electronics

MDPI

Article

Soft-Error Vulnerability Estimation Approach Based on the SET Susceptibility of Each Gate

Fábio Batagin Armelin [1,2,3,*], Lírida Alves de Barros Naviner [1] and Roberto d'Amore [2]

[1] LTCI, Télécom Paris, Institut Polytechnique de Paris, 19 Place Marguerite Perey, F-91120 Palaiseau, France
[2] Instituto Tecnológico de Aeronáutica, Pça. Mal. Eduardo Gomes 50, São José dos Campos 12228-900, Brazil
[3] Instituto Nacional de Pesquisas Espaciais, Av. dos Astronautas 1758, São José dos Campos 12227-010, Brazil
* Correspondence: fabio.armelin@inpe.br

Received: 1 May 2019; Accepted: 5 June 2019; Published: 2 July 2019

Abstract: Soft-Error Vulnerability (SEV) is a parameter used to evaluate the robustness of a circuit to the induced Soft Errors (SEs). There are many techniques for SEV estimation, including analytical, electrical and logic simulations, and emulation-based approaches. Each of them has advantages and disadvantages regarding estimation time, resources consumption, accuracy, and restrictions over the analysed circuit. Concerning the ionising radiation effects, some analytical and electrical simulation approaches take into account how the circuit topology and the applied input patterns affect their susceptibilities to Single Event Transient (SET) at the gate level. On the other hand, logic simulation and emulation techniques usually ignore these SET susceptibilities. In this context, we propose a logic simulation-based probability-aware approach for SEV estimation that takes into account the specific SET susceptibility of each circuit gate. For a given operational scenario, we extract the input patterns applied to each gate and calculate its specific SET susceptibility. For the 38 analysed benchmark circuits, we obtained a reduction from 15.27% to 0.68% in the average SEV estimation error, when comparing the estimated value to a reference obtained at the transistor level. The results point out an improvement of the SEV estimation process by considering the specific SET susceptibilities.

Keywords: Soft-Error; Single-Event Effect; Single-Event Transient; Soft-Error Rate; Soft-Error Vulnerability

1. Introduction

In digital systems, a Soft Error (SE) is a wrong value in a signal or data register that is not permanent, i.e., the system can be restored to its operational state after the treatment of the SE. In this work, we deal with the SEs caused by the Single-Event Effects (SEEs), that are the class of ionising radiation effect generated by a single ionisation particle.

The first SEE of interest is Single-Event Transient (SET), that is a temporary change in the logic value of a node, caused by an excursion on its voltage. This voltage excursion results from the transient current pulse produced by the recombination of the ions induced by the ionisation particle.

The SET is generated in the sensitive area (cross-section) of the transistors that compose the logic elements and can affect them in different ways. For the combinational elements (logic gates), the SET can propagate to the output or can be filtered out internally. On the other hand, for the sequential elements (registers—latches and flip-flops), the SET can also propagate to the output or be filtered, but these are minor effects. For them, the primary effect is the Single-Event Upset (SEU), that is the change of the stored value, that holds until the next write cycle.

The reliability assessment of digital systems usually adopts the Soft-Error Rate (SER) as the metric associated with the SEs. This parameter is intrinsically related to the rate of transient electrical pulses, R, generated in the transistors of the circuit, that can be calculated with the equation below [1],

$$R = \int_{E_{min}}^{E_{max}} \phi(E) \cdot \sigma(E) \cdot dE, \quad (1)$$

where ϕ is the particle flux, σ is the transistor cross-section, and E is the particle energy.

However, the circuit's SER is not the sum, or combination, of the R value of each transistor. The logical, temporal and electrical masking effects filter many transient pulses. Thus, applying the Equation (1) to estimate the device's SER would require an 'effective σ', including these masking effects. The difficulty in determining this effective σ led to the development of many SER estimation methods.

The SER estimation methods can be analytical, as in [2–4], simulated fault-injection, as in [5–8], emulated fault-injection, as in [9–11], or radiation tests, as in [12–14]. However, except for the radiation tests, they do not deal directly with the SER, as it requires the ionising particle flux. Instead, they deal with the probability of occurrence of a SE, adopting terms like SE sensitivity, susceptibility or vulnerability. In this work, we adopt two terms. For the probability of occurrence of SEs at the output of the gate-level components due to the SETs generated in their internal transistors, we use the term SET Susceptibility. On the other hand, for the probability of occurrence of SEs at the output of the circuit, we adopt the term Soft-Error Vulnerability (SEV).

We differentiate these two probabilities because we focus on the influence of the SET susceptibilities when evaluating the SEV at the gate level. This influence is taken into account by some classes of estimation methods (analytical and electrical simulation), but ignored by others (logic simulation and emulation), as follows.

In [7], the authors present a gate modelling and simulation method at the transistor level using SPICE. Their method intrinsically includes the internal logical masking, with the influence of both the gate topology and the input values (hereafter referred to as input patterns). It also includes the electrical masking, through the electrical modelling of the transistors and the SETs. Their results show that both logical and electrical masking effects affect the SET susceptibility of the analysed NAND2 gate.

The analytical method proposed in [3] also includes the internal masking effects of the gates. A major concern of the authors is how the input patterns influence the SET generation at the output of the gates. Their results also show that the SET susceptibilities are affected by these factors and, additionally, how they influence the circuit SEV.

The influence of the input values on the reliability of digital circuits is taken into account in other studies, concerning both transient and permanent faults. In [15], the authors present an algorithm based on signal probabilities and Probabilistic Transfer Matrices (PTMs). The authors of [16] describe a reliability evaluation method with Probabilistic Gate Models (PGMs), that also deals with signal probabilities. Finally, in [17], the authors introduce a stochastic computational approach that uses Stochastic Computational Models (SCMs) and signal probabilities embedded in pseudo-random binary streams.

On the other hand, the logic simulation-based methods presented in [5,6,8] and the emulation-based methods presented in [9–11] do not take the SET susceptibilities into account. Unlike the previously mentioned methods, these do not deal with transistors. They work directly at RTL level (for logic simulation) or gate level (for emulation), on the output of its components. As they inject the SEs uniformly, they intrinsically assume that all components have the same SET susceptibility.

In this context, we propose a logic simulation-based approach that takes into account the SET susceptibilities of the logic elements to estimate the SEV of the circuit. This approach is based on a simulation framework in which it is possible to inject SETs at the output of the logic elements. The SET injection process follows a weighted distribution, for which the weights are the SET susceptibilities of the logic elements.

Since this probability-aware approach requires accessing the output of the logic elements, the simulation requires the use of the post-synthesis back-annotated file, that is a netlist with the logic elements for the target technology. This approach can be applied to both ASICs and FPGAs. However,

in this work, we targeted only a specific FPGA family (presented in Section 3). For this reason, hereafter we refer to the logic elements directly as the CLBs (Configuration Logic Blocks of an FPGA).

The adoption of the proposed SEV estimation method requires the SET susceptibilities for the target technology. As this information is not available, we estimated the SET susceptibilities of the CLBs of the selected technology. This estimation needed some assumptions, as described in Section 3. They are reasonable for this study since they are consistently applied for the SET susceptibilities estimation and the SEV estimation at the transistor level, hereunder.

To evaluate the proposed approach, we estimated the SEV for a set of benchmark circuits, in four different ways. First, we estimated the SEV at the transistor level, to have a reference for comparison. This estimation process uses the same model adopted for the SET susceptibilities estimation. Second, we estimated the SEV at the CLB level (gate level) without taking into account the SET susceptibilities. Third, we estimated the SEV at the CLB level taking into account only the topology effect on the SET susceptibilities, ignoring the input patterns effect. Finally, we estimated the SEV with our proposed approach, that takes into account the topology and the input patterns effect on the SET susceptibilities of the CLBs.

The results show an improvement in the accuracy of the SEV estimation process when using the SET susceptibilities, with both topology and input patterns effect. For the 38 analysed benchmarks, the average error of the SEV obtained with the proposed approach regarding the reference obtained at the transistor level was 0.68%. On the other hand, the average error of the SEV obtained without taking into account the SET susceptibilities was 15.27% while ignoring only the input patterns effect leads to an error of 4.70%. Concerning the simulation time, the three SEV estimation methods at the CLB level have similar values. Which means there is no significant penalty in considering a weighted distribution instead of a uniform one. In contrast, at the transistor level, the simulation time is some orders of magnitude higher than others, depending on the circuit size.

It is important to note that our analysis focuses on the logical masking on both the transistor level, for the SET susceptibilities, and the CLB level, for the SEV. For this reason, the analysed benchmarks are only combinational circuits. However, the electrical and temporal masking effects could be included in another estimation of the SET susceptibilities that incorporates all the details of the technology, which were not available for this work.

A preliminary version of this study was presented in [18], in which we analysed the impact of the CLB configuration (topology) on the circuit's SEV. That preliminary analysis was performed for two small circuits and limited to the CLB configurations used by these circuits. The additional contributions of the current article are the inclusion of the input patterns effect; the analyses of all combinational configurations of the selected CLB; and the proposition of a structured methodology that can be applied to any circuit.

This paper is organised as follows. Section 2 describes our proposed simulation-based method for SEV estimation. Section 3 presents how we estimated the reference SET susceptibilities for the target technology. Section 4 describes the validation process for our proposed method. Section 5 presents the results. Finally, Section 6 discusses the proposed approach and the obtained results.

2. Simulation-Based Approach for SEV Estimation

The proposed logic simulation-based approach for SEV estimation relies on the comparison of the behaviours of the Design Under Test (DUT) with and without the injection of SETs in its CLBs. The distribution of the SET injection follows the SET susceptibility of each CLB. The greater the susceptibility, the greater the number of injected SET on a given CLB.

As the SET susceptibilities depend on the input patterns of the CLBs, the SEV is specific to an operational scenario, since different scenarios would lead to different input patterns distribution and, thus, different SET susceptibilities.

Additionally, the number of SETs injected for the SEV estimation must be parameterised in terms of SET per transistor, that are the sensitive elements. As a consequence, the number of SETs

injected in the CLBs of a circuit may differ between two operational scenarios, since they have different SET susceptibilities.

The proposed simulation framework that incorporates these features is described in Section 2.1. This framework is one of the tools used in the overall SEV estimation process detailed in Section 2.2. Besides the simulation itself, the SEV estimation process requires other steps to set the framework properly before the simulation. These steps are described in Sections 2.3–2.6.

2.1. Simulation Framework

The simulation framework is a testbench file, that can be used with any logic simulation tool. The testbench has the five elements illustrated in Figure 1.

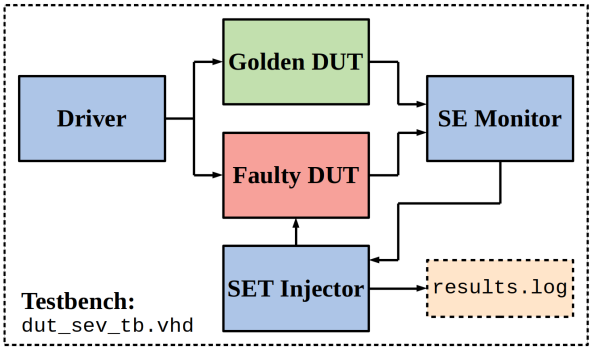

Figure 1. Proposed simulation framework for Soft-Error Vulnerability (SEV) estimation. It is a VHDL Testbench file that enables Single-Event Transient (SET) injection and Soft Error (SE) detection comparing the results of two versions of the Design Under Test (DUT), while running the same operational scenario.

The **Golden DUT** is the version of the DUT that is not subjected to SET injection, and the **Faulty DUT** is the version of the DUT in which the SETs are injected. The SET injection, at the output of the CLBs inside the **Faulty DUT**, is performed using a *saboteur* for each CLB. This element inverts the logic value of the CLB output while it is enabled and a SET is injected (see Section 2.3).

The **Driver** is responsible for the operational scenario. It applies the same sequence of test vectors to the inputs of both **Golden DUT** and **Faulty DUT**.

The **SE Monitor** compares the outputs of both DUTs, detecting the occurrence of a SE. When a SE occurs, the **SE Monitor** warns the **SET Injector**.

The **SET Injector** injects the SETs into the **Faulty DUT**, through the saboteurs. It controls when, where and how many SETs must be injected. Additionally, it generates the report for the SEV estimation process, with the total number of injected SETs and observed SEs. Section 2.6 describes how to generate this testbench, focusing on the **SET Injector**.

2.2. SEV Estimation Process

The overall SEV estimation process has seven steps, as illustrated in Figure 2. In this process, we adopted the VHDL language, but it could be used with another hardware description language (HDL), e.g., Verilog. This possibility is discussed by the end of this section.

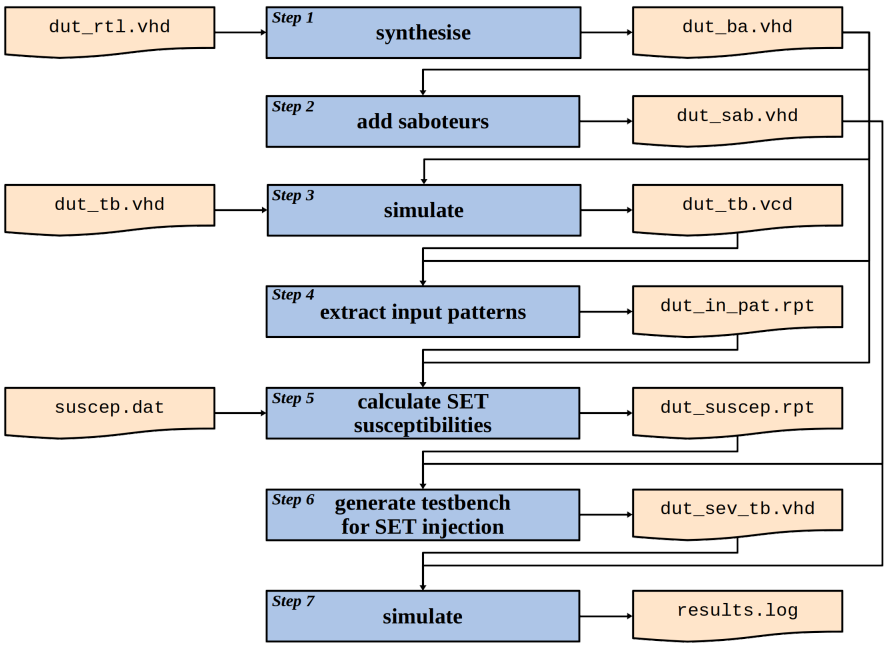

Figure 2. Proposed SEV Estimation Process. The overall process has seven steps that require three inputs to produce the SEV estimation and other six intermediate sub-products.

Step 1 is to synthesise the RTL description (`dut_rtl.vhd`) of the DUT to the target technology, generating the back-annotated post-synthesis netlist (`dut_ba.vhd`). It is essential to set the synthesis tool to do not insert I/O blocks so that the netlist will include only the CLBs. In this step, we used Synplify Pro L-2016.09M-2.

Step 2 is to insert the saboteurs into the DUT, generating the netlist for the **Faulty DUT** (`dut_sab.vhd`), as detailed in Section 2.3. The **Golden DUT** is the original post-synthesis netlist (`dut_ba.vhd`).

Step 3 is to simulate the DUT with the desired operational scenario to generate the Value Change Dump (VCD) file (`dut_tb.vcd`), from which the input patterns are extracted. The testbench (`dut_tb.vhd`) for this simulation must use the same **Driver** used in the simulation framework, discussed in Section 2.1, to assure the use of the same operational scenario. In this step, we used ModelSim SE-64 10.6a.

Step 4 is to extract the input patterns from the VCD file, generating a report (`dut_in_pat.rpt`) with the percentage of time that each CLB was submitted to each input pattern. The details of this process are presented in Section 2.4.

With the input patterns report (`dut_in_pat.rpt`) and the SET susceptibilities information (`suscep.dat`), the Step 5 is to calculate the specific SET susceptibilities of each CLB for the operational scenario. The result is a report file (`dut_suscep.rpt`) with the SET susceptibilities of each CLB used in the DUT. The calculus done in Step 5 is detailed in Section 2.5. The `suscep.dat` file is discussed in Section 3.

Step 6 is to generate the testbench used in the simulation framework (`dut_sev_tb.rpt`), incorporating the SET susceptibility of each CLB, as described in Section 2.6.

Finally, Step 7 is the actual simulation of the DUT injecting the weighted distribution of SETs, generating a report (`results.log`) with the estimated SEV for the circuit. In this step, we also used ModelSim SE-64 10.6a.

As illustrated in Figure 2, and described above, this SEV estimation process requires three inputs: (i) A synthesisable description of the DUT; (ii) a testbench that executes the intended operational scenario; and (iii) a reference database with the SET susceptibilities of each CLB as a function of the input patterns.

All seven steps can be automated by scripts. In fact, the validation of this approach, described in Section 4, used a set of scripts. The synthesis and simulations steps were automated using TCL (Tool Command Language) scripts. The steps for inserting saboteurs, extracting the input patterns, calculating the SET susceptibilities, and generating the testbench are based on file manipulation and basic mathematical operations. For them, we implemented scripts using Python 3.6, without any specific package.

Regarding the adoption of another HDL, e.g., Verilog, it is essential to note that the first Step is the synthesis, after that we only use the back-annotated post-synthesis netlist of the DUT. This way, the main concern about the language would be the testbench file. A significant issue would be the implementation of the weighted distribution of pseudo-random numbers, for which in VHDL we used a third party package, as described in Section 2.6. Additionally, the tools used for synthesis and simulation (Steps 1, 3, and 7) must support the selected language and the VCD file format. Finally, for the other Steps, the scripts would need to be adapted for the new language.

2.3. Adding Saboteurs

As described in Section 2.1, the saboteur is an element that inverts the logic value of an intercepted signal, when it is enabled, and a SET (fault) is injected, as represented in Equation (2). In the selected technology, described in Section 3.1, there is a CLB macro configuration that implements the logic function of Equation (2): AX1C. The saboteurs used in this work are instances of the AX1C macro.

$$output_{signal} = input_{signal} \oplus (sab\_enable \cdot fault) \tag{2}$$

The process of inserting saboteurs into the DUT consists of: (i) At the port of the entity, add the fault (SET) signal and the enable signals for each CLB; and, for each CLB, (ii) create a new signal, that replaces the original output of the CLB and is used as the input of the saboteurs; (iii) connects the output of the CLB to this new signal; and (iv) add an instance of the saboteur, mapping its input to the new signal, its output to the original output signal of the CLB, its enable signal to the respective enable port of the entity, and the fault signal to the fault port of the entity.

2.4. Extraction of the Input Patterns Distribution

The VCD file is a dump file generated by the logic simulation tools, usually used for waveform visualisation. It is an ASCII-based file that includes ascending time-ordered value changes of the signals [19]. Essentially, it has all the time marks in which there are value changes of at least one signal. Following a time mark, there is a list with all the signals that have a value change at that time, with their new values.

The information to be extracted from the VCD file is the percentage of time that each input pattern is applied to each CLB. The first step to extract it is to transform this textual representation into a table, with a list of time intervals and the values of each signal in the respective interval. Then, the time intervals in which each pattern is applied to the CLB is summed, generating the total time for each pattern. With the total time of the simulated scenario, the percentages can be calculated.

To exemplify how the VCD file store the information, and how the input patterns extraction occurs, consider the circuit and the graphic representation of a VCD file illustrated in Figure 3. For this VCD, the time marks are the instants 0, 1, 2, 3, 5, 6 and 8 µs. After the first time mark (0 µs), each signal receives its initial value (a = 0, b = 0, c = 0, d = 0, e = 0, f = 1, and g = 1). On the next time mark (1 µs), the signals with value changes receive their new values (a = 1, d = 1, f = 0, and g = 0). This process repeats for the others time marks, except for the last one (8 µs), that is not followed by any value change assignment.

The intermediate table, with time intervals and signals values for this example, are illustrated in Table 1. To improve its readability, it also includes columns for the inputs of each gate: {a b} for the AND, {d} for the Inverter, and {e c f} for the OR gate.

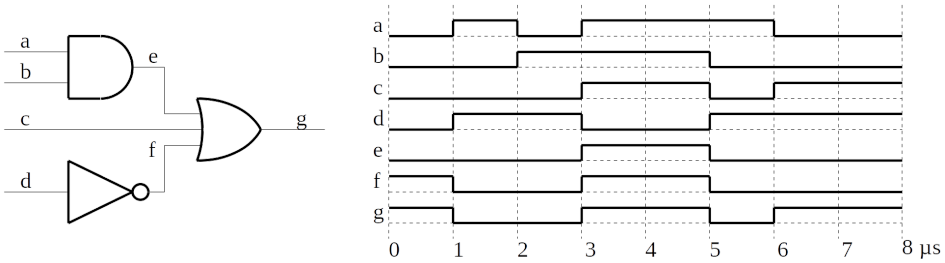

Figure 3. Input patterns extraction. Example circuit and its respective operational scenario used to explain the process to extract the input patterns distribution.

Table 1. Intermediate table of the input patterns extraction process. Tabular representation of the operational scenario illustrated in Figure 3.

Interval [µs]	a	b	c	d	e	f	g	{a b}	{d}	{e c f}
1	0	0	0	0	0	1	1	00	0	001
1	1	0	0	1	0	0	0	10	1	000
1	0	1	0	1	0	0	0	01	1	000
2	1	1	1	0	1	1	1	11	0	111
1	1	0	0	1	0	0	0	10	1	000
2	0	0	1	1	0	0	1	00	1	010

The AND gate has four input patterns 00, 01, 10, and 11. Assuming the time in which each pattern is applied is represented as $\{t_{00}, t_{01}, t_{10}, t_{11}\}$, processing the values presented in Table 1 results in {3, 1, 2, 2} µs. As the total time is 8 µs, the percentage of time in each input pattern $\{P_{00}, P_{01}, P_{10}, P_{11}\}$ is {37.5, 12.5, 25.0, 25.0}.

Similarly, the Inverter has two input patterns with time percentages of {37.5, 62.5}. Finally, the OR gate has eight input patterns with the time percentages of {37.5, 12.5, 25.0, 0.0, 0.0, 0.0, 0.0, 25.0}.

2.5. Calculation of the SET Susceptibilities

The actual SET susceptibility of each CLB of the DUT is calculated as a weighted sum of the SET susceptibilities of the respective CLB configuration for each input pattern. The weights are the percentage of time that each input pattern is applied to the CLB. So, the SET susceptibility of a specific instance of a CLB configuration can be obtained with the equation below,

$$\chi_{CLB_n} = \sum_{pattern=0_b}^{(2^i-1)_b} \chi_{CLB_{pattern}} \cdot P_{CLB_{n,pattern}}, \tag{3}$$

where χ_{CLB_n} is the SET susceptibility of the specific instance n of the CLB configuration; i is the number of inputs of the CLB configuration; *pattern* is the possible input patterns for the CLB configuration; $\chi_{CLB_{pattern}}$ is the SET susceptibility of the CLB configuration for the *pattern*; and $P_{CLB_{n,pattern}}$ is the percentage of time that the instance n of the CLB configuration is submitted to *pattern*.

To exemplify this calculus, consider two instances of the CLB configuration AND2 ($AND2_0$ and $AND2_1$). Additionally, consider the SET susceptibility of the four input patterns of the configuration AND2 to be {7%, 11%, 15%, 21%}. Finally, consider that the time distribution of the patterns are $P_{AND2_0} = \{10\%, 15\%, 25\%, 50\%\}$ and $P_{AND2_1} = \{40\%, 30\%, 15\%, 15\%\}$. The SET susceptibility of each instance can be calculated using Equation (4), expanded in Equation (5), resulting in $\chi_{AND2_0} = 16.6\%$ and $\chi_{AND2_1} = 11.5\%$ of the SET that produced in the transistors of the CLB.

$$\chi_{AND2_n} = \sum_{pattern=00_b}^{11_b} \chi_{AND2_{pattern}} \cdot P_{AND2_{n,pattern}} \tag{4}$$

$$\chi_{AND2_n} = \chi_{AND2_{00}} \cdot P_{AND2_{n,00}} + \chi_{AND2_{01}} \cdot P_{AND2_{n,01}} + \chi_{AND2_{10}} \cdot P_{AND2_{n,10}} + \chi_{AND2_{11}} \cdot P_{AND2_{n,11}} \tag{5}$$

2.6. Generating the Testbench for SEV Estimation

The testbench with the SEV estimation framework is composed of five elements, as detailed in Section 2.1. The generating process for this testbench consists in creating the specific **SET Injector** and interconnecting it with the other elements.

The **Driver** defines the operational scenario and must be the same used in the first simulation. It is an input for the overall SEV estimation process. The **Golden DUT** and the **Faulty DUT** are sub-products of the overall process. The first one is obtained with the synthesis, while the second is obtained with Step 2, by adding saboteurs. The **SE Monitor** is a parameterised comparator that can be used with any DUT, requiring only the adjustment of its input data width parameter to match the number of output signals from the DUT.

Regarding the **SET Injector**, it is specific for each operational scenario applied to the DUT. The SET distribution is performed using the Open Source VHDL Verification Methodology (OSVVM) [20]. This methodology includes the RandomPkg package, that provides a set of functions to generate pseudo-random numbers, including a weighted distribution function for integer values (DistValInt). The **SET Injector** relies on this function to select the CLB in which the SET will be injected.

Considering the two instances of the CLB configuration AND2 used as an example in Section 2.5 ($AND2_0$ and $AND2_1$), the selection of in which instance a SET would be injected can be obtained with the VHDL attribution exemplified in Listing 1.

Listing 1. DistValInt function of the RandomPkg package. Example of how to obtain a weighted distribution: The variable TargetCLB, used as an index to select the CLB in which the SET will be injected, can receive 0 or 1, for $AND2_0$ and $AND2_1$, with higher probability of receiving 0—for every 281 SETs, 166 are targeted to 0, and 155 to 1.

```
TargetCLB := RV.DistValInt(( (0, 166), (1, 115) ));
```

For the same example, the average susceptibility of the two instances of AND2 is 14.05%. Assuming that the SEV estimation process would inject 1000 SETs in the transistors of the CLB, on average, 14.05% of the 1000 SETs should be injected at the output of each CLB. With two instances, the total number of SETs that should be injected is 281, that would be distributed as 166 in $AND2_0$ and 115 in $AND2_1$. The Equation (6) presents the calculus to obtain the total number of SETs that needs to be injected, $N_{SET_{Total}}$,

$$N_{SET_{Total}} = N_{CLB} \cdot N_{Transistor_{CLB}} \cdot N_{SET_{Transistor}} \cdot \chi_{CLB_{average}} \quad (6)$$

where N_{CLB} is quantity of CLBs used in the DUT; $N_{Transistor_{CLB}}$ is the quantity of transistors per CLB, that in our model is 82; $N_{SET_{Transistor}}$ is the desired number of SETs at the transistor level; and $\chi_{CLB_{average}}$ is the average susceptibility of the CLBs used in the DUT.

In addition to where (in which CLB) and how many SETs are injected, the **SET Injector** also controls when they are injected. The SET injections are distributed evenly through the clock cycles of the operational scenario, and only one SET is injected per operational scenario execution.

3. Reference SET Susceptibilities

The proposed approach for the SEV estimation relies on the SET susceptibility of the elements that compose the circuit. In this section, we describe how we estimated the reference SET susceptibilities used in this work, and present the resulting values.

The analysis of the SET susceptibility of each CLB configuration takes into consideration one specific technology, presented in Section 3.1. The respective CLB were modelled in VHDL, as detailed in Section 3.2, to mimic the logic behaviour of the CLB and to enable the injection of the SETs at the transistor level. For this model, the configuration vectors, that determine the CLB combinational function, were defined as described in Section 3.3. These functions were grouped as components in

a VHDL library that can replace the original library of the chosen technology (see Section 3.4). Each CLB configured function contained in this library were submitted to the SET susceptibility estimation process described in Section 3.5, resulting in the susceptibilities presented in Section 3.6.

3.1. Analysed Technology

This SEV estimation approach could be applied to any FPGA or ASIC technology for which the respective SET susceptibilities, parameterised by the input patterns, are available. Unfortunately, this kind of information is currently unavailable. For this reason, we selected one technology for which we could estimate the SET susceptibilities and perform the comparative analyses presented in Section 4. We chose the ProASIC3E FPGA family [21] because its design data and internal structure enable the modelling of its CLB at the transistor level, with some reasonable assumptions, as described below.

The ProASIC3E FPGA family has 130 nm flash-based CMOS devices. The CLB of this family is called VersaTile. Unlike most other technologies, this CLB is not based on a sequential element and a Look-Up Table (LUT). Instead, it has a small set of logic gates that can be configured as combinational and sequential functions. Its internal structure is presented in [22] (p. 15), Figure 1-8.

The VersaTile has four inputs (X3Data, X2Clk, X1Enable, and XcClr) and one logic output, with two physical options: F2 for local nets and YL for long connections. This CLB is composed of 16 CMOS logic gates (eleven Inverters, four 2-Input Multiplexers, and one 2-Input NOR) interconnected through 32 flash-based switches. Its structure allows the configuration of any 3-input combinational function and a set of sequential functions, as latches and flip-flops, with enable and set/clear options [22]. The configurations available for this CLB are presented as macros in [23].

The design documents of this technology do not present the implementation details of the CLB at the transistor level. For this reason, we made some assumptions to make this study feasible. First, we assumed some CMOS topologies for the internal logic gates of the Versatile (described in Section 3.2). Second, we arbitrarily chose the values of the configuration vector for each function configured in a CLB (discussed in Section 3.3). Third, we assumed that all the transistors of the CLB have the same cross-section, without taking into account possible layout effects (described in Section 3.5). Finally, we did not model the electrical and timing characteristics, since the design documents do not provide the required information, and the logical masking is the most important for the analysis of the topology and input patterns effects. While these assumptions impact the quantitative values presented in this work, they do not undermine the general qualitative analyses.

3.2. CLB Simulation Model

The VersaTile CLB was modelled at the transistor level to reproduce its logical behaviour, with no concerns related to the electrical and timing characteristics. It was described in a structural VHDL as an interconnection of PMOS and NMOS transistors, implementing both logic gates and configuration switches.

For the logic gates, we assumed the topologies illustrated in Figure 4. This assumption results in 50 transistors for the logic gates of the CLB, being 25 PMOS and 25 NMOS (22 for the eleven Inverters, 24 for the four 2-Input Multiplexers, and 4 for the 2-Input NOR). Additionally, each configuration switch was modelled as an NMOS transistor. The resulting CLB model has 82 transistors.

The MOS transistors were modelled as unidirectional logic switches: the output (Drain) receives the input (Source) value when the switch control (Gate) is activated (low logic level for PMOS, and high logic level for NMOS). Additionally, a fault model is incorporated to enable the injection of the SETs: while the Fault signal is '1', the drain is forced to '0', for the NMOS, or '1', for the PMOS; otherwise, the transistor operates with no fault. This fault modelling is based on the electrical modelling discussed in [7], in which a transient current source is connected between the body and the drain of the MOS transistor. It implies that the drain of a NMOS can be forced only to '0', while the drain of a PMOS can be forced only to '1'.

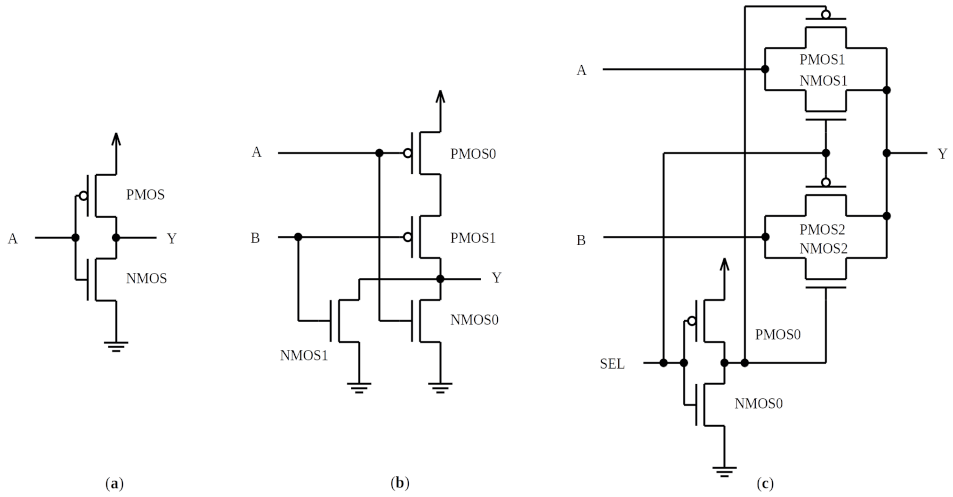

Figure 4. Adopted topologies for the internal logic gates of the Versatile CLB—(**a**) Inverter; (**b**) 2-Input NOR; and (**c**) 2-Input Multiplexer.

The model uses the IEEE std_logic_1164 multi-value logic system that defines different strengths for the signal values. The nominal operation of the CLB model is based on the weak values ('H' and 'L'), while the fault effect uses strong forcing values ('1' and '0'). In this way, the fault effect can be imposed on any signal.

For example, for an Inverter, as illustrated in Figure 4a, operating without fault (Fault = '0'), while the input value is 'L', the output value is 'H'. However, if a SET is injected in the NMOS transistor (Fault = '1'), the output will temporarily change to '0' during the desired pulse width of the SET, resulting in the sequence {'H';'0';'H'}. If a similar SET is injected in the PMOS transistor, the output will temporarily change to '1', resulting in the sequence {'H';'1';'H'}. In the first case, there is a temporary inversion of the logic value. In the last one, there is no inversion, but it is possible to identify that a SET was injected in that node.

This modelling avoids that a signal receives unknown values ('X' or 'W') when a SET is injected on its node. However, there is a specific situation in which an unknown value may appear. When a SET injected in the Inverter of the 2-Input Multiplexer (Figure 4c), causing the momentarily logic inversion of this Inverter output, the input and the output of the Inverter will have the same logic value. In this case, the transmission gates of both inputs of the Multiplexer will transmit their values to the output. If these inputs have different logic values, the output will be unknown ('W'). To avoid that this unknown value propagates outside the VersaTile model, a process handles this situation. Whenever an unknown value ('X' or 'W') would be propagated to the output, the output receives the logic inversion of its immediately previous value. For example, if the VersaTile output value is 'H', and during a SET injection it would temporarily change to 'W' (sequence {'H';'W';'H'}), it will in fact change to 'L' (sequence {'H';'L';'H'}).

3.3. Definition of the Configuration Vector of the CLBs

For the combinational functions, a VHDL script analyses all 2^{32} possible values for the configuration vector, validating two aspects. First, the switches that configure loops shall be open (SW1, SW3 and SW15). Second, every node has to have one, and only one, source. For example, considering the switches SW13, SW14 and SW16, if none of them is closed, the input 1 of the Mux2 will be left open. On the other hand, if more than one is closed, there will be a conflict between the values provided by the switches.

Then, the valid configurations are tested with all input patterns, and the results of the combinational function are logged in a csv file. There are many valid configuration vectors for each combinational function. For example, the AND2 macro has 1815 possible configurations, while the XOR3 has 32.

The selection of which possible configurations should be used followed these arbitrary criteria: (i) Have the least number of closed switches; (ii) use these inputs, in order, X3Data, X2Clk, X1Enable, and XcClr; and (iii) be the first in the list to meet the previous criteria.

Additionally, there are some combinational macro functions presented in [23] that has the same logic function. As an example, AND2 and NOR2B. The first one implements the function $Y = A \cdot B$, while the second implements $Y = \overline{\overline{A} + \overline{B}}$, that is the same. In these cases, we adopted the criteria presented above for the first macro (in alphabetical order). For the second one, we adopted the same criteria, except for the third item: be the *last* in the list to meet the previous criteria.

The VersaTile combinational functions are presented in the Supplementary Table S1. For each macro, it presents the configuration vector and the input mapping.

3.4. CLB Library

We created a VHDL library of components with all modelled CLB functions to replace the original library for the ProASIC3E family. Two versions of this library were created. In the first one (*proasic3e_nofault*), the macros have the same ports from the original library. They do not provide access to the transistor. Thus, with this library, it is not possible to inject the SETs. However, they are useful to verify the proper functionality of the macros.

In the second one (*proasic3e_faulty*), each macro has an additional Faults port, that gives access to the 82 transistors of the CLB model. As an example, Listing 2 presents the entity and the architecture for the macro AND2, with the Faults port.

The output of all macros in the library was mapped to the F2 output, with the YL output left opened.

Listing 2. Entity and architecture of the AND2 macro using the VersaTile model at the transistor level. The Faults port enable the access to inject the SETs in the transistors. Other ports are the same from the original macro.

```
entity AND2 is
    port(
        A           : in    std_ulogic;
        B           : in    std_ulogic;
        Y           : out   std_ulogic;
        Faults      : in    bit_vector( 81 downto 0 )
    );
end AND2;

architecture VersaTile_ST of AND2 is
begin
    CLB: VersaTile_82
    generic map(
        Config      => "11001000001110000010001100010101"
        )
    port map(
        X3Data      => A,
        X2Clk       => B,
        X1Enable    => 'Z',
        XcClr       => 'Z',
        F2          => Y,
        YL          => open,
        Faults      => Faults
        );
end VersaTile_ST;
```

3.5. Estimation of the SET Susceptibilities

We estimated the SET susceptibility of every macro contained in our version of the *proasic3e* library. The estimation process is based on VHDL simulations with SET injections at the transistor level.

In this process, we injected the same number of SETs in each transistor, adopting a uniform distribution of SETs. In other words, we assume that all transistors have the same cross-section. It is

a simplification, since, for example, F2 and YL signals requires different transistor characteristics. However, it is reasonable for this study, since it is consistently applied through all the work (for both CLB and circuit analysis).

Another simplification is that the transistors are treated separately as if they are part of a discrete circuit. As discussed in [7], for the integrated circuits, the cross-section (sensitive zone) depends on the circuit topology and layout. We do not consider these effects. However, for the same rationale presented above, it is reasonable for this study.

Regarding the SET pulse widths, there are some published radiation-test results for the selected technology. In [12], the authors reported that most of the observed SET pulse widths are wider than 1 ns, up to 4 ns. In this work, we performed all SET injections considering pulse widths of 1 ns.

The estimation of the SET susceptibility of every macro takes into account each input pattern independently. In this way, every combinational macro of n inputs is submitted to 2^n estimation processes, one for each input pattern.

3.6. Results

The estimated SET susceptibilities for the VersaTile combinational macros are presented in the Supplementary Table S2. For each macro, Table S2 presents the SET susceptibilities for each input pattern and their average SET susceptibility, considering the average of the values for each input pattern. The values are expressed as percentages. Taking into account the input patterns, the SET susceptibilities are in the range of ∼5–30%. For the average values, that considers only the topology influence, they are in the range of ∼10–27%. To illustrate this difference in SET susceptibilities, Figure 5 presents the obtained values for the 2-input macros.

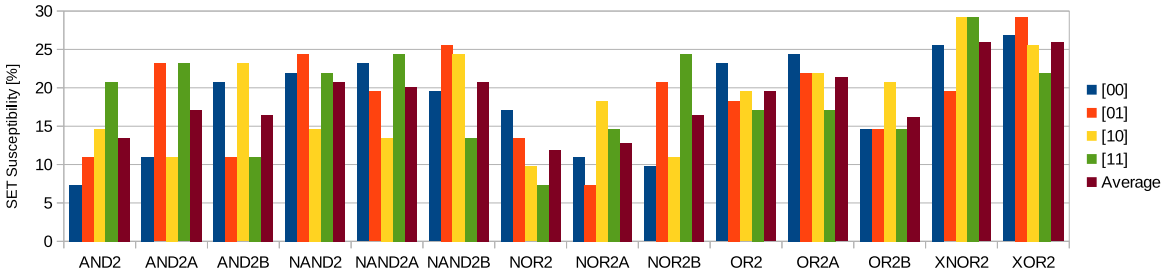

Figure 5. SET susceptibilities of the 2-input combinational functions. For each macro, it presents the susceptibility for each input pattern and their average value.

4. Validation of the Proposed Approach

We estimated the SEV for a set of benchmark circuits to validate the proposed approach. To have a reference for comparison, first, we estimated the SEV at the transistor level, using the model described in Section 3. Then we performed three estimations at the CLB level: (i) Ignoring the SET susceptibility of each CLB; (ii) considering only the topology effect on the SET susceptibilities; and (iii) considering the effects of both topology and input patterns on the SET susceptibilities, that is the proposed approach.

In this Section, we present the analysed benchmark circuits (Section 4.1), the evaluated operational scenarios (Section 4.2), and the four SEV estimation processes (Section 4.3).

4.1. Analysed Benchmarks

To evaluate the impact of considering the SET susceptibility of each CLB on the soft-error vulnerability of the circuits, we selected 38 combinational benchmark circuits from the LGSynth91 list [24] to be used as DUTs. We limited the analysis to the benchmarks that require less than 100 CLBs, due to the simulation time at the transistor level. Finally, the choice for the LGSynth91 list was arbitrary, but this selection is not important for this analysis since the proposed approach is applicable to any circuit.

The selected circuits have a wide range of complexity: the number of inputs varies from 3 to 47; the number of outputs varies from 1 to 36; and the number of CLBs, for the analysed technology, varies from 3 to 95. The details for each of the selected benchmark circuits are presented in Table 2.

Table 2. Combinational benchmark circuits—Characteristics and SEV estimation results. For each benchmark, the table presents (i) their number of inputs, outputs and CLBs required for the selected technology; and (ii) the number of observed SEs and the estimated SEV. Columns Trans. refers to the results at the transistor level; Unif., to uniform distribution, ignoring the SET susceptibilities; Topo., for considering only the topology effect; and Patt., for considering both Input patterns and topology effects.

Benchmark				Observed Soft Errors				Estimated Vulnerability [%]			
Name	**In**	**Out**	**CLB**	**Trans.**	**Unif.**	**Topo.**	**Patt.**	**Trans.**	**Unif.**	**Topo.**	**Patt.**
b1	3	4	3	56,274	42,267	56,501	56,274	22.88	17.18	22.97	22.88
cm82a	5	3	4	64,750	56,356	64,248	64,760	19.74	17.18	19.59	19.74
majority	5	1	5	29,154	27,743	25,442	29,484	7.11	6.77	6.21	7.19
tcon	17	16	8	102,504	112,713	102,998	102,532	15.63	17.18	15.70	15.63
cm152a	11	1	10	55,739	68,282	50,780	53,417	6.80	8.33	6.19	6.51
parity	16	1	10	199,540	140,892	199,374	199,538	24.33	17.18	24.31	24.33
cm42a	4	10	13	153,992	176,536	162,654	153,819	14.45	16.56	15.26	14.43
cm138a	6	8	13	121,281	133,799	133,016	121,104	11.38	12.55	12.48	11.36
t481	16	1	13	96,625	94,802	93,403	95,785	9.06	8.89	8.76	8.99
cmb	16	4	17	48,211	58,979	60,189	48,067	3.46	4.23	4.32	3.45
cm163a	16	5	19	113,452	128,205	110,202	110,635	7.28	8.23	7.07	7.10
9symml	9	1	21	85,420	94,073	83,462	85,332	4.96	5.46	4.85	4.96
cm85a	11	3	21	111,091	115,786	115,181	111,612	6.45	6.72	6.69	6.48
cm162a	14	5	21	136,472	150,604	131,212	135,535	7.93	8.75	7.62	7.87
cordic	23	2	22	95,024	119,779	93,648	95,421	5.27	6.64	5.19	5.29
decod	5	16	22	162,133	282,627	198,810	162,248	8.99	15.67	11.02	8.99
i1	25	16	22	155,015	196,462	152,493	156,465	8.59	10.89	8.45	8.67
z4ml	7	4	22	191,826	200,437	180,538	192,457	10.63	11.11	10.01	10.67
cu	14	11	24	175,629	240,885	189,681	175,590	8.92	12.24	9.64	8.92
pm1	16	13	24	258,680	267,277	273,931	256,468	13.14	13.58	13.92	13.03
pcle	19	9	27	242,821	274,071	257,181	240,347	10.97	12.38	11.62	10.86
cc	21	20	28	291,514	306,625	299,260	290,970	12.70	13.35	13.03	12.67
sct	19	15	28	243,121	275,822	258,943	243,129	10.59	12.01	11.28	10.59
x2	10	7	28	184,694	191,141	183,796	182,951	8.04	8.32	8.01	7.97
mux	21	1	29	58,684	64,726	53,283	55,647	2.47	2.72	2.24	2.34
f51m	8	8	32	319,996	282,514	322,889	319,432	12.19	10.77	12.31	12.17
frg1	28	3	35	147,293	169,121	151,430	148,369	5.13	5.89	5.28	5.17
lal	26	19	36	294,254	318,219	310,205	294,657	9.97	10.78	10.51	9.98
c8	28	18	44	330,306	379,865	339,958	331,218	9.15	10.53	9.42	9.18
unreg	36	16	48	358,056	393,843	366,789	358,096	9.10	10.01	9.32	9.10
pcler8	27	17	51	488,513	537,869	494,574	485,876	11.68	12.86	11.83	11.62
count	35	16	55	459,377	418,720	465,387	460,524	10.19	9.28	10.32	10.21
b9	41	21	59	510,557	525,984	511,516	513,199	10.55	10.87	10.57	10.61
comp	32	3	62	113,509	130,455	113,111	113,175	2.23	2.57	2.22	2.23
term1	34	10	68	260,615	314,802	263,336	260,610	4.67	5.65	4.72	4.67
cht	47	36	82	621,771	800,148	594,967	619,778	9.25	11.90	8.85	9.22
ttt2	24	21	85	601,227	683,075	586,786	593,077	8.63	9.80	8.42	8.51
my_adder	33	17	95	1,050,020	925,060	1,011,741	1,058,428	13.48	11.87	12.99	13.59

4.2. Operational Scenarios

The selected benchmark circuits do not have detailed documentation about their functionality. For this reason, in the **Driver**, we chose to use a pseudo-random sequence of test vectors to define the operational scenarios.

Instead of using the RandomPkg from the OSVVM, we decided to use Linear-Feedback Shift Registers (LFSR) to generate the pseudo-random sequence. Contrarily to the RandomPkg functions, the LFSR structures are synthesisable, which is interesting for an emulation-based SET injection approach, as proposed in [25].

We implemented an LFSR VHDL component with a configurable data width, n, ranging from 2 to 31 bits, according to the polynomials presented in [26]. In addition to the data width parameter, one can configure its initial value through the seed parameter.

For each of the analysed DUTs, the respective **Driver** is a set of these LFSRs, to fit the input data width of the DUT. The number of inputs of the DUT is divided by 31. The quotient gives the number of 31-bits LFSRs, while the remainder gives the width of the last LFSR. If the last LFSR would be one-bit

wide, the previous one is reduced to 30 bits, and the last is incremented to 2 bits. Finally, if more than one LFSR is needed, each of them is set to a different seed.

All operational scenarios span for 1000 cycles, i.e., they consist of a sequence of 1000 pseudo-random values. If the required LFSR data width is 9-bit or less, the LFSR sequence is restarted and repeated until achieving the 1000 cycles. In other cases, the sequence is truncated at 1000 cycles.

4.3. SEV Estimations

As already introduced, we performed four distinct SEV estimations, one at the transistor level, that provides the reference SEV, and three at the CLB level, to evaluate the effect of considering the SET susceptibilities of the CLBs.

All SEV estimations used the same structure of the simulation framework presented in Section 2.1, with some specificities for each category. For each DUT, the **Driver** and the **SE Monitor** were the same in the four categories.

For the **Golden DUT** and the **Faulty DUT**, we have two situations. The SEV estimation at the transistor level uses the library presented in Section 3.4 to model the DUT. The Faults port of the **Golden DUT** is set to all '0', while the Faults port of the **Faulty DUT** is used to inject the SETs. In this case, the netlist with the saboteurs is not used. On the other hand, the three SEV estimations at the CLB level use a variation of the original technology library without the VITAL timing model. As described in Section 2.1, the **Golden DUT** is the post-synthesis netlist, and the **Faulty DUT** is the netlist with the added saboteurs.

The **SET Injector** is the element that requires more adjustments from one category to other, especially regarding the number of injected SETs and the SET distribution.

At the transistor level, all SETs are distributed uniformly to each transistor, consistently to the SET susceptibility estimation (Section 3.5). At the CLB level, ignoring the SET susceptibilities, the SETs were also uniformly distributed to each CLB. On the other hand, the SEV estimations considering the SET susceptibilities used the weighted distribution described in Section 2.6. The SEV estimation process that takes into account only the topology effect uses the average SET susceptibility of each CLB. Finally, the SEV estimation process considering both topology and input patterns uses the SET susceptibilities obtained as described in Section 2.5.

For all SEV estimations, we injected the equivalent to 1000 SETs per transistor, that leads to 82,000 SETs inside each CLB, since the CLB model has 82 transistors. At the transistor level, the total number of injected SETs is 82,000 $\cdot N_{CLB}$. At the CLB level, we adopted the calculus presented in Section 2.6, but with different average Susceptibilities, $\chi_{CLB_{average}}$. For the process with uniform distribution, we adopted the average susceptibility of all configurations of the library. For the process with weighted distribution, we calculated the average susceptibility of the CLBs used in the DUT.

To exemplify the particularities of each SEV estimation, we describe the test data for the *majority* benchmark. This benchmark requires 5 CLBs (2 NOR3C, 1 OR3, and 2 OR2), identified as {$NOR3C_1$, $NOR3C_2$, $OR3_1$, $OR2_1$, $OR2_2$}. The average SET susceptibilities of each of these macros are {0.13567, 0.13567, 0.16768, 0.19512, 0.19512}, that leads to an average SET susceptibility of the circuit of 0.16585. The specific SET susceptibilities of each of these macros, for the adopted operational scenario, are {0.16206, 0.16203, 0.19674, 0.19391, 0.19395}, that leads to an average value of 0.18174. Finally, the average SET susceptibility of the entire library is 0.17182.

The quantity of SETs that needs to be injected ($N_{SET_{Total}}$) is obtained with Equation (6). The quantity of CLBs (N_{CLB}) is 5, the quantity of transistors per CLB ($N_{Transistor_{CLB}}$) is 82, and the quantity of SETs per transistor ($N_{SET_{Transistor}}$) is 1000. Thus, the difference between the SEV estimation processes is the average susceptibility of the CLBs ($\chi_{CLB_{average}}$). At the transistor level, we do not use the average susceptibility, so $N_{SET_{Total}}$ is 410,000. At the CLB level, ignoring the SET susceptibilities, $\chi_{CLB_{average}}$ is 0.17182 and $N_{SET_{Total}}$ is 70,446. Considering only the topology effect on the SET susceptibilities, $\chi_{CLB_{average}}$ is 0.16585 and $N_{SET_{Total}}$ is 67,999. Finally, considering both the topology and the input patterns effects on the SET susceptibilities, $\chi_{CLB_{average}}$ is 0.18174 and $N_{SET_{Total}}$ is 74,513.

5. Results

The obtained SEV estimations for each analysed benchmark circuit are presented in Table 2. For each benchmark, Table 2 includes the quantity of observed SEs in each category of SEV estimation, and the respective estimated SEV, in percentage. To improve readability, Table 2 does not include the total number of injected SETs. As explained in Section 4.3, in all cases it was injected an equivalent number of 1000 SETs per transistor.

Figure 6 presents the absolute value of the errors in the SEV estimation process. For the analysed benchmarks, in the chosen operational scenarios, the average error of the SEV estimation obtained ignoring the SET susceptibilities was 15.27%. Considering only the topology effect leads to an average error of 4.70%. Lastly, considering the topology and the input patterns effect leads to an average error of 0.68%. The calculus of these average errors takes into account the modulus of the errors presented in Table 2.

Complementary, ignoring the SET susceptibilities lead to an error of up to 74.32% (*decod* benchmark), while the best agreement was −1.89% of error (*t481* benchmark). Considering only the topology had a maximum error of 24.85% (*cmb* benchmark) and best agreement of −0.08% (*parity* benchmark).

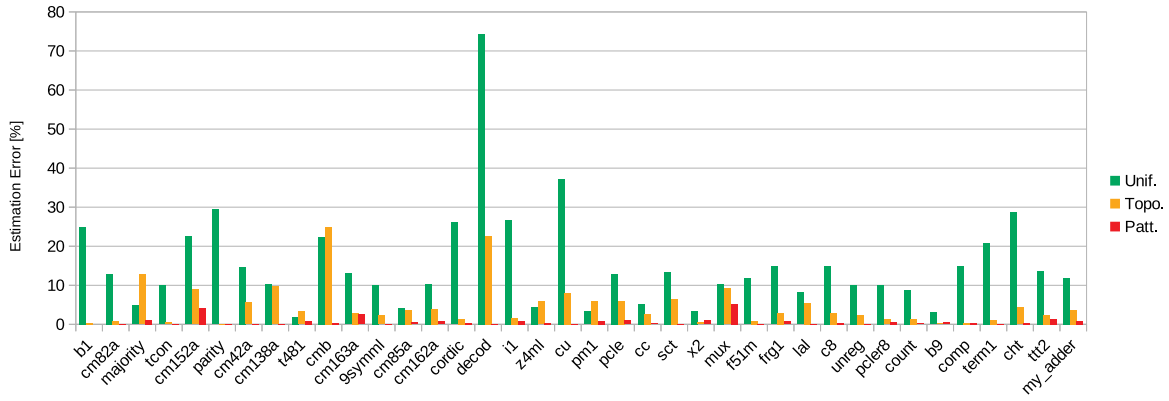

Figure 6. Errors of the SEV estimation processes. For each benchmark circuit, it presents the absolute value of the error of the three SEV estimations at the CLB level in comparison to the reference estimation at the transistor level.

Finally, considering both the topology and the input patterns effect leads to a maximum error of −5.17% (*mux* benchmark), and a minimum error of 0.00% (*b1*, *parity*, *sct*, and *term1* benchmarks).

Figure 7 presents the resulting simulation times. At the transistor level, there is an exponential growth of the time required to estimate the SEV as a function of the circuit size. At the CLB level, there is also a growth in the simulation time, as a function of the circuit size, but it is less accentuated. For the largest circuit analysed (*my_adder* benchmark), the simulation time for the estimation at the transistor level is three orders of magnitude higher than those at the CLB level. For the estimation methods at the CLB level, the simulation times are similar, but the required time for the methods with weighted distributions are higher than those with uniform distribution.

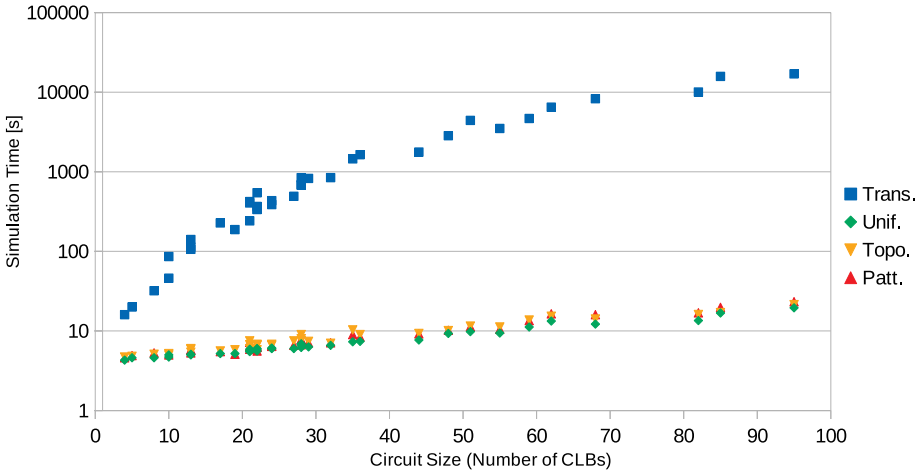

Figure 7. Simulation time of the SEV estimation processes. It presents the simulation time for the four SEV estimations as a functions of the circuit size.

6. Discussion

The main contribution of this work is the SEV estimation approach that incorporates the specific SET susceptibility of each gate (CLB). The evaluation of this proposed approach took into consideration the influence of the gate topology and the distribution of input patterns applied to them, focusing on the logical masking. The results (Section 5) show a significant reduction of the SEV estimation error by considering the gate topology. Moreover, it is possible to obtain an even better accuracy if the input patterns effect is also considered.

While our work focused on the logical masking, the simulation framework (Section 2.1) and estimation process (Section 2.2) can be used with SET susceptibilities that also include the electrical and temporal masking effects. The inclusion of the electrical masking effect on the SET susceptibilities would require the use of simulations with delays, to evaluate the SET propagation. In this work, we used zero-delay simulations because we focused on logical masking. Similarly, the inclusion of the temporal masking effect would also require the use of delays. However, this masking effect concerns only the sequential elements (memory cells and registers—latches and flip-flops), that was not considered in the analysis. The inclusion of sequential elements as the origin of induced SEs would additionally require that the simulation framework enables the injection of SEUs.

Both adaptions would be minor changes in the proposed approach, impacting mainly the simulation time. The major challenge would be to obtain the SET susceptibility of each element. Due to the electrical masking effect, the SET susceptibility of the combinational elements would be a function of the induced SET width at the transistor level. For the sequential elements, with the addition of the temporal masking, the SET susceptibility would also be a function of the Clock/Gate frequency.

The estimation of the SET susceptibilities of the CLBs of an FPGA technology, or the standard cells of an ASIC, would be obtained through electrical simulations, similar to the described in [7]. However, it requires the details of the technology implementation, that usually are not available. In this case, the source of these parameterised SET susceptibilities should be the device manufacturers.

As mentioned previously, the electrical simulation approach, as presented in [7], is interesting to estimate the SET susceptibilities. However, this approach may be impractical for the SEV estimation of large circuits, due to the simulation time at the transistor level.

The analytical approach presented by the authors of [3] also focuses on the combinational elements. While they include the analysis of sequential circuits, they focus on the SET generation and propagation through the combinational part of the circuit. Their analysis is strictly restricted to the combinational elements (even though they are used in sequential circuits). In contrast, as already discussed, our approach can be adapted to include the analysis of sequential elements.

As future perspectives, our simulation-based probability-aware approach can be extended to incorporate the injection of multiple SETs. Such method would be interesting for the analysis of multiple SETs generated by a single radiation particle, but also for the analysis of SoCs (Systems on Chip), for which the multiple SETs (or SEUs) would be injected at the outputs of the SoC functional blocks.

Finally, our proposed approach can also be adapted for an emulation-based fault-injection platform, as the one introduced in [25]. In this case, the emulation would replace the simulation framework, used in Step 7, while keeping the other steps. The emulation would reduce the estimation time, as the DUT would run nearly real-time.

Supplementary Materials: The following are available online at http://www.mdpi.com/2079-9292/8/7/749/s1, Table S1: Configurations of the VersaTile combinational functions, Table S2: SET susceptibilities of the VersaTile combinational functions.

Author Contributions: Conceptualization, F.B.A., L.A.d.B.N. and R.d.; methodology, F.B.A.; writing—original draft preparation, F.B.A.; writing—review and editing, L.A.d.B.N. and R.d.; supervision, L.A.d.B.N. and R.d.

Funding: This work was supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (grant number 207364/2015-0/SWE).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Slayman, C. JEDEC Standards on Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray Induced Soft Errors. In *Soft Errors in Modern Electronic Systems*; Nicolaidis, M., Ed.; Springer: New York, NY, USA, 2011; Chapter 3, pp. 55–76.
2. Asadi, H.; Tahoori, M.B.; Mullins, B.; Kaeli, D.; Granlund, K. Soft Error Susceptibility Analysis of SRAM-Based FPGAs in High-Performance Information Systems. *IEEE Trans. Nucl. Sci.* **2007**, *54*, 2714–2726. doi:10.1109/TNS.2007.910426.
3. Rezaei, S.; Miremadi, S.G.; Asadi, H.; Fazeli, M. Soft error estimation and mitigation of digital circuits by characterizing input patterns of logic gates. *Microelectron. Reliab.* **2014**, *54*, 1412–1420. doi:10.1016/j.microrel.2014.03.003.
4. Raji, M.; Pedram, H.; Ghavami, B. A practical metric for soft error vulnerability analysis of combinational circuits. *Microelectron. Reliab.* **2015**, *55*, 448–460. doi:10.1016/j.microrel.2014.11.004.
5. Jenn, E.; Arlat, J.; Rimen, M.; Ohlsson, J.; Karlsson, J. Fault injection into VHDL models: the MEFISTO tool. In Proceedings of the IEEE 24th International Symposium on Fault-Tolerant Computing, Austin, TX, USA, 15–17 June 1994; pp. 66–75. doi:10.1109/FTCS.1994.315656.
6. Baraza, J.; Gracia, J.; Gil, D.; Gil, P. Improvement of fault injection techniques based on VHDL code modification. In Proceedings of the Tenth IEEE International High-Level Design Validation and Test Workshop, Napa Valley, CA, USA, 30 November–2 December 2005; pp. 19–26. doi:10.1109/HLDVT.2005.1568808.
7. Buard, N.; Anghel, L. Gate Level Modeling and Simulation. In *Soft Errors in Modern Electronic Systems*; Nicolaidis, M., Ed.; Springer: New York, NY, USA, 2011; Chapter 4, pp. 77–102. doi:10.1007/978-1-4419-6993-4_4.
8. Lopes Filho, A.; d'Amore, R. Analysis of the error susceptibility of a field programmable gate array-based image compressor through random event injection simulation. *IET Comput. Digit. Tech.* **2012**, *6*, 160–165. doi:10.1049/iet-cdt.2011.0056.
9. Civera, P.; Macchiarulo, L.; Rebaudengo, M.; Sonza Reorda, M.; Violante, M. Exploiting circuit emulation for fast hardness evaluation. *IEEE Trans. Nucl. Sci.* **2001**, *48*, 2210–2216. doi:10.1109/23.983197.
10. Entrena, L.; Garcia-Valderas, M.; Fernandez-Cardenal, R.; Lindoso, A.; Portela Garcia, M.; Lopez-Ongil, C. Soft Error Sensitivity Evaluation of Microprocessors by Multilevel Emulation-Based Fault Injection. *IEEE Trans. Comput.* **2012**, *61*, 313–322. doi:10.1109/TC.2010.262.
11. Ebrahimi, M.; Mohammadi, A.; Ejlali, A.; Miremadi, S.G. A fast, flexible, and easy-to-develop FPGA-based fault injection technique. *Microelectron. Reliab.* **2014**, *54*, 1000–1008. doi:10.1016/j.microrel.2014.01.002.

12. Rezgui, S.; Wang, J.J.; Tung, E.C.; Cronquist, B.; McCollum, J. New Methodologies for SET Characterization and Mitigation in Flash-Based FPGAs. *IEEE Trans. Nucl. Sci.* **2007**, *54*, 2512–2524. doi:10.1109/TNS.2007.910126.
13. Bottoni, C.; Glorieux, M.; Daveau, J.; Gasiot, G.; Abouzeid, F.; Clerc, S.; Naviner, L.; Roche, P. Heavy ions test result on a 65nm Sparc-V8 radiation-hard microprocessor. In Proceedings of the 2014 IEEE International Reliability Physics Symposium, Waikoloa, HI, USA, 1–5 June 2014. doi:10.1109/IRPS.2014.6861096.
14. Cai, C.; Fan, X.; Liu, J.; Li, D.; Liu, T.; Ke, L.; Zhao, P.; He, Z. Heavy-Ion Induced Single Event Upsets in Advanced 65 nm Radiation Hardened FPGAs. *Electronics* **2019**, *8*, 323. doi:10.3390/electronics8030323.
15. Franco, D.T.; Vasconcelos, M.C.; Naviner, L.; Naviner, J.F. Reliability analysis of logic circuits based on signal probability. In Proceedings of the 2008 15th IEEE International Conference on Electronics, Circuits and Systems, St. Julien's, Malta, 31 August–3 September 2008; pp. 670–673. doi:10.1109/ICECS.2008.4674942.
16. Han, J.; Chen, H.; Boykin, E.; Fortes, J. Reliability evaluation of logic circuits using probabilistic gate models. *Microelectron. Reliab.* **2011**, *51*, 468–476. doi:10.1016/j.microrel.2010.07.154.
17. Han, J.; Chen, H.; Liang, J.; Zhu, P.; Yang, Z.; Lombardi, F. A Stochastic Computational Approach for Accurate and Efficient Reliability Evaluation. *IEEE Trans. Comput.* **2014**, *63*, 1336–1350. doi:10.1109/TC.2012.276.
18. Armelin, F.B.; Naviner, L.A.B.; d'Amore, R.; Azevedo, I.A. Impact evaluation of logic blocks configuration on FPGA's soft error rate estimation. In Proceedings of the 2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS), Monte Carlo, Monaco, 11–14 December 2016; pp. 277–280. doi:10.1109/ICECS.2016.7841186.
19. Institute of Electrical and Electronics Engineers. *1364-1995—IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language*; Technical Report; IEEE: New York, NY, USA, 1996.
20. Open Source VHDL Verification Methodology. Available online: http://www.webcitation.org/78vi82p4P (accessed on 6 June 2019).
21. MICROSEMI. *ProASIC3E Flash Family FPGAs with Optional Soft ARM Support*; Technical Report; Microsemi Corporation: Aliso Viejo, CA, USA, 2015.
22. MICROSEMI. *ProASIC3E FPGA Fabric User's Guide*; Technical Report; Microsemi Corporation: Aliso Viejo, CA, USA, 2012.
23. MICROSEMI. *IGLOO, ProASIC3, SmartFusion and Fusion Macro Library Guide for Software v10.1*; Technical Report; Microsemi Corporation: Aliso Viejo, CA, USA, 2010.
24. Yang, S. *Logic Synthesis and Optimization Benchmarks User Guide Version 3.0*; Technical Report; MCNC: Research Triangle, NC, USA, 1991.
25. Armelin, F.B.; Naviner, L.A.B.; d'Amore, R. Probability aware fault-injection approach for SER estimation. In Proceedings of the 2018 IEEE 19th Latin-American Test Symposium (LATS), Sao Paulo, Brazil, 12–14 March 2018; pp. 1–3. doi:10.1109/LATW.2018.8349692.
26. MAXIM INTEGRATED. *Application Note 4400—Pseudo Random Number Generation Using Linear Feedback Shift Registers*; Technical Report; Maxim Integrated: San Jose, CA, USA, 2010.