Fast and Accurate SER Estimation for Large Combinational Blocks in Early Stages of the Design

Martí Anglada, Ramon Canal, *Senior Member, IEEE*, Juan L. Aragón, and Antonio González, *Fellow, IEEE*

Abstract—Soft Error Rate (SER) estimation is an important challenge for integrated circuits because of the increased vulnerability brought by technology scaling. This paper presents a methodology to estimate in early stages of the design the susceptibility of combinational circuits to particle strikes. In the core of the framework lies *MASkIt*, a novel approach that combines signal probabilities with technology characterization to swiftly compute the logical, electrical, and timing masking effects of the circuit under study taking into account all input combinations and pulse widths at once. Signal probabilities are estimated applying a new hybrid approach that integrates heuristics along with selective simulation of reconvergent subnetworks. The experimental results validate our proposed technique, showing a speedup of two orders of magnitude in comparison with traditional fault injection estimation with an average estimation error of 5 percent. Finally, we analyze the *vulnerability* of the Decoder, Scheduler, ALU, and FPU of an out-of-order, superscalar processor design.

Index Terms—Combinational logic, microprocessor, reliability, soft errors

1 Introduction

Technology scaling has provided substantial benefits such as higher frequencies and increasing transistor density. On the other hand, scaling into sub-100 nm lithographies has brought new challenges, notably in the field of reliable circuit design. With lower supply voltages and node capacitances, integrated circuits have become more vulnerable to single-event transients (SETs): disturbances caused by particle strikes from ionizing radiation. Radiation-induced transients are primarily caused by thermal and high-energy neutrons coming from cosmic rays and by alpha particles coming from packaging materials [1]. Alpha particles collide with the silicon nuclei, while thermal and high-energy neutrons generate ionizing particles. In both cases, they deposit a dense track of electron-hole pairs as they pass through a *p*-*n* junction, and causing a current pulse at the node that collects the charge. If a sufficient amount of charge is collected by the junction, the SET results in a fault by flipping the logic state at the associated node. Bit flips may occur due to strikes in the transistors of a memory element or due to the propagation of strikes at the transistors of logic gates to latching elements. When a fault caused by a bit flip is captured by a storage element such as a memory cell or a register, it results in a single event upset (SEU). As opposed to events that may cause permanent failures, SEUs do not damage the device and are, therefore, called *transient* or *soft errors*. The Soft Error Rate (SER) is a metric to evaluate the vulnerability of a circuit against soft errors.

Historically, soft errors have been tackled in the context of memory because of the large area of the chip devoted to caches and the higher vulnerability of SRAM cells in comparison to combinational logic. However, technology scaling led to a reduction in the ratio between the minimum charge required to introduce a glitch (Q_{crit}) and the collection efficiency of transistors (Q_s), which increased the SER in logic circuits by five orders of magnitude [2], making it comparable to the SER of unprotected memory. While the adoption of FinFET technology has provided greater robustness against SEU in each generation [3], [4], the increase in transistor density maintains the overall system SER as an essential concern [5], [6].

With such sensitivity, mechanisms to protect circuits from soft errors are needed not only for traditional safety-critical applications but also for mainstream systems [7]. The drawback of effective protection mechanisms such as Quadded logic [8] is that they incur substantial performance, area and power overheads. Alternatively, there exist a set of *selective* hardening techniques that only protect the most sensitive elements of a combinational circuit, whether they are gates [9] or latches [10]. There is, therefore, a need for analysis tools that can provide the designer information about the most critical components of a circuit and the trade-off between the overheads of mechanisms for error resilience and the system-level SER value of the circuit. While reliability estimation techniques such as fault injection [11] already exist, the large

- *M. Anglada, R. Canal, and A. González are with the Departament d'Arquitectura de Computadors, Universitat Politècnica de Catalunya, c/Jordi Girona 1-3, Barcelona 08034, Spain.*
  E-mail: {manglada, rcanal, antonio}@ac.upc.edu.
- *J.L. Aragón is with the Computer Engineering Department, University of Murcia, Campus de Espinardo, Murcia 30100, Spain.*
  E-mail: jlaragon@ditec.um.es.

exploration space that needs to be covered in order to choose a proper hardening strategy makes the use of such slow, brute-force techniques impractical.

Modelling and analyzing the SER in logic is more complex than in memory elements, since there is an exponential number of input vectors, signal correlations due to reconvergence and combinations with pulse widths. Even more importantly, there exist some *masking effects* that reduce the likelihood that a particular SET within a circuit will be latched and causing an error. These masking effects are commonly classified as [12]:

- *Logical masking*: Transient faults are masked by gates whose output is independent of the faulty input.
- *Electrical masking*: The pulse is attenuated (either its amplitude is reduced or its 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.

A correct modelling of these effects is required for an accurate SER estimation. Recent works show the consequences of not considering them: if *logical* masking is not taken into account, the SER could be overestimated by 25 times [13]. Omitting *electrical* masking could yield a pessimistic result of more than 3 times [14], while omitting *timing* masking implies considering erroneously the contribution to the total SER of 62 percent of the gates of the circuit [15].

This work presents an accurate and fast framework to estimate the SER in combinational logic considering the three aforementioned masking effects and taking into account all possible input combinations and pulse widths. In the core of the framework lies the proposed MASkIt approach, an algorithm that traverses a circuit backwards from outputs to inputs and computes at each gate the probability that a strike in that gate results in an SEU. Signal probabilities are used to accurately compute the effect of logical masking of all possible input vectors at once, instead of iterating through input combinations. A pre-characterized cell library is applied in the computation of electrical and timing masking effects, taking into account all defined pulse widths and amplitudes simultaneously to avoid simulating glitch propagation for each set of pulse characteristics. Note, however, that the presence of reconvergent subnetworks generates dependences among signals, which invalidates the use of trivial procedures based on path probabilities to compute the SER of a circuit. MASkIt employs two different mechanisms to deal with reconvergences to fast and accurately estimate signal probabilities and vulnerability:

- The estimation of signal probabilities is done using a novel hybrid method that merges static analysis with selective subcircuit simulation: all small-sized reconvergent subnetworks of the circuit are exhaustively simulated whereas a linear-time estimation algorithm is run to obtain the signal probability for the rest of the gates.
- Reconvergences during vulnerability estimation are handled using Binary Decision Diagrams (BDD), a well-known data structure to operate on boolean functions, but we have limited their use to reconvergent subnetworks in order to achieve scalability.

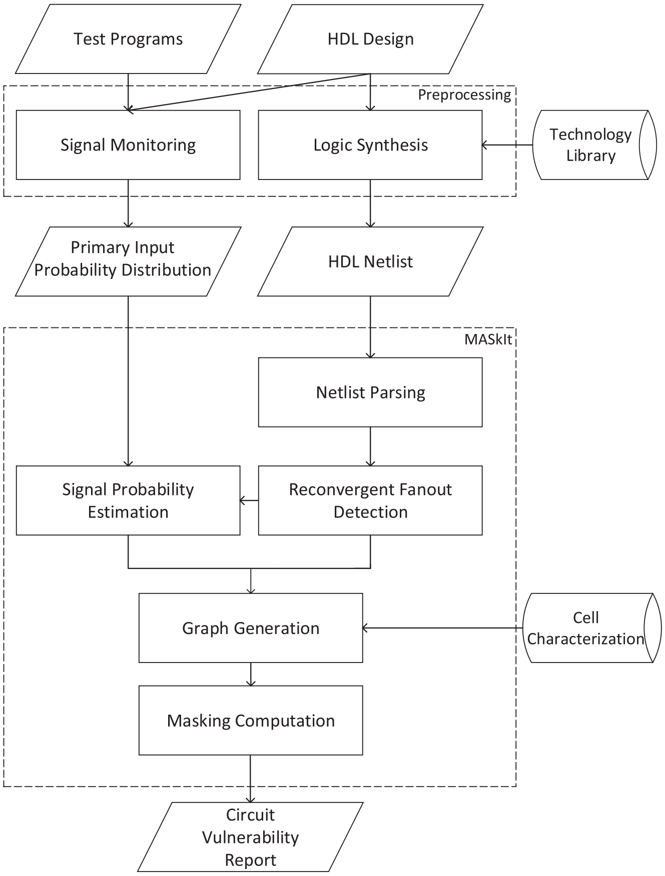


Fig. 1. Overview of the proposed MASkIt framework.

The main algorithm is devised to be easily embedded within a larger toolflow in order to accurately estimate the SER of combinational blocks of a microprocessor in early design stages. Fig. 1 shows an example of such toolflow. With a logic synthesis tool, several alternate netlists can be generated from the same HDL design of the combinational block by modifying configuration parameters, such as clock period or technology process. The primary input probability distribution for the netlist can be obtained through a tracing tool by running a set of test programs and getting the signals corresponding to the primary inputs of the block under study. Such primary input probability distributions are the basis to estimate the probabilities of the signals of the whole circuit using the procedure detailed in Section 5. With the approach described in Section 4, MASkIt uses the computed signal probabilities along with a cell characterization library to estimate the vulnerability of the circuit taking into account the three masking effects. Due to MASkIt's ability to quickly compute the SER and provide a vulnerability report for a particular configuration of parameters, a large design space can be explored as early in the design process as an HDL design is available.

The remainder of this paper is organized as follows. Section 2 reviews previous work on estimating SER in combinational logic. Sections 3, 4, 5, and 6 are dedicated to the description and validation of MASkIt, the core algorithm of the model. In particular, Section 3 formulates the SER estimation and modeling approach. Section 4 presents the proposed approach for computing the masking effects. Section 5 describes the mechanism to estimate signal probabilities.

Section 6 reports the experimental results of the validation process. Finally, Section 7 depicts a use case of the entire framework for analyzing the vulnerability of a state-of-the-art, out-of-order processor design. The main conclusions of the work are summarized in Section 8.

2 Background and Related Work

In recent years, SER in combinational logic has been an area of extensive research, and a variety of methodologies for its estimation have been developed. These techniques can be classified in two wide categories: dynamic and static.

Dynamic techniques use Monte Carlo fault injection, a classic approach that implies generating random input vectors, injecting a transient fault in the output of a gate and observing if the output values of the circuit differ from the ones obtained in a fault-free simulation.

There are several works that use this methodology to compute the SER in logic. IBM presented a program [16] to determine if a design meets SER specification based on layout, technology information and Q_{crit} values. Wang et al. [17] injected faults in the state elements of a whole processor pipeline to analyze how many transient faults resulted in soft errors. The work of Zhang et al. [18] combines graph theory and fault injection to emulate the logical masking effect. In [19], path-based analysis and cell-precharacterization are used in conjunction with fault injection to compute the system-level SER. A variety of modifications to the classic techniques have been proposed as well. In [20], a way to easily inject a wider fault model set in RTL code with small temporal overhead is introduced. In [21], SET are only simulated when they can not be affected by timing masking and when the circuit has stabilized after a new input vector, achieving simulation speedup by reducing the number of time instances a pulse has to be injected. Kuo et al. [22] build table-based models to characterize particle strikes, propagation paths and latches, and combine them with a heuristic for the generation of random sequences in order to speed up the simulations. In [23], the authors propose to reduce execution time by integrating the RTL and gate-level models in an FPGA to accurately inject faults while being able to quickly propagate them across clock cycles.

Even using such optimizations, the major drawback of these approaches is that a large number of error injections have to be performed in order to obtain statistically significant results, which results in long execution times.

Other works disfavor the use of fault injection, but propose methodologies that still require sampling and simulating input vectors. SEAT-LA [24] pre-characterizes the current-voltage transfer characteristics and the delay of the cells to model glitch propagation. However, it simulates the circuit across multiple input combinations in order to compute the logic state of the nodes to consider logical masking. The proposal of Krishnaswamy et al. [25] reduces the state explosion that appears when considering the different sensitization paths using signature-based analysis, but the computation of signatures entails simulation of all input vectors.

On the other hand, relying on circuit simulation to compute the SER in combinational logic is impractical due to its exponential complexity. In pursuance of smaller execution times, a plethora of static (also known as analytical) approaches that trade accuracy for speedup have been presented.

Static approaches can be, to a large extent, classified into two categories: symbolic and probabilistic. Symbolic techniques abstract the gates of the circuit to data structures which are efficient to manipulate. This way, computations are performed orders of magnitude faster than fault injection campaigns with minimal loss of accuracy. FASER [13] uses Binary Decision Diagrams to describe Boolean functions at circuit nodes and to represent nodes that have been struck. Krishnaswamy et al. propose in [26] to use Probabilistic Transfer Matrices (PTM) and Algebraic Decision Diagrams (ADD) to compute the logical masking in circuit nodes concurrently considering all input combinations. MARS-C [27] also uses BDD and ADD to encode the circuit, but extends previous works to offer a unified treatment of all three masking factors, while the methodology in [28] employs PTM to analyze gates under stuck-on faults. The work of Abdollahi [29] presents Probabilistic Decision Diagrams, an exact representation that implicitly considers signal correlations, allowing fast computations due to its compactness. The main drawback associated to the aforementioned techniques is their lack of scalability: the diagram representation requires a considerable amount of memory and simulation time to enumerate the massive input space in large circuits.

Probabilistic approaches, alternatively, develop error propagation rules to compute the reliability of the circuit by using them in sensitized paths, usually found using static analysis. The method in [30] uses an enhanced static timing analysis to derive all possible waveforms propagated from a struck gate and combines it with error propagation rules to calculate the probability of storing an incorrect value in all reachable latches. BDEC, the error calculator presented in [31], introduces a probabilistic gate level error propagation model based on Boolean difference. Using primary input signals, error probabilities and gate error probabilities, the reliability of the circuit is computed in linear-time complexity with the number of gates of the circuit. In [32], Han et al. also construct a gate reliability model that derives approximate reliability results by bounding gate errors at much smaller time and space complexities than PTM. Upper and lower bounds are used in [33] to improve scalability as well, where they are used in the context of gate observability. The work of Rejimon and Bhanja [34] adopts Bayesian Networks to capture the dependencies among signals, constructing for each node of the network a function for the truth table of the gate including error probabilities. CEP [35], presents rules for the propagation of signal and error correlations in addition to the traditional propagation rules, allowing an accurate computation of the error probability of the circuit. The proposal in [36] extends the work in CEP by using a SAT solver to compute signal probabilities in reconvergent nodes, which reduce the accuracy of the correlation rules. Cai and Chen also propose to use signal correlations in [37], where they introduce a new set of rules that achieve excellent accuracy. The probabilistic techniques on these works, however, only demonstrate their feasibility on small-sized circuits and the required analysis to run them would require a significant analysis time for medium and large-sized circuits.

TABLE 1
Related Work Summary

Circuit size	Number of considered masking effects		
	1 effect	2 effects	3 effects
~10 gates	[26], [28], [32]	[41], [42]	
~100 gates	[29], [33], [34]	[36]	[18], [21]
~1000 gates	[31], [37], [38], [39]	[2], [15], [27], [40]	[22], [24], [43]

There are two major drawbacks concerning the majority of the works in the literature. The first one is the lack of a framework capable of computing all three masking factors in a holistic way, which impacts the correctness of those works. The techniques presented in [31], [33], [37], [38] and [39] only take into account logical masking. On the other hand, approaches such as [15], [27], [40], [41] or [36] do not compute the electrical masking effect, while [42] and [2] overlook the logical masking effect. The second drawback is that a large body of the presented techniques use only small circuits as benchmarks, which raises concerns regarding the scalability of the works. Benchmarks classified as *small* include a range from less than ten gates [32], [42], tens of gates [26], [41], a few hundred of gates [18], [21], [29], [34] and a thousand gates [24], [27], [31]. Table 1 summarizes the discussed literature works within the two aforementioned drawback dimensions.

Several works provide a higher-level error estimation, considering the effects that hardware faults may cause at the software layer. Rehman et al. [44] and Shafique et al. [45] propose compiler-based mechanisms that take into account the masking provided by program instructions. Brosch et al. [46] present a reliability model that integrates system usage and execution environment. Carbin et al. [47] introduce a programming language that allows to exploit unreliable hardware for faster and more energy-efficient computations. As circuit-level SER estimation is orthogonal to software-level techniques such as the ones described, it can be used to enhance the overall system SER estimation.

The algorithm presented in this work is able to model the three masking effects and does not require an exponential running time, as it will be shown in Section 4. The validation of the model (Section 6) is performed using the ISCAS'85 benchmark suite, which contains circuits ranging from hundreds of gates to a few thousands. To prove the scalability and effectiveness of our MASkIt framework, Section 7 presents as use case considering circuits comprised of more than 10,000 gates: some major combinational blocks of a state-of-the-art, out-of-order processor, including the instruction decoder, the instruction scheduler, the Arithmetic-Logical Unit and the Floating Point Unit.

3 MASkIt Overview

3.1 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

$$SER_{circuit} = \sum_{i=1}^{\#gates} SER_{gate_i}. \tag{1}$$

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 Soft Error Transient (SET) 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

$$SER_{gate_i} = \sum_{c=1}^{\#charges} R_{PH}(q_c) * V(gate_i, q_c), \tag{2}$$

where V is the vulnerability of the gate and R_{PH} is the particle hit rate, defined in [2] 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 [13] 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

$$V(gate_i, q_c) = LM(i,c) * EM(i,c) * TM(i,c), \tag{3}$$

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).

3.2 Netlist Modeling

Given a combinational circuit, our approach needs three inputs: its netlist, 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 primary inputs of the circuit and the set Ω of primary 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 ith gate of G and Γ_i is a set of gates $\{g_m^a, g_n^b, g_o^c, ...\}$, 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 ith gate of G and a corresponds to input number a ($0 \leq a < k$) from g_i.
- Each element of Ω is pair $<i, f>$, which indicates that gate g_i of G is connected to the output flip-flop f of the circuit.

Every element i from the input set I has associated a probability P_i, which corresponds to the likelihood of input i to be 0.

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 percent of V$_{dd}$ and from 90 to 10 percent of V$_{dd}$, respectively).
- Every element g_i of G has associated a propagation delay δ, the minimum switching delay among all input combinations of the gate.
- 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.

MASkIt computes a set of vulnerability probabilities V using Equation (3) 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, taking into account the three masking effects. Finally, it calculates the vulnerability of the whole circuit based on those probabilities using Equations (1) and (2).

4 Masking Probability Computation

4.1 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 LM, the probability that a gate is not affected by the logical masking factor, 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 as SET for a 2-input NAND gate can be expressed as

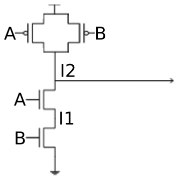
$$LM_{NAND2} = \\ P_0 * (1 - P_1) * T_0^{in}[0 \rightarrow 1] \tag{4a}$$

$$+ (1 - P_0) * P_1 * T_1^{in}[0 \rightarrow 1] \tag{4b}$$

$$+ (1 - P_0) * (1 - P_1) * (T_0^{in}[1 \rightarrow 0] + T_1^{in}[1 \rightarrow 0]), \tag{4c}$$

where (4a) and (4b) correspond to case 2-b) in the previous example while (4c) 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.

These transition probabilities take into account two phenomena regarding particle strikes in logic gates. The first one is that strikes affect particular internal nodes of transistors, activating off transistors. Depending on which transistors of the gate are active, different internal nodes will be sensitive, which affects the likelihood of a strike upsetting the state.



A	B	Out	Out|I1	Out|I2
0	0	1	1	1
0	1	1	1	0
1	0	1	0	1
1	1	0	0	1

Fig. 2. NAND-2 sensitivity analysis. When Out|I1 or Out|I2 differ from Out, there is a probability that a strike in node I1 or I2 would cause a bit-flip in the output of the gate.

To illustrate this effect, Fig. 2 depicts a 2-input NAND gate at a transistor level. Since strikes activate off transistors, if the input combination is {A=1,B=1 } and a particle strikes at internal node I1 (Out | I1), the output of the gate will not be affected. On the other hand, if a particle strikes at internal node I2 (Out | I2), the output will be upset and suffer a transition. If the input combination is {A = 0, B = 0} neither a particle strike at internal node I1 nor a strike at internal node I2 will generate an undesired transition.

The second phenomena that transition probabilities take into account is that the larger the fanout of a gate is, the smaller the likelihood of a strike upsetting its output becomes. These probabilities are extracted from the cell characterization library, which is performed using the methodology in [48].

In a similar fashion to Equation (4), equations to measure the logical masking effect for any kind of gate (not only NANDs) can be constructed, in order to calculate the LM term in Equation (3). These equations are applied in the backward traversal algorithm of MASkIt to compute the logical masking effect of each gate in the circuit.

4.2 Electrical Masking

Electrical masking occurs because of the natural attenuation of the undesired pulse during its propagation along 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 an SET at that gate would be 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) 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 an output 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 using the well-known glitch propagation model in [24]. This model has also been adopted by a variety of recent works in the literature, such as [49], [50], [51] and [52]. The 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.

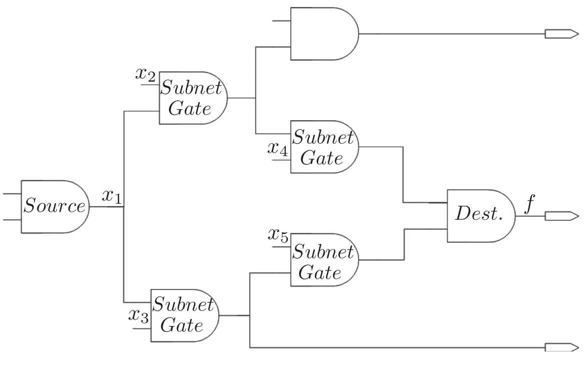


Fig. 3. Example of a reconvergent fanout subnet.

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 an 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 (3).

4.3 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 (3).

4.4 Reconvergent Fanouts

When a logic signal splits into multiple branches (fanout) and later reconverges in two or more inputs of a gate, the subnet thus formed is referred to as a *reconvergent fanout* (RFON). Reconvergent fanouts need to be taken into account because they break the assumption that signal paths are independent. Fig. 3 shows an example of this: f is the output of a gate whose inputs are not independent because both are functions of the same signal, x_1. We identify the gate that produces the branching signal as the *Source* of the reconvergence, while we identify the gate that receives the dependent inputs as the *Destination* of the dependence. Additionally, it is not an uncommon issue, since it has been reported that in current VLSI designs about half of the nodes in a circuit cause a reconvergence [53].

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. The Boolean difference of a function f with respect to input x_i is

$$\frac{df}{dx_i} = f(x_1, \ldots, x_i = 0, \ldots, x_n) \oplus f(x_1, \ldots, x_i = 1, \ldots, x_n). \quad (5)$$

When the difference equals 1, the result is all input combinations for which a change in x_i also changes the output of f. Note that this is equivalent to listing all input combinations[1] of the subnetwork for which a SET in its source node would cause a change in the logical value at the output of its destination node. Since the signal probabilities of all the inputs of the subnetwork are known, we can obtain the overall probability that, given a SET at the source of a reconvergence, the value at the destination is altered. An efficient structure to represent and operate with Boolean functions is a Binary Decision Diagram [54]. 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 to 0 and another one corresponding to the evaluation of the variable to 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 [54]. 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

[1] We define as *inputs* the leads reaching the subnetwork and coming from nodes not contained in the subnetwork. The inputs of the subnetwork in Fig. 3 are $x_2 \ldots x_5$.

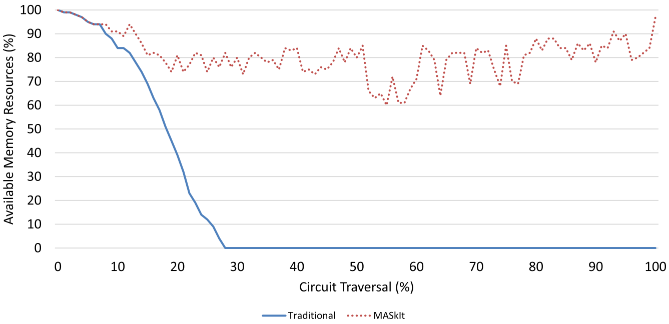


Fig. 4. Comparison of BDD memory usage in a traditional approach and in the MASkIt approach.

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 [13] or [55], in MASkIt the diagrams are only created to handle RFON subnets and, consequently, the exponential nature of the *Satisfy all* procedure is bounded for a small subset of gates of the circuit. Moreover, 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. Thus, less memory usage than in previous works is required.

Fig. 4 shows an example of this difference in memory usage using circuit c7552 of the ISCAS'85 benchmarks, which contains 3,512 gates. In a traditional approach, the exponential nature of the graph size leads to a swift unavailability of resources: before 30 percent of the circuit has been visited, the construction of the BDD has consumed the entirety of free memory, greatly impacting the performance of the later traversal and computations. On the other hand, the approach adopted by MASkIt of only constructing BDDs in reconvergent subnets vastly reduces the resource readiness issue since, at any given point in time, at most 40 percent of the memory is used by BDDs.

4.5 MASkIt Algorithm

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 5 (line 1). Afterwards, it executes a Breadth-First Search (lines 3 to 22) 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 the gates and visit them in the proper order.

In every iteration of the main loop, one gate whose successors have already been visited is chosen to compute its vulnerability. This selection is performed by popping the first element of the queue (line 4). That gate is then marked as visited (line 5) and removed from the queue. The predecessors of the gate are queued 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. The computation of the vulnerability of the connection at the output of the gate is done considering all alternative paths from the gate 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. The computation of the vulnerability of a successor path is performed according to whether or not the visited gate is a source of reconvergence.

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:    $gate$ pop 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:       $V_j(gate) \leftarrow$ Compute vulnerability of any path from
          $gate$ to a circuit output via $j$ using the $LM$, $EM$ and
          $TM$ effects of $j$ and the vulnerability of $j$
13:     end for
14:     Compute vulnerability of the connection at the output
        of $gate$ with all $V_j(gate)$ values
15:   else
16:     for $j \leftarrow$ {Reconvergent subnets in $gate$} do
17:       Traverse subnet $j$, building BDDs for each gate, until
          the reconvergence destination of $j$ is reached
18:       $V_j(gate) \leftarrow$ Compute vulnerability of any path from
          $gate$ to a circuit output via $j$ using the sensitization
          BDD of $j$ and the vulnerability of the reconvergence
          destination of $j$
19:     end for
20:     Compute vulnerability of the connection at the output
        of $gate$ with all $V_j(gate)$ values
21:   end if
22: end while
23: Compute the vulnerability of the circuit based on the
    vulnerabilities of all connections
```

The majority of gates are not sources of reconvergence. In that case (line 10), every input from all the successor gates of the gate being visited are considered (line 11). The probability $V_j(g)$ that a SET at the output of a gate g reaches a latch through a path traversing successor j corresponds to the probability that the SET is not affected by the masking effects of gate j nor by the masking effects of the path from j to the output, i.e., the vulnerability $V(j)$ of gate j. Since the masking effects of gate j and its vulnerability are independent, we can express the vulnerability of a path from g to the output of the circuit traversing j as

$$V_j(g) = LM(j) * EM(j) * TM(j) * V(j). \qquad (6)$$

Elements LM, EM, and TM are computed using the mechanisms described in Sections 4.1, 4.2 and 4.3, respectively.

Then, we compute the vulnerability of the connection at the output of the gate as the union of the path vulnerabilities for its s independent successors (line 14)

$$V(g) = \bigcup_{j=1}^{s} V_j(g). \tag{7}$$

When the gate being visited is a source of a reconvergence, the vulnerability of the paths of its successors is not independent and, therefore, we cannot use the method described in lines 10 through 14. As explained before, our solution is to use Binary Decision Diagrams on the reconvergence subnetwork. To do that, we construct the BDD of the reconvergence destination node by traversing the whole subnetwork and building the BDD for each gate (line 17). 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 vulnerability value $V_{BDD}(j)$ of the reconvergence subnework j is computed. To compute the vulnerability of a gate g, 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 (line 18). Since both probabilities are independent, we can express the vulnerability of a path from g to the output of the circuit traversing the destination of reconvergence of j as

$$V_j(g) = V_{BDD}(j) * V(destination(j)). \tag{8}$$

This process is repeated for the s subnetworks for which the gate is a source (line 16) and combined into one probability for the node (line 20) using Equation (7), which assumes independence among all subnetworks.

Finally, in line 23, 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.

5 Estimation of Signal Probabilities

As explained previously, the very first step of MASkIt is to compute the signal probabilities of all the gates (nodes) in a circuit. 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 [56] and, therefore, the usual approach is to estimate such probabilities instead of running an unduly complex algorithm to achieve accuracy.

There are two widely used techniques to solve the problem. The first one consists of sampling input vectors and execute a logical simulation per sample in order to obtain the state at each node. The final probability is computed as the sum of logic states divided by the number of samples. This technique is adopted in SER analysis works such as [24], [57] and [18].

The second approach uses heuristic algorithms in order to avoid the long execution times caused by the large number of samples and simulations needed for high coverage. These algorithms traverse the circuit from primary inputs to primary outputs, computing the signal probability at each gate using the signal probability of the inputs of the gate. The computation is based on the fact that, if input signals are independent, the following elemental rules can be applied:

1) For a NOT gate, the probability P_{NOT} of its output to be 0 given the probability P of having a 0 at the input is
$$P_{NOT} = 1 - P. \tag{9}$$
2) For a 2-input NAND gate, the probability P_{NAND} of its output to be 0 given the probability P_i of having a 0 at input i is
$$P_{NAND} = (1 - P_0) * (1 - P_1). \tag{10}$$
3) For a 2-input OR gate, the probability P_{OR} of its output to be 0 given the probability P_i of having a 0 at input i is
$$P_{OR} = P_0 * P_1. \tag{11}$$

These rules can be extended to any kind of logic gate with an arbitrary number of inputs. Traversing a circuit applying these rules is known as the *0-Algorithm* [58]. The 0-Algorithm runs in linear time and yields exact signal probabilities whenever the network is *free* from reconvergent fanouts. Since reconvergent fanouts break the assumption that signal paths are independent, using the previous elemental rules for gates that are destination of a reconvergence introduces some error in the computation of signal probabilities, which discards the 0-Algorithm as a precise mechanism to estimate signal probabilities.

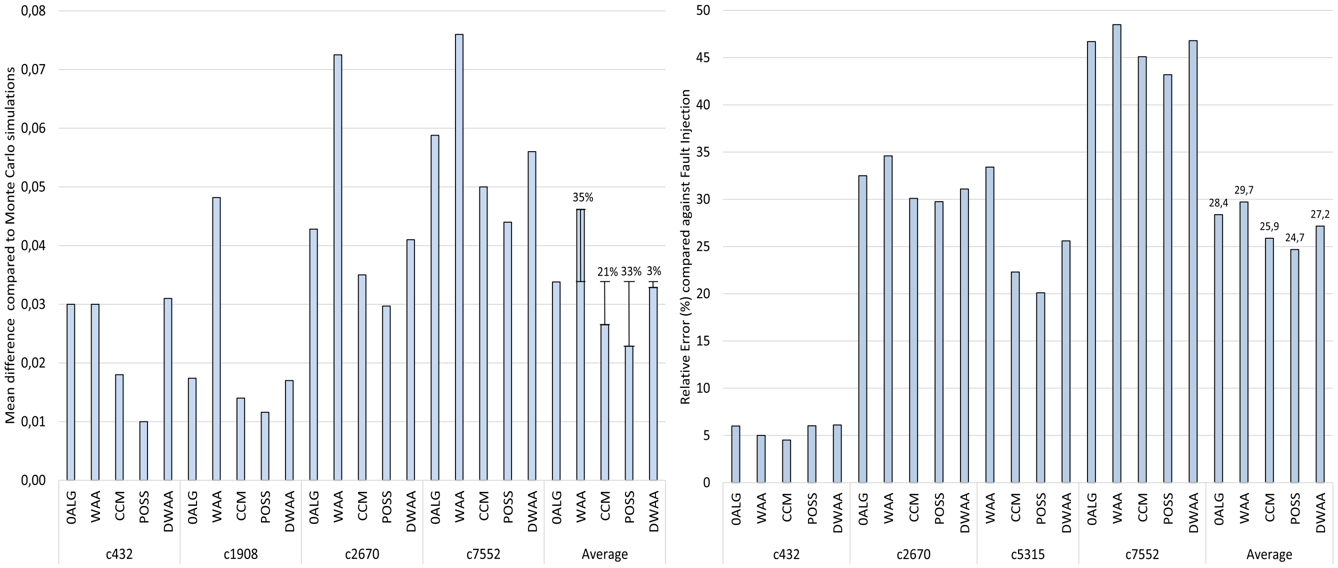
Several alternatives have been proposed in the literature to enhance the accuracy of the 0-Algorithm while remaining in polynomial execution time, such as the Weighted Averaging Algorithm (WAA) [59], the Dynamic Weighted Averaging Algorithm (DWAA) [58] and the Possibilistic Algorithm (POSS) [60]. These algorithms identify nodes that are destinations of reconvergence and compute an estimate for their signal probability as well as an estimate for their influence to succeeding gates according to a series of heuristic weights. While every algorithm derives different heuristics, they all are based on the observation that the signal probability $P(j)$ of a node j can be expressed by

$$P(j) = P(j|f=0) * P(f=0) + P(j|f=1) * P(f=1), \tag{12}$$

where f is a reconvergent fanout node of the fanin cone of j. If f were the only RFON in the circuit, $P(j)$ could be exactly calculated by using formulas such as (9), (10), and (11) 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.

A different approach is the Correlation Coefficient Method (CCM) [58], 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 a signal probability by adding a correlation coefficient to each formula and also proposes a new set of rules to propagate them.

These algorithms have been applied in several recent works in the literature to approximate signal probabilities, such as the use of DWAA in CASSER [61] and the work of



(a) Signal probability estimation error for different algorithms

(b) Model estimation error using different algorithms to estimate signal probabilities

Fig. 5. Accuracy comparison of signal probability and SER estimation for different algorithms.

Franco et al. [62] or the use of CCM in CEP [35], the framework of Li and Draper [63] and the technique of Yoshida et al. [36]. However, we have found these estimates to be inadequate for overall circuit SER computation. Fig. 5a shows the error of the signal probability estimation of the aforementioned algorithms applied to a subset of the ISCAS'85 benchmark circuits. In average, POSS estimates signal probabilities 33 percent more accurately than the 0-Algorithm whereas the estimation by CCM is 21 percent more accurate. DWAA estimates signal probabilities with just 3 percent more accuracy while WAA even worsens the accuracy by increasing the error by 35 percent. While POSS yields a reasonable signal probability estimate with a mean absolute error of only 2 percent, it is important to note that the effect of that error is largely intensified when computing the overall SER of a circuit, as it can be seen in Fig. 5b, which shows the relative error of each signal probability estimation algorithm against fault injection.

A mean error of 2 percent per node implies that the SER computation will be at least 2 percent inaccurate. But, since the SER algorithm computes the vulnerability of a particular gate based on the vulnerability of its successors, the error increases at each step of the traversal of the circuit, resulting in large errors (e.g., 24.7 percent when the signal probabilities are estimated with POSS) when the traversal completes.

In consequence, we propose a new approach to accurately estimate signal probabilities. Our approach is based on the observation that the majority of reconvergent subnetworks in a circuit have sizes that are modest in comparison with the total number of gates of the circuit.[2] Therefore, exhaustive simulation of those reconvergent subnetworks is a feasible task that produces an exact signal probability for the destination nodes without significant execution time overhead.

Fig. 6 shows a histogram of the size of all the reconvergent subnetworks within the ISCAS'85 benchmark circuits. More than 90 percent of the RFON subnetworks have size 30 or less, hence exhaustively simulating all of them results in a good signal probability estimate.

Further analysis of the benchmark circuits reveals that the average number of inputs for reconvergent subnetworks of size 30 is 21. Generating and simulating the 2^{21} possible inputs for a subnetwork of that size is a swift process that can be repeated for all subnetworks without substantial execution time overhead. Furthermore, as seen in Fig. 6, 70 percent of the subnetworks have size 15 or less, which entails negligible simulation time for a great part of the subnetwork population. This allows the entire process of simulating all subnetworks smaller than 30 to be achievable in acceptable execution times.

Algorithm 2 details our proposed approach to estimate signal probabilities, which builds on the Possibilistic algorithm. As a preprocess, reconvergent fanouts are searched using the algorithm of Roberts and Lala to detect all reconvergent fanouts [64]. The circuit is then traversed from inputs to outputs, visiting every node. In the case that the visited node is not a destination of reconvergence, the signal probability for that node is estimated according to the rules of the Possibilistic

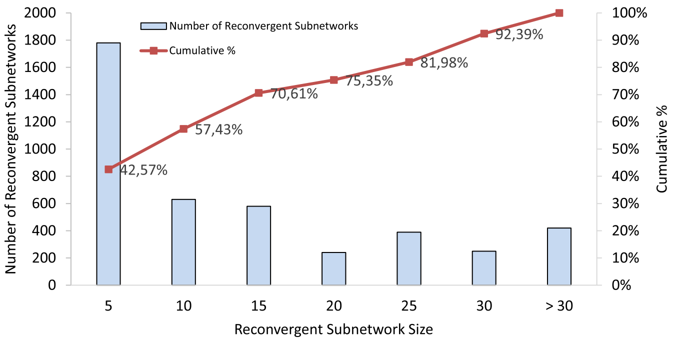


Fig. 6. Histogram of ISCAS'85 reconvergent subnetwork sizes.

[2] We define *reconvergent subnetwork size* as the number of gates covering all paths between the source and the destination of reconvergence.

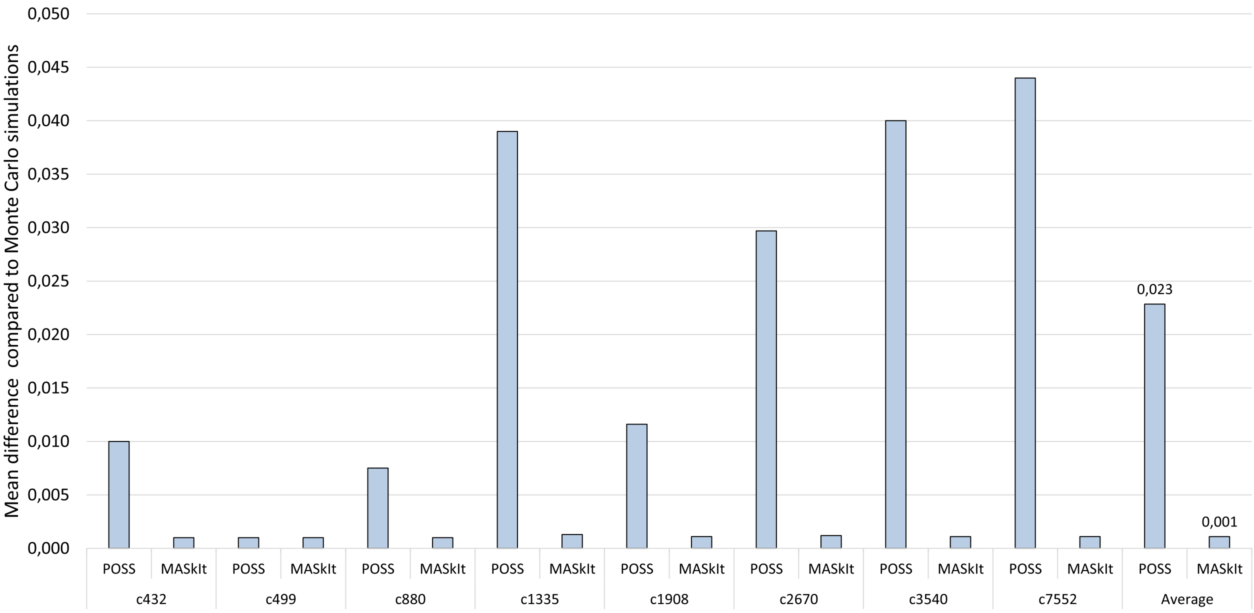


Fig. 7. Comparison among the Possibilistic algorithm and MASkIt to estimate signal probabilities.

algorithm. Those heuristic rules are also used in the case that the node is the destination of reconvergence from a subnetwork of size greater than 30. Only in the case that the node is the destination of reconvergence from a subnetwork of size 30 or smaller, we generate the 2^n inputs of the reconvergent subnetwork. Then, we simulate the subcircuit 2^n times and average the 0-to-1 ratios of the nodes by the likelihood of each input vector. The weight associated to each input vector is computed as the union of the signal probabilities of every signal in the vector, assuming that all of them are independent.

Algorithm 2. Algorithm to Estimate Signal Probabilities

```
$R \leftarrow$ Detect all RFON in the circuit
Traverse the circuit from inputs to outputs, setting in all
  gates the heuristic values needed for POSS
Traverse the circuit from inputs to outputs. **for each** $gate$
  **if** $gate$ is not a destination in $R$ **or** $gate$ is a destination
    with size $> 30$ in $R$ **then**
    SignalProbability($gate$) $\leftarrow$ POSS Heuristic($gate$)
  **else**
    $R_{gate} \leftarrow$ Subnetwork in $R$ whose destination is $gate$
    $IV \leftarrow$ Generate all input vectors in $R_{gate}$
    **for** $i = \{i_1, i_2, \ldots, i_n\} \leftarrow$ Vector in $IV$ **do**
      $O_i$   Signal value of $gate$ after simulating
        $R_{gate}$ with $i$
      $P_i \leftarrow \prod_{j=1}^{n} i_j$
    **end for**
    SignalProbability($gate$) $\leftarrow \sum_{i=1}^{|IV|} O_i \times P_i$
  **end if**
**end for**
```

6 Validation

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.

6.1 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.

Fig. 7 compares the signal probability estimation error obtained using the Possibilistic algorithm (the best performing algorithm from the literature, as seen in Section 5) with the error obtained using the MASkIt approach. Adding the exhaustive simulation of reconvergent subnetworks on top of the Possibilistic algorithm yields a 96 percent reduction of estimation error. The remaining absolute error on average is only 0.1 percent per node, which is considered adequate enough to build the vulnerability estimation upon and also produces positive results for the whole reliability estimation model.

Fig. 8 plots the trade-off between the accuracy obtained simulating fanout subnetworks up to a particular size and the execution time of those simulations.

If larger subnets are considered, gates with big fanin 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

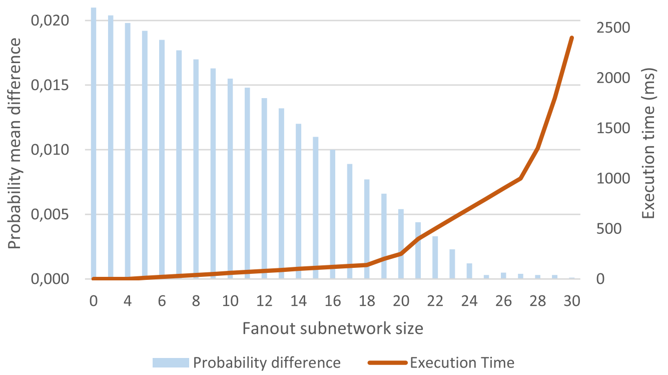


Fig. 8. Comparison between fanout cone size, accuracy, and execution time.

execution time and it provides a significant improvement in the computation of signal probabilities.

6.2 Model Validation

For the validation of our framework, we compare MASkIt against the results of fault injection. 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 15 nm Nangate Open Cell Library [65] as selected technology. The cell characterization of that library is performed using the methodology in [48].

The fault injection approach consists in changing the logic value at the output of a gate. To perform an experiment, a particular 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 by Equation (13) [66]

$$n = \frac{z^2_{\alpha/2} * S^2_n}{\epsilon^2}, \tag{13}$$

where n is the number of iterations needed, $z_{\alpha/2}$ is the critical value of the normal distribution for $\alpha/2$, S_n is the estimated deviation at the output and ϵ is the desired margin of error.

For our experiments, we chose an interval of confidence of 95 percent. Since the outputs of the experiment are probabilities representing vulnerability, we chose a margin of error of 1 percent. 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 [67]). Therefore, according to [66], the number of Monte Carlo trials is 3,934. We rounded this number to 4,000 iterations.

Table 2 shows that the proposed MASkIt 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 percent with an average of 5 percent, while the average absolute error is 1.9 percent and the maximum is 3.1 percent. If the circuit contains few reconvergent fanouts, such as c499, the model outputs results almost identical to those obtained with fault injection. However, the size of the circuit does not correlate with the precision of the algorithm. Finally, in terms of execution time, MASkIt is 350 times faster on average than classic fault injection, making our proposal a fast yet accurate approach for circuit SER estimation also considering masking effects.

7 Use Case: SER Estimation for Combinational Blocks of a Microprocessor

This section demonstrates the use of MASkIt for large, characteristic circuits. To this end we use OPA, an open-source, VHDL-coded, FPGA-synthesizable, out-of-order superscalar processor [68] as a token of state-of-the-art circuit design in microarchitecture. As a case study, we use our entire methodology to compute the vulnerability of four critical components of the OPA processor: the instruction decoder, the instruction scheduler, the Arithmetic and Logic Unit

TABLE 2
Validation Experiments for ISCAS'85 Benchmarks

Circuit details		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 seconds	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

TABLE 3
Vulnerability Results for Several Combinational Blocks of a Superscalar Processor

Circuit details				MASkIt Vulnerability			Execution time
Name	Primary inputs	Primary outputs	Gate count	Addition	Multiplication	Bubblesort	
Decoder	168	317	12448	0.56	0.55	0.52	2.5 hours
Scheduler	287	151	12311	0.36	0.35	0.32	2.5 hours
ALU	188	96	2781	0.41	0.46	0.49	50 mintues
FPU	227	166	1072	0.20	0.25	0.24	8 mintues

(ALU) and the Floating Point Unit (FPU). To the best of our knowledge, this is the first work to report reliability figures on such kind of circuits.

7.1 Preprocessing

The preprocessing before SER estimation begins with the behavioral description in HDL code of each block circuit under study. To obtain the signal probabilities of the circuits primary inputs, logical simulations using ModelSim are performed. The logical simulation consists in executing several benchmarks and monitoring at each cycle the logical values of each primary input. The signal probability used for the soft error estimation is the 0-to-1 ratio of these values. In our experiments, we have used three microbenchmarks that are representative of the typical workloads these circuits would execute. The test programs are the following: accumulation of the values of a vector of 1,000 elements, multiplication of two matrices of 10x10 elements, and a bubblesort of a vector of 1,000 elements. We perform 100 executions for each program, initializing the input sets with random values on each run. The OPA processor block circuits are then synthesized using a subset of elementary gates from NanGate 45 nm's technology library [65] that have been previously characterized. The synthesis constraints are: voltage of 0.7 V, temperature of 75°C and frequency of 1 GHz. The generated netlist is parsed and mapped to the corresponding graph structure, in which each node has data fed from the cell characterization library and the primary inputs also have their signal probabilities set.

7.2 Results

Table 3 shows the results of running MASkIt for the four aforementioned block circuits after the preprocessing stage. The table reports the vulnerability for the three benchmark applications as well as the execution time. Each circuit is also characterized by the number of primary inputs, primary outputs and total gates. Several revealing observations can be extracted. To begin with, masking effects have a tremendous effect in the reduction of the SER. The weakest circuit in our experiments is the *Decoder* while running the addition benchmark with a vulnerability of 56 percent, which implies that 44 percent of particle strikes are masked simply because of the inherent properties of the circuit design. For other circuits the effect of masking can be highly significant. It is the case of the FPU implemented in OPA, which is the most resilient of the evaluated circuits, with a vulnerability of just 20 percent, meaning that 80 percent of particle strikes might be masked.

A second interesting observation is that the impact that different input distributions have in the reliability of a circuit is not very large but it is not negligible. The circuit that presents the widest variability is the ALU, which has a vulnerability of 41 percent with the input distribution obtained by running the Addition benchmark and a vulnerability of 49 percent while running the Bubblesort benchmark. The vulnerability of the other three components (Decoder, Scheduler and FPU) is more contained towards a mean value: the Decoder and the Scheduler have absolute variations of only 4 percent whereas the FPU has an absolute variation of 5 percent.

This study serves to illustrate MASkIt's capability to estimate the SER of several alternatives of the same combinational circuit in early stages of design. With the synthesis constrains previously described, the netlist obtained for the Decoder has a substantial amount of XOR and XNOR gates, which do not contribute to mask particle strikes through Logical Masking. As a consequence, the Decoder exhibits a relatively higher vulnerability in comparison with the other three circuits. If further synthesis constraints are added to prevent the use of these kind of gates, the vulnerability of the Decoder is reduced almost 10 percent: to 47 percent in Addition and Multiplication and to 43 percent in Bubblesort.

The proposed MASkIt approach makes possible to perform early vulnerability studies by iterating across multiple variations of a targeted circuit due to the achieved efficiency and scalability. While we have not executed fault injection for the use case circuits of the OPA processor, based upon the results obtained for ISCAS'85 circuits (Table 2), we estimate that using a traditional approach on circuits with more than 10,000 gates would require at least 6 weeks of execution time while MASkIt estimates the vulnerability for circuits of such size in only 2.5 hours (this the case of the Decoder, composed of 12,448 gates) which represents a time improvement of two orders of magnitude with respect to fault injection. Moreover, a comparison between running MASkIt in *c7752*, the largest circuit in the ISCAS'85 benchmark, and the Decoder, the largest circuit in this case study, reveals that, while the number of gates in the circuit increases by 3.5x, the execution time only increases by 1.6x. Therefore, it is shown that the execution time of MASkIt is not exponential even though it computes the effect of all input vectors and pulse widths simultaneously.

8 Conclusions

In this paper, we have introduced a novel methodology that allows to estimate the vulnerability of combinational logic in early stages of the design. The basis of such methodology is MASkIt, an algorithm that traverses any circuit from outputs to inputs and incrementally evaluates the overall SER by computing the effects of logical, electrical and timing masking for all gates. All possible input widths and input vectors are considered at once due to the use of a cell technology library and signal probabilities. These signal probabilities are approximated using a novel approach that combines an heuristic algorithm along with the simulation of selected reconvergent subnets within the circuit. Experiments on benchmark circuits show that the proposed algorithm offers two orders of magnitude speedup compared with a fault-injection based SER estimation methodology with less than 5 percent inaccuracy. We have also shown that our technique can be applied to swiftly and accurately estimate the vulnerability of state-of-the-art logic designs. As a case study, we have studied several combinational block circuits from an out-of-order superscalar. We show how MaskIt can handle circuits of 12 K gates in over 2 hours, making it the first method to achieve such turn around times. Consequently, MaskIt is as an effective tool to help circuit designers choose the least vulnerable circuit implementation in early design stages.

Acknowledgments

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] R. C. Baumann, "Soft errors in advanced semiconductor devices-part I: The three radiation sources," *IEEE Trans. Device Mater. Rel.*, vol. 1, no. 1, pp. 17–22, Mar. 2001.

[2] P. Shivakumar, et al., "Modeling the effect of technology trends on the soft error rate of combinational logic," in *Proc. Int. Conf. Depend. Syst. Netw.*, 2002, pp. 389–398.

[3] N. Seifert, et al., "Soft error susceptibilities of 22 nm Tri-Gate devices," *IEEE Trans. Nucl. Sci.*, vol. 59, no. 6, pp. 2666–2673, Dec. 2012.

[4] T. Uemura, et al., "Investigation of logic circuit soft error rate (SER) in 14nm FinFET technology," in *Proc. IEEE Int. Rel. Physics Symp.*, 2016, pp. 3B-4–1–3B-4-4.

[5] J. Meza, et al., "Revisiting memory errors in large-scale production data centers: Analysis and modeling of new trends from the field," in *Proc. 45th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, 2015, pp. 415–426.

[6] S.-L. Gong, et al., "DRAM scaling error evaluation model using various retention time," in *Proc. 45th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, 2017, pp. 177–183.

[7] P. N. Sanda, et al., "Soft-error resilience of the IBM Power6 processor," *IBM J. Res. Develop.*, vol. 52, no. 3, pp. 275–284, 2008.

[8] J. Han, et al., "A fault-tolerant technique using quadded logic and quadded transistors," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 23, no. 8, pp. 1562–1566, Aug. 2015.

[9] A. T. Sheikh, et al., "A fault tolerance technique for combinational circuits based on selective-transistor redundancy," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 25, no. 1, pp. 224–237, Jan. 2017.

[10] A. Yan, et al., "Double-node-upset-resilient latch design for nanoscale CMOS technology," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 25, no. 6, pp. 1978–1982, Jun. 2017.

[11] J. Arlat, et al., "Fault injection for dependability validation: A methodology and some applications," *IEEE Trans. Softw. Eng.*, vol. 16, no. 2, pp. 166–182, Feb. 1990.

[12] S. Mukherjee, *Architecture Design for Soft Errors*. San Mateo, CA, USA: Morgan Kaufmann, 2011.

[13] B. Zhang, et al., "FASER: Fast analysis of soft error susceptibility for cell-based designs," in *Proc. 7th Int. Symp. Quality Electron. Des.*, 2006, pp. 755–760.

[14] N. Miskov-Zivanov and D. Marculescu, "Formal modeling and reasoning for reliability analysis," in *Proc. 47th Des. Autom. Conf.*, 2010, pp. 531–536.

[15] S. Krishnaswamy, et al., "On the role of timing masking in reliable logic circuit design," in *Proc. 45th ACM/IEEE Des. Autom. Conf.*, 2008, pp. 924–929.

[16] P. C. Murley and G. R. Srinivasan, "Soft-error Monte Carlo modeling program, SEMM," *IBM J. R+D*, vol. 40, no. 1, pp. 109–118, 1996.

[17] N. J. Wang, et al., "Characterizing the effects of transient faults on a high-performance processor pipeline," in *Proc. Int. Conf. Depend. Syst. Netw.*, 2004, pp. 61–70.

[18] M. Zhang and N. R. Shanbhag, "Soft-error-rate-analysis (SERA) methodology," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 25, no. 10, pp. 2140–2155, Oct. 2006.

[19] D. Holcomb, et al., "Design as you see fit: System-level soft error analysis of sequential circuits," in *Proc. Conf. Des. Autom. Test Europe*, 2009, pp. 785–790.

[20] J.-C. Baraza, et al., "Enhancement of fault injection techniques based on the modification of VHDL code," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 16, no. 6, pp. 693–706, Jun. 2008.

[21] D. Alexandrescu, E. Costenaro, and M. Nicolaidis, "A practical approach to single event transients analysis for highly complex designs," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotech. Syst.*, 2011, pp. 155–163.

[22] Y. Kuo, et al., "Accurate statistical soft error rate (SSER) analysis using a quasi-Monte Carlo framework with quality cell models," in *Proc. Int. Symp. Quality Electron. Des.*, 2010, pp. 831–838.

[23] L. Entrena, et al., "Soft error sensitivity evaluation of microprocessors by multilevel emulation-based fault injection," *IEEE Trans. Comput.*, vol. 61, no. 3, pp. 313–322, Mar. 2012.

[24] R. Rajaraman, et al., "SEAT-LA: A soft error analysis tool for combinational logic," in *Proc. 19th Int. Conf. VLSI Des.*, 2006, Art. no. 4.

[25] S. Krishnaswamy, et al., "Signature-based SER analysis and design of logic circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 28, no. 1, pp. 74–86, Jan. 2009.

[26] S. Krishnaswamy, et al., "Accurate reliability evaluation and enhancement via probabilistic transfer matrices," in *Proc. Conf. Des. Autom. Test Europe*, 2005, pp. 282–287.

[27] N. Miskov-Zivanov and D. Marculescu, "MARS-C: Modeling and reduction of soft errors in combinational circuits," in *Proc. Des. Autom. Conf.*, 2006, pp. 767–772.

[28] R. B. Schivittz, et al., "A probabilistic model for stuck-on faults in combinational logic gates," in *Proc. 17th Latin-Amer. Test Symp.*, 2016, pp. 39–44.

[29] A. Abdollahi, "Probabilistic decision diagrams for exact probabilistic analysis," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2007, pp. 266–272.

[30] H. Asadi, et al., "Efficient algorithms to accurately compute derating factors of digital circuits," *Microelectron. Rel.*, vol. 52, no. 6, pp. 1215–1226, 2012.

[31] N. Mohyuddin, et al., "Probabilistic error propagation in logic circuits using the boolean difference calculus," in *Proc. IEEE Int. Conf. Comput. Des.*, 2008, pp. 7–13.

[32] J. Han, et al., "Reliability modeling of nanoelectronic circuits," in *Proc. Conf. Nanotechnol.*, 2005, pp. 269–272.

[33] A. Stempkovskiy, et al., "Practical metrics for evaluation of fault-tolerant logic design," in *Proc. IEEE Conf. Russian Young Res. Elect. Electron. Eng.*, 2017, pp. 569–573.

[34] T. Rejimon and S. Bhanja, "Scalable probabilistic computing models using Bayesian networks," in *Proc. 48th Midwest Symp. Circuits Syst.*, 2005, pp. 712–715.

[35] L. Chen, et al., "CEP: Correlated error propagation for hierarchical soft error analysis," *J. Electron. Testing*, vol. 29, no. 2, pp. 143–158, 2013.

[36] S. Yoshida, et al., "An soft error propagation analysis considering logical masking effect on re-convergent path," in *Proc. IEEE 22nd Int. On-Line Testing Symp.*, 2016, pp. 13–16.

[37] J. Cai and C. Chen, "Circuit reliability analysis using signal reliability correlations," in *Proc. IEEE Int. Conf. Softw. Quality Rel. Secur. Companion*, 2017, pp. 171–176.

[38] S. Z. Shazli and M. B. Tahoori, "Using boolean satisfiability for computing soft error rates in early design stages," *Microelectron. Rel.*, vol. 50, no. 1, pp. 149–159, 2010.

[39] T. Takata, et al., "A robust algorithm for pessimistic analysis of logic masking effects in combinational circuits," in *Proc. IEEE 17th Int. On-Line Testing Symp.*, 2011, pp. 246–251.

[40] H. Asadi and M. B. Tahoori, "Soft error derating computation in sequential circuits," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2006, pp. 497–501.

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

[42] R. Garg, C. Nagpal, and S. P. Khatri, "A fast, analytical estimator for the SEU-induced pulse width in combinational designs," in *Proc. Des. Autom. Conf.*, 2008, pp. 918–923.

[43] G. B. Hamad, et al., "Efficient multilevel formal analysis and estimation of design vulnerability to single event transients," in *Proc. IEEE 21st Int. On-Line Testing Symp.*, 2015, pp. 1–6.

[44] M. Shafique, et al., "Reliable software for unreliable hardware: Embedded code generation aiming at reliability," in *Proc. 7th IEEE/ACM/IFIP Int. Conf. Hardware/Softw. Codes. Syst. Synthesis*, 2011, pp. 237–246.

[45] M. Shafique, et al., "Exploiting program-level masking and error propagation for constrained reliability optimization," in *Proc. 50th Annu. Des. Autom. Conf.*, 2013, pp. 1–9.

[46] F. Brosch, et al., "Architecture-based reliability prediction with the palladio component model," *IEEE Trans. Softw. Eng.*, vol. 38, no. 6, pp. 1319–1339, Nov./Dec. 2012.

[47] M. Carbin, et al., "Verifying quantitative reliability for programs that execute on unreliable hardware," *ACM SIGPLAN Notices*, vol. 48, no. 10, pp. 33–52, 2013.

[48] M. Riera, et al., "A detailed methodology to compute soft error rates in advanced technologies," in *Proc. Conf. Des. Autom. Test Europe*, 2016, pp. 217–222.

[49] M. Ebrahimi, et al., "Layout-based modeling and mitigation of multiple event transients," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 35, no. 3, pp. 367–379, Mar. 2016.

[50] M. Ebrahimi, et al., "Comprehensive analysis of alpha and neutron particle-induced soft errors in an embedded processor at nanoscales," in *Proc. Conf. Des. Autom. Test Europe*, 2014, pp. 1–6.

[51] S. Kiamehr, et al., "Chip-level modeling and analysis of electrical masking of soft errors," in *Proc. 31st VLSI Test Symp.*, 2013, pp. 1–6.

[52] M. Amin Sabet, B. Ghavami, and M. Raji, "A scalable solution tosoft error tolerant circuit design using partitioning-based gate sizing," *IEEE Trans. Rel.*, vol. 66, no. 1, pp. 245–256, Mar. 2017.

[53] I. M. Ratiu, et al., "VICTOR: A fast VLSI testability analysis program," in *Proc. IEEE Int. Test Conf.*, 1982, pp. 397–401.

[54] R. E. Bryant, "Symbolic Boolean manipulation with ordered binary-decision diagrams," *ACM Comput. Surveys*, vol. 24, no. 3, pp. 293–318, 1992.

[55] N. Miskov-Zivanov and D. Marculescu, "Circuit reliability analysis using symbolic techniques," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 25, no. 12, pp. 2638–2649, Dec. 2006.

[56] R. G. Michael, et al., *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA, USA: Freeman, 1979.

[57] F. Kriebel, et al., "ACSEM: Accuracy-configurable fast soft error masking analysis in combinatorial circuits," in *Proc. Conf. Des. Autom. Test Europe*, 2015, pp. 824–829.

[58] S. Ercolani, et al., "Estimate of signal probability in combinational logic networks," in *Proc. Eur. Test Conf.*, 1989, pp. 132–138.

[59] B. Krishnamurthy and I. G. Tollis, "Improved techniques for estimating signal probabilities," *IEEE Trans. Comput.*, vol. 38, no. 7, pp. 1041–1045, Jul. 1989.

[60] M. A. Al-Kharji and S. A. Al-Arian, "A new heuristic algorithm for estimating signal and detection probabilities," in *Proc. 7th Great Lakes Symp. VLSI*, 1997, pp. 26–31.

[61] A. C.-C. Chang, et al., "CASSER: A closed-form analysis framework for statistical soft error rate," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 21, no. 10, pp. 1837–1848, Oct. 2013.

[62] D. T. Franco, et al., "Signal probability for reliability evaluation of logic circuits," *Microelectron. Rel.*, vol. 48, no. 8, pp. 1586–1591, 2008.

[63] J. Li and J. Draper, "Accelerated soft-error-rate (SER) estimation for combinational and sequential circuits," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 22, no. 3, 2017, Art. no. 57.

[64] M. W. Roberts and P. K. Lala, "Algorithm to detect reconvergent fanouts in logic circuits," *IEEE Proc. Comput. Digit. Techn.*, vol. 134, no. 2, pp. 105–111, Mar. 1987.

[65] Nangate 15nm open cell library, (2014). [Online]. Available: http://www.nangate.com/?page_id=2328, Accessed on: 20-Dec-2018.

[66] R. R. Robey and R. S. Barcikowski, "Type I error and the number of iterations in Monte Carlo studies of robustness," *Brit. J. Math. Statistical Psychology*, vol. 45, no. 2, pp. 283–288, 1992.

[67] W. L. Winston, *Simulation Modeling Using@ RISK*. Duxbury, MA, USA: Brooks/Cole, 2000.

[68] W. W. Terpstra, "OPA: Out-of-order superscalar soft CPU," in *Proc. Ontology Router Configuration*, 2015.

Martí Anglada received the BS degree in computer engineering, in 2013, and the MSc degree in high performance computing, in 2015, from the Universitat Politècnica de Catalunya (UPC-BarcelonaTech). He is currently working toward the PhD degree. He joined the UPC-BarcelonaTech ARCO Research Group in July 2014. His research is focused on low-power, resilient architectures.

Ramon Canal is an associate professor with the Computer Architecture Department, Universitat Politècnica de Catalunya. His research focuses on resilient, energy efficient architectures. He has an extensive list of publications and several invited talks. He has been a program committee member for several editions of HPCA, MICRO, ISCA, HiPC, IPDPS, and ICCD. He has been co-general chair of IOLTS 2012 and HPCA 2016. He is a senior member of the IEEE.

Juan L. Aragón received the PhD degree in computer engineering from UMU, in 2003, followed by a 1-year postdoctoral stay as a visiting assistant professor and researcher with UC Irvine. He is an associate professor with the Computer Architecture Department, University of Murcia, Spain. He has also been a visiting researcher with EPFL (Switzerland), in 2013 and with Princeton University (USA), in 2015 and 2017, respectively. He has co-authored 50 research papers in major conferences and journals.

Antonio González received the PhD degree, in 1989. He is a full professor with the Computer Architecture Department, Universitat Politècnica de Catalunya, Barcelona, Spain, and the director of the Microarchitecture and Compilers Research Group. He was the founding director of the Intel Barcelona Research Center from 2002 to 2014. His research interests focus on computer architecture and code optimization. He is a fellow of the IEEE.