MASkIt: Soft Error Rate Estimation for Combinational Circuits

Martí Anglada[1], Ramon Canal[1], Juan L. Aragón[2], and Antonio González[1]

[1]Universitat Politècnica de Catalunya {*manglada, rcanal, antonio*}*@ac.upc.edu*
[2]Universidad de Murcia *jlaragon@ditec.um.es*

***Abstract*—Integrated circuits are getting increasingly vulnerable to soft errors; as a consequence, soft error rate (SER) estimation has become an important and very challenging goal. In this work, a novel approach for SER estimation of combinational circuits is presented. The proposed framework is divided in two stages. First, signal probabilities are computed via a hybrid approach combining heuristics and selective simulation of reconvergent subnets. In the second stage, signal probabilities are used to compute the vulnerability of all the gates in a combinational block using a backward-traversing algorithm that takes into account logical, electrical and timing masking factors. Experimental results show that our signal probability estimation approach, in comparison with similar techniques in the literature, reduces innacuracy by 96% while adding minimal execution time overhead. In addition, results indicate that our framework is two orders of magnitude faster than traditional Monte Carlo-based fault injection with minor loss in accuracy in both signal probability and SER estimation (average error of 5%).**

I. Introduction

Semiconductor technology scaling has brought tremendous improvement in both performance and energy efficiency. Device scaling translates into a reduction in feature size and voltage levels, which results in smaller devices that require less current to switch; and thus, they can be operated at higher frequencies. On the other hand, scaling in the sub-100 nm lithographies has dramatically increased reliability issues. In particular, nanoscale integrated circuits have become highly sensitive to single-event transients (SET): disturbances caused by particle strikes from ionizing radiation. Since the faults caused by such radiation are not permanent, they are called *soft errors*. Historically, soft errors have been tackled in the context of memory cells because of the large area of the chip devoted to caches and the higher vulnerability of SRAM cells in comparison with combinational logic. In this way, their negative effect in memory has been kept under control due to the use of protection mechanisms with low overhead, such as error-correcting codes. On the other hand, the expected soft error rate (SER) of combinational logic has been reported [34] to increase by five orders of magnitude in eight technology nodes (from 600nm to 50nm). As a consequence, methodologies to accurately deal with SER in combinational logic have become an important area of study. In order to develop efficient protection mechanisms, accurate tools to measure its effect are in dire need.

A variety of techniques have been proposed to estimate the SER of combinational logic. Several works [4], [11], [23], [38] are based on Monte Carlo fault injection simulations. These approaches suffer from large simulation times to reach stable results and they are not practical for the size of contemporary circuits. Therefore, most efforts nowadays are based on static, analytical methods.

Modeling and analyzing the SER in logic is more complex than in memory elements, since there are some well-known masking effects that reduce the overall likelihood that a pulse caused by a particle strike is latched and results in an error. These masking effects are commonly classified as [26]:

- *Logical masking*: Transient faults are masked by gates whose output is independent of the faulty input (e.g., an OR gate with an input set to 1).
- *Electrical masking*: The pulse is attenuated (either its amplitude is reduced or rise/fall times are increased) by the electrical properties of the gates throughout the logic chain, and the resulting magnitude is insufficient to change the value that is latched.
- *Timing masking*: The pulse arrives at a state-holding element out of its latching-time window.

Due to the complexity of the problem, a tradeoff between accuracy, efficiency and scope has to be made. Analytical techniques that compute the SER at a system level usually rely on Architecturally Correct Execution analysis [27], which in some cases can lead to a 7x overestimation [14]. For a more accurate static analysis, circuit-level approaches can be adopted. However, their use comes with several shortcomings and drawbacks. Some fast methods [3], [19], [20], [22], [36] only take into account one or two masking effects. A number of works report great accuracy on their estimation of the soft error rate [28], [37], [41], but they need to perform their analysis for a large number of input vectors to achieve a high coverage which, in turn, significantly impacts execution time.

Another important challenge that analytical mechanisms need to cope with is reconvergence. Whenever a logic signal splits into multiple branches and later reconverges in two or more inputs of a gate, it creates dependencies between signals that complicate the computation of the different masking effects. Although there are several accurate techniques that use path extraction or try to analyze multilevel correlations [2], [9], [10], [35], they suffer from a high computational cost and are only feasible for small-sized circuits. On the other hand, there are a few works that try to use heuristics [8], [13], [16] or specific data structures [15], [18], [30], [40] to estimate the effects of reconvergent subnets in the reliability of the circuit.

In this work, we present MASkIt: an accurate and fast SER estimation framework based on signal probabilities that considers the effects of the three masking factors and signal reconvergence. Signal probabilities are estimated using a novel hybrid method that merges static analysis with selective sub-circuit simulation: we first exhaustively simulate all small-sized reconvergent subnetworks of the circuit and then, we run an estimation algorithm. In this way, we are able to incorporate accurate signal probabilities in those nodes of the circuit whose estimation by the heuristic algorithm is inaccurate without losing its defining speed benefits. Soft Error Rate is estimated by traversing the circuit from outputs to inputs, computing the vulnerability of each gate by combining signal probabilities, data from cell characterization and the incrementally-gained information about the masking capabilities of the different paths. Reconvergent fan-outs are handled with Binary Decision Diagrams (BDD), but their use is limited to small-sized subnets of the circuit. Therefore, we bypass one of the greatest weakness of BDDs -scalability- and achieve an accurate and fast SER estimation.

To validate the proposed approach, we compare the output of our analysis against Monte Carlo based fault injection using the ISCAS'85 benchmarks [5], and we show that MASkIt achieves about the same accuracy (average error of 5%) with two orders of magnitude less computing time. We study several signal probability estimation algorithms presented in the literature and show that the accuracy of our technique is significantly better (96% error reduction) while adding minimal execution time overhead.

The remainder of this paper is organized as follows. Section II formulates the SER estimation and modeling approach. Section III describes the mechanism to estimate signal probabilities. Section IV presents the proposed approach for computing the masking effects. Section V reports the experimental results of the validation process. Section VI summarizes the major conclusions of the work.

II. MASkIt: OVERVIEW OF THE PROPOSED APPROACH

A. SER estimation

The overall SER of the circuit can be computed as the accumulation of the individual SER of all the gates in the circuit. Each of those SER, SER_{gate_i}, is defined as the probability that a particle with a particular charge q strikes at $gate_i$ and the SET (Soft Error Transient) originated is not masked. This is computed by integrating the products of particle-hit rate and masking probability over a range of charges q_{min} to q_{max}. For practicality, the integral is often approximated as a discrete sum:

$$SER_{gate_i} = \sum_{c=1}^{\#charges} R_{PH}(q_c) * Vul(gate_i, q_c) \quad (1)$$

where Vul is the vulnerability of the gate and R_{PH} is the particle hit rate, defined in [34] as a function of neutron flux, area and slope of charge collection. q_c corresponds to a discrete charge selected from the continuous range: $q_c = c * (q_{max} - q_{min})/\#charges$. In [40] it is shown that $\#charges = 5$ yields to an accurate enough SER estimation in comparison with SPICE models.

The vulnerability of a gate is defined as the probability that a soft error propagates up to a latch. It can be formulated as:

$$Vul(gate_i, q_c) = LM(i,c) * EM(i,c) * TM(i,c) \quad (2)$$

which corresponds to the probability that the SET with charge q_c at gate $gate_i$ is neither affected by the logical masking factor (LM) nor the electrical masking factor (EM) nor the timing masking factor (TM).

B. Netlist modeling

Given a combinational circuit, our approach needs three inputs: the netlist of a combinational circuit, a set of input probabilities and a cell characterization library. The netlist of the circuit is a list of gates and their connectivity, which corresponds to a directed acyclic graph defined by the union of the list of gates G (vertices), the set of connections C (directed edges), the set I of inputs of the circuit and the set Ω of outputs of the circuit.

- Each element of G is formed by a pair $<\varphi, k>$, where φ is a Boolean operation (such as NOT or NAND) and k is the number of inputs of the gate.
- Each element of C is a connection between one output of a gate and one or more inputs of succeeding gates. This is formally defined as a pair $< g_i, \Gamma_i >$, where g_i is the i^{th} gate of G and Γ_i is a set of gates $\{g_m^a, g_n^b, g_o^c, \ldots\}$, where subscripts m, n and o indicate a particular gate from G and superscripts a, b and c indicate the input (0, 1, ..., k-1) of the gate that is connected to the output of g_i.
- Each element of I is a pair $<i, a>$, where i designates the i^{th} gate of G and a corresponds to input number a ($0 \leq a < k$) from g_i.
- Each element of Ω is pair $<i, f>$, indicating that gate g_i is connected to the output flip-flop f of the circuit.

Every element i from I has associated an input probability P_i, which corresponds to the probability that the value of i is 0. In addition, from the cell characterization library, we extract the following variables of interest:

- Every element g_i of G has associated two transition probabilities $T_i^{out}[0 \rightarrow 1]$ and $T_i^{out}[1 \rightarrow 0]$ which correspond to the probability that, given a strike, the output at g_i transitions, respectively, from 0 to 1 and from 1 to 0.
- Every element g_i of G has associated a rise time t_r and a fall time t_f, which correspond to the slopes of the output pulse generated when g_i is struck (from 10% to 90% of V_{dd} and from 90% to 10% of V_{dd}, respectively).
- Every element g_i of G has associated a propagation delay δ.
- Every element g_i of G has associated a first transition delay d_1 and a second transition delay d_2.
- Every element g_i of G has associated a width PW_{min}, which corresponds to the minimum width the striking pulse needs in order to flip the logic value of g_i.

- Every element of Ω has associated a time τ_s and a time τ_h, which correspond, respectively, to the setup time and the hold time of the flip-flop.

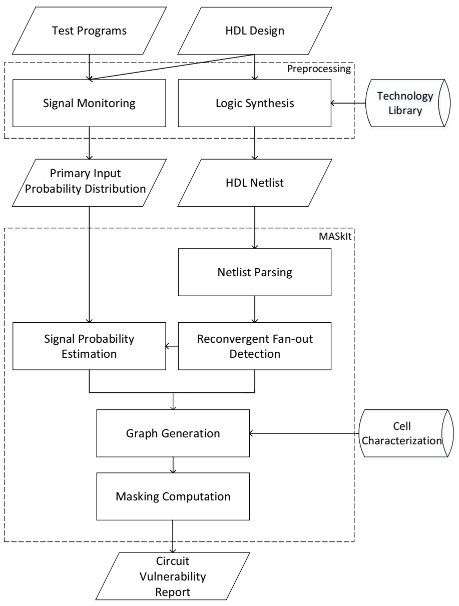

Fig. 1. Overview of the MASkIt tool flow.

MASkIt computes a set of vulnerability probabilities Vul which indicate, for every gate in G, the probability that a strike in that gate will be propagated to one or more outputs in Ω. Finally, it computes the vulnerability of the whole circuit based on the above probabilities. Figure 1 shows the flow of the proposed MASkIt framework, which consists of a preprocessing stage and two main computing stages:

1) **Signal Probability Estimation** (described in Section III): Propagates the signal probabilities of the primary inputs of the circuit to all of the nodes based on the functionality of each gate and minimal simulations.
2) **Masking Effect Calculation** (described in Section IV): Computes the vulnerability of all gates of the circuit based on signal probability and parameters extracted from cell characterization.

III. Estimation of Signal Probabilities

The signal probability P_i of a node i is the probability that the signal value of the node i will be a 0 under a random assignment of an input vector. It has been long established that computing signal probabilities is a #P-complete problem [24] and, therefore, the usual approach is to estimate such probabilities instead of running an unduly complex algorithm to achieve accuracy.

A frequently adopted technique [20], [28], [41] consists in generating a number of random input vectors and simulating the circuit using tools such as ModelSim to obtain the state at each node. To avoid the expensive execution time needed for high coverage, heuristic algorithms are also used.

The basis for such algorithms is the fact that the signal probability for a gate can easily be computed according to the following elemental rules:
The probability P_{NAND} that at the output of a 2-input NAND gate there is a 0 knowing the probability P_i that there is a 0 at the input i is:

$$P_{NAND} = (1 - P_0) * (1 - P_1) \tag{3}$$

The probability P_{OR} that at the output of a 2-input OR gate there is a 0 knowing the probability P_i that there is a 0 at the input i is:

$$P_{OR} = P_0 * P_1 \tag{4}$$

This approach can be extended to any kind of gate. Traversing all the gates of the circuit from inputs to outputs using these elemental rules is referred to as 0-Algorithm [12].

The 0-Algorithm runs in linear time and yields exact signal probabilities whenever the network is free from reconvergent fan-outs (nodes with fan-out > 1 whose branches reconverge within the circuit). In the general case, however, a circuit has dependencies within input leads of each gate. In fact, it has been reported that in current VLSI designs about half of the nodes in a circuit cause a reconvergence [29].

There are a few widely-adopted, polynomial-time algorithms that try to reduce the lack of precision induced by the 0-Algorithm by using some heuristics: the Weighted Averaging Algorithm (WAA) [21], the Dynamic Weighted Averaging Algorithm [12] and the Possibilistic Algorithm (POSS) [1]. While each approach deals with the problem differently, they all rely on the observation that the signal probability $P(j)$ of a node j can be expressed by:

$$P(j) = P(j|f = 0) * P(f = 0) + P(j|f = 1) * P(f = 1) \tag{5}$$

where f is a reconvergent fan-out node (RFON) of the fan-in cone of j. If f were the only RFON in the circuit, $P(j)$ could be exactly calculated by using formulas such as (3-4) and applying three times the 0-Algorithm: first to evaluate $P(f)$, then forcing the logical value of f to 0 and 1 to calculate the conditional probabilities.

Another popular approach is the Correlation Coefficient Method (CCM) [12], which computes the signal probability of a gate not only using the probability of the inputs but also explicit ratios that express the correlation between each pair of inputs. CCM modifies the elemental rules to compute signal probability by adding a correlation coefficient to each formula and also proposes a new set of rules to propagate them.

Several works within the literature apply these procedures as a way to estimate signal probability: CASSER [7] and the work of Franco et al. [13] use DWAA, while CEP [8] and the work of Hunag et al. [16] take advantage of CCM.

Since, to the best of our knowledge, there are no comparisons in the literature of the accuracy of the four aforementioned algorithms with circuits with sizes larger than a few hundreds of nodes, we have implemented them in order to decide which one to use in our framework.

As detailed in Section V, the use of these heuristics improves the accuracy of the signal probability computation in comparison with the 0-Algorithm, but it is still insufficient considering the importance of the role that signal probability plays in our reliability estimation algorithm.

Therefore, we propose a new approach that combines the Possibilistic Algorithm with the exhaustive simulation of only a critical subset of reconvergent subnets. A simulation of this kind consists in generating all input vectors for the subnet. We define its inputs as the leads reaching the subnet and coming from nodes not contained in the subnet. We propose to generate the 2^n inputs, simulate the circuit 2^n times and average the 0-to-1 ratios of the nodes by the likelihood of each input vector.

In order to perform these simulations, all RFON are searched using Robert's algorithm to detect all reconvergent fan-outs [32], and only the ones with a reconvergent subnet size of 30 or less gates are simulated. We define a reconvergent subnet size as the number of gates among all paths between the source and the destination of reconvergence. Once the reconvergent subnets are simulated, the Possibilistic algorithm is run to calculate the rest of the nodes' signal probabilities. Section V contains the justification on the threshold value for the size of the subnets that are considered.

IV. Masking Probability Computation

A. Logical Masking

Logical masking occurs when the generated current pulse due to a strike arrives at the input of a gate whose output signal happens to be logically independent of the faulty input at that time, i.e., the output is determined by the rest of input signals. We estimate the logical masking effect of each individual gate by computing the likelihood of controlling input vectors and combining them with the transition probabilities for each input combination.

To better illustrate the process, let us consider a 2-input NAND gate:

1) If both inputs are set to 0, a transient fault at only one of the inputs $(0 \rightarrow 1)$ does not affect the output.
2) If one input is 1 and the other is 0:
   a) If the input at 1 suffers a transition $(1 \rightarrow 0)$, the output is not affected.
   b) If the input at 0 suffers a transition $(0 \rightarrow 1)$, the output does change.
3) If both inputs are 1, a transition at any input $(1 \rightarrow 0)$ changes the output.

Therefore, the effect of not logically masking a SET for a 2-input NAND gate can be expressed as:

$$NotMasking_{NAND2} =$$

$$P_0 * (1 - P_1) * T_0^{in}[0 \rightarrow 1] \tag{6a}$$

$$+ (1 - P_0) * P_1 * T_1^{in}[0 \rightarrow 1] \tag{6b}$$

$$+ (1 - P_0) * (1 - P_1) * (T_0^{in}[1 \rightarrow 0] + T_1^{in}[1 \rightarrow 0]) \tag{6c}$$

where (6a) and (6b) correspond to case 2-b) in the previous example while (6c) corresponds to case 3. $T_j^{in}[n \rightarrow m]$ indicates the probability of a transition from n to m at input j, which corresponds to $T_i^{out}[n \rightarrow m]$ from the cell characterization, where i is the gate whose output is connected to input j.

Similarly, equations to measure the logical masking factor for any kind of gate (not only NANDs) can be constructed, in order to calculate the overall LM term in equation (2).

B. Electrical Masking

Electrical masking occurs because of the attenuation of the transient pulse during its propagation alongside a chain of gates. In particular, the rise and fall time of the pulse increases while its amplitude decreases. We estimate the electrical masking effect of each individual gate by determining if a strike in that gate is wide enough to be latched when reaching an output.

A strike in a gate directly connected to a primary output of the circuit needs to be stable from the setup time (t_s) and until the hold time (t_h) of the output latch. However, a strike in an intermediate gate (with other gates as successors) is more unlikely propagated to the outputs since it needs to be wide enough to be stable from t_s until t_h but also taking into account the attenuation effect caused throughout the propagation path to the output latch.

At each visited gate, we calculate how its electrical properties affect the shape of the pulse using parameters t_r, t_f, d_1 and d_2 from the cell characterization within the glitch propagation model in [28]. This model contains a set of equations to initially calculate the output voltage (that depends on the rise and fall times and also the input pulse width) and then compute the pulse width at the output using the output voltage.

MASkIt applies those equations in its backward search algorithm to extract the input pulse width of the gate under consideration from the propagated output pulse, which corresponds to a previously-computed input pulse from a successor gate. The resulting input width represents the *minimum width* of a SET in order to traverse the path from the struck gate to any output and be latched. The probability that a strike generates a pulse of that width corresponds to the probability that the strike becomes electrically masked. The complement of that probability is the term EM in equation (2).

C. Timing Masking

Timing masking occurs because latches are insensitive to signals that arrive out of their latching window. We estimate the timing masking effect by computing the likelihood that a particle strikes at such a time that the SET produced cannot reach an output at an appropriate time to be latched.

Note that a strike in a gate directly connected to an output needs to be stable before the setup time (t_s) of the latch. A strike in a gate that has other gates as successors needs to be stable before the setup time of the latch in addition to the propagation time from its successors to the latch.

In its backward traversal from outputs to inputs, MASkIt computes at each gate the *maximum time* within a clock cycle the SET would need to occur in order to traverse the path from the struck gate to any output and be latched. To do so, every gate propagates the maximum time computed by its successors and adds its own cell-characterized delay δ. The probability that a particle strikes at that time or earlier in the cycle corresponds to the probability that the strike becomes timing masked. The complement of that probability is the TM term in equation (2).

D. Reconvergent fan-outs

When a logic signal splits into multiple branches (fan-out) and later reconverges in two or more inputs of a gate, the subnet thus formed is referred to as *reconvergent fan-out* (RFON). Reconvergent fan-outs need to be taken into account because they break the assumption that signal paths are independent.

In this work, we propose to obtain the Boolean difference of the function at the destination of the reconvergence and use it in conjunction with the vulnerability of the destination node to compute the vulnerability at the source node.

When the Boolean difference of a function f with respect to input x_i equals 1, the result is all input combinations for which a change in x_i also changes the output of f. An efficient structure to represent and operate with Boolean functions is a Binary Decision Diagram (BDD) [6]. A BDD is a rooted, directed acyclic graph with one or two terminal nodes labeled '0' and '1' and with a set of non-terminal nodes labeled with a Boolean variable. Each non-terminal node has exactly two edges from that node to others: one corresponding to the evaluation of the variable as 0 and another one corresponding to the evaluation of the variable as 1.

A BDD needs to be manipulated in order to represent the Boolean difference of a function. This can be achieved with a subset of the basic operations presented in [6]. In particular, only three procedures are used:

- *Apply*: Takes two functions f_1, f_2 and a Boolean operator $< op >$ and produces a BDD representing the function $f_1 <op> f_2$.
- *Restrict*: Transforms the graph representing a function f into one where argument x_i is replaced with some constant b.
- *Satisfy all*: Lists all argument values for which a function f evaluates to 1.

Binary Decision Diagrams and its operations are used in MASkIt as follows: When a source of reconvergence is visited, an initial BDD is created. Then, all the nodes from the reconvergent subnet are visited until the destination of the reconvergence is reached. For every gate in the subnetwork, the *Apply* procedure constructs a BDD representing the function computed at the output of each logic gate. When the traversal of the subnet is completed, the destination node contains a BDD which represents its Boolean function as a function of all the subnets inputs. Two *Restrict* operations are performed to the final BDD: one replacing the source node variable with the constant 0 and another replacing the source node variable with the constant 1. Using the *Apply* procedure, the XOR of these two restriction BDDs is computed, resulting in a BDD that represents the Boolean difference of the destination node with respect to the source node. Applying the *Satisfy all* procedure on the Boolean difference BDD yields all input patterns that cause a change in the output of the destination if the value of the source changes. Since the signal probability for every signal is known, the probability of those input vectors can also be calculated, i.e., the probability that a strike at the source is not masked within the subnet. That probability is independent from the vulnerability of the destination and, therefore, the vulnerability of the source node can be easily computed by combining the two. If the source node feeds several subnets, the process is repeated for all of them, obtaining vulnerabilities for the different paths. Finally, the different vulnerabilities are combined as the union of non-mutually exclusive events. Unlike previous works that also use BDDs such as [40] or [25], in MASkIt the diagrams are only created to handle RFON subnets and, consequently, less memory usage is required. Whenever the vulnerability of a subnet has been computed, the BDD structure representing that subnet is no longer needed and it can be freed from memory.

E. MASkIt Algorithm

Algorithm 1 Algorithm to estimate the SER in a circuit

```
1: Compute signal probabilities of all gates
2: Inititialize $Queue$ with output gates
3: while $Queue$ is not empty do
4:     $gate \leftarrow$ first element of $Queue$
5:     mark $gate$ as visited
6:     for $parent \leftarrow \{$Gates in Predecessors($gate$)$\}$ do
7:         if all gates in Successors($parent$) have been
           visited then
8:         $Queue \leftarrow Queue \cup parent$
9:     end for
10:    if $gate$ is not source of reconvergence then
11:        for $j \leftarrow \{$Inputs in Successors($gate$)$\}$ do
12:            $Prob_j \leftarrow$ Compute $LM$, $EM$ and $TM$ of
               any path from $gate$ to a circuit output that
               includes the connection from $gate$ to $j$
13:        end for
14:    else
15:        for $j \leftarrow \{$Reconvergent subnets in $gate\}$ do
16:            Traverse subnet $j$, building BDDs for each
               gate, until the reconvergence destination of
               $j$ is reached
17:            $Prob_j \leftarrow$ Compute $LM$, $EM$ and $TM$ of
               $j$ using the sensitization BDD and the vul-
               nerability of the reconvergence destination
18:        end for
19:    end if
20:    Compute vulnerability of the connection at the output
   of $gate$ with all $Prob$ values
21: end while
22: Compute the vulnerability of the circuit based on the
   vulnerabilities of all connections
```

Algorithm 1 presents the procedure to compute the masking probabilities of each gate in the circuit. The algorithm starts by estimating the signal probabilities of all gates using the method described in Section III (line 1). Afterwards, it executes a Breadth-First Search (lines 3 to 21) starting from the outputs and then moving backwards until the inputs of the circuit are reached. A queue, created at line 2, is used to store in the proper order the gates whose connections at the output are ready to be visited.

In every iteration of the main loop, one gate whose successors have already been visited is selected to compute its vulnerability. That gate is removed from the queue (line 4) and marked as visited (line 5). The predecessors of the gate are enqueued to be examined later (line 8) only if all their successors have been visited (line 7). After the queue structure has been updated, the vulnerability of the gate is computed according to its successors and whether or not it is a source of reconvergence.

The majority of gates are not sources of reconvergence. In that case, every input from all the successor gates of the gate being visited are considered (line 11). In line 20, we compute the vulnerability of the connection at the output of the gate for all alternative paths to the output of the circuit. For this purpose, we first compute the individual vulnerability for each set of paths that include the connection from the output of the gate to a particular input of a successor gate (line 12). This corresponds to computing LM, EM and TM in equation (2) by the mechanisms previously described. Then we compute the vulnerability of the connection at the output of the gate as a combination of the individual vulnerabilities, assuming that the probability of masking is independent for different paths.

When the gate being visited is a source of a reconvergence, the previous independence assumption is never true and, therefore, we cannot use the method described in lines 10 through 13. The solution proposed in our methodology is to use Binary Decision Diagrams (BDD) on the reconvergence subnet. To do that, we construct the BDD of the reconvergence destination node by traversing the whole subnet and building the BDD for each gate (line 16). Then, applying BDD operations, the list of all input patterns that cause a change in the output of the destination if the value of the source changes is found. With the probabilities computed in line 1, the masking value of the reconvergence subnet is computed (line 17). To compute the vulnerability of the gate, we need to combine that value with the vulnerability of the reconvergence destination, i.e., the probability that the fault is not masked from the destination until an output of the circuit. This process is repeated for all subnets for which the gate is a source (line 15) and combined into one probability for the node in line 20.

Finally, in line 22, we compute the vulnerability of the whole circuit by combining the vulnerabilities of all its connections, assuming that the strikes are equiprobable in all connections. Therefore, the vulnerability of the circuit is the arithmetic mean of the vulnerabilities of all connections.

V. Experimental Results

In this section, we report the validation results for our signal probability and circuit SER estimation methods. The experiments were performed on a machine with an Intel Core i7-4500U running at 2 GHz with 8 GB of RAM.

A. *Signal probability estimation*

In order to analyze the accuracy of the several signal probability computation methods, we compare their estimation with the output of circuit simulation. To do so, random input vectors are sampled from an input distribution, the circuit is simulated and the value at each node is noted. After 10^7 simulation runs, the reference ratio between 0 and 1 (the probability that there is a 0 at that node) at each node is obtained. Then, the different algorithms, whose outputs are signal probabilities per node, are executed. Finally, the difference between the reference probability and the algorithm probability is measured for each node. The quality of each algorithm is based on the arithmetic mean of those differences. We used circuits from the ISCAS'85 benchmarks to conduct these experiments. 100 different input distributions were tested to extract conclusions.

Figure 2 shows that previous algorithms (0-ALG, WAA, DWAA, CCM and POSS) do not produce accurate enough estimations. On average, the signal probability estimation produced by POSS is 33% better than the 0-Algorithm and the estimation by CCM is a 21% better. The Dynamic WAA only estimates a 3% better while WAA introduces an additional 35% error. The Possibilistic algorithm is the one that yields the best results in these experiments but its estimation comes, on average, with a 0.023 difference per node in comparison with Monte Carlo signal probability simulation, which still is a very high error. On the other hand, with the methodology proposed in this work, the remaining error is only 0.1% (a 96% reduction with respect to POSS), which is considered good enough to build the vulnerability estimation upon.

Figure 3 plots the trade-off between the accuracy obtained simulating fan-out subnetworks up to a particular size and the execution time of those simulations.

A detailed analysis of the benchmark circuits reveals that the majority of reconvergent fan-out subnets contain 30 nodes or less, and that the average number of inputs per fan-out cone of that size is 21. Simulating 2^{21} inputs can be done in milliseconds and, since the circuits have an average of 450 reconvergent fan-out subnets, the simulation of all the cones is completed in a few seconds.

If larger subnets are considered, gates with big fan-in counts (5 or more) become more frequent, and the number of inputs reaching the subnet increases dramatically with the subnet size. Moreover, despite the extra computation effort, the differences between the simulated probabilities and the computed probabilities are not reduced significantly. We conclude that simulating the small subnets is feasible in terms of execution time and it provides a significant improvement in the computation of signal probabilities.

B. *Model validation*

For the validation of our framework, we compare the output of the algorithm with the results of fault injection experiments. The experiments consist of three variables: the input probability distribution, the circuit upon which the injection is performed and the number of iterations. The input distribution consists of a random number between 0 and 1 for each primary input of the circuit. The circuits correspond to the ISCAS'85 benchmarks synthesized using the Design Compiler from Synopsys with 15nm Nangate Open Cell Library [17] as selected technology. The cell characterization of that library is performed using the methodology in [31].

The fault injection approach consists in changing the bit at the output of a gate. To perform an experiment, a particular

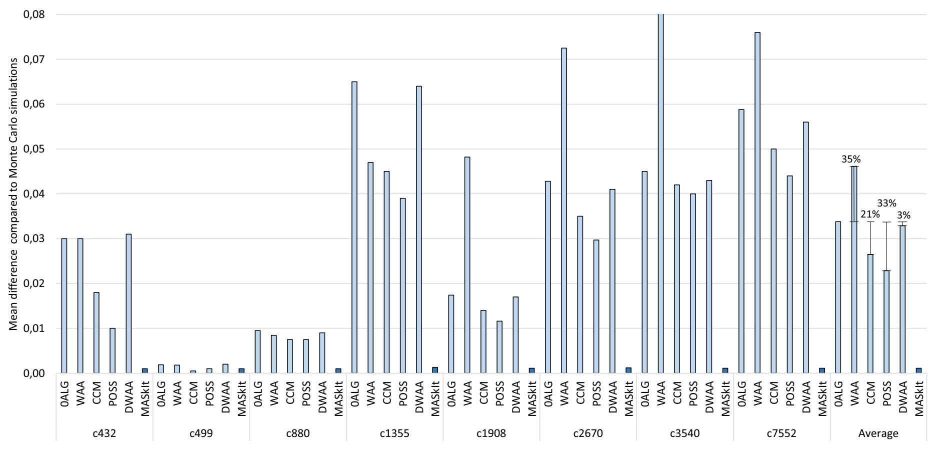

Fig. 2. Comparison among methodologies to estimate signal probabilities.

TABLE I. Validation experiments results for ISCAS'85 Benchmarks

Circuit		Fault injection		MASkIt		
Name	Gate count	Vulnerability	Execution time	Vulnerability	Execution time	Relative error (%)
c432	160	0.257	1 hour	0.256	15 seconds	0.5
c499	202	0.467	1.5 hours	0.468	22 seconds	0.004
c880	383	0.53	4.7 hours	0.515	33 seconds	2.7
c1355	546	0.385	10 hours	0.365	46 minutes	5.4
c1908	880	0.398	22 hours	0.369	3 minutes	7.3
c2670	1193	0.361	40 hours	0.339	9 minutes	6.14
c3540	1669	0.275	80 hours	0.249	22 minutes	9.5
c5315	2307	0.432	7.5 days	0.409	45 minutes	5.2
c7752	3512	0.375	1.5 weeks	0.344	1.5 hours	8.5

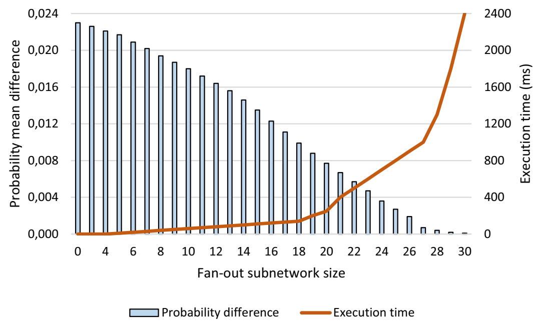

Fig. 3. Comparison between fan-out subnet size, accuracy and runtime.

input vector from an input distribution is generated and the circuit is simulated. Then, faults are injected iteratively to all gates and outputs are observed for changes with respect to the original simulation. If there are no differences among the original outputs and the fault-injected ones, the fault was masked. If one or more outputs have been changed, the fault was not masked and a counter, representing vulnerability for the gate, is incremented. This process is repeated for a number of iterations, inside which new input vectors are sampled from the input distribution. When a sufficiently large number of input vectors have been simulated, the vulnerability counter at each of the gates is divided by the number of iterations, thus computing the ratio of experiments in which a fault in the gate was not masked, i.e., its vulnerability. These actions are repeated for 100 different input probability distributions.

The number of samples in a Monte Carlo experiment is defined in [33] as a factor of the interval of confidence, the margin of error and the estimated deviation at the output. For our experiments, we chose an interval of confidence of 95%. Since the outputs of the experiment are probabilities representing vulnerability, we chose a margin of error of 1%. Even though we do not know the standard deviation of our population, it can be estimated by a large number of beforehand preliminary simulations (such as 10^7, as suggested in [39]). Therefore, according to [33], the number of Monte Carlo trials is 3934. We rounded this number to 4000 iterations.

Table I shows that the proposed approach achieves excellent accuracy with two orders of magnitude reduction in execution time with respect to the classic fault injection technique. The maximum relative error is below 10% with an average of 5%, while the average absolute error is 1.9% and the maximum is 3.1%. If the circuit contains few reconvergent fan-outs, such as c499, the model outputs results almost identical to those obtained with fault injection. On the other hand, the size of the circuit does not correlate with the precision of the algorithm. The average MASkIt runtime, which includes signal probability estimation, is 350 times faster than classic fault injection.

VI. Conclusion

In this paper, we have presented MASkIt: a Soft Error Rate estimation methodology for combinational logic. Our approach is based on a backward-search traversal of the circuit that computes the different masking factors for all gates by combining signal probabilities with a pre-characterized technology library. Signal probabilities are approximated using a novel approach that integrates an heuristic algorithm along with the simulation of selected RFON subnets within the circuit. Experimental results indicate that MASkIt achieves a speedup of two orders of magnitude over Monte Carlo fault injection campaigns while incurring in an average error of 5%, and it is much more accurate than previous techniques with similar computational requirements, mainly due to the high-accuracy of the novel approach to compute signal probabilities.

Acknowledgement

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness and Feder Funds under grant TIN2013-44375-R, by the Generalitat de Catalunya under grant FI-DGR 2016 and by the FP7 program of the EU under contract FP7-611404 (CLERECO)

References

[1] Al-Kharji, M. A., and Al-Arian, S. A. A new heuristic algorithm for estimating signal and detection probabilities. In *7th Great Lakes Symp. on VLSI* (1997), IEEE, pp. 26–31.

[2] Asadi, G., and Tahoori, M. B. An analytical approach for soft error rate estimation in digital circuits. In *Int. Symp. on Circuits and Systems* (2005), IEEE, pp. 2991–2994.

[3] Asadi, H., and Tahoori, M. B. Soft error derating computation in sequential circuits. In *Proc. of the 2006 IEEE/ACM Int Conf. on Computer-Aided Design* (2006), ACM, pp. 497–501.

[4] Baraza, J.-C., et al. Enhancement of fault injection techniques based on the modification of vhdl code. *IEEE Trans. on VLSI 16*, 6 (2008), 693–706.

[5] Bryan, D. The iscas'85 benchmark circuits and netlist format. *North Carolina State University* (1985), 25.

[6] Bryant, R. E. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys 24*, 3 (1992), 293–318.

[7] Chang, A. C.-C., et al. Casser: a closed-form analysis framework for statistical soft error rate. *IEEE Trans. on VLSI 21*, 10 (2013).

[8] Chen, L., Ebrahimi, M., and Tahoori, M. B. Cep: correlated error propagation for hierarchical soft error analysis. *Journal of Electronic Testing 29*, 2 (2013), 143–158.

[9] Chen, L., and Tahoori, M. B. An efficient probability framework for error propagation and correlation estimation. In *18th Int. On-Line Testing Symp.* (2012), IEEE, pp. 170–175.

[10] Ebrahimi, M., et al. Class: Combined logic and architectural soft error sensitivity analysis. In *18th Asia and South Pacific Design Automation Conf.* (2013), IEEE, pp. 601–607.

[11] Entrena, L., et al. Soft error sensitivity evaluation of microprocessors by multilevel emulation-based fault injection. *IEEE Trans. on Computers 61*, 3 (2012), 313–322.

[12] Ercolani, S., Favalli, M., Damiani, M., Olivo, P., and Ricco, B. Estimate of signal probability in combinational logic networks. In *European Test Conference, 1989., Proceedings of the 1st* (1989), IEEE, pp. 132–138.

[13] Franco, D. T., et al. Signal probability for reliability evaluation of logic circuits. *Microelectronics Reliability 48*, 8 (2008), 1586–1591.

[14] George, N. J., et al. Transient fault models and avf estimation revisited. In *IEEE/IFIP Int. Conf. on Dependable Systems and Networks* (2010), IEEE, pp. 477–486.

[15] Hamad, G. B., et al. Characterizing, modeling, and analyzing soft error propagation in asynchronous and synchronous digital circuits. *Microelectronics Reliability 55*, 1 (2015), 238–250.

[16] Huang, R. H.-M., and Wen, C. H.-P. Advanced soft-error-rate (ser) estimation with striking-time and multi-cycle effects. In *51st ACM/EDAC/IEEE Design Automation Conf.* (2014), IEEE, pp. 1–6.

[17] Inc, N. Nangate 15nm open cell library, 2014.

[18] Khalid, U., et al. Reliability-evaluation of digital circuits using probabilistic computation schemes. In *National Postgraduate Conf.* (2011), IEEE, pp. 1–4.

[19] Kimi, Y., et al. An accurate soft error propagation analysis technique considering temporal masking disablement. In *21st Int. On-Line Testing Symp.* (2015), IEEE, pp. 23–25.

[20] Kriebel, F., et al. Acsem: Accuracy-configurable fast soft error masking analysis in combinatorial circuits. In *Proc. of the Conf. on Design, Automation and Test in Europe* (2015), IEEE, pp. 824–829.

[21] Krishnamurthy, B., and Tollis, I. G. Improved techniques for estimating signal probabilities. *IEEE Trans. on Computers 38*, 7 (1989).

[22] Krishnaswamy, S., et al. Signature-based ser analysis and design of logic circuits. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems 28*, 1 (2009), 74–86.

[23] Kuo, Y.-H., et al. Accurate statistical soft error rate (sser) analysis using a quasi-monte carlo framework with quality cell models. In *11th Int. Symp. on Quality Electronic Design* (2010), IEEE, pp. 831–838.

[24] Michael, R. G., and David, S. J. Computers and intractability: a guide to the theory of np-completeness. *WH Free. Co., San Fr* (1979).

[25] Miskov-Zivanov, N., and Marculescu, D. Circuit reliability analysis using symbolic techniques. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems 25*, 12 (2006), 2638–2649.

[26] Mukherjee, S. *Architecture design for soft errors*. M. Kauf., 2011.

[27] Mukherjee, S. S., et al. A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor. In *Proc. 36th Annu. IEEE/ACM Int. Symp. on Microarchitecture* (2003), IEEE, p. 29.

[28] Rajaraman, R., et al. Seat-la: a soft error analysis tool for combinational logic. In *19th Int. Conf. on VLSI Design* (2006), IEEE.

[29] Ratiu, I. M., et al. Victor: A fast vlsi testability analysis program. In *ITC* (1982), pp. 397–403.

[30] Rejimon, T., et al. Probabilistic error modeling for nano-domain logic circuits. *IEEE Trans. on VLSI Systems 17*, 1 (2009), 55–65.

[31] Riera, M., et al. A detailed methodology to compute soft error rates in advanced technologies. In *Proc. of the Conf. on Design, Automation and Test in Europe* (2016), European Design and Automation Assoc.

[32] Roberts, M., and Lala, P. Algorithm to detect reconvergent fanouts in logic circuits. *IEEE Proc. Comp. and Digital Tech.* (1987), 105–111.

[33] Robey, R. R., and Barcikowski, R. S. Type i error and the number of iterations in monte carlo studies of robustness. *British Journal of Math. and Statistical Psychology 45*, 2 (1992), 283–288.

[34] Shivakumar, P., et al. Modeling the effect of technology trends on the soft error rate of combinational logic. In *Proc. Int. Conf. on Dependable Systems and Networks* (2002), IEEE, pp. 389–398.

[35] Takata, T., and Matsunaga, Y. A robust algorithm for pessimistic analysis of logic masking effects in combinational circuits. In *17th Int. On-Line Testing Symp.* (2011), IEEE, pp. 246–251.

[36] Wang, F., and Agrawal, V. D. Soft error rate determination for nanometer cmos vlsi logic. In *40th Southeastern Symp. on System Theory* (2008), IEEE, pp. 324–328.

[37] Wang, F., and Xie, Y. Soft error rate analysis for combinational logic using an accurate electrical masking model. *IEEE Trans. on Dependable and Secure Computing 8*, 1 (2011), 137–146.

[38] Wang, N. J., et al. Characterizing the effects of transient faults on a high-performance processor pipeline. In *Int. Conf. on Dependable Systems and Networks* (2004), IEEE, pp. 61–70.

[39] Winston, W. L. *Simulation modeling using@ RISK*. Duxbury, 2000.

[40] Zhang, B., et al. Faser: Fast analysis of soft error susceptibility for cell-based designs. In *Proc. of the 7th Int. Symp. on Quality Electronic Design* (2006), IEEE, pp. 755–760.

[41] Zhang, M., and Shanbhag, N. R. Soft-error-rate-analysis (sera) methodology. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 25*, 10 (2006), 2140–2155.