

Timing-Logic Derating Computation Using Event Propagation Probabilities

Hossein Asadi

ECE Dept., Northeastern Univ.
gasadi@ece.neu.edu

Mehdi B. Tahoori

ECE Dept., Northeastern Univ.
mtahoori@ece.neu.edu

Abstract—Timing derating is one of the key factors to compute the soft error rate of a sequential circuit. We present a very fast and accurate approach based on enhanced static timing analysis and signal probability to estimate the probability of latching an incorrect value in the system bistables (*timing derating*). Experimental results and comparison with Monte-Carlo simulations show that the accuracy of our approach is within 2% of the simulation-based method while orders of magnitude faster.

I. INTRODUCTION

Soft errors due to *Single Event Upsets* (SEU) are the main reliability threat of digital systems. *Soft Error Rate* (SER) for a device is defined as the error rate due to SEUs. Accurate SER estimation is essential to develop efficient soft error tolerant schemes and to determine the contribution of circuit nodes to the overall system SER. Estimation of SER in sequential circuit is very challenging since computation of the probability of erroneous system state requires dynamic analysis of transients. To compute the error rate of a node n_i in a digital circuit, three probability factors have to be computed [6]: *Nominal FIT* \times *Logic Derating* \times *Timing Derating*.

An erroneous system state occurs in the following scenario. particle strike must cause a glitch at the output of the gate (Nominal FIT), this glitch has to propagate through the combinational logic to the flip-flop inputs (Logic Derating), and finally this erroneous glitch must be captured in a flip-flop, i.e. the erroneous transient must have a sufficient overlap with the latching window of the flip-flop (Timing Derating).

Timing derating depends on the width of a glitch caused by a particle strike at the output of the gate, the latching windows of reachable flip-flops, and the propagation delay from the output of the victim gate to the input of reachable flip-flops. Moreover, the propagation probability of the erroneous glitch (from logic perspective) must be also carefully considered since propagation of transients through reconvergent paths might be blocked depending on the logic state of the circuit. Unlike logic derating which requires static analysis, timing derating requires dynamic analysis of transient propagation. Therefore, fault injection method for timing derating estimation requires timing-accurate simulation, which makes this approach orders of magnitude more tedious (and less accurate) than logic derating estimation.

Previous logic derating estimation methods use fault injection based on random vector simulation approaches [5], [6], [7], [8]. The execution time for logic derating estimation of a node in large circuits exponentially increases with the size of the circuit. Hence, logic derating estimation of larger circuits becomes intractable and very inaccurate us-

ing fault injection techniques. An analytical approach to accurately estimate logic derating in combinational circuits was proposed in our previous work [1], [2]. The proposed method gives linear computational complexity and computes the logic derating factor orders of magnitude faster than simulation-based methods.

There has been also previously published methods on probabilistic testability analysis [3], [4], which can be utilized to compute the logic derating factor. These methods either sacrifice accuracy in reconvergent fanouts or give non-linear computational complexity [3], [4].

In this paper, we present a technique for logic-timing derating estimation of sequential circuits. We use an enhanced static timing analysis method to compute all propagated waveforms from a struck gate to reachable flip-flops and calculate the probability of latching an incorrect value in a flip-flop (i.e. incorrect system state). We also exploit a technique based on signal probability to estimate propagation probabilities. The rest of this paper is organized as follows. In Sec. II, the proposed logic and timing derating estimation approach is presented. In Sec. III, experimental results are presented. Finally, Sec. IV concludes the paper.

II. TIMING-LOGIC DERATING ESTIMATION

If a particle with sufficient energy hits a particular gate and cause a bit flip at the output of this gate, we call this gate as *error site*. Based on structural paths from the error site to reachable primary outputs and flip-flops, we can categorize nets (signals) and gates in the circuit as follows [1], [2]. An *on-path* signal is a net on a path from the error 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.

Assume that a particle strike creates a glitch with pulse width w at time t at the output of gate g_i . Also, assume that there is only one path from this gate to flip-flop FF_j . Depending on the value of other signals in the circuit, this erroneous transient may or not propagated to the input of FF_j . If propagates, 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 g_i to FF_j , and w' depends on the various rise and fall transition delays for the gates along this path. The other glitch parameter is pulse magnitude which may be attenuated when propagating along the logic gates. If the glitch is attenuated by 40% below the VDD voltage, the glitch doesn't pass through the next logic gates.

For computing the *propagation probability* (PP) of the erroneous glitch, we use the estimation method presented in our previously work [1], [2]. Consider the example shown

in Fig 1 in which there is only one path from the error site to an output. As we traverse this path gate by gate, the propagation probability from an on-path input of a gate to its output depends on the type of the gate and the signal probability of other off-path signals. In this example, the propagation probability for the glitch to the output of the gate D (AND gate) is the product of the probability of the output of gate B being 1 and the propagation probability at its input ($1 \times 0.2 = 0.2$). Similarly, propagation probability at the output of the gate E (OR gate) is calculated as $0.2 \times (1 - SP_C) = 0.2 \times 0.6 = 0.12$. 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.

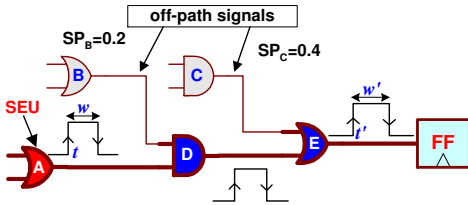


Fig. 1. Propagation of transient through a unique path to a flip-flop

Once the duration of the propagated erroneous glitch to the input of a flip-flop is calculated, the *latching probability* (LP), the probability that an erroneous value is captured in the flip-flop, can be calculated based on the setup (S) and hold (H) time of the flip-flop, glitch width (W), and clock period (T). Figure 2 shows the latching window. *Error propagation probability* (EPP) is calculated as the product of propagation probability and latching probability, i.e. $EPP = PP \times LP$.

$$LP = \frac{S + H + W}{T} \quad (1)$$

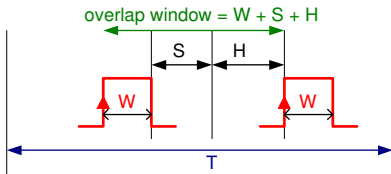


Fig. 2. Latching window

In the general case, there can be multiple paths from gate g_i (error site) to flip-flop FF_j . In this case, there is at least one gate along the path in which the transient can appear on at least two inputs of that gate. In this situation, the shape of the propagated 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.

Figure 3 shows an example in which there are multiple paths from the error site to the flip-flop. There are three

possible propagation scenarios: 1) propagation through NAND gate only, 2) propagation through OR gate only, and 3) propagation through both paths. Even if we consider simple gate delay model (the delay of each gate is shown inside the gate in this figure), there are five possible waveforms that can appear at the input of the flip-flops. The top waveform at the input of the flip-flop, as an example, is due to the propagation through NAND gate only.

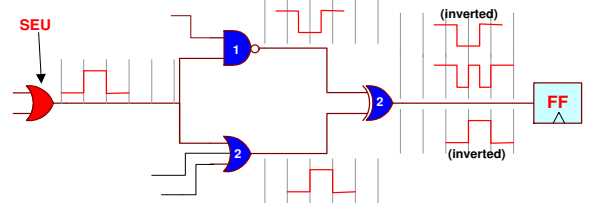


Fig. 3. Propagation of transient through reconvergent paths

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

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

Next, we explain how to compute all possible erroneous waveforms and propagation probabilities.

A. Propagation Probability

Here we explain how to perform a static error propagation analysis. In Sec. B, we expand this approach to perform a dynamic error propagation analysis, i.e. propagation of erroneous transients (glitches).

In general network of logic gates in which there are reconvergent paths from an error site to a particular flip-flop or output, the polarity of propagated erroneous value (with respect to the erroneous value at the error site) must be considered. Therefore, the propagation probability from the error site to the output of a reconvergent gate depends on not only the type of the gate and the signal probabilities of the off-path signals, but also the polarities of the propagated error on the on-path signals. In the presence of errors, each signal can take four values:

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

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	$P_0(out) = \prod_{i=1}^n P_0(X_i)$
	$P_a(out) = \prod_{i=1}^n [P_0(X_i) + P_a(X_i)] - P_0(out)$
	$P_{\bar{a}}(out) = \prod_{i=1}^n [P_0(X_i) + P_{\bar{a}}(X_i)] - P_0(out)$
	$P_1(out) = 1 - [P_0(out) + P_a(out) + P_{\bar{a}}(out)]$

TABLE I

OUTPUT PROPAGATION PROBABILITY RULES FOR ELEMENTARY GATES

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

- $P_a(U_i)$ ($P_{\bar{a}}(U_i)$) is the probability that the erroneous value is propagated from the error site to U_i with an even (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, i.e., the error is masked.

$$P(U_i) = P_a(U_i) + P_{\bar{a}}(U_i) + P_1(U_i) + P_0(U_i) = 1 \quad (2)$$

The propagation computation rules for *AND*, *OR*, and *NOT* gates are shown in Table I. Similarly, propagation rules for other logic gates can be derived easily. More detailed description can be found in [1] and [2].

B. Latching Probability

The objective here is to compute all possible erroneous waveform 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 caused by a SEU. Note that the 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. A glitch at the output of gate g_i starting at time t with pulse width w can be expressed as two transition events one at time t and other one at time $t + w$ on the output of this gate. Depending upon the polarity of the glitch, the first event is a rising (falling) transition 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 original events. Each event is described as a pair of time and polarity (falling or rising). This can be done by levelizing the gates reachable from g_i . The gates are processed based on their combinational levels in the 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.

Since all possible events will be considered in the event list for each gate, one could argue that the size of this list

could be excessively large. We looked at the maximum size of the event list for some of the simulated circuits in our experiments. Our results show that the maximum sizes of event list for benchmark circuits s298, s386, and s526 from ISCAS'89 benchmark suite are 13, 9, and 13, respectively. Therefore, the size of the event lists is not an issue.

Since the error-free state of gate g_i is a statistical variable, the erroneous transient could be either a positive or 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, we can describe an erroneous transient without specifying the error-free state of g_i . We use a notation similar to what is used in Sec. A. We denote the first event of the glitch as a and the second glitch as \bar{a} (as the opposite of the first event). So, we put the events (a, t) and $(\bar{a}, t + w)$ at the output of gate g_i to represent an erroneous transient with pulse width w . We then use the same propagation rules presented in Table I level by level starting from the error site to all reachable flip-flop. However, we need to perform these propagation rules on timed events. This way, we can calculate the event list $Event_List(g)$ for each on-path gate g .

Once the list of all possible events at the input of each flip-flop FF_j is calculated, we can generate all possible waveforms that can be resulted from propagation of $[(a, t), (\bar{a}, t + w)]$ from gate g_i . A possible waveform can be resulted from an event a to an \bar{a} event (or alternatively from an event \bar{a} to an a event) in $Event_List(FF_j)$. A recursive search generates all valid waveforms from a list of a and \bar{a} events.

Figure 4 shows how we use the this approach to propagate all events from the error site to the reachable flip-flop. Once all timed a and \bar{a} events are propagated to the input of the flip-flop, all possible waveform and propagation probabilities of each of them are calculated.

III. EXPERIMENTAL RESULTS

In order to verify the accuracy of the proposed technique, we have also developed a simulation-based fault injection engine using Monte-Carlo 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. The Monte-Carlo simulation 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 3%). However, since the Monte-Carlo simulation is too time consuming, we stop the simulation once the number of simulated gates exceeds some pre-defined threshold. For our simulation, this bound is 10,000 iterations.

The proposed approach was implemented and applied to ISCAS89 sequential benchmark circuits. Table II shows the run time for both Monte-Carlo simulation and our pro-

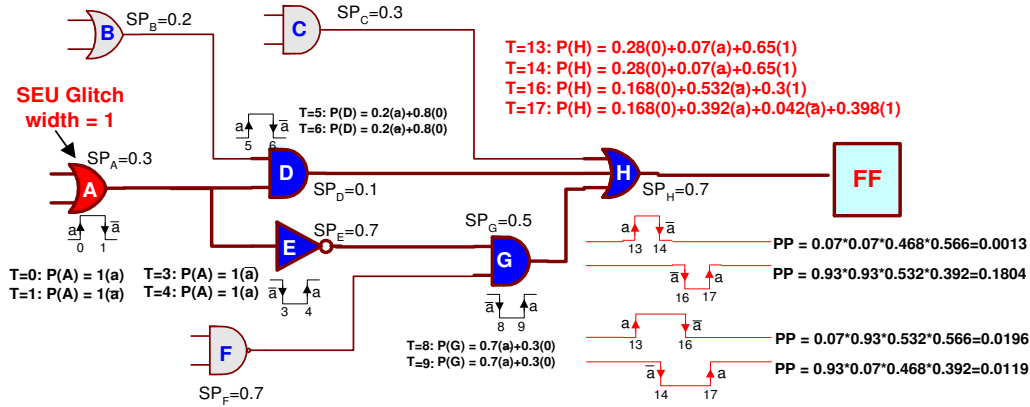


Fig. 4. Example: Event propagation, generation of all possible propagated waveform, and propagation probabilities

Circuit	#Gates	Run time(sec)		Speedup	w=50ps		w=70ps	
		MC Sim	Proposed		% Diff	Sim Status	% Diff	Sim Status
s298	119	4751	0.26	18272	0.47	C	5.03	NC
s344	160	2517	0.35	7191	1.81	C	3.30	NC
s386	159	4187	0.15	27913	0.91	C	0.51	C
s400	164	4237	1.5	2825	2.45	NC	1.01	NC
s526	193	7848	1.6	4905	2.59	NC	3.48	NC
s1196	529	3328	2.2	1513	0.45	C	1.47	C
s1238	508	9174	3.3	2780	1.49	C	0.92	C
s1423	657	17238	3.9	4420	1.11	C	2.25	C
s1488	653	11562	4.8	2409	1.12	C	1.29	C
s35932	16065	N/A	537	-	-	NC	-	NC
s38417	19253	N/A	671	-	-	NC	-	NC
s38584	22179	N/A	645	-	-	NC	-	NC
average	-	7204	155	8025	1.37	-	2.14	-

TABLE II

PROPOSED APPROACH VERSUS MONTE CARLO SIMULATION

C: MONTE CARLO SIMULATION COMPLETED WITHIN 3% CONFIDENCE INTERVAL, NC: MONTE CARLO SIMULATION ABORTED

posed approach as well as the accuracy of the proposed approach for different pulse widths (50 psec due to particle charge of 0.12pC and 70 psec due to particle charge of 0.13pC in 250nm technology). Note that the maximum difference between the proposed method and Monte Carlo simulation happens when the simulation didn't reach the confidence interval. Therefore, this difference is mainly due to inaccuracy of Monte Carlo simulation. Monte-Carlo fault injection method was completely intractable for larger circuits (Not Available (N/A) entries in this table). However, the run time of our approach for the largest ISCAS'89 circuits is only 11 minutes.

IV. CONCLUSIONS

In this paper, we have proposed a logic and timing de-rating estimation method in sequential circuits. The proposed technique 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. We also exploit a technique based on signal probability to estimate propagation probabilities. Experimental results and comparison versus timing accurate Monte Carlo simulation show that our proposed technique is 4-5 order of magnitude faster

than Monte-Carlo simulation method while the difference is less than 2% on average.

REFERENCES

- [1] H. Asadi and M. B. Tahoori, "An Accurate SER Estimation Method Based on Propagation Probability," Proc. Design Automation and Test in Europe (DATE) Conf., pp.306-307, 2005.
- [2] H. Asadi and M. B. Tahoori, "An Analytical Approach for Soft Error Rate Estimation In Digital Circuits," Proc. IEEE Int'l Symp. on Circuits and Systems, 2005.
- [3] S. K. Jain and V. D. Agrawal, "STAFAN: An Alternate to Fault Simulation," Proc. of the 21st Design Automation Conference, pp. 18-23, 1984.
- [4] W. B. Jone and S. R. Das, "CALOP - A Random Pattern Testability Analyzer," IEEE Trans. on Systems, Man and Cybernetics, vol. 25, May 1995.
- [5] K. Mohanram and N. A. Touba, "Cost-Effective Approach for Reducing Soft Error Failure Rate in Logic Circuits," Proc. Int'l Test Conf., pp. 893-901, 2003.
- [6] H. T. Nguyen and Y. Yagil, "A Systematic Approach to SER Estimation and Solutions," Proc. Intl. Reliability Physical Symp., 2003.
- [7] M. Zhang and N. R. Shanbhag, "A Soft Error Rate Analysis (SERA) Methodology," Proc. Intl. Conf. on Computer-Aided Design (ICCAD), pp. 111-118, 2004.
- [8] Q. Zhou and K. Mohanram, "Cost-Effective Radiation Hardening Technique for Combinational Logic," Proc. Int'l Conf. on Computer-Aided Design, 2004.