Efficient Algorithms to Accurately Compute Derating Factors of Digital Circuits

Hossein Asadi
Department of Computer Engineering
Sharif University of Technology, Tehran, Iran
asadi@sharif.edu

Mehdi B. Tahoori
Chair of Dependable Nano Computing
Karlsruhe Institute of Technology, Germany
mehdi.tahoori@kit.edu

Mahdi Fazeli
Department of Computer Engineering
Iran University of Science & Technology, Tehran, Iran
m_fazeli@iust.ac.ir

Seyed Ghassem Miremadi
Department of Computer Engineering
Sharif University of Technology, Tehran, Iran
miremadi@sharif.edu

***Abstract*—Fast, accurate, and detailed soft error rate (SER) estimation of digital circuits is essential for cost-efficient reliable design. A major step to accurately estimate a circuit SER is the computation of failure probability, which requires the computation of three derating factors, namely logical, electrical, and timing derating. The unified treatment of these derating factors is crucial to obtain accurate failure probability. Existing SER estimation techniques are either unscalable to large circuits or inaccurate due to lack of unified treatment of all derating factors. In this paper, we present fast and efficient algorithms to estimate SERs of circuit components in the presence of single event transients by unified computation of all derating factors. The proposed algorithms, based on propagation of error probabilities and shape of erroneous waveforms, are scalable to very large circuits. The experimental results and comparisons with statistical fault injections (SFI) using Monte-Carlo simulations confirm the accuracy (only 2% difference) and speedup (five to six orders of magnitudes) of the proposed technique.**

I. Introduction

Despite advances in device scaling, transistor density, and system frequency, CMOS technology fabricated on 65nm process or smaller has become sensitive to radiation-induced transient errors. These transient errors, also called *Single Event Upsets* (SEU) or *soft errors*, are the main reliability threat of digital systems. Recent studies show that in addition to DRAMs and SRAMs, combinational logic, latches, and flip-flops have also become sensitive to cosmic rays at terrestrial levels [1], [2], [3], [4].

When an energetic particle strikes a sensitive region in a combinational logic, it may produce a transient voltage, called *Single Event Transient* (SET). A SET may result in a soft error if captured by circuit bistables. SETs occurring in a logic gate may logically be masked (*logical masking*); their magnitude may be attenuated by logic gates such that they cannot pass through successive logic stages (*electrical masking*); or they may not arrive at appropriate latching-window of a bistable (*timing masking*).

To design a cost-effective soft error hardening technique, detailed and accurate *Soft Error Rate* (SER) of circuit components are needed. By having accurate SERs of each individual component (logic gates and bistables), designers can selectively protect the most susceptible parts of a circuit with minimal performance and power penalty. A circuit SER directly depends on the occurrence rate and the failure probability of SETs. The most challenging issue to estimate a circuit SER is computation of the circuit failure probability. To accurately compute the failure probability of a logic gate, it is essential to unifiedly compute all masking (derating) factors together; i.e., separate computation of logical, electrical, and timing derating factors would result in inaccurate estimation of a circuit SER [5].

Extracting an exact probability of *Logic Derating* (LD) for a circuit with reconvergent fanouts is an NP-complete problem as the number of paths from the fault site to the circuit outputs exponentially increases within the circuit. This makes existing LD computation techniques very time-consuming. Computation of *Timing Derating* (TD) depends on the shape of a glitch caused by a particle strike at the output of the gate, the latching windows of reachable bistables, and the propagation delay from the output of the faulty gate to the inputs of reachable bistables. Unlike LD which requires static analysis, TD requires dynamic analysis of transient propagation. This makes TD estimation more time-consuming than LD. Lastly, *Electrical Derating* (ED) can be computed by extracting the shape (magnitude, width, rise/fall time) of the propagated SET pulse at the output of each logic gate. The rise/fall time and the width of a SET can modify the shape of the propagated glitch and this can directly affect the other derating factors (i.e., LD and TD).

This paper presents efficient algorithms and a unified framework, called *LETD* (Logic-Electrical-Timing Derating computation tool), to accurately estimate LD, ED, and TD of digital circuits. We present an analytical technique for logic-electrical-timing derating estimation which eliminates the need for time-consuming (fault) simulations. The proposed technique takes into account the effect of reconvergent fanouts and scales very well for large logic blocks. In our proposed technique, we use an enhanced static timing analysis to compute all propagated waveforms from a struck gate (fault site) to reachable flip-flops and calculate the probability of latching an incorrect value in a flip-flop (i.e. incorrect system state). While propagating transient glitches, computation of electrical masking effect is integrated with error propagation probability scheme.

We have validated our proposed algorithms by a reference model built on *Statistical Fault Injection* (SFI) using Monte-Carlo (MC) timing accurate simulations. The experimental results show that the accuracy of our proposed technique is within 2% of the results obtained by the reference model while orders of magnitude faster. We also investigate the time-complexity of the proposed approach. Our analysis shows that the proposed approach has polynomial complexity. This makes the proposed approach a viable solution to accurately measure the SER of industrial-scale circuits.

Briefly, the main contribution of our proposed technique over previous work is as follows. Compared to previous fault injection based techniques, our proposed technique can compute all three derating factors of very large circuits (which contain reconvergent fanouts) orders of magnitude faster. Note that *Binary Decision Diagrams* (BDD) and *Algebraic Decision Diagrams* (ADD) based approaches or simulation-based fault-injection techniques are able to compute these three derating factors for reconvergent fanout nodes but the simulation time is intractable for very large circuits. In particular, although the methods presented in [5], [6], and [7] can accurately compute SERs in reconvergent fanouts, they are not scalable, making them inapplicable to very large-size circuits. On the other hand, the techniques presented in [8], [9], and [10] can accurately compute SERs in reconvergent fanouts applicable to large circuits but these techniques do not consider all three masking factors.

The rest of this paper is organized as follows. Sec. II reviews the previous work on SER estimation techniques. Sec. III proposes our SER estimation approach in digital circuits. In Sec. IV, experimental results are presented. Finally, Sec. V concludes the paper.

II. Related Work

Briefly, previous SER estimation methods can be categorized into four large groups: 1) FI based on random vector simulation or based on gate/path pre-characterization approaches; 2) BDD and/or ADD based approaches; 3) Boolean Difference or satisfiability based approaches; and 4) Error probability propagation based approaches. Next we discuss previous SER estimation methods in detail.

A. Fault-Injection (FI) Based on Random Vector Simulation or Gate/Path Pre-Characterization

The methods in this category rely on fault injection and logic simulation to determine the probability that a SET is captured by circuit flip-flops. This is typically achieved by injecting several SETs at the output of all susceptible gates and simulating the circuit using all possible input vectors. Although FI methods offer a high level of accuracy, they are very time-consuming and are not tractable for large-scale circuits. In particular, FI methods become more intractable when considering the electrical derating of logic gates. To overcome this issue, different methods have been proposed to use pre-characterization. The main aim of pre-characterization is to employ fault injection in order to determine the response of all library gates or sensitized paths of a circuit to SETs. Such information can be used to expedite FI experiments when propagating SETs along logic gates. A common shortcoming of pre-characterization-based methods is that the masking factors are calculated separately and the correlation between masking factors is not considered in the overall SER.

In the work presented in [11], logic gates are pre-characterized first and then SERs of logic gates are computed and stored in a table for each input vector. Logic simulations are performed in order to compute SERs of sensitized paths from each logic gate to bistables. SERs of a set of paths are pre-computed and in order to compute the SER of a specific path, the path SER is approximated by the closest path in the set. The proposed technique is time-consuming for large circuits as it uses logic simulations. Moreover, it introduces inaccuracies as it 1) neglects pulse attenuation, 2) does not consider the effect of re-convergent fan-outs, and 3) does not consider the unified treatment of the three masking mechanisms.

Another method based on pre-characterization has been presented in [7]. The proposed method uses MC-based SPICE simulations to extract probability distributions of SET pulse widths. To do this, a large amount of samples including random paths of logic gates are generated first and then different glitches are injected and propagated along the paths. Electrical, logical, and timing masking factors of the paths are then computed and stored in tables. The SER of a logic gate for a given SET pulse width is estimated by looking up in the tables. The proposed method requires very large amount of samples to build the lookup tables. In addition, this method does not consider the unified treatment of the masking factors.

A *Soft Error Rate Analysis* (SERA) methodology for combinational and memory circuits has been presented in [12]. In SERA, graph theory and fault simulation are used to investigate logical masking and also to extract equivalent inverter chain for each logically sensitized path. To measure the electrical masking, SPICE simulation of inverter chain is exploited. The authors also use different heuristics to accelerate the SER estimation process. Logical masking factor is computed by random vector simulations which can be very time-consuming for large circuits. Additionally, the presence of re-convergence significantly increases the number of paths in a circuit, thereby limiting the applicability of this method. Moreover, similar to other pre-characterization methods, it is assumed that the masking factors are independent and computed separately.

A hierarchical soft error estimation tool called HSEET has been proposed in [6]. In HSEET, a design employing hierarchical architecture is first defined in terms of basic building blocks. Basic building blocks are then pre-characterized for charge generation and errors are propagated using logic-level simulation [13]. The proposed technique, however, works well only for designs employing hierarchical architecture such as adders and multipliers. Additionally, using logic simulations makes the proposed technique very time-consuming for large-size circuits.

A SER estimation approach based on parameterized descriptor has been proposed in [14]. SETs are modeled in terms of a parametric Weibull function and a rate function described by a discrete set of error rate numbers. SETs are injected at each node and are propagated through each sensitized path in the design. Simulation of large or limited number of input vectors using FI experiments makes the proposed approach either time-consuming or inaccurate, respectively. The error threshold and the confidence levels for FI experiments have not been reported and as a result, the corresponding accuracy is unknown. The cancellation effect of the propagated glitches has not been also considered in the re-convergent paths.

An analysis to model the effect of SETs using a series of matrix transformations has been proposed in [15]. In this technique, both circuits and SETs are modeled in the form of matrices. All gates are pre-characterized for their responses to different SETs and the information is stored in a matrix format. Since the proposed technique is not based on dynamic logic simulations, it runs very fast. The effect of reconvergent paths is not considered in this technique and the accuracy has not been reported for large circuits.

A signature-based SER estimation technique is presented in [16], [9]. A signature of an internal node of a circuit is a bit pattern produced by applying different input vectors to primary inputs of the circuit. This technique has two main drawbacks: 1) it does not consider the electrical and timing derating factors in SER estimation process; 2) To extract signatures, it is required to apply random input vectors to the primary inputs and perform logic simulations. In order to extract SERs more accurately, further logic simulations are required, which makes the proposed technique very time-consuming for large circuits.

Briefly, the pre-characterization-based FI methods significantly reduce the runtime of FI experiments by sacrifying the overall accuracy. The common shortcomings of these methods are: 1) they are not applicable for SER estimation of large-scale circuits due to their runtime; 2) they do not consider the cross effects of masking factors in re-convergent fanouts resulting in inaccurate SER estimation of very large size circuits including many re-convergent paths.

B. BDD/ADD-Based Approaches

The main aim of such approaches is to encode and propagate SETs at the gate level using BDDs or ADDs. Some of the well-known BDD/ADD-based methods are detailed and discussed next.

In [17], a statistical method based on BDD to analyze the susceptibility of combinational circuits to SEUs is presented. Sensitized path information and SEU characteristics are represented by BDD. To speed up the SER estimation process, a heuristic is used to partition a circuit into smaller parts. The proposed method relies on explicit enumeration of BDDs corresponding to all input conditions and assumes simple superposition of re-convergent glitches, without considering their possible mutual masking.

A SER estimation technique to compute masking factors in a unified framework using BDDs and ADDs is presented in [5]. The proposed technique models and propagates SETs in both steady-state and transient state of circuit operation. Although the re-convergent fanout has been addressed in this technique, it still suffers from large run-time especially for large circuits including many *primary inputs* (PIs) and bistables. As reported by the authors, the run-time becomes an issue for circuits with an overall number of PIs and bistables greater than 40 or 50 [5]. To overcome this issue, the authors have suggested to divide the large circuits into smaller sub-circuits and to measure the overall SER by summing up the SERs of the sub-circuits.

A general computational framework based on probabilistic transfer matrices (PTMs) to estimate the effects of soft errors on logic circuits is developed in [18]. ADDs are used to implement PTMs. Since the size of decision diagrams grows exponentially with the circuit size, this approach is not applicable for large circuits.

Briefly, BDD-based methods suffer from the following shortcomings: 1) since they rely on implicit enumeration of the input vector space, they are time-consuming for large-scale circuits; 2) despite the use of circuit partitioning techniques, BDD-based algorithms are inherently limited due to the memory blowup problems associated with them; 3) the unified treatment of masking factors is not typically considered in such methods.

C. Boolean Difference or Satisfiability-Based Approaches

Boolean satisfiability is well suited to calculate logical masking but it is unable to model the main characteristics of a transient glitch including pulse width and height to compute electrical and timing masking factors. A SER estimation method based on Boolean difference is presented in [8]. The circuit netlist is traversed once using a post-order traversal to calculate error probabilities at the output of each gate using Boolean difference. The complexity of the proposed method is $O(N)$, where N is the number of the gates in the circuit. A similar technique based on Boolean satisfiability is also presented in [10].

D. Error Probability Propagation-Based Approaches

Developing error propagation rules for library gate and using static analysis of error propagation probabilities in the sensitized paths is another SER estimation approach found in the literature. An analytical approach to accurately estimate static logic derating in combinational circuits was proposed in our previous work [19], [20], [21]. The proposed approach gives linear computational complexity and computes the logic derating factor orders of magnitude faster than simulation-based methods. However, the proposed approach does not consider the effect of timing and electrical masking factors on the overall circuit SER. A similar technique to estimate timing derating was presented in [22].

A technique to rank system bistables according to contribution of each bistable to the overall circuit SER is presented in [23]. This is done by computing the error propagation probabilities from the fault site to circuit outputs in multi-cycle circuit operation. This analysis is only provided for circuit bistables and it does not consider electrical and timing masking factors.

We will further discuss previous techniques and compare them with our proposed technique in Sec. VI.D.

III. Proposed SER Modeling in Digital Circuits

In this section, we present an analytical technique based on an enhanced static timing analysis to unifiedly estimate all derating factors. Consider a particle strike hits gate A (as a candidate *fault site*) and creates a full swing glitch with pulse width w at time t at the gate output, as shown in Fig. 1. Based on structural paths from the fault site (the node at which the particle strike happens) to reachable primary outputs and flip-flops, we can categorize nets (signals) and gates in the circuit as follows [19], [20]. An *on-path* signal is a net on a path from the fault site to a reachable output. Also, an *on-path gate* is defined as the gate with at least one on-path input. Finally, an *off-path* signal is a net that is not on-path and is an input of an on-path gate. Off-path and on-path signals are shown in Fig. 1. The darkened gates in this figure are on-path gates.

Depending on the value of off-path signals in the circuit, the erroneous transient may or may not propagate to the input of FF_j. If it propagates, then a glitch with width w' at time t' will appear at the input of this flip-flop. $t' - t$ depends on the propagation delay along the path from gate A to FF_j, and w'

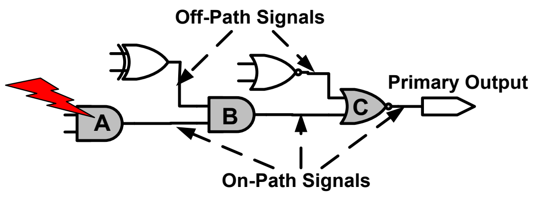


Fig. 1. Propagation of a transient through a unique path to outputs

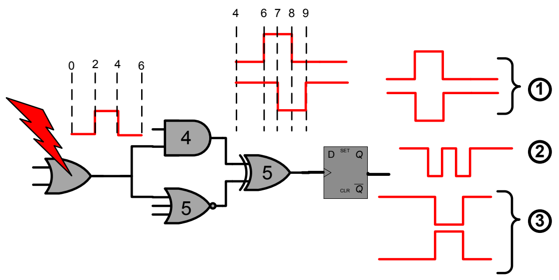


Fig. 2. Propagation of a transient through reconvergent paths

depends on the various rise and fall transition delays for the gates along this path.

For computing the *propagation probability* (PP) of the erroneous glitch, we use the estimation method presented in our previous work [19], [20]. Note that we assume that the value of all signals other than on-path signals (which propagated the erroneous glitch) are stable, i.e. no other signal is making a transition. We use this assumption throughout the paper. The effect of this assumption on the accuracy of the overall SER will be addressed in Sec. IV.C.

The *latching probability* (LP) is defined as the probability that an erroneous value is captured in a reachable flip-flop. An erroneous glitch can happen at any point during the clock period. Therefore, based on the delay of the path from a fault site to a flip-flop input, it can in turn appear at the input of a flip-flop any time during the clock period. However, an erroneous glitch can be latched and cause an error if it overlaps with the latching window of the flip-flop. For edge-triggered D flip-flops, the latching window is the sum of setup and hold times around the latching edge of the clock. Once the duration of the propagated erroneous glitch to the input of a flip-flop is obtained, LP can be calculated based on the setup and hold time of the flip-flop, glitch width, and clock period. Note LP can be derived using different equations presented in previous work [7], [24], [6], [15], [11]. Lastly, *Error Propagation Probability* (EPP) can be calculated as the product of propagation probability and latching probability, i.e., $EPP = PP \times LP$ [22].

In general, there can be multiple paths from gate g_i (fault site) to flip-flop FF_j. In this case, there is at least one gate along the path in which the transient appears on at least two inputs of that gate. In this situation, the shape of the propagated erroneous waveform due to a simple glitch at the output of g_i may not be a simple glitch. The shape of the propagated waveform depends on the particular paths which propagate the transient and relative propagation delays of these paths.

Fig. 2 shows an example in which there are multiple paths from the fault site to the flip-flop. There are three possible propagation scenarios: 1) propagation through only the AND gate, 2) propagation through only the NOR gate, and 3) propagation through both gates (or paths). Even if one considers a simple gate delay model (the gate delays are shown inside the gates in Fig. 2), there are five possible waveforms that can appear at the input of the flip-flops, plus one case of no propagation at all. The top two waveforms (case 1) at the input of the flip-flop are due to the propagation through only the AND gate, where the output of the NOR gate is either 0 or 1. If the glitch is propagated through both paths, then the shape of the waveform is not a single glitch (case 2). Finally, if the glitch is propagated through only the NOR gate, then one of the two bottom waveforms (case 3) will appear at the input of the flip-flop, depending on the output value of the AND gate.

GATE	RULE
AND	$P_1(out) = \prod_{i=1}^{n} P_1(X_i)$
	$P_a(out) = \prod_{i=1}^{n} [P_1(X_i) + P_a(X_i)] - P_1(out)$
	$P_{\bar{a}}(out) = \prod_{i=1}^{n} [P_1(X_i) + P_{\bar{a}}(X_i)] - P_1(out)$
	$P_0(out) = 1 - [P_1(out) + P_a(out) + P_{\bar{a}}(out)]$
OR	$P_0(out) = \prod_{i=1}^{n} P_0(X_i)$
	$P_a(out) = \prod_{i=1}^{n} [P_0(X_i) + P_a(X_i)] - P_0(out)$
	$P_{\bar{a}}(out) = \prod_{i=1}^{n} [P_0(X_i) + P_{\bar{a}}(X_i)] - P_0(out)$
	$P_1(out) = 1 - [P_0(out) + P_a(out) + P_{\bar{a}}(out)]$
3-State BUF	$P_1(out) = P_1(input) \times P_1(enable)$
	$P_a(out) = [P_1(input) + P_a(input)] \times [P_1(enable) + P_a(enable)] - P_1(out)$
	$P_{\bar{a}}(out) = [P_1(input) + P_{\bar{a}}(input)] \times [P_1(enable) + P_{\bar{a}}(enable)] - P_1(out)$
	$P_0(out) = 1 - [P_1(out) + P_a(out) + P_{\bar{a}}(out)]$

TABLE I
COMPUTING ERROR PROPAGATION PROBABILITY AT THE OUTPUT OF A GATE IN TERMS OF ITS INPUTS

This simple example shows that depending upon the possible propagation paths from the fault site to a reachable flip-flop, various waveforms can appear at the input of the flip-flop. For each propagation scenario, the error probability is the product of the propagation probability and the latching probability for that particular case. The overall EPP is calculated as follows:

$$EPP_{g_i \rightarrow FF_j} = 1 - \prod_{all\ propagated\ waveforms\ i} (1 - PP_i \times LP_i) \quad (1)$$

In the following subsections, we explain how to compute all possible erroneous waveforms and their corresponding propagation probabilities.

A. Modeling Logical Masking

To model logical masking, we use error propagation rules for logic gates similar to our previous work presented in [19], [20]. Here we explain how to perform a static error propagation analysis. In Sec. III.B, we expand this approach to perform a dynamic error propagation analysis, i.e. propagation of erroneous transients (glitches). In Sec. III.C, we explain the proposed model for measuring the contribution of the electrical masking in the overall circuit SER.

In the presence of errors, the status of each signal can be expressed with four values:

- *0:* no error is propagated to this signal line and the signal has an error-free value of 0.
- *1:* no error is propagated to this signal line and it has the logic value of 1.
- *a:* the signal has an erroneous value with the same polarity as the original erroneous value at the fault site (denoted by a).
- *$\bar{a}$:* the signal has an erroneous value, but the erroneous value has an opposite polarity compared to the erroneous value at the fault site (denoted by $\bar{a}$).

Based on this four-value logic, we can define error propagation rules for each logic gate. These probabilities, denoted by $P_a(U_i)$, $P_{\bar{a}}(Ui)$, $P_1(U_i)$, and $P_0(U_i)$, are defined as follows:

- $P_a(U_i)$ and $P_{\bar{a}}(Ui)$ are defined as the probabilities of the output of node U_i being a and $\bar{a}$, respectively. In other words, $P_a(U_i)$ is the probability that the erroneous value is propagated from the fault site to U_i with an even number of inversions, whereas $P_{\bar{a}}(Ui)$ is the similar propagation probability but with an odd number of inversions.
- $P_1(U_i)$ and $P_0(U_i)$ are the probabilities of the output of node U_i being 1 and 0, respectively. In these cases, the error is masked and not propagated.

According to the above definitions, for on-path signals, $P_a(U_i) + P_{\bar{a}}(Ui) + P_1(U_i) + P_0(U_i) = 1$ and for off-path signals, $P_1(U_i) + P_0(U_i) = 1$. To propagate the erroneous values through on-path gates, we use *Signal Probability* (SP) of off-path signals to include their corresponding masking factor. SP of a line k (SP_k) indicates the probability of line k having logic value one. To further clarify how signal probabilities of off-path signals can affect the masking factor, let consider an example in which an erroneous value is propagated through a two-input OR gate (IN1: off-path and IN2: on-path). In this example, a greater signal probability at IN1 will reduce the propagation probability of the erroneous value through this gate. In case SP_{IN1} is equal to one, the propagation probability from IN2 to the gate output becomes zero. To include the masking factor of off-path signals, we have developed propagation computation rules for elementary gates (AND and OR) as well as a 3-state buffer as shown in Table I. In these propagation rules, since the polarities of propagated errors are taken into account, propagation probabilities at the output of reconvergent gates are accurately estimated.

In the proposed approach, we first extract on-path gates and signals from the fault site to reachable flip-flops and primary outputs. Afterwards, starting from the fault site, we apply the propagation rules level by level for each on-path gate until we reach all reachable flip-flops and primary outputs.

B. Modeling Timing Masking

In order to model timing masking, we extract all possible erroneous waveforms at the input of each reachable flip-flop FF_j due to a glitch (with a particular width w) at the output of a gate g_i (fault site) caused by a SEU. Note that the initial transient pulse width can be determined based on the energy of the particle (the amount of injected charge), type and size of the gate, and the technology parameters [25], [1]. A glitch at the output of gate g_i starting at time t with pulse width w can be expressed as two transition events at time t and $t + w$ on the fault site, respectively. Depending upon the polarity of the glitch, the first event is a rising (falling) and the second event is a falling (rising) transition.

We use a modified version of static timing analysis in which we compute all events at the outputs of all on-path gates due to these two events at the fault site. Each event is described as a pair of time and polarity (falling or rising). Since the error-free state of gate g_i is a statistical variable, the erroneous transient could either be a positive or a negative glitch. Therefore, the injected glitch can be expressed by two events as follows: The first event can be either a falling or a rising transition. The second event has to be the opposite of the first event. This way, an erroneous transient can be described without specifying the error-free state of g_i.

Here, we use two techniques to model an injected SET. We have verified both these two techniques with a reference model which will be presented in Sec. IV.A The computed SERs using these two techniques are similar. In the first modeling technique, we denote the first event of the glitch as a and the second event as $\bar{a}$ (as the opposite of the first event). So, the events (a, t) and $(\bar{a}, t + w)$ are put at the output of gate g_i to represent an erroneous transient with pulse width w. In the second modeling technique, we denote the first event of the glitch as a and the second event as the error-free probability of the fault site which is equal to the signal probability of the fault site (SP_{g_i}). So, the events (a, t) and $(SP_{g_i}, t + w)$ are inserted at the output of gate g_i to represent an erroneous transient with pulse width w.

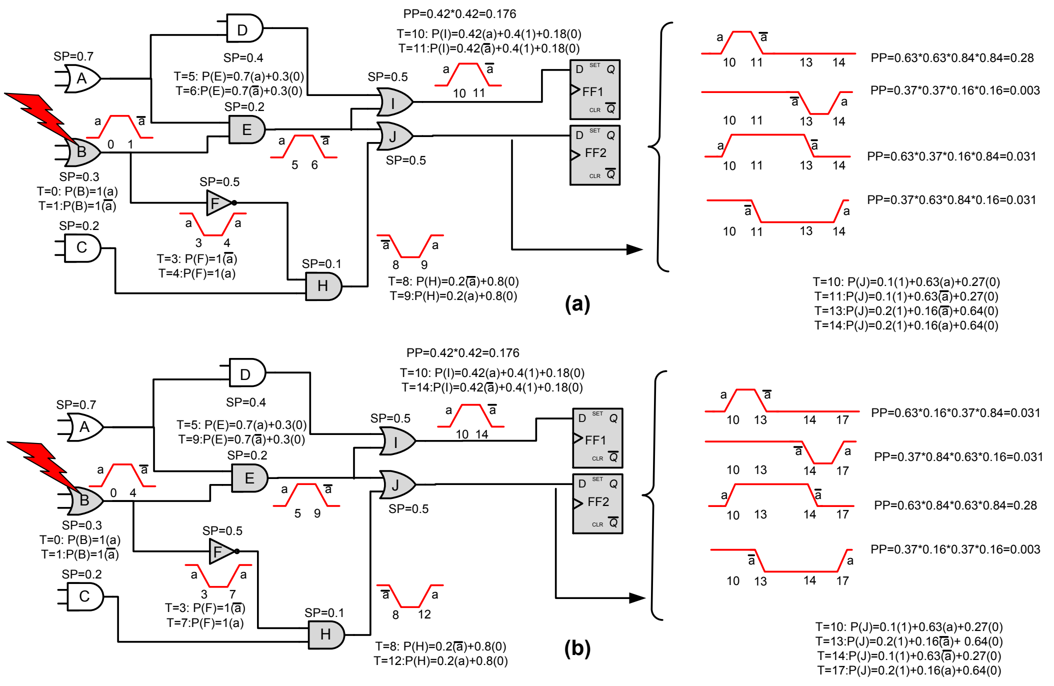
The events are propagated level by level, based on their distance from g_i. The level of each gate is defined as one plus the maximum level of its input, assuming that the level of g_i is zero. The same propagation rules presented in Table I are used starting from the fault site to all reachable flip-flops. However, we need to perform these propagation rules on timed events. The gates are processed based on their levels in their increasing order. The events at the output of each gate can be determined based on the events at its input, type of the gate, and the gate delay model. This way, one can calculate the event list $Event\_List(g_i)$ for each on-path gate g_i.

Once the event list at the input of each flip-flop FF_j is extracted, one can generate all possible waveforms that can result from propagation of an injected SET. If we use the first SET injection modeling technique ([(a, t), $(\bar{a}, t + w)$] at g_i), a propagated waveform at the FF_j input can be obtained from a series of a-event to $\bar{a}$-event (or alternatively from $\bar{a}$-event to a-event) in $Event\_List(FF_j)$. By enumerating all such series, all propagated waveforms will be calculated. As an example, consider the following event list at a flip-flop input: $\{(a, t_1), (\bar{a}, t_2), (a, t_3), (\bar{a}, t_4)\}$, where $t_1 < t_2 < t_3 < t_4$. Possible waveforms include $[(a, t_1), (\bar{a}, t_2)]$,$[(a, t_3), (\bar{a}, t_4)]$, $[(a, t_1), (\bar{a}, t_4)]$,$[(\bar{a}, t_2), (a, t_3)]$, and $[(a, t_1), (\bar{a}, t_2), (a, t_3), (\bar{a}, t_4)]$. However, $[(a, t_1), (\bar{a}, t_2), (a, t_3)]$ is not a valid waveform since it starts and ends by a-event. If one uses the second SET injection modeling technique ([(a, t), $(SP_{g_i}, t + w)$] at g_i), a propagated waveform at the FF_j input can be obtained from a series of a-event to error-free $0/1$-event (or alternatively from $\bar{a}$-event to $0/1$-event) in $Event\_List(FF_j)$.

Since all possible events will be considered in the event list of each gate, one could argue that the size of this list could be excessively large. The maximum size of the event lists has been extracted for some of the simulated circuits in our experiments. The results show that the maximum size of event lists for ISCAS'89 benchmark circuits varies between 13 (for $s298$) to 217 (for $s35932$). Therefore, the size of the event lists is tractable.

B.1 Example: SET Propagation in Reconvergent Fanouts

Fig. 3 shows two examples with different SET widths, which illustrates how we use the proposed approach to propagate a SET from a fault site to the reachable flip-flops. Note in the examples given in Fig. 3, the first SET injection modeling technique discussed earlier has been used. Therefore, the prop-



PP=0.42*0.42=0.176
T=10: P(I)=0.42(a)+0.4(1)+0.18(0)
T=11:P(I)=0.42($\bar{a}$)+0.4(1)+0.18(0)
SP=0.7
A
D
SP=0.4
T=5: P(E)=0.7(a)+0.3(0)
T=6:P(E)=0.7($\bar{a}$)+0.3(0)
SP=0.2
E
SP=0.5
I
J
SP=0.5
FF1
FF2
B
SP=0.3
T=0: P(B)=1(a)
T=1:P(B)=1($\bar{a}$)
SP=0.2
C
SP=0.5
F
T=3: P(F)=1($\bar{a}$)
T=4:P(F)=1(a)
SP=0.1
H
T=8: P(H)=0.2($\bar{a}$)+0.8(0)
T=9:P(H)=0.2(a)+0.8(0)
(a)
PP=0.63*0.63*0.84*0.84=0.28
PP=0.37*0.37*0.16*0.16=0.003
PP=0.63*0.37*0.16*0.84=0.031
PP=0.37*0.63*0.84*0.16=0.031
T=10: P(J)=0.1(1)+0.63(a)+0.27(0)
T=11:P(J)=0.1(1)+0.63($\bar{a}$)+0.27(0)
T=13:P(J)=0.2(1)+0.16($\bar{a}$)+0.64(0)
T=14:P(J)=0.2(1)+0.16(a)+0.64(0)
PP=0.42*0.42=0.176
T=10: P(I)=0.42(a)+0.4(1)+0.18(0)
T=14:P(I)=0.42($\bar{a}$)+0.4(1)+0.18(0)
T=5: P(E)=0.7(a)+0.3(0)
T=9:P(E)=0.7($\bar{a}$)+0.3(0)
T=3: P(F)=1($\bar{a}$)
T=7:P(F)=1(a)
T=8: P(H)=0.2($\bar{a}$)+0.8(0)
T=12:P(H)=0.2(a)+0.8(0)
(b)
PP=0.63*0.16*0.37*0.84=0.031
PP=0.37*0.84*0.63*0.16=0.031
PP=0.63*0.84*0.63*0.84=0.28
PP=0.37*0.16*0.37*0.16=0.003
T=10: P(J)=0.1(1)+0.63(a)+0.27(0)
T=13:P(J)=0.2(1)+0.16($\bar{a}$)+ 0.64(0)
T=14:P(J)=0.1(1)+0.63($\bar{a}$)+0.27(0)
T=17:P(J)=0.2(1)+0.16(a)+0.64(0)

Fig. 3. Example: computing the propagation probability of all possible waveforms to flip-flops: a) SET width=1, b) SET width=4

agated waveforms in Fig. 3 correspond to this SET injection modeling technique.

For the sake of simplicity, we have used a simple delay model in these examples (gates delay: OR=5, AND=5, and NOT=3). Assume an energetic particle strikes gate B and it generates a SET at the output of this logic gate. In Fig. 3a, it is assumed that the particle causes a SET with the size (or width) of one unit of time while in Fig. 3b the size of the produced SET is assumed to be four units of time. Initially, the events $(a, 0)$ and $(\bar{a}, 1)$ are inserted at the output of gate B (according to the first SET modeling technique) to represent an erroneous transient in Fig. 3a. These two events are propagated through logic gates E, F, H, I, and J according to propagation rules presented in Table I. As an example, the probability that these two events are propagated through gate E depends on the signal probability of gate A. The corresponding probabilities at time 5 and 6 are shown in Fig. 3a. Similarly, the error propagation probabilities for all logic gates in the forward cone of gate B have been shown in Fig. 3a.

Once all timed a-events and $\bar{a}$-events are propagated to the inputs of the flip-flops, the next step would be finding all possible waveforms in the inputs of all circuit flip-flops and computing the corresponding event propagation probabilities. In this example, only one waveform is propagated from the fault site (output of gate B) to the input of $FF1$. The propagation probability of this waveform is computed by the product of a-event and $\bar{a}$-event probabilities at time 10 and 11. In the given example, four possible waveforms are propagated from the fault site to the input of $FF2$. The propagation probability of each waveform is computed as shown in Fig. 3a. As an example, the first waveform is propagated to the input of FF2 if an a-event occurs at time 10, an $\bar{a}$-event occurs at time 11, and neither a-event nor $\bar{a}$-event occur at time 13 and 14. Therefore, the propagation probability of this waveform is computed by the product of the probabilities of an a-event at time 10 and $\bar{a}$-event at time 11 and the probability of having neither a-event nor $\bar{a}$-event at times 13 and 14.

As it can be seen in Fig. 3a, the SET in the output of gate B produces two glitches in the output of gate E and gate H. These two glitches converge at gate J. As the initial size of the SET is one unit of time, these two glitches have not timely overlapped in this reconvergent gate. Hence, none of these glitches affects the propagation of the other one.

To better understand how the proposed approach can accurately compute propagation probability of waveforms at reconvergent fanouts, let's increase the width of SET from one unit to four units of time. By increasing the width of the SET, the propagated waveforms from the fault site will overlap at the reconvergent point, i.e., at the inputs of gate J during the time interval of (8,9). As shown in Fig. 3b, four possible waveforms can propagate to the output of gate J. As it is shown in this figure, the output value will be a-event if the output value of gate H is zero. The output value of gate J becomes $\bar{a}$-event at time 13, if the output value of gate E is zero during this time period. During the time period 8 to 9, the SET propagates through both gate E and H and the propagated glitches converge at the inputs of gate J. Here, the output value of gate E is a-event and the output value of gate H is $\bar{a}$-event. As a result, the output value of gate J will be one as they have opposite polarities. Hence, the two glitches will cancel out each other during this time period. Note in general, propagation probability of some of waveforms can be zero. That is depending on the signal probability of off-path signals and the polarity of waveforms, not all possible waveforms can propagate to the next stage of a reconvergent fanout.

B.2 Discussion

In our proposed methodology, signal probabilities are computed using a first-order approximation in which the cross correlation of different signals are not considered. In other words, although the polarity of propagated erroneous waveforms is taken into account in the reconvergent paths, the correlation of off-path signals in the reconvergent paths is not taken into ac-

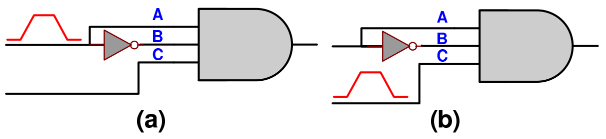


Fig. 4. Correlation of the signal values of the off-path signals

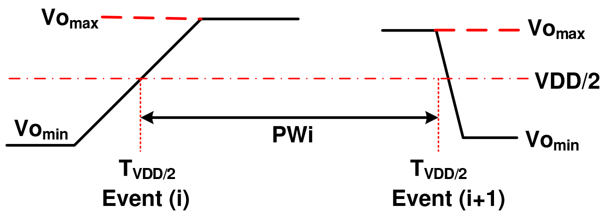


Fig. 5. SET pulse model used in the proposed approach

count in our proposed approach. To further clarify this, let's consider an example given in Fig. 4. In Fig. 4.(a), our proposed approach takes into account the polarity of the propagated glitches to A and B in the reconvergent point (the AND gate) but in Fig. 4.(b), our proposed approach does not take into account the correlation of the signal values of the off-path signals (A and B).

In addition, we have assumed that during the propagation of a transient glitch, the logic values of the off-path signals remain unchanged. In other words, all off-path signals are assumed stable all over the circuit while injecting and propagating the erroneous single event transient. Such assumption will introduce an inaccuracy in the overall failure probability computed by our proposed technique. Hence, our proposed technique is not an ”exact” method. An assumption of independent signal probabilities as well as the persistence of the values in off-path signals are the main sources of inaccuracy for our proposed technique. However, these approximations, as will be shown in Sec. IV.C, work quite accurately for even the largest circuits of ISCAS89 circuits (98% accuracy compared to the SFI timing-accurate reference model).

In our proposed technique, we also propagate erroneous waveforms to bistable inputs independent of relative timing to the clock edge. Then, we calculate the latching probability based on the overlap of propagated waveform and setup/hold time of the clock cycle. This means that in the proposed approach, we do not inherently deal with the exact timing of side inputs. Fortunately, this approximation does not introduce a significant inaccuracy in our proposed technique as it will be shown in Sec. IV.C.

Lastly, in our proposed technique, we consider the probability of capturing an incorrect value caused by a SET in system states, i.e. the error is either stored in a system bistable or propagated to a primary output. The analysis of error propagation within system bistables, in multiple clock cycles, is beyond the scope of this paper.

C. Modeling Electrical Masking

To model electrical attenuation of injected charges, we propose a technique based on the mathematical equations presented in [13]. Mathematical equations are used to model the propagation of a transient pulse from gate inputs to the gate outputs. To incorporate this electrical masking model in our framework, library-cells are pre-characterized so that the fall and rise time of library gates are pre-determined.

In this technique, the amplitude of the output pulse is calculated based on the input pulse width (PW_i) and the slopes of the output pulse (fall time, t_f, and rise time, t_r). The SET pulse is considered as a trapezoid pulse as depicted in Fig. 5. In this figure, PW is measured between 50% of signal amplitudes of rise time and fall time. Both input pulse width (PW_i) and output pulse width (PW_o) are measured using this convention, as shown in Fig. 5. In order to compute the output pulse width (PW_o), we use the analytical equations presented in [13]. After computation of the output pulse width, we add a new event to the event-list of the gate output with the corresponding output pulse width.

Note in previous study presented in [13], only attenuation of transient glitches is modeled. Attenuation is modeled by a parameterized representation of the glitch and a transfer function is used to extract the output glitch shape as it passes through combinational logic. In our proposed methodology, we use the same mathematical equations to compute the output pulse width when propagating a glitch through logic gates. In other words, attenuation of output glitch is represented by changes in the output pulse width. Once we computed possible change in the output pulse width, we feed this information into our timing-logic derating algorithm to compute the new propagation probability of waveforms.

As an example, let's consider a waveform with five units of pulse ($t1$,$t1+5$) at the input of a logic gate with an intrinsic delay of td. If this waveform is attenuated by the logic gate by one unit, we would have a waveform of ($t1+td$,$t1+td+4$) at the output of this logic gate.

In general where the input voltages are not full swing, we use the minimum output voltage (Vo_{min}) and the maximum output voltage (Vo_{max}) of the gate inputs to compute the output pulse width as follows:

- First, the maximum Vo_{min} and the minimum Vo_{max} of the gate inputs are selected as representative of the inputs Vo_{min} and Vo_{max}, respectively.
- Second, the input pulse width (PW_i) is determined by the maximum Vo_{min} and the minimum Vo_{max} of two consecutive events at the gate inputs considering the rise/fall time of the driving gate. The input pulse width is computed as the time difference of two consecutive events at the gate inputs where the voltage reaches to VDD/2. This has been illustrated in Fig. 5
- Third, the output fall/rise time (t_f and t_r) is determined based on the gate intrinsic delay, the gate load dependent delay (or transition delay) and the load capacitance using non-linear delay model which will be detailed in Sec. IV.B.
- Fourth, Vo_{min} and Vo_{max} of the gate outputs are then computed according to Equation 2 and Equation 3 [13]. In case, rise and fall time is measured between 10% to 90% of signal amplitudes, a scaling factor of 1.25 ($\frac{1}{90\%-10\%} = 1.25$) is used to account for complete swing. Note here we assume that the gate output voltage changes from Vo_{min} to Vo_{max} (or vice versa) linearly with a specific slope that is calculated using the output rise and fall time.

$$\begin{aligned} Vo_{min} &= VDD \times (1 - PW_i/(t_f \times 1.25)),\ PW_i < t_f \times 1.25 \\ &= 0, \qquad\qquad\qquad\qquad\quad PW_i > t_f \times 1.25 \end{aligned} \tag{2}$$

$$\begin{aligned} Vo_{max} &= VDD \times PW_i/(t_r \times 1.25),\ PW_i < t_r \times 1.25 \\ &= VDD, \qquad\qquad\qquad PW_i > t_r \times 1.25 \end{aligned} \tag{3}$$

- Lastly, the output pulse width (PW_o) is calculated based on the output Vo_{min} for $0 \to 1$ transition (or the output Vo_{max} for $1 \to 0$ transition), and the first and the second transition delays ($TDelay1$ and $TDelay2$) of the output pulse using Equation 4. Note in Equation 4, the scaling factor in the second term of this equation ($[VDD/2 - Vo_{min}]/VDD/2$ for $0 \to 1$ transition and $[Vomax - VDD/2]/VDD/2$ for $1 \to 0$ transition) is used since the output voltages can swing to Vo_{min} or Vo_{max}.

$$\begin{aligned} PW_o &= (PW_i - TDelay1) + \\ TDelay2 &\times (VDD/2 - Vo_{min})/(VDD/2), \quad 0 \to 1 \\ &= (PW_i - TDelay1) + \\ TDelay2 &\times (Vo_{max} - VDD/2)/(VDD/2), \quad 1 \to 0 \end{aligned} \tag{4}$$

While traversing the event list of logic gates, this electrical masking model is applied to the injected charge. The electrical masking at each stage can affect the output pulse widths and can change the circuit derating factor.

Note there are few techniques that can be used to determine the shape of radiation-induced voltage pulses from the deposited charge [26], [27]. Our proposed electrical masking model can be further extended to get an induced charge as the input and use the method of [26], [27] to obtain the shape of voltage pulse. Further discussion and trend of SET pulse widths can be found in [1], [25], [26], [27].

D. *Logic-Electrical-Timing Derating Algorithms*

Algorithm 1 shows the overall procedure to compute logic, electrical, and timing derating in our proposed technique. For each logic gate, considered as a fault site, all its structurally reachable logic gates and flip-flops are extracted first. Then, the extracted list is sorted using the topological sort algorithm (line 5). Initially, a SET with a pulse width of w and the corresponding Vo_{min} and Vo_{max} is inserted at the output of the fault site (line 6). Then, the event list as well as the probability of each event is propagated from the fault site to reachable flip-flops (lines 7 through 11). To propagate events for each logic gate (G_j), all events are added to the gate event list (line 8). Then, error propagation rules are applied for each event according to Table I (line 9). Additionally, the electrical masking of each event is computed and the corresponding pulse width is updated to the output event list (line 10).

Based on the event list at the input of each flip-flop, the possible waveforms are extracted and the corresponding waveform probability is computed to obtain the propagation and latching probabilities (lines 13 through 16). The logic-electrical-timing derating from gate G_i to an arbitrary flip-flop FF_j, denoted by $LET_{G_i \to FF_j}$, is calculated by summing up the propagation and latching probabilities of all events from gate G_i to flip-flop FF_j (line 14). Then, the overall derating factor of gate G_i, denoted by LET_{G_i}, is calculated based on the derating factor from this gate to all flip-flops (line 15).

Electrical masking of each event is computed according to Algorithm 2. In this algorithm, the maximum Vo_{min} and the minimum Vo_{max} of each valid waveform are computed first (lines 9 and 10). The input pulse width (PW_i) is calculated next as described in Sec. III.C (line 11). Next, the output fall/rise times (t_f and t_r) are computed (line 12). Vo_{min} and Vo_{max} of the gate outputs are then computed according to Equation 2

```
1  Algorithm:Computing Overall Derating Factor
2  $LET(G_i)$: Logic-Electrical-Timing derating factor of gate $G_i$
3  $LET_{G_i \to FF_j}$: Logic-Electrical-Timing derating factor from $G_i$ to
   $FF_j$
4  for each gate $G_i$ do
5      List($G_i$) $\leftarrow$ Sorted list of on-path gates reachable from $G_i$;
6      Event_List($G_i$) $\leftarrow$ Add_Initial_Events();
7      for each gate $G_j$ in List($G_i$) do
8          Event_List($G_j$).Add_event_list($G_j$ inputs);
9          Apply_propagation_rules($G_j$ input events); //Table I
10         Compute_Electrical_Masking($G_j$ input events); //Alg.2
11     end
12     LET($G_i$) $\leftarrow$ 1;
13     for each flip-flop ($FF_j$) in List($G_i$) do
14         Compute $LET_{G_i \to FF_j}$ using Event_List($FF_j$);
15         $LET(G_i) \leftarrow LET(G_i) \times (1 - LET_{G_i \to FF_j})$ ;
16     end
17     $LET(G_i) \leftarrow 1 - LET(G_i)$;
18 end
   algocf
```

Algorithm 1: Computing overall derating factors

and Equation 3 (lines 13 and 14). Lastly, the output pulse width (PW_o) is calculated as described in Sec. III.C (line 15). Description of the steps taken in this algorithm has been detailed in Sec. III.C.

Lastly, Algorithm 3 is used to compute the propagation probability of all possible waveforms from a fault site to the inputs of flip-flops. The propagation probability of a waveform k is calculated by multiplying the probability of all events within the waveform (lines 4 through 11). Note here the propagation probability of each waveform, denoted by PP_k, is computed based on the first SET injection modeling technique, explained in Sec. III.B.

```
1  Algorithm: Computing Electrical Masking Factor
2  $Vo_{min}(G_j, input_k)$: Minimum voltage of input $k$ of $G_j$
3  $Vo_{max}(G_j, input_k)$: Maximum voltage of input $k$ of $G_j$
4  $Vo_{min}(G_j)$: Minimum voltage of $G_j$ output
5  $Vo_{max}(G_j)$: Maximum voltage of $G_j$ output
6  $PW_o$: Output pulse width
7  for each gate $G_j$ in List($G_i$) do
8      for each valid waveform ($W_l$) in Event_List($G_j$) do
9          $Vo_{min}(G_j, inputs) \leftarrow Max(Vo_{min}(G_j, input_k))$;
10         $Vo_{max}(G_j, inputs) \leftarrow Min(Vo_{max}(G_j, input_k))$;
11         $PW_i(W_l) \leftarrow$ Time difference of $W_l$ and the previous event
           at $VDD/2$;
12         Compute $t_f$ and $t_r$; //NLDM delay model
13         Compute $Vo_{min}(G_j)$; //Equation 2
14         Compute $Vo_{max}(G_j)$; //Equation 3
15         Compute $PW_o$; //Equation 4
16     end
17 end
   algocf
```

Algorithm 2: Computing SET electrical masking

To compute the SER of a logic gate, each event in the gate event list is propagated through the gates in the forward cone of the fault site. Then, the SER of the logic gate is computed based on the events propagated to the inputs of the flip-flops. In general, the following steps are followed to compute the failure probability of a gate G_i:

1. *Path Construction*: Extract all on-path signals (and gates) from G_i to every reachable primary output PO_j and/or

```
Algorithm: Computing Waveform Propagation Probability
$PP_k$: Waveform $k$ Propagation Probability
$PP_k$ = 1; /*Initialize waveform propagation probability*/
for j=0; j < EventListSize($FF_i$); j = j+1 do
    if Event[j]=0 or Event[j]=1 then
        $PP_k \leftarrow PP_k \times (EventList[j].p_0 + EventList[j].p_1)$;
    end
    if Event[j]=$a$ or Event[j]=$\bar{a}$ then
        $PP_k \leftarrow PP_k \times (EventList[j].p_a + EventList[j].p_{\bar{a}})$;
    end
end
algocf
```

Algorithm 3: Computing waveform propagation probability for all possible waveforms reaching the flip-flop inputs

bistable FF_k . This is achieved using the forward *Depth-First Search* (DFS) algorithm [28].

2. *Ordering*: Prioritize signals on these paths based on their distance level using the *topological sorting* algorithm [28]. Topological sort of a directed acyclic graph is defined as an ordered list of vertices such that if there is an edge (u, v) in the graph, then u appears before v in the list [28].
3. *Propagation Probabilities Computation*: Traverse the paths in the topological order and apply the corresponding propagation rule (presented in Table I) to compute the probability for each on-path node (logic gate). Moreover, when propagating the injected pulse through the constructed path, the effect of electrical and timing masking is measured.

In the following subsection, we will explain these three steps in more detail.

D.1 Path Construction and Ordering Algorithm

1. Construct the directed graph $G(V, E)$ corresponding to the combinational part of the circuit. V is set of graph vertices (i.e., gates and flip-flops) and E is set of graph edges (i.e., nets connecting gates and flip-flops).
2. Perform the DFS algorithm to find all reachable logic gates or flip-flops of the circuit from the erroneous logic gate, G_i. This subset of nodes is denoted by V1.
3. Construct the graph $G(V1, E1)$ as defined as follows: E1 is the list of all edges (u, v) of $G(V, E)$, where both vertices u and v are in V1, i.e., $E1 = \{(u, v)|u, v \in V1\}$
4. Apply the topological sorting algorithm on graph $G(V1, E1)$. $V1_{sort}$ is the ordered list of all nodes (logic gates or flip-flops) of this graph with respect to the topological sorting. $V1_{sort} = \{U_1, U_2, U_3, ..., U_n\}$.

D.2 Error Propagation Computation

1. Start at node U_1, assuming that there is a SET pulse in one of its inputs stating with a or $\bar{a}$ event.
2. Traverse all nodes $U_i \in V1_{sort}$ from U_1 to U_n.
3. For every input signal (X_j) of node U_i and for every event in its event list do:
   - If $X_j \in V1_{sort}$, X_j is an on-path signal and $P_1(X_j)$, $P_0(X_j)$, $P_a(X_j)$, and $P_{\bar{a}}(X_j)$ have already computed.
   - If $X_j \notin V1_{sort}$, X_j is an off-path signal, i.e., X_j is not reachable from the erroneous node G_i. In this case, $P_a(X_j) = 0$, $P_{\bar{a}}(X_j) = 0$, $P_1(X_j) = SP_{X_j}$ and $P_0(X_j) = 1 - SP_{X_j}$.
4. Compute propagation probabilities according to Table I.

D.3 Time Complexity of the Proposed Algorithms

Here, we discuss the time complexity of the proposed algorithms. The construction of graph $G(V, E)$ is done in $O(|V| + |E|)$, where $|V|$ is the number of logic gates and flip-flops and $|E|$ is the number of connecting nets. Also, the complexity of both DFS and topological sorting algorithms is $O(|V| + |E|)$. So, the path construction and the ordering can be done in $O(|V| + |E|)$. As mentioned earlier, when a SET is injected to a fault site, the netlist that is topologically sorted is traversed level by level and the probability propagation rule is used to propagate the glitches to the output of the gates in each level. As the topological sort is done in $O(|V| + |E|)$, traversing of the paths and applying the propagation rules in a topologically sorted list can also be done in $O(|V| + |E|)$.

Therefore, the complexity of computing the SER of a logic gate is $O((|V| + |E|).|AVG(EventSize)|)$, where $AVG(EventSize)$ is the average size of event lists when propagating a SET from a fault site to flip-flops and primary outputs. The worst-case complexity of computing the SER of a logic gate is $O((|V| + |E|).|MAX(EventSize)|)$, where $|MAX(EventSize)|)$ is the maximum size of event lists during SET propagation along logic paths. The complexity of computing the SER of all logic gates and the overall circuit SER is $O((|V| + |E|).|V|.|AVG(EventSize)|)$. Similarly, the worst-case complexity of computing the SER of all logic gates and the overall circuit SER is $O((|V| + |E|).|V|.|MAX(EventSize)|)$.

It should be noted that both average and maximum event size is extremely lower than the total number of gates of a circuit. For example, our experimental results show that the maximum and average event size in s35932 circuit is 217 and 8, respectively, while it contains 16065 logic gates and 1728 flip-flops. Therefore, if one can ignore the event size in the above computation, the overall complexity of the proposed algorithms would be equal to $O((|V| + |E|).|V|)$.

IV. Experimental Results

A. Reference Model

In order to verify the accuracy of the proposed technique, we have developed an *Statistical Fault Injection* (SFI) engine based on Monte-Carlo (MC) simulation. For a given glitch width, we have randomly injected glitches at the output of random gates at random time during the clock period (i.e. the random variables are the stricken gate and the time of the glitch). Timing accurate simulation determines if the injected glitch can be propagated and captured in any flip-flop. In this reference model, we perform timing-accurate simulations meaning that the signals are transitioning all over the circuit based on timing of various paths with erroneous pulse injection. In other words, a SET is injected to the fault site while other signals are not stable yet. The SFI technique terminates if the accuracy of the estimated derating falls within a pre-defined confidence interval (in our experiments, the maximum variance of the estimated value is 2% and the confidence level is 99%). Note all three derating factors, logical, electrical, and timing derating, have been incorporated in the SFI technique. Therefore, results obtained by the SFI technique is considered as a reference for other proposed methodologies.

Using this reference model, we compute the three derating

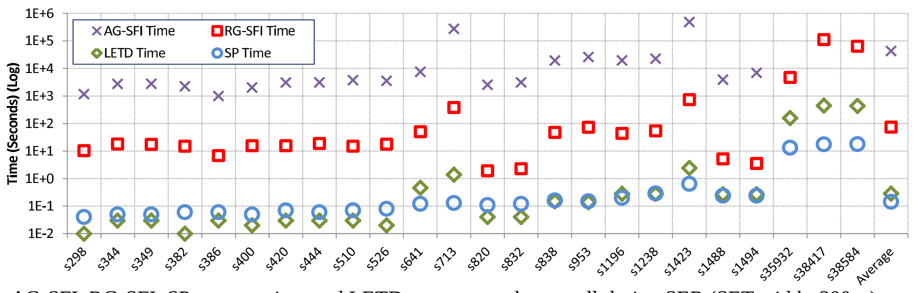


Fig. 6. Execution times for AG-SFI, RG-SFI, SP computation, and LETD to compute the overall design SER (SET width=300ps)

factors and the overall SER of small-size as well as larger-size circuits and compare them with results obtained using our proposed technique. Note even for small-size circuits, the search space is uncountable considering logical, timing, and electrical derating factors.

B. Non-Linear Delay Model

In our proposed technique, circuit delays are measured in different process technology nodes using the Standard Delay Format (SDF) [29] supported by variety of CAD tools. For calculating circuit delays, LETD supports two delay models including generic linear delay model and non-linear delay model. In linear delay model, the total delay is calculated by summing up intrinsic delay and transition delay. Intrinsic delay is cell input to cell output delay and transition delay is caused by output load capacitances. Based on the linear delay model, the propagation delay can be written as $D_{Gate} = D_{Transition} + D_{Intrinsic}$.

In this equation, $D_{Transition}$ is the transition (effort) delay and $D_{Intrinsic}$ is the intrinsic (parasitic) delay. The transition delay of gate is calculated as the product of the gate logical and electrical efforts. The electrical effort equals to $\frac{C_{load}}{C_{in}}$ where C_{in} is the input capacitance and C_{load} is the sum of the load capacitance at the gate output. The logical effort, intrinsic delay, and gate capacitance are provided by the cell library. Consequently, the rise and fall delays can be expressed as Equation 5. In this equation, $D_{Intrinsic}$ is the rise (fall) intrinsic delay, C_{Load} is the total load capacitance of the gate output, and R is the equivalent output resistance.

$$\begin{aligned} D_{rise} &= D_{Intrinsic(rise)} + R \times C_{Load} \\ D_{Fall} &= D_{Intrinsic(Fall)} + R \times C_{Load} \end{aligned} \tag{5}$$

Most of current standard cell libraries provided by foundries or standard cell design houses (including libraries of UDSM technology) include table models to specify the delays of various timing arcs of a cell. These table models are usually known as *Non-Linear Delay Model* (NLDM) since non-linear variation of delay with input transition time and load capacitance are expressed in such tables [29]. In *NLDM*, the cell delay is modeled as a function of input transition time and output load capacitance. This data is stored in a two-dimensional lookup-table where the table row shows the input transition time and the table column indicates the output load capacitance.

C. Results

The proposed technique was implemented and applied to ISCAS89 sequential benchmark circuits for a 45nm standard library. All experiments have been performed on a system equipped with an eight-core Intel XEON processor (running at 2GHz), 8MB L2 cache system, and 4GB main memory. We have extracted the failure probability or EPP with respect to logical, timing, and electrical derating factors using the SFI technique (the reference model detailed in Sec.IV.A) in two cases: a) Random Gates SFI (RG-SFI) in which a random subset of gates in the circuit is selected and fault injection is performed only for a subset of gates in order to obtain overall EPP; and b) All-Gates SFI (AG-SFI) in which per gate soft error rate for all gates in the design is calculated using SFI (by injecting erroneous pulse at random time under random vectors applied) and the overall EPP is calculated as $\frac{1}{n}\sum_{i=1}^{n} EPP(G_i)$, where n is the total number of logic gates.

Fig. 6 shows the run-time for SFI techniques (RG-SFI and AG-SFI) and LETD (as well as the SP computation time). Note that the Y-axis in this figure is logarithmic. Compared to RG-SFI, LETD is two to three orders of magnitude faster. LETD is five to six orders of magnitude faster than AG-SFI. Please note that AG-SFI is intractable for larger benchmark circuits (those not reported in Fig. 6). For example, it takes more than four weeks for largest ISCAS'89 circuits while LETD takes only few minutes.

Fig. 7 shows accuracy of LETD compared to the SFI technique. The accuracy is reported in terms of the failure probability computed by LETD and the reference SFI technique. The accuracy is compared using various SP variances (0.02, 0.04, and 0.08). Note that the run-time for SP estimation is exponentially related to the required accuracy of the values. However, these results confirm that the overall accuracy of the proposed approach is not considerably sensitive to the accuracy of the SP values. Hence, SPs with higher variances, extracted in less run-time, can be used to achieve the overall failure probability with a reasonable accuracy.

According to the results reported in Fig. 7, the average difference between the failure probabilities obtained by LETD and the SFI technique is 0.02. Note the inaccuracy results can be reported in two ways: a) absolute difference between the failure probabilities and b) percentage difference of the failure probabilities. In the context of SER estimation, the absolute difference of the failure probability is more meaningful than the inaccuracy percentage (%) as reporting the inaccuracy percentage does not reflect how the proposed method follows the reference model.

Fig. 8 shows the failure probability for different pulse widths including 200psec, 400psec, and 600psec injected on ISCAS89 circuits implemented using a 45nm standard library. As it may be difficult to accurately provide the pulse width for a particular process, one can obtain the minimum and maximum failure probability with respect to a minimum and maximum interval for the duration of the injected erroneous pulse. A straightforward solution is to execute the proposed approach twice, once

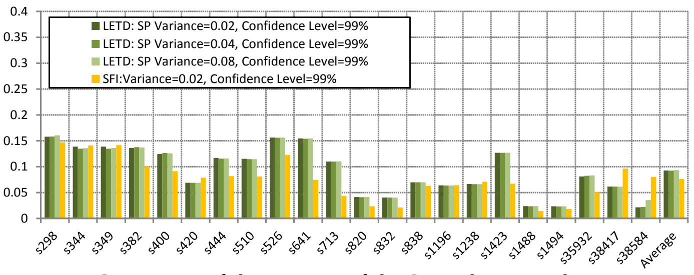


Fig. 7. Comparison of the accuracy of the SFI technique with LETD using different SP variances (SET width=300ps)

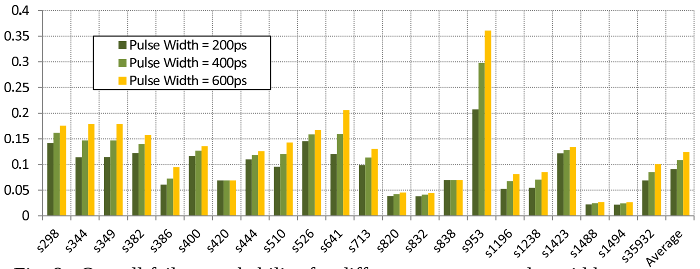


Fig. 8. Overall failure probability for different erroneous pulse widths

with w_{min} and another time with w_{max}, where w_{min} and w_{max} are the minimum and maximum widths of the injected pulse due to a SEU, and obtain two (minimum and maximum) failure probabilities.

Fig. 9 shows the contribution of each derating factor in the overall failure probability for a 300psec SET injected on ISCAS89 circuits implemented using a 45nm standard library. In this figure, we report the failure probability of ISCAS89 circuits with respect to: a) logical derating, b) logical and timing derating, and c) logical, timing, and electrical derating. As expected, the failure probability reported with respect to logical derating is greater than the failure probability reported with respect to both logical and timing derating factors. Also, the failure probability with respect to all derating factors is smaller compared to the former cases. The results confirm that considering only logical derating or logical and timing derating factors can result in considerable over-estimation of the design SER.

D. *Comparison of Our Proposed technique with Other Methods*

Here, we compare existing SER estimation methods with our proposed technique considering the following factors. The comparison is shown in Table II. The head row of the table lists SER estimation methods discussed in Sec. II. The first column of the table lists important features that a SER estimation should possess.

Since previous methods have used different measures to report the inaccuracy, we have converted all inaccuracy results to absolute difference in Table II. Briefly, the main conclusions from Table II can be summarized as follows:

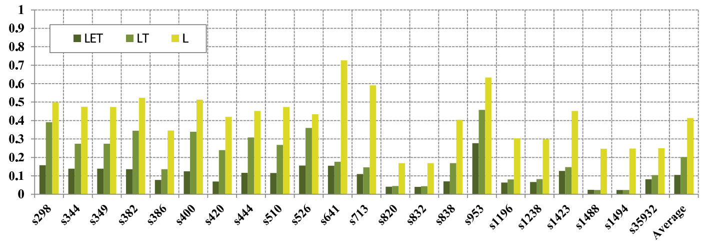


Fig. 9. Contribution of derating factors in the overall failure probability (SET width=300ps)

- All previous methods except [5] and [14] do not compute the overall SER in a unified framework.
- Although the methods in [5], [6], [7] can accurately compute SERs in reconvergent fanouts, they fail to scale for large circuits.
- Although the methods presented in [8] and [16] can accurately compute SERs in reconvergent fanouts for large circuits, they only consider logical masking.
- All previous methods except the method presented in [7] have reported the inaccuracy only for small circuits including very few number of reconvergent paths.

Please note the method presented in [14] uses SFI and logic simulation to account for logical masking. Since SFI can be intractable for large circuits, a limited number of input vectors is used for fault simulation. Note SFI with lower confidence levels and smaller variances is exponentially less time-consuming. In addition, the proposed method in [14] considers the effect of three derating factors in their SER estimation separately (i.e., not in a unified framework). Additionally, the inaccuracy results have been reported only for small circuits including very few number of reconvergent paths. As the main source of inaccuracy in the proposed technique in [14] is disregarding the effect of reconvergent paths, the inaccuracy would significantly increase for SER estimation of large circuits including very large number of reconvergent paths.

V. Conclusions and Future Work

In this paper, we have presented a methodology and algorithms to accurately compute logic, electrical, and timing derating of digital circuits. The proposed method uses an enhanced static timing analysis to derive all possible waveforms propagated from a struck gate to reachable flip-flops and calculates the probability of latching an incorrect value in a flip-flop. Experimental results and comparison versus timing accurate SFI technique show that our proposed technique is orders of magnitude faster than the SFI technique while the average difference is almost 2%.

As an extension of this work, the unified treatment of all derating factors can be considered in the presence of multiple event transients. Additionally, the proposed technique can be further extended to compute SERs multi cycles of circuit operation after incidence of single or multiple event transients.

References

[1] M.J. Gadlage, J.R. Ahlbin, B. Narasimham, V. Ramachandran, C. A. Dinkins, N.D. Pate, B.L. Bhuva, R.D. Schrimpf, L.W. Massengill, R.L. Shuler, and D. McMorrow. Increased single-event transient pulsewidths in a 90-nm bulk cmos technology operating at elevated temperatures. *IEEE Transactions on Device and Materials Reliability*, 10(1):157–163, March 2010.

[2] R.C. Baumann. Radiation-induced soft errors in advanced semiconductor technologies. *IEEE Transactions on Device and Materials Reliability*, 5(3):305–316, September 2005.

[3] B. Narasimham, M.J. Gadlage, B.L. Bhuva, R.D. Schrimpf, L.W. Massengill, W.T. Holman, A.F. Witulski, and K.F. Galloway. Test circuit for measuring pulse widths of single-event transients causing soft errors. *IEEE Transactions on Semiconductor Manufacturing*, 22(1):119–125, February 2008.

[4] Q. Zhou and K. Mohanram. Gate sizing to radiation harden combinational logic. *IEEE Transactions on CAD*, pages 155–166, January 2006.

[5] N. Miskov-Zivanov and D. Marculescu. Modeling and optimization for soft-error reliability of sequential circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27:803–816, May 2008.

Feature/Method	[7]	[16]	[13]	[11]	[8]	[5]	[12]	[14]	[6]	[15]	LETD
Reconv. Fanout	Yes	Yes	No	No	Yes	Yes	No	No	Yes	No	Yes
LET Derating†	LET	L	LET	LET	L	LET	LET	LET	LET	LET	LET
Unified	No	No	No	No	No	Yes	No	Yes	No	No	Yes
Run-Time$\star$	$6m$	$0.08s$	$9h$	$< 4m$	$< 1ms$	NR$**$	$< 2m$	$0.02s$	$25m\odot$	$< 10s$	$< 1s$
Scalability‡	1s:11h 4n:7kn$\bigtriangledown$	$< 12s$ 100n:125kn	2h:9h 92n:1kn	4m:4h 2kn:16kn	$< 10ms$ 22n:5kn	1s:70s 10n:0.5kn	1m:10h 5n:25kn	$< 1s$ 59n:25kn	6s:2h 12n:2kn	16s 3kn	1s:7m 133n:20kn
Inaccuracy:sm.$\lhd$	$< 1\%$	$< 1\%\otimes$	<2%	NR	$< 1\%\otimes$	$< 2\%$	$< 1\%\otimes$	$< 3\%$	NR	$< 1\%$	$< 2.5\%$
Inaccuracy:lg.$\rhd$	$< 1\%$	NR	NR	NR	NR	NR	NR	NR	NR	NR	$< 2\%$

TABLE II

COMPARISON OF OUR PROPOSED TECHNIQUE (LETD) WITH PREVIOUS SER ESTIMATION METHODS

† L: Logic derating; E: Electrical derating; T: Timing derating $**$ NR: Not Reported
$*$ Run-time for circuits including about 1000 gates. $\bigtriangledown$ n: Node, kn: 1000 Nodes
‡ Minimum and maximum size of simulated circuits $\odot$ Works only for hierarchical designs (e.g., adders)
$\lhd$ Inaccuracy results for circuits with less than 1000 gates $\rhd$ Inaccuracy results for circuits with more than 10,000 gates
$\otimes$ Accuracy compared with a reference model that considers only logic derating

[6] K. Ramakrishnan, R. Rajaraman, N. Vijaykrishnan, Y. Xie, M. J. Irwin, and K. Unlu. Hierarchical soft error estimation tool (hseet). In *Proceedings of the 9th International Symposium on Quality Electronic Design (ISQED)*, pages 680–683, 2008.

[7] Yu-Hsin Kuo, Huan-Kai Peng, and C.H. Wen. Accurate statistical soft error rate (sser) analysis using a quasi-monte carlo framework with quality cell models. In *Proc. of the IEEE Intl. Symposium on Quality Electronic Design (ISQED)*, pages 831–838, 2010.

[8] N. Mohyuddin, E. Pakbaznia, and M. Pedram. Probabilistic error propagation in logic circuits using the boolean difference calculus. In *Proc. of the IEEE Intl. Conference on Computer Design (ICCD)*, pages 7–13, 2008.

[9] S. Krishnaswamy, I.L. Markov, and J.P. Hayes. On the role of timing masking in reliable logic circuit design. In *Proc. of the ACM/IEEE Design Automation Conference (DAC)*, pages 924–929, 2008.

[10] S.Z. Shazli and M.B. Tahoori. Using boolean satisfiability for computing soft error rates in early design stages. *Microelectronics Reliability*, 50(1):149–159, 2010.

[11] D. Holcomb, W. Li, and S. A. Seshia. Design as you see fit: System-level soft error analysis of sequential circuits. In *Proceedings of the IEEE/ACM International Conference on Design, Automation and Test in Europe (DATE)*, pages 785–790, April 2009.

[12] M. Zhang and N. R. Shanbhag. Soft-error-rate-analysis (sera) methodology. *IEEE Transactions on CAD*, 25(10):2140–2155, October 2006.

[13] R. Rajaraman, J.S. Kim, N. Vijaykrishnan, Y. Xie, and M.J. Irwin. Seat-la: A soft error analysis tool for combinational logic. In *Proceedings of the 19th International Conference on VLSI Design*, January 2006.

[14] R.R. Rao, K. Chopra, D.T. Blaauw, and D.M. Sylvester. Computing the soft error rate of a combinational logic circuit using parameterized descriptors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(3):468–479, March 2007.

[15] C. Zhao, X. Bai, and S. Dey. Evaluating transient error effects in digital nanometer circuits. *IEEE Transactions on Reliability*, 56(3):381–391, September 2007.

[16] S. Krishnaswamy, S.M. Plaza, I.L. Markov, and J.P. Hayes. Signature-based ser analysis and design of logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28:74–86, 2009.

[17] B. Zhang, W.S. Wang, and M. Orshansky. Faser: Fast analysis of soft error susceptibility for cell-based designs. In *Proceedings of the 7th International Symposium on Quality Electronic Design (ISQED)*, pages 760–765, March 2006.

[18] S. Krishnaswamy, G.F. Viamontes, I.L. Markov, and J.P. Hayes. Accurate reliability evaluation and enhancement via probabilistic transfer matrices. In *Proc. of the IEEE/ACM Intl. Conference on Design, Automation and Test in Europe (DATE)*, pages 282–287, 2005.

[19] G. Asadi and M. B. Tahoori. An accurate ser estimation method based on propagation probability. In *Proceedings of the IEEE/ACM International Conference on Design, Automation and Test in Europe (DATE)*, pages 306–307, March 2005.

[20] G. Asadi and M. B. Tahoori. An analytical approach for soft error rate estimation in digital circuits. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 3, pages 2991–2994, May 2005.

[21] H. Asadi and M. B. Tahoori. Soft error modeling and remediation techniques in asic designs. *Microelectronics Journal*, 41(8):506–522, 2010.

[22] H. Asadi and M. B. Tahoori. Soft error derating computation in sequential circuits. In *Proc. of the IEEE Intl. Conference on Computer Aided Design (ICCAD)*, pages 497–501, November 2006.

[23] H. Asadi and M. B. Tahoori. Soft error modeling and protection for sequential elements. In *Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, pages 463–471, October 2005.

[24] H.T. Nguyen and Y. Yagil. A systematic approach to ser estimation and solutions. In *Proc. of the Intl. Reliability Physical Symposium (IRPS)*, pages 60–70, 2003.

[25] M.J. Gadlage, J.R. Ahlbin, B. Narasimham, V. Ramachandran, C. A. Dinkins, B.L. Bhuva, R.D. Schrimpf, and R.L. Shuler. The effect of elevated temperature on digital single event transient pulse widths in a bulk cmos technology. In *IEEE International Reliability Physics Symposium*, pages 170–173, 2009.

[26] R. Garg, C. Nagpal, and S. P. Khatri. A fast, analytical estimator for the seu-induced pulse width in combinational designs. In *Proceedings of Design Automation Conference (DAC)*, pages 918–923, 2008.

[27] R. Garg and S.P. Khatri. Efficient analytical determination of the seu-induced pulse shape. In *Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 461–467, 2009.

[28] T. H. Cormen, C. L. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press & McGraw-Hill, 2nd edition, 2001.

[29] http://www.opensourceliberty.org. 2009.