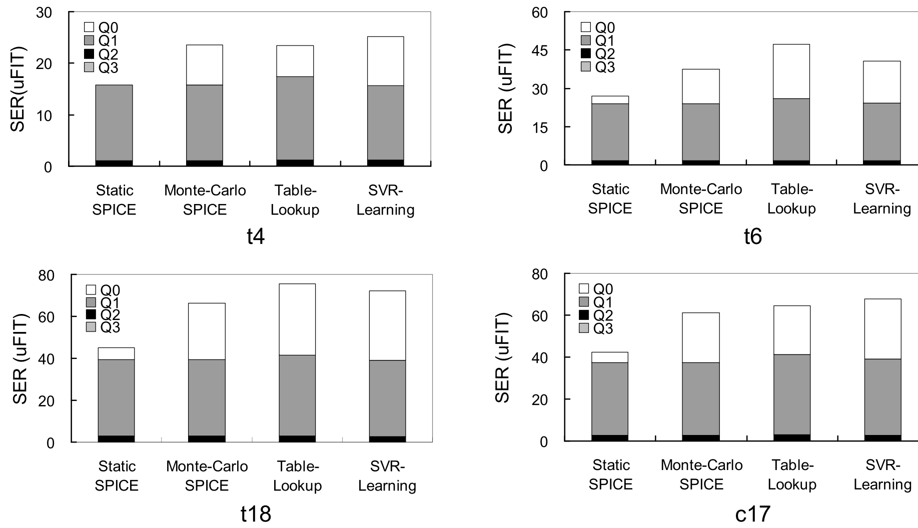

Fig. 19. SER breakdown by charge strength.

SVR-learning framework runs even faster ($<$ 0.01 second on the four cases), and the average speedup is of the order of 10^7X.

Figure 18 compares the SER analysis results where three facts are observed: (1) under 5% process-variation, the SER obtained by Monte-Carlo SPICE simulation are 35% $\sim$ 52% above that obtained by static SPICE analysis. Since the process variations worsen the stability of circuits beyond the deep submicron regime, statistical analysis methods should be used to avoid increasingly underestimated circuit SER. (2) The table-lookup framework underestimates t4, t18, and c17, but overestimates t6 meanwhile, with the maximum error difference being 26.27%. (3) The SVR-learning framework yields SER's slightly above the result using Monte-Carlo SPICE simulation and the maximum difference is $<$ 9.0%.

To more closely investigate the SER difference between static and statistical analyses, we breakdown the results in Figure 18 by charge strength levels, and present the results in Figure 19. Comparing the results between static and Monte-Carlo SPICE simulations across all test circuits, it is observed that the results of the two SPICE simulations and the two proposed frameworks are very similar for $Q_1 \sim Q_3$ parts (difference $<$ 5%). For the Q_0 part, indicated by the white bars, however, the static SPICE simulation constantly underestimate the SER. The table-lookup framework performs

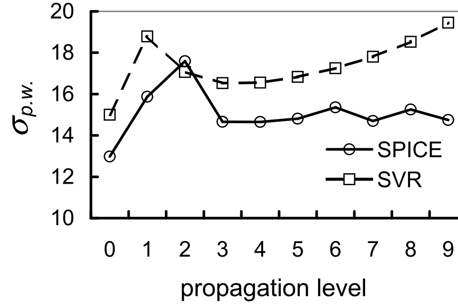

Fig. 20. σ_{pw} propagation along a path.

Table III. Benchmark Circuits, SER and Runtime from Table-Lookup Monte-Carlo and SVR-Learning Frameworks

circuit	N_{node}	N_{po}	N_{pred}(k)	table-lookup (tbl) SER(FIT)	table-lookup (tbl) time(s)	SVR-learning (svr) SER(FIT)	SVR-learning (svr) time(s)	tbl/svr spdup(X)
t4	4	1	0.1	2.18E-05	0.2	2.52E-05	< 0.01	>20.0
t6	6	2	0.2	4.73E-05	0.3	4.06E-05	< 0.01	>30.0
t18	12	3	0.4	6.05E-05	0.7	7.21E-05	< 0.01	>70.0
c17	12	3	0.5	5.31E-05	0.7	6.66E-05	< 0.01	>70.0
c432	233	7	362.5	1.25E-04	114.4	1.48E-04	5.9	19.4
c499	638	32	2939.3	1.15E-04	870.6	1.59E-04	42.9	20.3
c880	443	26	402.5	1.52E-04	173.2	2.18E-04	6.1	28.4
c1355	629	32	3013.2	1.19E-04	891.8	1.36E-04	43.5	20.5
c1908	425	25	1240.3	2.12E-04	365.1	2.27E-04	18.4	19.8
c2670	841	157	570.8	3.48E-04	401.0	3.40E-04	9.6	41.8
c3540	901	22	3142.4	7.41E-04	1070.6	6.67E-04	39.8	26.9
c5315	1806	123	2272.2	1.15E-03	818.2	1.09E-03	35.1	23.3
c6288	2788	32	43776.4	6.86E-04	15703.1	8.45E-04	501.5	31.3
c7552	2114	126	3704.8	1.04E-03	1406.7	8.89E-04	97.4	14.4
mul_4	158	8	145.4	1.58E-04	98.8	1.79E-04	2.4	60.6
mul_8	728	16	2960.3	4.14E-04	710.2	6.06E-04	45.1	15.7
mul_16	3156	32	52348.1	1.48E-03	9565.0	1.47E-03	784.7	12.2
mul_24	7234	48	273008.7	2.63E-03	39628.5	2.35E-03	3553.2	11.2
mul_32	13017	64	890360.5	2.82E-03	131535.6	3.21E-03	11142.1	11.8
							Average	28.8

better than the static SPICE simulation but worse than the SVR-learning framework. Overall, the SVR-learning framework can give a slightly larger but closer (and more stable) results as Monte-Carlo SPICE simulation.

To further investigate the 9% SER over-estimation of the SVR learning framework, one transient fault along a path is specifically identified in Figure 20. The X-axis and Y-axis denote the propagation level and the standard deviation of the pulse width (σ_{pw}) of this transient fault, respectively. After two propagations, the σ_{pw} drops sharply and has not yet been fully captured by the current learning model. This behavior does not seem to correlate to any of our existing input variables, and caused larger SER estimations according to Eq. (2) and Figure 4. Such an issue will be another topic worth exploring.

Since running Monte-Carlo SPICE simulation with process variations for large circuits will be prohibitively time-consuming, we can only run both frameworks on large benchmark circuits. However, the entire set of circuits includes the ones used in Figure 18, ISCAS 85 benchmark circuits [Brglez and Fujiwara 1985], and a series of multipliers. Table III first lists the name, the total number of nodes,the total number of outputs, and the total number of predictions for each circuit. The latter columns in the table report the SER and runtime from both the table-based and the SVR-learning frameworks. Accordingly, SER is clearly proportional to the number of nodes and primary outputs of a circuit, which correspond, respectively, to the possibility of the circuit struck by radiation particles and the possibility of the resulting transient faults observed in primary outputs. The runtime, however, does not only depend on the number of strike nodes, but also depend on the number of convolutions between these nodes. For example, $c3540$ (an ALU with control) has fewer nodes than $c5315$ (another ALU), whereas its runtime is larger. This property is also observed from the large runtime of multiplier circuits, in which every primary output depends on each primary input. Finally, the SVR-learning framework runs faster than the table-based framework by 11.8X-70.0X, with an average of 28.8X.

7. CONCLUSIONS

Traditional SER analysis techniques try to mimic the results of static SPICE simulation. However, static analysis tends to increasingly underestimate true SERs in the presence of process variations. In this article, we first examine the soft-error effect beyond deep submicron technologies considering process variations. From the statistical point of view, we found that transient faults are not always diminishing in pulse width after propagation, and may even become larger when reaching flip-flops. We also showed that soft errors originating from particle strikes with small charges can easily escape from the traditional static analysis.

To cope with these sophisticated issues, a table-lookup Monte-Carlo framework and a SVR-learning framework are proposed, respectively. The first framework captures the change of transient-fault distributions implicitly, using the Monte-Carlo method, whereas the second does the same task explicitly, using support vector regression. Experimental results show that both frameworks are capable of more accurately estimating SERs when comparing to the static SPICE simulation. Moreover, the SVR learning framework outperforms the table-lookup framework in terms of both SER accuracy and runtime.

Statistical soft error rate (SSER) is an emerging topic. As the IC technology keeps evolving beyond deep submicron, we envision SSER analysis becoming increasingly critical for reliable scaled designs. A few future directions of SSER research include (1) deriving more accurate learning models for σ_{pw}; (2) developing a faster Monte-Carlo framework with high accuracy; and (3) applying statistical circuit optimization.

REFERENCES

AMUSAN, O. A., MASSENGILL, L. W., BHUVA, B. L., DASGUPTA, S., WITULSKI, A. F., AND AHLBIN, J. R. 2007. Design techniques to reduce set pulse widths in deep-submicron combinational logic. *IEEE Trans. Nuclear Sci. 54*, 6, 2060–2064.

BARTLETT, W. AND SPAINHOWER, L. 2004. Commercial fault tolerance: A tale of two systems. *IEEE Trans. Depend. Secure Comput. 1,* 1, 87–96.

BATES, D. M. AND WATTS, D. G. 1988. *Nonlinear Regression Analysis and Its Applications.* Wiley.

BORKAR, S., KARNIK, T., NARENDRA, S., TSCHANZ, J., KESHAVARZI, A., AND DE, V. 2003. Parameter variations and impact on circuits and microarchitecture. In *Proceedings of the Design Automation Conference.* 338–342.

BOWMAN, K., DUVALL, S., AND MEINDL, J. 2002. Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. *IEEE J. Solid-State Circuits 37*, 2, 183–190.

BRGLEZ, F. AND FUJIWARA, H. 1985. A neural netlist of ten combinational benchmark circuits and translator in Fortran. In *Proceedings of the International Symposium on Circuits and Systems*.

CHA, H. AND PATEL, J. H. 1993. A logic-level model for particle hits in CMOS circuits. In *Proceedings of the International Conference on Circuit Design*. 538–542.

CHOI, S. H., PAUL, B. C., AND ROY, K. 2004. Novel sizing algorithm for yield improvement under process variation in nanometer technology. In *Proceedings of the Design Automation Conference*. 454–459.

CRISTIANINI, N. AND SHAWE-TAYLOR, J. 2002. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

DODD, P. E. AND MASSENGILL, L. W. 2003. Basic mechanisms and modeling of single-event upset in digital microelectronics. *IEEE Trans. Nuclear Sci. 50,* 3, 583–602.

EDAMATSU, H., HOMMA, K., KAKIMOTO, M., KOIKE, Y., AND TABUCHI, K. 1998. Pre-layout delay calculation specification for CMOS ASIC libraries. In *Proceedings of the Asian South Pacific Design Automation Conference (ASP-DAC)*. 241–248.

FERLET-CAVROIS, V., PAILLET, P., MCMORROW, D., FEL, N., BAGGIO, J., GIRARD, S., DUHAMEL, O., MELINGER, J. S., GAILLARDIN, M., SCHWANK, J. R., DODD, P. E., SHANEYFELT, M. R., AND FELIX, J. A. 2007. New insights into single event transient propagation in chains of inverters-evidence for propagation-induced pulse broadening. *IEEE Trans. Nuclear Sci. 54,* 6, 2338–2346.

GARG, R., NAGPAL, C., AND KHATRI, S. P. 2008. A fast, analytical estimator for the seu-induced pulse width in combinational designs. In *Proceedings of the Design Automation Conference.* 918–923.

KRISHNASWAMY, S., MARKOV, I., AND HAYES, J. P. 2008. On the role of timing masking in reliable logic circuit design. In *Proceedings of the Design Automation Conference.* 924–929.

MISKOV-ZIVANOV, N. AND MARCULESCU, D. 2006. Mars-c: Modeling and reduction of soft errors in combinational circuits. In *Proceedings of the Design Automation Conference.* 767–772.

MISKOV-ZIVANOV, N., WU, K.-C., AND MARCULESCU, D. 2008. Process variability-aware transient fault modeling and analysis. In *Proceedings of the International Conference on Computer Aided Design*. 685–690.

MITRA, S., SEIFERT, N., ZHANG, M., SHI, Q., AND KIM, K. S. 2005. Robust system design with built-in soft error resilience. *IEEE Trans. Computer 38,* 2, 43–52.

MOHANRAM, K. 2005. Closed-form simulation and robustness models for seu-tolerant design. In *Proceedings of the VLSI Test Symposium.* 327–333.

MUKHERJEE, S., KONTZ, M., AND REIHARDT, S. 2002. Detailed design and evaluation of redundant multithreading alternatives. In *Proceedings of the International Symposium on Computer Architecture*. 99–110.

NANGATE INC. 2008. Nangate 45nm Open Library. http://www.nangate.com/.

NANOSCALE INTEGRATION AND MODELING GROUP. 2008. *Predictive Technology Model*. Nanoscale Integration and Modeling Group. http://www.eas.asu.edu/ ptm/.

NATARAJAN, S., BREUER, M., AND GUPTA, S. 1998. Process variations and their impact on circuit operation. In *Proceedings of the International Symposium on Defect and Fault Tolerance in VLSI Systems*. 73–81.

OMANA, M., PAPASSO, G., ROSSI, D., AND METRA, C. 2003. A model for transient fault propagation in combinational logic. In *Proceedings of the International On-Line Testing Symposium.* 111–115.

RAJARAMAN, R., KIM, J. S., VIJAYKRISHNAN, N., XIE, Y., AND IRWIN, M. J. 2006. Seat-la: A soft error analysis tool for combinational logic. In *Proceedings of the International Conference on VLSI Design*. 499–502.

RAMAKRISHNAN, K., RAJARAMAN, R., SURESH, S., VIJAYKRISHNAN, N., XIE, Y., AND IRWIN, M. J. 2007. Variation impact on ser of combinational circuits. In *Proceedings of the International Symposium on Quality Electronic Design*. 911–916.

RAO, R., CHOPRA, K., BLAAUW, D., AND SYLVESTER, D. 2006. An efficient static algorithm for computing the soft error rates of combinational circuits. In *Proceedings of the Design Automation and Test in Europe Conference.* 164–169.

SALZMANN, J., SILL, F., AND TIMMERMANN, D. 2007. Algorithm for fast statistical timing analysis. In *Proceedings of the International Symposium on System-on-Chip*. 1–4.

SEMICONDUCTOR ROADMAP COMMITTEE OF JAPAN. 2003. Parameters of low power SoC design. http://strj-jeita.elisasp.net/pdf-nenjihou_koku-0303-roadmap/3-13_setsukeitask_force.pdf.

SHIVAKUMAR, P., KISTLER, M., KECKLER, S. W., BURGER, D., AND ALVISI, L. 2002. Modeling the effect of technology trends on the soft error rate of combinational logic. In *Proceedings of the International Conference on Dependable Systems and Networks*. 389–398.

SMOLA, A. J., SCHOLKOPF, B., AND OLKOPF, B. S. 2003. A tutorial on support vector regression. Tech. rep., Statistics and Computing.

TOSAKA, Y., HANATA, H., ITAKURA, T., AND SATOH, S. 1999. Simulation technologies for cosmic ray neutron-induced soft errors: Models and simulation systems. *IEEE Trans. Nuclear Sci. 46,* 3, 774–780.

VAPNIK, V. N. 1995. *The Nature of Statistical Learning Theory*. Springer, Berlin.

WEISBERG, S. 2005. *Applied Linear Regression* 3rd Ed., Wiley.

WESTE, N. H. E. AND HARRIS, D. 2005. *CMOS VLSI Design - A Circuit and Systems Perspective* 3rd Ed., Addison Wesley, Boston.

ZHANG, B., WANG, W.-S., AND ORSHANSKY, M. 2006. Faser: Fast analysis of soft error susceptibility for cell-based designs. In *Proceedings of the International Symposium on Quality Electronic Design*. 755–760.

ZHANG, M. AND SHANBHAG, N. 2004. A soft error rate analysis (sera) methodology. In *Proceedings of the International Conference on Computer Aided Design*. 111–118.

ZHANG, M., MAK, T., TSCHANZ, J., KIM, K., SEIFERT, N., AND LU, D. 2007. Design for resilience to soft errors and variations. In *Proceedings of the International On-Line Test Symposium.* 23–28.

Received June 2010; revised May 2011; accepted August 2011