Tracking Uncertainty with Probabilistic Logic Circuit Testing

Smita Krishnaswamy, Igor L. Markov, and John P. Hayes
University of Michigan

Editor's note:
The diverse nature of the faults and defects that may occur at nanoscale ranges necessitates new techniques for ATPG. This article proposes an efficient technique that relies on a probabilistic approach to detect and diagnose nontraditional faults and defects.
—Fabrizio Lombardi, Northeastern University

The evolution of IC technology has included changes, sometimes subtle ones, in the types of defects prevalent in circuits. Therefore, designers must occasionally reconsider fault modeling and testing in order to incorporate relevant new sources of circuit error. At the deep-submicron scale, VLSI circuits have higher feature density, lower supply voltage, and higher operating frequency, all contributing to greater signal noise. High feature density increases capacitive and inductive crosstalk between neighboring signals, which can result in delay and logic faults. Furthermore, process variations are more common because of the discrepancy between lithography wavelengths and circuit feature sizes. These phenomena lead to a considerable amount of randomness in actual threshold voltages and gate delays observed after manufacturing.

Decreased threshold voltages leave circuits susceptible to soft faults caused by external radiation. When primary radiation particles enter the atmosphere, neutrons and other secondary particles can strike a critical circuit node, leaving behind an ionized track in silicon, called a single-event upset. An SEU can flip a gate's output signal. If this effect propagates to a flip-flop, that flip-flop can capture (latch) it, forming a soft, but persistent, error. However, because of three masking mechanisms, not all SEUs cause circuit errors: An SEU is logically masked if it appears in an unsensitized portion of the circuit. It is electrically masked if its amplitude becomes lower than the threshold voltage as it propagates through a gate. Temporal masking occurs if an SEU arrives at a flip-flop during a nonlatching portion of the clock cycle. The probability of an SEU at a gate depends on gate area, neutron flux, altitude, and other environmental factors.[1]

New device technologies such as quantum and nanocircuits exhibit probabilistic behavior because the scale of interaction is often that of subatomic particles. For instance, quantum bits exist in superposition states that collapse to either 0 or 1 with different probabilities upon measurement. In many nanoelectronic devices, the difference between logical states approaches the thermal limit. The behavior of such devices will therefore be inherently probabilistic. Furthermore, defects become more common in nanoscale devices because of manufacturing difficulties. For example, devices such as single-electron transistors and quantum cellular automata store state in a few electrons, so missing electrons and misplaced cells are common faults in these devices. Additionally, electromagnetic noise and other types of interference affect nanoscale systems easily. In summary, most defects in nanocircuits are either inherently probabilistic or modeled probabilistically when deterministic models are impractical.

Several soft-error-rate analyzers have been released recently.[2,3] These tools electrically characterize specific technology nodes and estimate the overall error rate of combinational circuits. However, they are difficult to use for ATPG because they don't offer discrete logic-

0740-7475/07/$25.00 © 2007 IEEE Copublished by the IEEE CS and the IEEE CASS

Table 1. Key differences between deterministic and probabilistic testing.

Attribute	Deterministic testing	Probabilistic testing
Fault occurrence	Deterministic	Transient or intermittent
Fault model	Stuck-at, bridging, transition	Probabilistic generalization
Test inputs	Set of input vectors	Multiset of input vectors (multitest)
Coverage	Each test either detects a fault or not (binary)	Tests detect faults with varying probabilities
Goal	Detect fault presence or conclude fault absence	Estimate fault probability

level fault models. Also, the user cannot specify faults for test generation, and the tools don't handle multiple SEUs or multiple faults in general because these tools emphasize radiation-induced SEUs in VLSI circuits. To address these issues, we propose a general fault-modeling framework that can capture both probabilistic faults such as SEUs and deterministic faults such as stuck-at faults.

Generating tests for probabilistic models is fundamentally different from previous testing techniques. Traditionally, the goal of testing has been to detect the presence of faults. Traditional testing applies a test pattern set to a circuit's inputs and then compares the resultant outputs with correct precomputed outputs to determine whether a fault is present. In contrast, the goal of probabilistic testing is to estimate fault probability, or in other words to track uncertainty. Probabilistic testing requires a multiset (a set with repetitions) of test patterns because a given fault is present for only a fraction of the computational cycles. We call this multiset a *multitest*. Another difference is that some test vectors are more likely than others to detect transient faults, because of path-dependent effects such as electrical masking. Therefore, in probabilistic testing, we must consider the likelihood of detecting a fault—in other words, a test vector's sensitivity to a fault. Table 1 summarizes these differences.

PTM fault-modeling framework

Traditional testing uses a deterministic fault model to represent faults and derive test vectors that propagate fault effects (errors) to outputs. In our framework, we use a probabilistic analog called the probabilistic transfer matrix (PTM) method, which is technology independent but well-suited to representing faults and errors in nanotechnology.[4]

The PTM framework describes a gate or circuit's faulty behavior using a PTM, M, a matrix whose (j, k)th entry represents the conditional probability of output signals $Out = o_0, o_1, \ldots, o_n$ having value k if input signals $In = i_0, i_1, \ldots, i_m$ have value j—that is, $p(k|j)$. We treat row and column indices j, k as bit vectors whose entries represent values of inputs and outputs. For instance, $p(3|2)$ represents the probability that two output variables (o_0, o_1) have values (1, 1) given that two input variables (i_o, i_1) have values (1, 0). A fault-free circuit has an ideal transfer matrix (ITM), which is a PTM in which the output's correct value occurs with probability 1. Figure 1 shows the ITM and a PTM for a two-input AND gate. In the PTM, $AND_2(p)$, the probability of an output error is p for each input combination.

We treat wiring subnetworks as a special class of gates, which have permutation matrices as ITMs. We form wire PTMs by permuting bits of the identity matrix's row and column indices. Figure 2 shows the wire swap matrix for two adjacent wires $Swap_2$, a fan-out matrix F_2, and an identity matrix representing a wire or buffer I_1. In working with n-bit buses, it is useful to define n-bit versions of the wiring ITMs in Figure 2—for example, $I_{1,n}$, $F_{2,n}$, and $Swap_{2,n}$. The first subscript is the number of output buses, and the second is the bus width. $I_{1,n}$ is a $2^n \times 2^n$ identity matrix, and $F_{2,n}$ is a $2^n \times 2^{2n}$ matrix that copies an n-bit signal twice.

In the PTM framework, we represent a signal with a 1×2 row vector, $v = [p_0\ p_1]$, where p_0 is the

$$\begin{array}{c|cc} 00 & 1 & 0 \\ 01 & 1 & 0 \\ 10 & 1 & 0 \\ 11 & 0 & 1 \end{array}$$

(a)

$$\begin{array}{c|cc} 00 & 1-p & p \\ 01 & 1-p & p \\ 10 & 1-p & p \\ 11 & p & 1-p \end{array}$$

(b)

Figure 1. ITM AND_2 (a) and PTM $AND_2(p)$ (b) for an AND gate. The PTM describes output error probability p for each input combination.

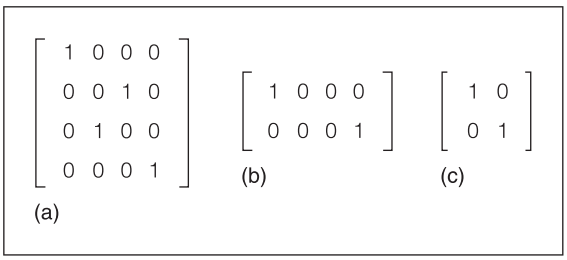


Figure 2. Wiring ITMs: *Swap*$_2$ (a) is an adjacent swap of two wires; F_2 (b) is a fan-out with two branches; and I_1 (c) is an identity matrix on one wire.

probability of the signal's being 0, and p_1 is the probability of its being 1. Multiple signals can be jointly represented. For example, input vector v_{in} (to a gate or a circuit) is a row vector representing the joint probability distribution of its input signals. The ith entry of v_{in}, denoted $v_{in}(i)$, gives the probability that input signals have values represented by bit vector i.

We can combine independent signals by using the tensor product of individual signal vectors. Given two matrices M_1 and M_2 of dimensions $2^k \times 2^l$ and $2^m \times 2^n$, tensor product $M = M_1 \otimes M_2$ is a $2^{km} \times 2^{ln}$ matrix whose entries are

$$M(i_0 \ \ldots \ i_{k+m-1}, j_0 \ \ldots \ j_{l-1}) =$$
$$M_1(i_0 \ \ldots \ i_{k-1}, i_0 \ \ldots \ j_{l-1}) \times$$
$$M_2(i_k \ \ldots \ i_{k+m-1}, j_l \ \ldots \ j_{l+n-1})$$

The output probability distribution after input vector v_{in} is evaluated on gate G with PTM M_G is $v_{in} \times M_G$.

Example 1. If inputs i_1 and i_2 of an ideal AND gate are described by $v_{i1} = [0.5\ 0.5]$ and $v_{i2} = [0.5\ 0.5]$, then the input row vector is $v_{in} = v_{i1} \otimes v_{i2} = [0.25\ 0.25\ 0.25\ 0.25]$. The output distribution is given by $(v_{in} \times AND_2) = [0.75\ 0.25]$.

PTMs can represent a wide variety of deterministic and probabilistic faults. Figure 3 shows a subset of gate faults that PTMs can capture.

Thus far, we've described signals by their logic values. We now expand signal representation to incorporate necessary electrical characteristics, while retaining our model's discreteness. For instance, we can differentiate signals of long and short duration just as we differentiate signals of high and low amplitude (with their logic values). We represent a signal by vector w, which has four entries instead of two: $w = [p_{0S}\ p_{0L}\ p_{1S}\ p_{1L}]$. The second bit of the row index represents short (S) or long (L) duration, so p_{0S} is the probability of a logic 0 with short duration. Extraneous glitches such as those induced by SEUs are likely to have short durations, whereas driven logic signals are likely to have relatively long durations.

Each gate in a circuit has the probability of experiencing an SEU strike that depends on environmental factors such as neutron flux and temperature. We call this the probability of occurrence for gate (or node) G: $p_{occur}(G)$. SEU strikes create glitches that can be differentiated through a combination of shape and amplitude. These differentiations are important in a glitch's propagation through circuit gates. Therefore, we use a modified identity matrix called $I_{1,n}(p_{occur})$ to represent the probability distribution of a glitch induced by an SEU strike.

We use the specific glitch propagation model of Omana et al. to determine which signal characteristics to capture.[5] A different model might require that other characteristics be represented. The model we use classifies glitches into three types, depending on their duration D and amplitude A relative to gate propagation delay T_P and threshold voltage V_{TH}. We assume that signals change from the logic-low value to the logic-high value only when a glitch occurs (but these values can later be inverted). The following are the three types of glitches:

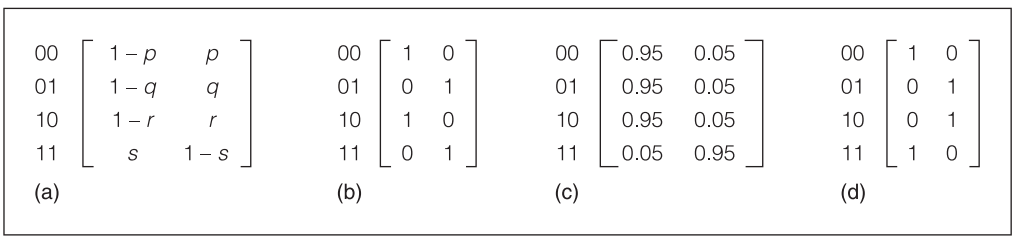


Figure 3. Various faults in an AND gate: input-dependent error (a), first input signal stuck-at 1 (b), probabilistic output bit-flip with $p = 0.05$ (c), and wrong-gate error with AND gate replaced by an XOR gate (d).

- *Type 1*. These glitches have amplitude $A > V_{TH}$ and duration $D > 2T_P$, and they propagate without attenuation.
- *Type 2*. These have $A > V_{TH}$ and $2T_P > D > T_P$, and they propagate with attenuated amplitude $A' < A$.

- *Type 3*. Glitches in this category have $A < V_{TH}$ and don't propagate at all.

Amplitude is already indicated by the logic value, so we need an additional bit to indicate whether the duration is longer or shorter than the gate's propagation delay (when amplitude is higher than threshold voltage). Duration is irrelevant for glitches with amplitudes lower than the threshold voltage, because these amplitudes are likely to be attenuated. Figure 4b shows the probability distribution of an SEU strike when the correct logic value is 0. Type 1 glitches are indicated by row labels 11, type 2 glitches are indicated by labels 10, and type 3 glitches are indicated by 01. Figure 4b assumes a uniform distribution with respect to glitches.

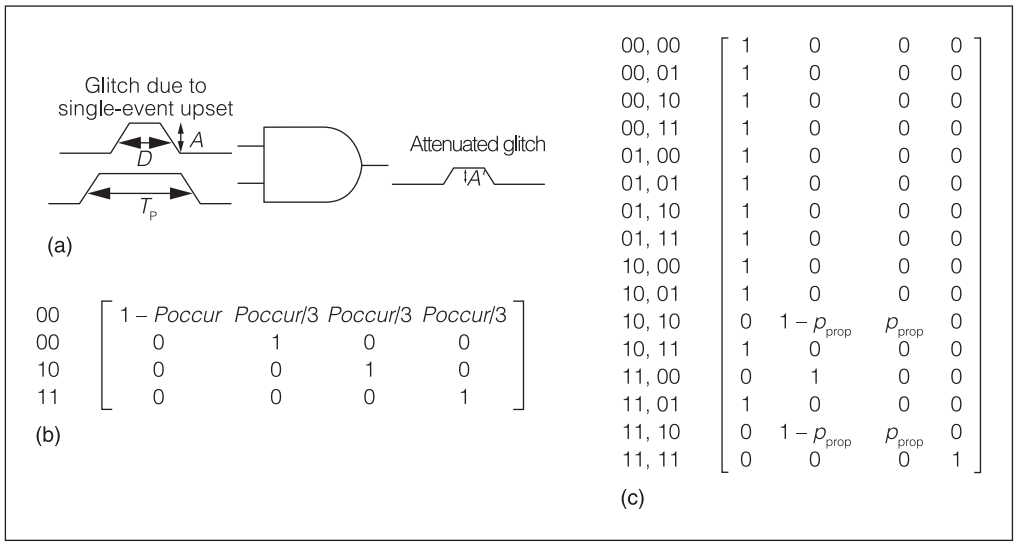


Figure 4. PTMs for SEU modeling: glitch with duration *D* and amplitude *A* attenuated to duration *D*′ and amplitude *A*′ as it passes through a gate with propagation delay T_p (a); probability distribution $I_{2,2}(p_{occur})$ of the energy of an SEU strike at a gate output, assuming a uniform energy distribution (b); and SEU-induced glitch propagation $AND_{2,2}(p_{prop})$ for a two-input AND gate (c). Row labels indicate input signal type. Type 2 glitches become attenuated to Type 3 with a probability of 1 – p_{prop}. (In (b), division by 3 is due to the uniform distribution assumption.)

Once an SEU strikes a gate and induces a glitch, the gate's electrical characteristics determine whether the glitch is propagated. Glitches with long duration and high energy relative to gate propagation delay and threshold voltage are usually propagated; others are quickly attenuated. We call the probability that a glitch is propagated $p_{prop}(g)$. We represent a logic gate's relevant glitch transfer characteristics with a modified gate PTM. For example, Figure 4c shows a modified AND PTM, denoted $AND_{2,2}(p_{prop})$.

In the selected glitch model, attenuation transforms sensitized Type 2 glitches, with a certain probability, into Type 3 glitches. All other signals retain their original output value given by the gate's logic function. The PTM in Figure 4c describes this transfer function. It shows an AND gate that propagates an input glitch (only if the other input has a noncontrolling value), with certainty if the glitch is Type 1 (and thus is indistinguishable from a driven logic value), or with probability p_{prop} if the glitch is Type 2.

When using 2-bit signal representations, we compute a signal's probability of having a logic 1 value by marginalizing or summing out over the second bit. For instance, if a signal has the 2-bit distribution [0.2 0.1 0.3 0.4], because the second bit indicates duration, the probability of a logic 0 is 0.2 + 0.1, and the probability of a logic 1 is 0.3 + 0.4.

Test vector sensitivity

To discuss the sensitivity of test vectors to faults, we begin with an example of a circuit in which a fault is detected by different vectors with different probabilities. Then, we present PTM-based algorithms for calculating test vector sensitivity.

Example 2. Consider the circuit in Figure 5. Suppose an SEU periodically occurs at input *b*. The test vectors that propagate the induced glitch to the outputs are t_1 = 001 (to output *z*), t_2 = 100 (to output *y*), and t_3 = 101 (to both *y* and *z*). In deterministic testing, we can choose any of these test vectors. However, error attenuation along sensitized paths affects the propagation of probabilistic errors. If the propagation probability is p_{prop} at each gate, t_1 has probability $p_{t1} = p^2_{prop}$ of propagating the error to an output, t_2 has probability $p_{t2} = p_{prop}$, and t_3 has probability $p_{t3} = p_{t1} + p_{t2} - p_{t1}p_{t2}$. For a fault that occurs with probability p_f, a vector t_i must be repeated $\lceil 1/(p_{ti} * p_f) \rceil$ times for one expected detection. Therefore, test application time is shortest for

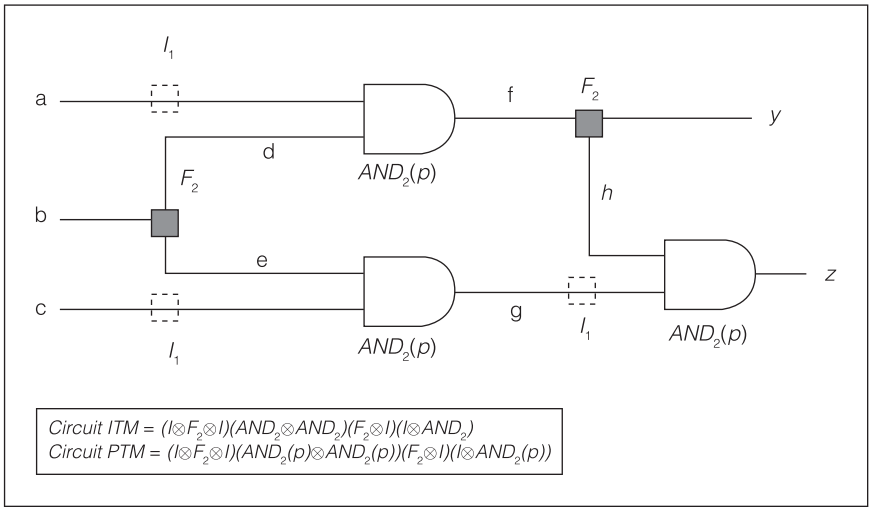


Figure 5. Sample circuit with its ITM and PTM computations.

test vector t_3, which is the most sensitive to the transient fault in question.

There are two ways to identify sensitive test vectors. The first is circuit PTM computation, a general method for deriving circuit reliability information.[4] This method combines the gate PTMs to form a circuit PTM, and the most sensitive test vectors can be read off from the circuit PTM. Starting with the gate PTMs, we calculate the circuit PTM using a few construction rules and two matrix operations:

- If two gates G1 and G2 with PTMs P_1 and P_2 are connected in series, their combined PTM is given by their matrix product P_1P_2.
- If G1 and G2 are connected in parallel, their combined PTM is given by their tensor product $P_1 \otimes P_2$.

Figure 5 shows an algebraic expression for a circuit PTM directly calculated from the circuit diagram with the preceding rules. Figure 6 shows the same circuit with a circuit PTM for SEU-induced probabilistic faults.

Given the notion of a circuit's PTM, we can now define a test vector's sensitivity to faults in the circuit. We define the sensitivity of test vector t to fault set $F = \{f_1, f_2, \ldots f_n\}$, which occurs with probability $P = \{p_1, p_2, \ldots p_n\}$ in circuit C with PTM M_F and ITM M, as the total probability that the output under t is erroneous given that faults F exist with probability P. Test vector t can be represented by vector v_t with 0s in all but the index corresponding to the input assignments of t. For instance, if test vector t assigns 0s to all input signals and C has three inputs, then $v_t = [1\,0\,0\,0\,0\,0\,0\,0]$. The sensitivity of t is the probability that the ideal and faulty outputs are different, and we compute it by taking the L_1 norm of the element-wise product (denoted $.*$) of the correct and faulty output vectors. (The L_1 norm of a vector is simply the sum of its elements.)

$$sens(F, t) = 1 - \|(v_t M_f).*(v_t M)\|L_1 \quad (1)$$

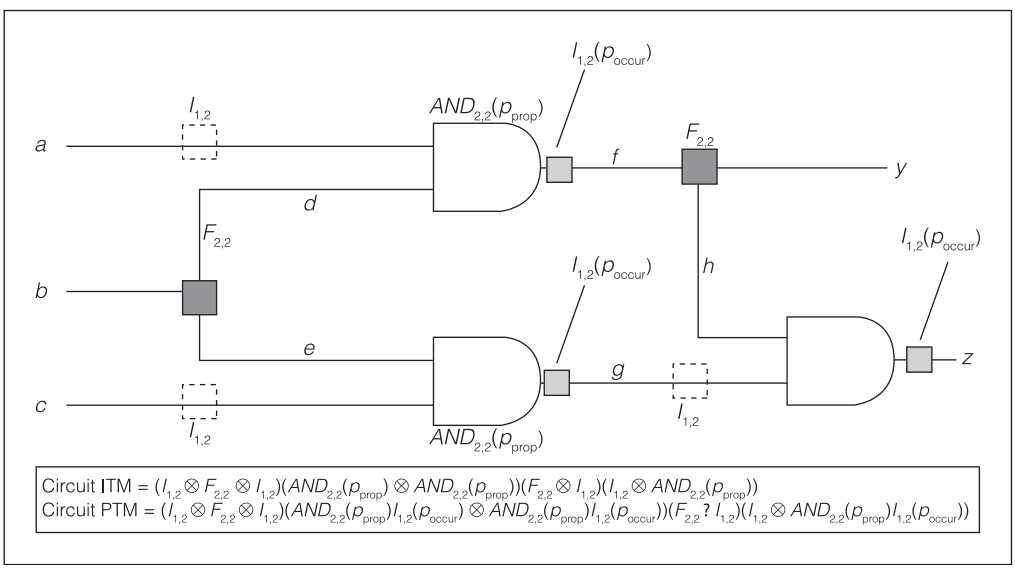


Figure 6. Circuit with ITM and PTM computations describing an SEU strike and propagation with multibit signal representations.

Example 3. We compute the sensitivity of test vector $v_t = [1\,0\,0\,0\,0\,0\,0\,0]$ for the circuit in Figure 5, with error probability $p = 0.1$, using the circuit's PTM M_F and ITM M, as shown in Figure 7.

Note that v_tM_f and v_tM must be marginalized if there is a multibit signal representation. For example, in the case of the SEU model described in the previous section, we must sum out the second bit for both vectors to obtain the correct sensitivity.

The second method of sensitivity computation is output distribution computation for a particular test vector. We begin with a preselected complete set of test vectors for the permanent stuck-at faults corresponding to those in F. For each test vector in this set, we compute the faulty output (v_tM_f) and the ideal output (v_tM) by propagating signal distributions through each gate using vector-PTM multiplication.

Example 1 illustrates this process. The difference in this sensitivity computation method is that we avoid explicitly computing the circuit PTM and ITM, which is computationally expensive. Then, we use Equation 1 to compute the sensitivity.

Fan-outs result in inseparable probability distributions of the branch signals. Marginalizing these signals or treating them as separate can cause inaccuracies in the output probabilities. A possible way to handle this problem is simply to keep these signals together (joint representation) and enlarge any gate PTM that any of the signals pass through, thus encapsulating the fan-out. We enlarge gates by adding inputs that pass through unchanged. This corresponds to tensoring the gate matrix with an identity matrix.

$$v_t \times M_F = v_t \times \begin{bmatrix} 0.810 & 0.090 & 0.082 & 0.018 \\ 0.810 & 0.090 & 0.082 & 0.018 \\ 0.810 & 0.090 & 0.082 & 0.018 \\ 0.810 & 0.090 & 0.018 & 0.082 \\ 0.810 & 0.090 & 0.082 & 0.018 \\ 0.810 & 0.090 & 0.082 & 0.018 \\ 0.090 & 0.010 & 0.738 & 0.162 \\ 0.090 & 0.010 & 0.162 & 0.738 \end{bmatrix} = [\ 0.8100 \quad 0.0900 \quad 0.0820 \quad 0.0180\ ]$$

(a)

$$v_t \times M = v_t \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [\ 1 \quad 0 \quad 0 \quad 0\ ]$$

(b)

$$sens\,(F, t) = 1 - ||[1\ 0\ 0\ 0] * [0.81\ 0.09\ 0.082\ 0.018]|| = 1 - ||[0.81\ 0\ 0\ 0]|| = 1 - (.81) = 0.9$$

(c)

Figure 7. Matrix equations to compute the sensitivity of test vector v_t for the circuit in Figure 5 with error probability $p = 0.1$, using the circuit's PTM M_F and ITM M. The first matrix equation multiplies input vector v_t by ITM M, resulting in an output vector (a). The second matrix equation multiplies input vector v_t by faulty matrix M_F, resulting in an erroneous output vector (b). Combining these two matrix equations, we compute the sensitivity of this input vector (c).

In Example 4, we compute signal vectors through the circuit in topological order to obtain output vectors. At each step, we compute the appropriate joint input probability distribution for the next gate in topological order. However, because of inseparable signal distributions, we must often enlarge gates with identities. This method's complexity is linear in the number of gates in a circuit, if gate size (including enlargement) is bounded by a constant. Therefore, this method scales much farther than PTM computation. The gate size is limited to less than 10 inputs in most circuit synthesis techniques. We can contain the enlargement by imposing a limit on the number of signals and marginalizing any distributions that become larger.

Example 4. Consider the circuit in Figure 5 with the primary inputs described by vectors v_a, v_b, and v_c, and the two-input AND gates described by PTM $AND_2(p)$. We compute the faulty output vector in the following steps:

1. Multiply v_b by F_2 to obtain fan-out branches d and e: $v_{d,e} = v_b \times F_2$.
2. Enlarge the AND gate with inputs a and d to include e, which is inseparable from d, as an input and an output: enlarged_$AND_2(p)$ = $AND_2(p) \otimes I_1$.
3. Compute the joint probability of a, d, and e, using the tensor product to form the inputs for the enlarged gate of the previous step: $v_{a,d,e} = v_a \otimes v_{d,e}$.
4. Compute the joint probability of f with e by vector matrix multiplication: $v_{f,e} = v_{a,d,e} \times$ enlarged_$AND_2(p)$.
5. Signals f and e are now inseparable. Therefore, enlarge the AND gate with inputs c and e,

```
Start with R = F and no tests selected
While R ≠ Ø
   Select text t_i from T that minimizes R when selected
for each fault f_j calculate d_j = Π_i(1 - sens(f_j, t_i))
Delete f_j ∈ R such that 1 - d_j > P_th
EndWhile
Return the selected tests
```

Figure 8. Greedy algorithm for minimizing the number of test vectors (with repetition) required for fault detection.

and compute outputs f and g: $v_{f,g} = (v_{f,e} \otimes v_c) \times (I_1 \otimes AND_2(p))$.

6. Multiply the joint vector of f and g by an enlarged F_2 to produce y, x, and g: $v_{y,h,g} = v_{f,g} \times (F_2 \otimes I_1)$.
7. Multiply $v_{y,h,g}$ by an enlarged AND gate to produce y, z: $v_{y,z} = v_{y,h,g} \times (I_1 \otimes AND_2(p))$.
8. To find the ideal output vector, change all instances of $AND_2(p)$ to AND_2.

This computation incorporates effects of the reconvergent fan-out from b to z because signals in the fan-out branch (d, g, f, x, z) are always jointly represented and processed. However, storing joint probability distributions can be computationally expensive, so we sometimes trade a loss of accuracy for a reduction in memory complexity.

Compact multitest generation

Multitest generation is closely related to the standard set-cover problem, in which a set's elements are ''covered'' by subsets. The difference in multitest generation is that set members (faults) are detected by a test (covered) only probabilistically. This lets us modify algorithms designed for set cover and introduce integer linear-programming (ILP) formulations whose linear-programming (LP) relaxations can be solved in polynomial time. Furthermore, modifying the multitest objective simply amounts to altering the ILP objective function.

Suppose single fault f in circuit C has estimated occurrence probability p. We confirm this probability as follows:

1. Derive test vector t with high sensitivity $sens(f, t)$.
2. Apply t to C, $k = \lfloor 1/(sens(f, t)) \rfloor$ times for one expected detection.
3. If there are $d(f) >> 1$ detections, we conclude that the actual probability of f is higher, and we reject the estimated probability. We estimate the probability that there are $d(f)$ detections in k trials using the binomial theorem. If the probability of $d(f)$ detections is low, the actual fault probability is likely to be greater than estimated probability p.

To extend this argument to multiple faults, we consider two assumptions about faults:

- *Assumption 1*. There are several possible probabilistic faults, yet the circuit experiences only a single fault in any given clock cycle. This is a common assumption in techniques focused on SEUs and is justified by the relatively low frequency of particle strikes.
- *Assumption 2*. Each circuit component (gate) has an independent fault probability; that is, multiple faults at different locations can occur in the same clock cycle. This assumption applies to nanotechnologies in which large physical defects or random device behavior can lead to multiple faults in different circuit locations. Here, the probability of two faults is given by the product of individual fault probabilities.

Our goal in either case is to pick a multitest of vectors $T' \subset T = \{t_1, t_2, \ldots t_m\}$ such that $|T'|$ is minimal. Recall that each test vector t_i represents a subset of F (that is, a subset of faults).

Under Assumption 1, we can reduce multitest size by using test vectors that are either especially susceptible to one fault or somewhat sensitive to many faults. Therefore, to obtain the minimum detection probability, p_{th}, we need n tests, where n satisfies $(1 - p)^n \leq 1 - p_{th}$.

Figure 8 shows the greedy algorithm for generating such a multitest, starting from a compacted set of test vectors for the corresponding deterministic faults. Intuitively, compacted test sets are likely to contain many sensitive test vectors because each test vector can detect multiple faults. However, we get better results if we start with a larger set of test vectors, such as the union of different compact test sets. The set R in the algorithm stores the uncovered faults in any iteration—undetected faults—with a minimum probability of p_{th}. As before, $T = \{t_1, t_2 \ldots t_n\}$ is

the test set, and F is the fault set. Kleinberg and Tardos proved that the approximation factor for the related set-cover algorithm is $O(\log(|multitest|))$.[6] The runtime is lower-bounded by the multitest's size, which is the number of iterations through the while loop.

$$\begin{array}{l} \text{Minimize } \sum_{i=0}^{m} x_i \\ \text{subject to} \\ \forall j,\ (\sum_{i=1}^{0} x_i \times sens(f_j, t_i)) \geq n \\ \forall (i, x_i) \geq 0,\ x_i \text{ is an integer} \end{array}$$

(a)

$$\begin{array}{l} \text{Minimize } \sum_{j=0}^{n} (\sum_{i=0}^{m} x_i \times sens(f_j, t_i)) \\ \text{subject to} \\ \forall j,\ (\sum_{i=1}^{m} x_i \times sens\ (f_j, t_i)) \geq n \\ \forall (i, x_i) \geq 0,\ x_i \text{ is an integer} \end{array}$$

(b)

Figure 9. ILP formulations for test set generation for a fixed number of expected detections: minimizing the number of test vectors required (a) to maximize fault resolution (minimize overlap) (b).

We also formulate this problem as the ILP formulation shown in Figure 9a. The challenge in adapting the ILP algorithms for set cover or set multicover is that there is no notion of a probabilistic cover in known set-cover formulations. In our case, each test detects each fault with a different probability, $sens(f_j, t_i)$. If we required a minimum detection probability p_{th}, as in Figure 8, the constraint that $\forall f_j$, $\Pi_j(1 - sens(f_i, t_j)) < 1 - p_{th}$ wouldn't be linear. We alter this constraint and linearize it by observing that each repetition of test t_i can be regarded as an independent, identically distributed binomial variable for each fault f_j. Therefore, if a test repeats x_i times, the expected detections for fault f_j are $x_i \times sens(f_j, t_i)$—that is, the expected value of a binomial random variable with parameters $(x_i, sens(f_j, t_i))$. Because expectation is linear, we can add the contributions of all test vectors for each fault f_j as $\Sigma_i(x_i \times sens(f_j, t_i))$, leading to the constraint in line 3 of Figure 9a. We can show that this ILP formulation reduces to the multiset-multicover problem. The LP relaxation, along with randomized rounding, gives a solution of this problem, which is within a log factor of optimal.[7] In randomized rounding, each x_i is rounded up with a probability equal to the fractional part of x_i.

Assumption 2 generalizes the single-fault case. We can treat the fault set as a single fault with multiple locations and introduce fault probabilities in all gates' PTMs simultaneously. We denote this fault F'. Then, we simply pick the test vector t that is most sensitive to the combination of simultaneous faults, using the output-distribution-based test vector sensitivity computation. We repeat t a total of $k/(sens(F', t))$ times for k expected detections. Table 2 represents a situation in which each gate in the circuit has a small error probability, $p = 10^{-5}$. The average difference between the number of repetitions needed by a random vector and the number needed by the most sensitive test vector is 53.3%. This implies a proportional decrease in test application time.

In addition, we can diagnose the probabilistic faults in Assumption 1. In other words, we select test vectors that minimize ambiguity about which fault is detected. For this purpose, we modify the objective to that of Figure 9b. Intuitively, we know that achieving the required detection probability minimizes the total number of extra detections. This is equivalent to minimizing the

Table 2. Number of repetitions required for one expected detection: random vectors versus maximally sensitive test vectors.

Circuit	Average sensitivity	No. of repetitions	Maximum sensitivity	No. of repetitions	Improvement (%)
9symml	3.99×10^{-5}	2.51×10^{4}	7.99×10^{-5}	1.25×10^{4}	50.00
Alu4	5.78×10^{-4}	1.73×10^{3}	1.82×10^{-3}	549	68.20
i1	6.65×10^{-5}	1.50×10^{4}	9.99×10^{-5}	1.00×10^{4}	33.40
b9	7.70×10^{-5}	1.30×10^{4}	1.10×10^{-4}	9.09×10^{3}	30.00
C880	5.38×10^{-4}	1.86×10^{3}	9.39×10^{-4}	1.07×10^{3}	42.70
C1355	1.03×10^{-3}	970	1.27×10^{-2}	78	91.80
C499	2.76×10^{-4}	3.62×10^{3}	1.27×10^{-3}	787	78.27
x2	3.39×10^{-5}	2.95×10^{4}	4.99×10^{-5}	2.00×10^{4}	32.10
Average					53.30

Table 3. Number of test vectors required to detect input signal faults with various threshold probabilities: randomly chosen test vectors versus our algorithm (Figure 8).

	Random	Algorithm	Random	Algorithm	Random	Algorithm	Random	Algorithm
Circuit	p_{th} = 0.05		p_{th} = 0.75		p_{th} = 0.95		p_{th} = 0.99	
c6288	377	56	782	112	1,266	236	1,998	360
c432	731	462	1,415	924	2,696	1,947	3,797	2,970
c499	1,643	518	2,723	1,036	4,448	2,183	8,157	3,330
c3540	907	411	1,665	817	3,589	1,716	4,975	2,615
c5315	2,669	854	4,691	1,708	8,961	3,599	13,359	5,490
c7552	3,729	1,680	6,824	3,364	12,210	7,082	18,314	10,805
c2670	3,650	884	5,699	1,770	11,104	3,729	15,961	5,682
Improvement (%)		64.5		59.7		53.71		53.05

overlap in the subsets represented by the test vectors. In contrast to the previous formulation, this problem is related to the multiset-exact-multicover problem.

In practice, the number of test vectors needed for probabilistic testing can be quite small because testers are likely to be mainly concerned with the most frequently occurring faults. The number of repetitions of a test vector for n expected detections is $\propto n/p_f$, where p_f is the fault probability. Therefore, multitest size decreases with expected fault probability.[8] Also, if application time is limited, we can select test vectors that maximize the expected detection rate. To do this, we use a binary search for the largest value of n achievable with m test vectors. Because the program in Figure 9a attempts to minimize the number of test sets selected, it also maximizes the faults covered by each test.

In summary, test generation for probabilistic faults consists of the following steps:

- Generate test set T for the corresponding deterministic faults in F.
- Evaluate the sensitivity of each test in T with respect to each fault in F, using the output distribution computation.
- Execute the greedy algorithm (Figure 8) or the ILP formulations (Figure 9).

Table 3 shows the number of test vectors required to detect input signal probabilistic stuck-at-0 faults (of probability $p_f = 0.05$) using the method shown in Figure 8. These results show that our algorithm requires more than 50% fewer test vectors than random selection. Our base set is a complete test vector set generated by the Atalanta ATPG software.[9]

Once a multitest is generated, we can use Bayesian learning to estimate the actual error probability. This well-established AI technique uses observation (data) and prior domain knowledge to predict events.[10] In our case, the prior domain knowledge is the expected or modeled fault probabilities in a circuit, and the data comes from test results.

As circuit reliability becomes a primary concern in submicron VLSI and nanotechnology, testing will take on an increasingly important role in ensuring the correct functionality of combinational circuits. Furthermore, the types of faults that occur in circuits will change because of the greater impact of transient faults and inherently probabilistic effects in circuit technology. Our general probabilistic fault model and algorithms address these concerns. Our methods are suitable for the following uses: First, after manufacture, they can be used to assess circuits' susceptibility to transient faults and to determine yield. They can also be used in conjunction with radiation and high-temperature testing for further acceleration. Second, during deployment, our methods can serve to test the effects on the circuit of high-radiation environments and to turn on redundant computation units for error correction if necessary. Our future research in probabilistic testing will include the development of BIST techniques that can be used efficiently in these situations. ■

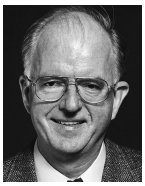
References

1. P. Shivakumar et al., "Modeling the Effect of Technology Trends on Soft Error Rate of Combinational Logic," *Proc. Int'l Conf. Dependable Systems and Networks* (DSN 02), IEEE CS Press, 2002, pp. 389-398.

2. M. Zhang and N.R. Shanbhag, "A Soft Error Rate Analysis (SERA) Methodology," *Proc. IEEE/ACM Int'l Conf. Computer-Aided Design* (ICCAD 04), IEEE CS Press, 2004, pp. 111-118.
3. B. Zhang, W.S. Wang, and M. Orshansky, "FASER: Fast Analysis of Soft Error Susceptibility for Cell-Based Designs," *Proc. 7th Int'l Symp. Quality Electronic Design* (ISQED 06), IEEE CS Press, 2006, pp. 755-760.
4. S. Krishnaswamy et al., "Accurate Reliability Evaluation and Enhancement via Probabilistic Transfer Matrices," *Proc. Design, Automation and Test in Europe Conf.* (DATE 05), IEEE CS Press, 2005, vol. 1, pp. 282-287.
5. M. Omana et al., "A Model for Transient Fault Propagation in Combinatorial Logic," *Proc. 9th Int'l On-Line Testing Symp.* (IOLTS 03), IEEE Press, 2003, pp. 111-115.
6. J. Kleinberg and E. Tardos, *Algorithm Design*, Addison-Wesley, 2005.
7. V.V. Vazirani, *Approximation Algorithms*, Springer-Verlag: New York, 2001.
8. I. Polian et al., "Transient Fault Characterization in Dynamic Noisy Environments," *Proc. IEEE Int'l Test Conf.* (ITC 05), IEEE Press, 2005, pp. 40.1-40.10.
9. H.K. Lee and D.S. Ha, *On the Generation of Test Patterns for Combinational Circuits*, tech. report 12-93, Dept. of Electrical Engineering, Virginia Polytechnic Inst. and State Univ., 1993.
10. M. DeGroot, *Optimal Statistical Decisions*, McGraw-Hill, 1970.

Smita Krishnaswamy is a PhD candidate in the Department of Computer Science and Engineering at the University of Michigan, Ann Arbor. Her research interests include CAD, circuit test, and logic synthesis. Krishnaswamy has an MSE in computer engineering from the University of Michigan, Ann Arbor. She is a student member of the IEEE and the ACM.

Igor L. Markov is an associate professor of electrical engineering and computer science at the University of Michigan, Ann Arbor. His primary research interest is in combinatorial optimization applied to the design and verification of VLSI circuits. Markov has a PhD in computer science from the University of California, Los Angeles. He is a senior member of the IEEE.



John P. Hayes is a professor of electrical engineering and computer science at the University of Michigan, Ann Arbor, where he holds the Claude E. Shannon Chair of Engineering Science. His research interests include computer-aided design and testing, fault-tolerant systems, and quantum computing. Hayes has a PhD in electrical engineering from the University of Illinois at Urbana-Champaign. He is a Fellow of the IEEE and the ACM.

■ Direct questions and comments about this article to Smita Krishnaswamy, Advanced Computer Architecture Laboratory, University of Michigan, 2260 Hayward St., Ann Arbor, MI 48109-2121; smita@eecs.umich.edu.

For further information on this or any other computing topic, visit our Digital Library at http://www.computer.org/publications/dlib.