Soft Error Derating Computation in Sequential Circuits

Hossein Asadi
Northeastern University, ECE Dept.
Boston, MA 02115
gasadi@ece.neu.edu

Mehdi B. Tahoori
Northeastern University, ECE Dept.
Boston, MA 02115
mtahoori@ece.neu.edu

ABSTRACT

Soft error tolerant design becomes more crucial due to exponential increase in the vulnerability of computer systems to soft errors. Accurate estimation of soft error rate (SER), the probability of system failure due to soft errors, is a key factor in design of cost-effective soft error resilient systems. We present a very fast and accurate approach based on enhanced static timing analysis and signal probabilities to estimate the probability of latching an incorrect value in the system bistables (*timing derating*). Experimental results and comparison with fault injections using timing accurate Monte-Carlo simulations show that the accuracy of our approach is within 1% while orders of magnitude faster.

Categories and Subject Descriptors

B.2.3 [**Performance and Reliability**]: Reliability, Testing, and Fault-Tolerance

1. INTRODUCTION

Improvements in device scaling, transistor density and system speed of CMOS technology come at the expense of increased vulnerability of these systems to soft errors. *Soft errors*, also known as *Single Event Upsets* (SEUs), are the main reliability threat of digital systems. *Soft Error Rate* (SER) is defined as the system failure rate due to SEUs. Even if SER per bit remains constant with technology scaling, the SER per chip will increase exponentially due to the increase in the number of transistors per chip, i.e. Moore's law. Recent studies show that the soft error vulnerability of combinational logic components will soon become comparable with that of sequential elements (SRAM cells, flip-flops, and latches) [5]. Accurate SER estimation is essential to develop efficient soft error tolerant schemes and to determine the contribution of design components to the overall system SER.

An erroneous system state occurs in the following scenario [4]. A particle strike causes a glitch at the output of the gate (Nominal FIT), this glitch propagates through the combinational logic to the flip-flop inputs (Logic Derating), and finally this erroneous glitch is 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). Therefore, the error rate of a node in a digital circuit is computed as $Nominal\ FIT \times Logic\ Derating \times Timing\ Derating$.

Unlike logic derating estimation which only requires static analysis, estimation of timing derating needs dynamic analysis of transient propagation. Specifically, for logic derating estimation based on fault injection, a sample of fault sites (e.g. gate outputs) are selected and for each error site, a sample of input vectors are fault simulated. However, for timing derating estimation, a new dimension is added in which the erroneous transient pulse at the fault site has to be injected at a random time within the clock period. As a result, fault injection for timing derating estimation is orders of magnitude more tedious and less accurate than logic derating estimation.

This work focuses on estimation of timing derating factor in sequential circuits in SER estimation flow. We present an analytical technique for logic-timing derating estimation which eliminates the need for time-consuming (fault) simulations. The proposed technique is based on an enhanced static timing analysis method to compute all propagated waveforms from a struck gate (error site) to reachable flip-flops and calculate the probability of latching an incorrect value in a flip-flop. We also exploit a technique based on signal probability values to estimate the propagation probabilities of erroneous values (or transient pulses) from the error site to reachable latches and flip-flops. Algorithms for the estimation of the latching probabilities of erroneous transients are provided. We also analyze the dependency of the overall accuracy of the proposed method to the accuracy of signal probability values.

The rest of this paper is organized as follows. Sec. 2 reviews the previous work on SER estimation techniques. In Sec. 3, the proposed logic and timing derating estimation approach is presented. In Sec. 4, experimental results are presented. Finally, Sec. 5 concludes the paper.

2. PREVIOUS WORK

Previous logic-timing derating estimation methods can be categorized into two groups. The first group uses fault injection based on random vector simulation approaches [3, 4, 6, 8]. Since the accuracy of such approaches depends on the ratio of the number of injected faults and simulated vectors to the total number of possible error sites and vector space, it is very hard to achieve a reasonable accuracy using these techniques. 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 using fault injection techniques. The second group uses an approach based on Binary Decision Diagram (BDD) for SER estimation [7]. Although this approach might be able to achieve more accurate results compared to simulation-based methods, it has still exponential time complexity for large circuits, especially with reconvergent fanouts.

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

3. TIMING -LOGIC DERATING ESTIMATION

If a particle with sufficient energy hits a particular gate and causes a bit flip at the output of this gate, we call this gate as the *error site*. Based on structural paths from the error site to the reachable primary outputs and flip-flops, we can categorize nets (signal lines) 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 a gate with at least one on-path input. 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 full swing glitch with pulse width w at time t at the output of gate g_i, as shown in Fig 1. Also, assume that there is only one path from this gate to a flip-flop FF_j. Depending on the value of other signals in the circuit, this 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 g_i to FF_j, and w' depends on the various rise and fall transition delays for the gates along this path.

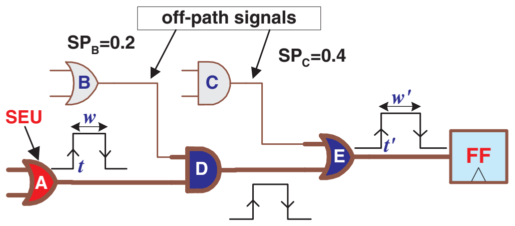

Figure 1: Propagation of transient through a unique path to a flip-flop

For computing the *propagation probability* (PP) of the erroneous glitch, we use the estimation method presented in our previous 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 the other off-path signals. In this example, the propagation probability for the glitch to propagate to the output of 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, the propagation probability at the output of 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 are stable, i.e. no other signal is making a transition. This assumption is used throughout the paper.

The *latching probability* (LP) is defined as the probability that an erroneous value is captured in a reachable flip-flop. 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 (S) and hold (H) time of the flip-flop, glitch width (W), and clock period (T): $LP = \frac{S+H+W}{T}$. Figure 2 shows the latching window. *Error propagation probability* (EPP) is calculated as the product of propagation probability and latching probability, i.e. $EPP = PP \times LP$.

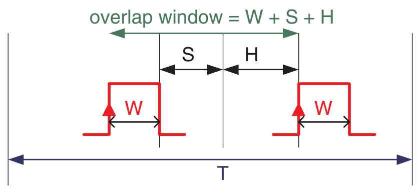

Figure 2: Latching window

In general, 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 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.

Figure 3 shows an example in which there are multiple paths from the error site to the flip-flip. There are three possible propagation scenarios: 1) propagation through only the NAND gate, 2) propagation through only the OR gate, and 3) propagation through both paths. Even if we consider a 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, plus one case of no propagation at all. The top waveform at the input of the flip-flop is due to the propagation through only the NAND gate. If the output of the OR gate is 0, then the same waveform is propagated since the reconvergent gate is XOR. If the output of the OR gate is 1, then the inverted waveform will be propagated (not shown). If the waveform is propagated through both paths, then the shape of the waveform is not a single glitch (middle waveform). Finally, if the glitch is propagated through only the OR gate, then the bottom waveform or its inverted will appear at the input of the flip-flop, depending on the output value of the NAND gate.

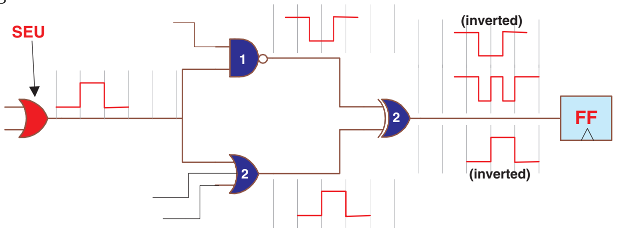

Figure 3: Propagation of transient through reconvergent paths

This simple example shows that depending upon the 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, 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:

$$EPP_{g_i \to FF_j} = 1 - \prod_{all\ propagated\ waveforms\ k} (1 - PP_k \times LP_k)$$

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

3.1 Propagation Probability

For estimation of the propagation probability, we use an approach similar to [1, 2]. Here we explain how to perform a static error propagation analysis. In Sec. 3.2, this is expanded for 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 reachable flip-flop or primary output, the polarities of propagated erroneous values, 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, 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 logic value of 1.
- *a:* the signal has an erroneous value with the same polarity as the original erroneous value at the error site (denoted by a).
- *ā:* 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 $\bar{a}$).

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

- $P_a(U_i)$ ($P_{\bar{a}}(Ui)$) is the probability that the erroneous value is propagated from the error site to U_i with an even (odd) number of inversions.
- $P_1(U_i)$ ($P_0(U_i)$) is the probability of node U_i being 1 (0). In this case, the error is masked and not propagated.

Note that $P(U_i) = P_a(U_i) + P_{\bar{a}}(Ui) + P_1(U_i) + P_0(U_i) = 1$. Since the polarities of propagated errors are considered, propagation probabilities at the output of reconvergent gates are correctly calculated. The propagation computation rules for elementary gates, *AND*, *OR*, and *NOT*, are shown in Table 1. Propagation rules for other logic gates can be derived accordingly.

These propagation rules are used for each gate reachable from the error site in a level by level order. Therefore, all error propagation probabilities can be calculated in only one pass. More details can be found in [1, 2].

Table 1: Output propagation probability rules for elementary gates

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	$P_0(out) = \prod_{i=1}^{n} P_0(X_i)$
	$P_a(out) = \prod_{i=1}^{n} [P_0(X_i) + P_a(X_i)] - P_0(out)$
	$P_{\bar{a}}(out) = \prod_{i=1}^{n} [P_0(X_i) + P_{\bar{a}}(X_i)] - P_0(out)$
	$P_1(out) = 1 - [P_0(out) + P_a(out) + P_{\bar{a}}(out)]$
NOT	$P_1(out) = P_0(input), \quad P_0(out) = P_1(input)$
	$P_a(out) = P_{\bar{a}}(input), \quad P_{\bar{a}}(out) = P_a(input)$

3.2 Latching Probability

The objective here is to compute 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 (error site) caused by an 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. 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 error 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 error 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. We use a notation similar to what we used in Sec. 3.1. 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.

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 1 are used starting from the error site to all reachable flip-flop. 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, we can calculate the event list $Event\_List(g)$ for each on-path gate g.

Once the event list at the input of each reachable flip-flop FF_j is calculated, we can generate all possible waveforms that can be resulted from propagation of $[(a, t), (\bar{a}, t + w)]$ at g_i. A propagated waveform at FF_j input can be obtained from a series of a to $\bar{a}$ events (or alternatively from $\bar{a}$ to a events) 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 starts and ends by a events. Figure 4 shows an example of this approach to propagate all events from the error site to the reachable flip-flop.

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. We looked at the maximum size of the event lists for some of the simulated circuits in our experiments. Our 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.

3.3 Algorithm

Algorithm 1 shows the overall procedure as explained in Sec. 3.2. For each gate (considered as an error site) in the circuit, all its structurally reachable flip-flips are extracted. The event list as well as the probability of each event is propagated from the error site to reachable flip-flops. Based on the event list at the input of each flip-flop, the possible waveforms are computed to obtain propagation and latching probabilities. Timing-logic derating due to SEUs at this error site is calculated based on these probabilities. The overall circuit derating can be obtained based on the derating of each individual gate. Note that the glitch pulse width must be specified as an input to this procedure.

```
1  Algorithm:Timing Derating Computation
2  $w$: Glitch-Width
3  $TD$: Timing-Derating factor
4  for each gate $G_i$ do
5      List($G_i$) ← Extract on-path gates reachable from $G_i$
6      List($G_i$) ← Sort List($G_i$) based on distance from $G_i$
7      Event_List($G_i$) ← Add_Event($a$,time=$t$);
8      Event_List($G_i$) ← Add_Event($\bar{a}$, time=$t + w$);
9      for each gate $G_j$ in List($G_i$) do
10         for each input ($k$) of gate $G_j$ do
11             Event_List($G_j$).Add_event_list($k$);
12         end
13         for each event $E$ in Event_List($G_j$) do
14             Apply_propagation_rules($E$); /*see Table 1*/
15         end
16     end
17     TD($G_i$) ← 1;
18     for each Flip-Flop ($FF_j$) in List($G_i$) do
19         $TD_{G_i \to FF_j}$ ← 0;
20         for each valid waveform ($p_k$) in
           Event_List($FF_j$) do
21             $PP_k$ ← Propagation_Probability($p_k$);
22             $LP_k$ ← Latching_Probability($p_k$);
23             $TD_{G_i \to FF_j} \leftarrow TD_{G_i \to FF_j} + PP_k \times LP_k$;
24         end
25         $TD(G_i) \leftarrow TD(G_i) \times (1 - TD_{G_i \to FF_j})$ ;
26     end
27     $TD(G_i) \leftarrow 1 - TD(G_i)$;
28 end
```
Algorithm 1: Timing Derating Computation

3.4 Electrical Masking Effect

The magnitude (height) of the erroneous glitch can be attenuated while propagating through logic stages. This is known as *electrical masking* and affects SER. To consider this effect, logic library cells can be pre-characterized for different particle charge values. For each library cell, an *electrical attenuation factor* lookup table can be obtained based on cell fanout capacitance and SEU pulse width and height. The logical masking factor needs to be multiplied by the attenuation factor when computing $P_a(U_i)$ and $P_{\bar{a}}(Ui)$. As propagation probability table (Table 1) is used to obtain the propagation probabilities at the output of each gate using the corresponding values at the inputs of that gate, the attenuation lookup tables are used to compute the magnitudes of the propagated values at the output of the gate based on the magnitudes of the transients at the inputs of the gate and the fanout of the gate.

4. EXPERIMENTAL RESULTS

In order to verify the accuracy of the proposed technique, we have developed a fault injection engine based on Monte-Carlo (MC) simulations. For a given glitch width, we have injected glitches at the output of gates at different times during the clock period. The random variables are the struck gate and the time of the glitch. Timing accurate logic simulation determines if the injected glitch can be captured in any flip-flop. The MC simulation terminates if the accuracy of the estimated derating falls within a pre-defined confidence interval (in our experiments, the maximum variance is 5% and the confidence level is 99%).

The proposed approach was implemented and applied to ISCAS'89 sequential benchmark circuits. All experiments have been performed on the DELL Precision 450 system equipped with 2 GB main memory. Figure 5 shows the run time for both Monte-Carlo simulation and our proposed approach including signal probability (SP) calculation time. Note that the Y-axis in this figure is logarithmic. On average, the proposed method is 31,000 times faster than the MC simulation approach. The run time of our approach for the largest ISCAS'89 circuits is only 5 minutes.

Figure 6 shows the accuracy of the proposed approach compared to the MC simulation method. The accuracy is compared using different variances (accuracies) of SP values used in the proposed method. 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 method is not considerably sensitive to the accuracy of SP values. In other words, it is possible to use less accurate SP values (tractable for large circuits) to achieve a reasonably accurate SER. The results show that the accuracy of our presented approach is within 1% of the MC approach.

5. CONCLUSIONS

Soft errors due to single event upsets are the main reliability threat of digital systems. Estimation of soft error rate in sequential circuits is very challenging since computing the probability of an erroneous system state requires dynamic analysis of transients. As a result, fault injection methods become completely intractable.

In this paper, we have proposed a combined logic and timing derating estimation method in sequential circuits. The proposed technique uses an enhanced static timing analysis to derive all possible erroneous waveforms propagated from a struck gate to reachable flip-flops and calculates the probability of latching an incorrect value in flip-flops. We

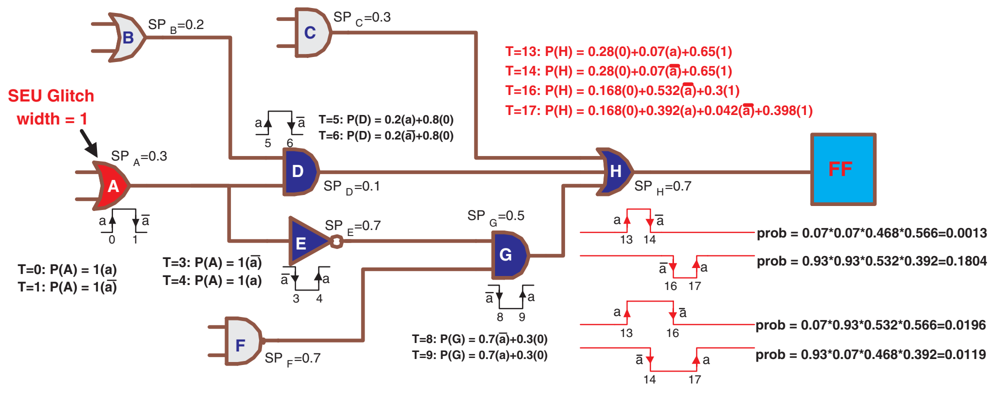
SEU Glitch width = 1
T=13: P(H) = 0.28(0)+0.07(a)+0.65(1)
T=14: P(H) = 0.28(0)+0.07(ā)+0.65(1)
T=16: P(H) = 0.168(0)+0.532(ā)+0.3(1)
T=17: P(H) = 0.168(0)+0.392(a)+0.042(ā)+0.398(1)
T=5: P(D) = 0.2(a)+0.8(0)
T=6: P(D) = 0.2(ā)+0.8(0)
T=0: P(A) = 1(a)
T=1: P(A) = 1(ā)
T=3: P(A) = 1(ā)
T=4: P(A) = 1(a)
T=8: P(G) = 0.7(ā)+0.3(0)
T=9: P(G) = 0.7(a)+0.3(0)
prob = 0.07*0.07*0.468*0.566=0.0013
prob = 0.93*0.93*0.532*0.392=0.1804
prob = 0.07*0.93*0.532*0.566=0.0196
prob = 0.93*0.07*0.468*0.392=0.0119
FF

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

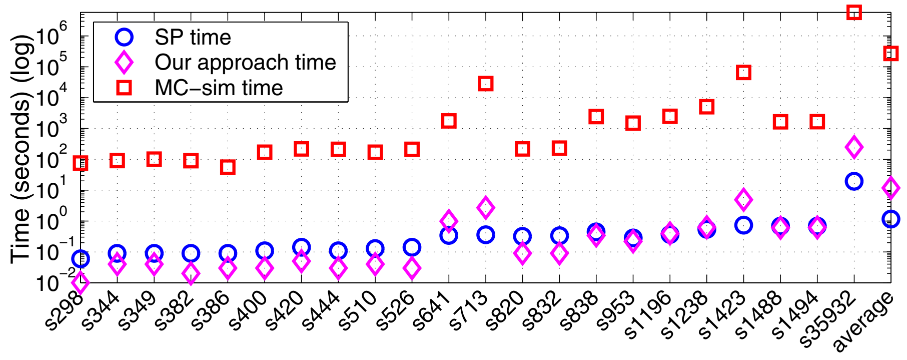

Figure 5: Execution times for the MC simulation approach, SP computation, and the proposed method (for an injected pulse width of 50 ps)

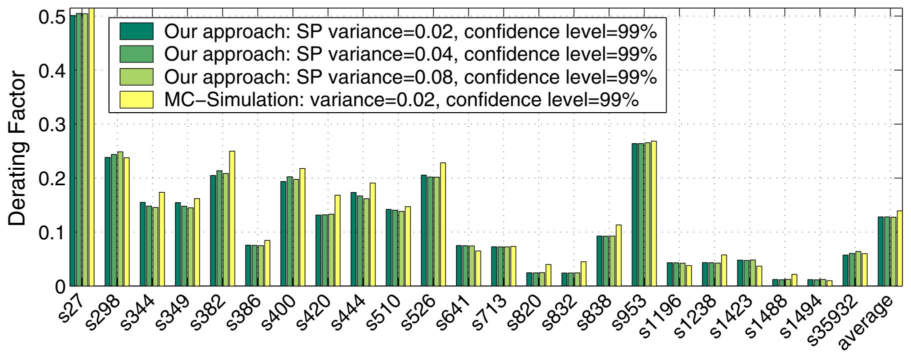

Figure 6: Comparison of the accuracy of the MC simulation with our approach using different SP variances (for an injected pulse width of 50 ps)

also exploit a technique based on signal probability to estimate propagation probabilities. Experimental results and a comparison with timing accurate Monte-Carlo simulations show that our proposed technique is 4-5 orders of magnitude faster while the difference in accuracy is almost 1% on average.

6. REFERENCES

[1] G. Asadi and M. B. Tahoori,"An Accurate SER Estimation Method Based on Propagation Probability," Proc. Design Automation and Test in Europe Conf., pp.306-307, March 2005.

[2] G. Asadi and M. B. Tahoori,"An Analytical Approach for Soft Error Rate Estimation In Digital Circuits," Proc. Intl. Symp. on Circuits and Systems, pp. 2991 - 2994, May 2005.

[3] 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.

[4] H. T. Nguyen and Y. Yagil, "A Systematic Approach to SER Estimation and Solutions," Proc. Intl. Reliability Physical Symp., pp. 60-70, 2003.

[5] P. Shivakumar, M. Kistler, S.W. Keckler, D. Burger, and L. Alvisi, "Modeling the Effect of Technology Trends on the Soft Error Rate of Combinatorial Logic," Proc. Int'l Conf. on Dependable Systems and Networks, pp. 389-398, 2002.

[6] M. Zhang and N. R. Shanbhag, "A Soft Error Rate Analysis (SERA) Methodology," Proc. Intl. Conf. on Computer-Aided Design, pp. 111 - 118, Nov. 2004.

[7] B. Zhang and M. Orshansky, "Symbolic Simulation of the Propagation and Filtering of Transient Faulty Pulses," Proc. SELSE Workshop, Urbana-Champaign, April 2005.

[8] Q. Zhou and K. Mohanram, "Cost-Effective Radiation Hardening Technique for Combinational Logic," Proc. Intl. Conf. on Computer-Aided Design, pp. 100-106, Nov. 2004.