micromachines

Article

A Fast Simulation Method for Evaluating the Single-Event Effect in Aerospace Integrated Circuits

Xiaorui Zhang [1], Yi Liu [1,*], Changqing Xu [1,2,*], Xinfang Liao [1], Dongdong Chen [1] and Yintang Yang [1]

[1] Laboratory of Digital IC and Space Application, School of Microelectronics, Xidian University, Xi'an 710071, China
[2] Guangzhou Institute of Technology, Xidian University, Guangzhou 510555, China
* Correspondence: yiliu@mail.xidian.edu.cn (Y.L.); cqxu@xidian.edu.cn (C.X.)

Abstract: With the continuous progress in integrated circuit technology, single-event effect (SEE) has become a key factor affecting the reliability of aerospace integrated circuits. Simulating fault injection using the computer simulation technique effectively reflects the SEE in aerospace integrated circuits. Due to various masking effects, only a small number of faults will result in errors; the traditional method of injecting one fault in one workload execution is inefficient. The method of injecting multiple faults in one workload execution will make it impossible to judge which fault results in errors because the propagation characteristic of SEE and faults may affect each other. This paper proposes an improved multi-point fault injection method to improve simulation efficiency and solve the problems of the general multi-point fault injection method. If one workload execution does not result in errors, multiple faults can be verified by one workload execution. If one workload execution results in errors, a specific grouping method can be used to determine which faults result in errors. The experimental results show that the proposed method achieves a good acceleration effect and significantly improves the simulation efficiency.

Keywords: aerospace integrated circuits; single-event effect; multi-point fault injection method; grouping method

Citation: Zhang, X.; Liu, Y.; Xu, C.; Liao, X.; Chen, D.; Yang, Y. A Fast Simulation Method for Evaluating the Single-Event Effect in Aerospace Integrated Circuits. *Micromachines* **2023**, *14*, 1887. https://doi.org/10.3390/mi14101887

Academic Editor: Piero Malcovati

Received: 28 August 2023
Revised: 28 September 2023
Accepted: 28 September 2023
Published: 30 September 2023

1. Introduction

Due to the existence of the earth radiation belt proton, galactic cosmic ray, solar cosmic ray, and so on, the radiation effect in the space environment seriously affects the reliability of aerospace integrated circuits [1]. With the continuous progress in integrated circuit technologies, single-event effect (SEE) has become a key factor affecting the reliability of aerospace integrated circuits [2,3]. SEE results from the generation of electron-hole pairs in the sensitive area of the device when high-energy particles bombard microelectronic devices. These charges are collected by the electrodes of the sensitive device, affecting the working state of the aerospace integrated circuits [4–6]. The impact of SEE on aerospace integrated circuits will lead to system functional errors and even system functional failure in severe cases. So, it is necessary to evaluate the SEE in aerospace integrated circuits.

The evaluation methods of SEE mainly include onboard experiments, ground experiments [7], and computer simulation techniques [8]. The experiment data obtained from onboard experiments and ground experiments can reflect realistic changes in aerospace integrated circuits in the radiation environment, but there are disadvantages, such as long test cycles and high costs. In contrast, computer simulation technology has the characteristics of low cost and high efficiency, which can be used as a supplement to onboard experiments and ground experiments.

The evaluation method of SEE based on computer simulation technique mainly includes device-level simulation method, circuit-level simulation method, and system-level simulation method according to the hierarchy.

Device-level simulation method uses relevant parameters to build device models and then adds the radiation model to simulate the impact of SEE [9–11]. This method can reflect the physical mechanism of SEE with high accuracy, but the simulation speed is slow, so it is only suitable for small-scale circuits. Circuit-level simulation object is the circuit netlist described by SPICE. It needs to obtain the transient current source model generated by the device-level simulation result, and then the transient current source model is added to the circuit node by PWL or Verilog-A [12–14], etc. Although this method improves the simulation speed, it still cannot meet the speed requirements of large-scale digital circuits. So, it is suitable for analog circuits and small-scale digital circuits. System-level simulation method modifies the relevant signal to simulate the impact of SEE in the RTL model or higher hierarchy [15–17]. The simulation speed is fast, but it cannot reflect the physical mechanism of SEE and it cannot be compared to the SEE experiment result.

Therefore, this paper uses a mixture simulation method. The simulation object is divided into different modules according to the execution function. The module to add the transient current source model is described by SPICE; the other modules are described by RTL. Different types of modules are connected through A/D and D/A interfaces, as shown in Figure 1.

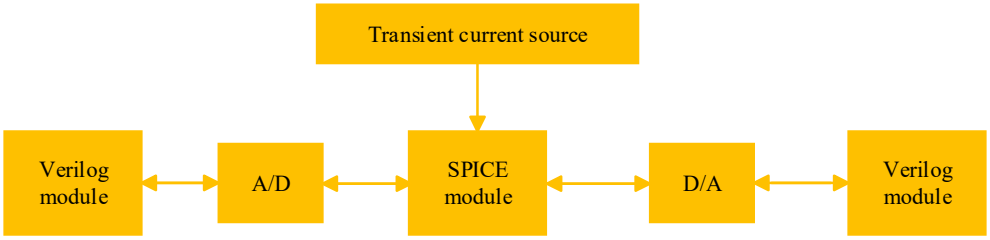

Figure 1. Mixture simulation method.

The transient current source model adopted in this paper is obtained from previous work [18] and is added by PWL.

With the development of integrated circuit technology, the fault injection set becomes larger and larger. To effectively reflect the SEE in aerospace integrated circuits, tens of thousands of faults need to be verified at least [19]. The traditional fault injection method injects only one fault in one workload execution. However, due to the propagation characteristics of SEE and the design characteristics of aerospace integrated circuits, most faults will not result in errors [20,21]. The traditional fault injection method is inefficient.

Injecting multiple faults in one workload execution has the problem of being unable to judge which fault results in errors because the errors continue to propagate as the aerospace integrated circuits run. So, it will be impossible to judge whether the subsequent faults injected after errors occurred result in errors. In addition, injecting multiple faults in one workload execution may result in additional errors compared to injecting one fault.

The current research direction is mainly to improve efficiency by reducing the fault injection set, which is that some faults can be judged whether they result in errors without simulation. Some studies analyze SEE by analyzing the instruction program to determine whether faults will result in errors [22–24]. Some studies analyze from the perspective of logic masking [25,26], register reading, and writing [27].

Although these methods can directly improve simulation efficiency, all methods are based on injecting only one fault in one workload execution. Otherwise, the methods of analyzing the instruction program makes accuracy decline. The methods of analyzing logic masking consider logic masking without considering sequential circuits. The method of analyzing register reading and writing only considers the circuits with read and write ports.

So, an improved multi-point fault injection method is proposed in this paper. This method not only solves the problems existing in the general multi-point fault injection method but also can be applied to any aerospace integrated circuits. The improved multi-point fault injection method injects multiple faults in one workload execution. If the

workload execution does not result in errors, it is proved that the faults injected in this workload execution would not result in errors. If the workload execution results in errors, a grouping method is used to find faults that result in errors. Simulation results based on the Leon2 system show that the proposed method can achieve a speedup of 60.69× at most.

2. Proposed Method

2.1. Motivation

The general multi-point fault injection method cannot determine which fault is an error fault and the grouping method can solve the problem. The purpose of grouping is to find the error faults.

Once the workload execution results in errors, the injected faults will be grouped. If there are no errors, the injected faults would not be grouped. Each group corresponds to a workload execution with injected faults. Considering the propagation characteristics of SEE, determining whether a fault is an error fault can only be done on the premise that only one fault is injected in one workload execution. Therefore, the grouping ends until the workload execution does not result in errors or the number of injected faults in one workload execution is 1.

Injecting multiple faults does result in additional errors compared to injecting one fault, even if the probability of this happening is low. However, the number of injected faults must be one when judging whether one fault is an error fault or not by using the grouping method, so this shows that even if it occurs, only increasing the number of workload executions will not affect the correctness of the results.

Based on the above analysis, the multi-point fault injection method based on the grouping method can solve the problems of general multi-point fault injection and improve the simulation efficiency.

2.2. The Introduction of One Multi-Point Fault Injection Process

Before analyzing SEE, it is necessary to run one workload execution completely without injecting faults to obtain the correct simulation data, called golden data, which is used to judge whether the workload execution that injected faults results in errors. The multi-point fault injection process proposed in this paper needs to inject a certain number of faults into one workload execution first; these faults are called the first layer faults, and this workload execution is called the first layer workload execution. The simulation results of the first layer workload execution will be compared with the golden data. The specific grouping method is obtained in Section 2.3. Each group is called one fault group. After the process of the first layer ends, the process of the second layer begins. First, it is necessary to judge the number of fault groups and the number of faults contained in each group in the second layer, which depends on the grouping method of the previous layer. A multi-point fault injection process ends until a layer contains no fault group. The process of deciding whether to group the injected faults is shown in Figure 2.

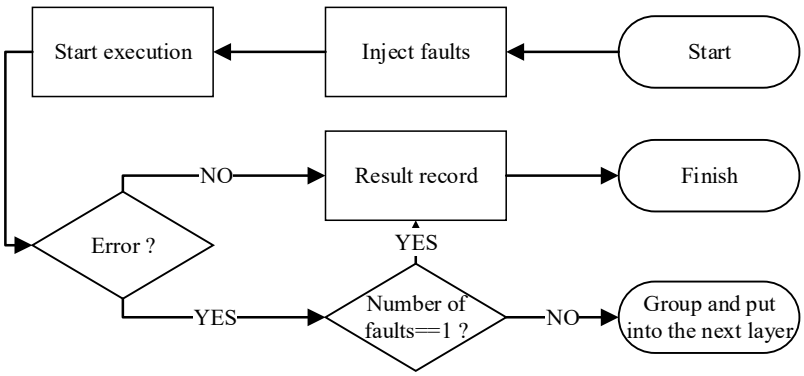

Figure 2. The process of deciding whether to group the injected faults.

Take the example of injecting eight faults in one workload execution and the grouping method is to divide the faults as equally as possible into two groups. Assume that there is one and only one error fault in eight faults. The number of the first layer faults is eight. In the first layer, eight faults are injected in one workload execution. The comparison results are inconsistent, and the number of faults injected is greater than one, the eight faults will be equally divided into two groups of four faults each and are defined as the second layer fault groups. The second layer includes two fault groups, so the number of the second layer workload executions is two and one workload execution results in errors. The above process is repeated until the multi-point injection process ends. The structure of this multi-point fault injection process is shown in Figure 3. The number inside the rectangle indicates the number of faults injected into one workload execution. Red means the workload execution results in errors; yellow means the workload execution does not result in errors.

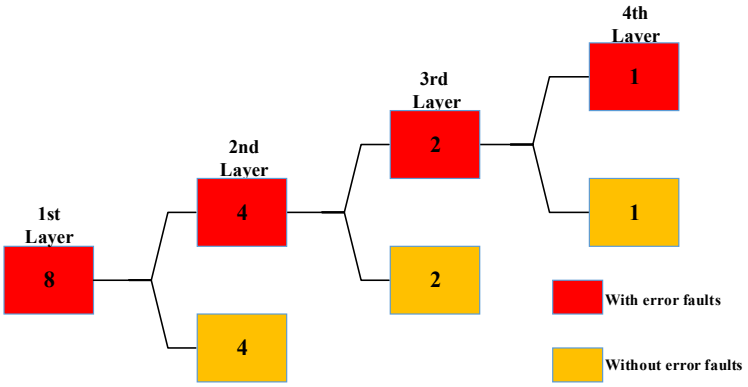

Figure 3. The structure of one multi-point fault injection process.

2.3. Determination of Relevant Parameters

The four cases are shown in Figure 4. The number of the first layer faults in each case is eight. In Figure 4a, the first layer workload execution does not result in errors because there are no error faults. However, the number of workload executions required increases as the number of error faults increases. In Figure 4b–d, 7, 9 and 11, workload executions are required, and the corresponding numbers of error faults are 1, 2, and 2. When the number of error faults increases to a certain number, the proposed method will be less efficient than the traditional fault injection method.

From the above analysis, it can be known that it is necessary to control the number of error faults in the first layer faults, which is related to the error probability. Otherwise, the number of fault groups divide, and the number of faults contained in each fault group are different with different grouping methods. So, one multi-point fault injection process requires two input parameters: the number of the first layer faults and the specific grouping method. Next, this paper will explain how to choose the two parameters through theoretical analysis.

Assume that the number of the first layer faults is k, and the error probability is p. The number of error faults is a random variable with a binomial distribution. Therefore, the probability that the number of error faults = i is:

$$P(x = i) = C_k^i \times p^i \times (1 - p)^{k-i} \quad (1)$$

C_k^i denotes the binomial coefficient. The number of error faults can vary from 0 to k theoretically, so the number of workload executions required for one multipoint fault injection process is:

$$sum = 1 + \sum_{i=1}^{k} [R(k,i) \times P(x = i)] \quad (2)$$

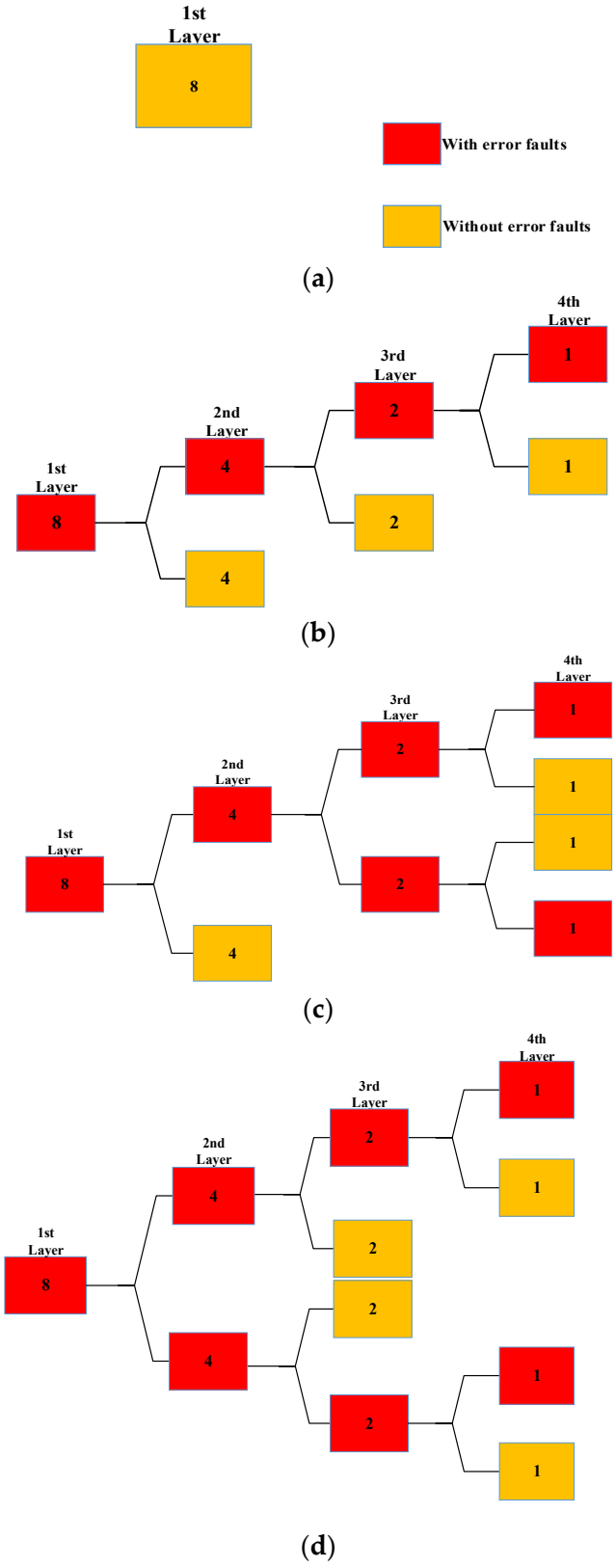

Figure 4. There are four structures and the number of the first layer faults is 8: (**a**) the number of error faults is 0 and the number of workload executions required is 1; (**b**) the number of error faults is 1 and the number of workload executions required is 7; (**c**) the number of error faults is 2 and the number of workload executions required is 9; (**d**) the number of error faults is 2 and the number of workload executions required is 11.

R (k, i) denotes the number of workload executions required when the number of the first layer faults is k, and the number of error faults is i. If i = 0, the number of workload executions required is one.

Then, it only needs to determine the grouping method to calculate R (k, i). To determine the grouping method, it is necessary to obtain the complete structure of this multi-point fault injection process. The complete structure refers to the number of layers, the number of fault groups contained in each layer, and the number of faults contained in each fault group. To obtain the complete structure, it is assumed that each workload execution will result in errors. Therefore, if the number of faults in one fault group is greater than 1, it will be grouped until the number of faults is 1 for all fault groups in one layer.

The grouping method in this paper only considers four grouping methods, which is dividing the faults in one fault group as equally as possible into two groups, three groups four groups, or five groups. Four complete structures can be obtained for the same number of the first layer faults.

Some special grouping situations need to be explained to ensure that the number of faults in each group is as equal as possible, for example, if the number of faults in one fault group is 7 and the faults are divided into three groups, the number of faults in the three group is 3, 2, and 2. In addition, if the number of faults is 3 but it needs to be divided into four groups, it will only be divided into three groups and each group contains one fault.

The probability pro that one fault group contains error faults is:

$$\text{pro} = 1 - \frac{C_{k-n_{m_h}}^{i}}{C_{k}^{i}} \tag{3}$$

C_{k}^{i} and $C_{n_{m_h}}^{i}$ denotes the binomial coefficient. m_h denotes the m-th fault group of the h-th layer. n_{m_h} denotes the number of faults contained in the m-th fault group of the h-th layer.

When a fault group contains error faults, it will be grouped and will need additional workload executions. The number of additional workload executions depends on the grouping method; if the fault group is divided into two groups, the number of additional workload executions is 2; if the fault group is divided into three groups, the number of additional workload executions is 3. The other grouping methods are the same. Summing up the number of additional workload executions required for all fault groups, the number of workload executions required can be obtained when the number of error faults is i ($i \geq 1$), which is:

$$R(k,i) = \sum_{h=1}^{H} \sum_{m=1}^{M_h} \{[1 - \frac{C_{k-n_{m_h}}^{i}}{C_{k}^{i}}] \times \text{group}\} \tag{4}$$

H denotes the number of layers, M_h denotes the number of fault groups at the h-th layer, and group denotes the number of additional workload executions required when a fault group contains error faults.

Therefore, the number of workload executions required for one multi-point fault injection process is shown:

$$\begin{aligned} \text{sum} &= 1 + \sum_{i=1}^{k} [P(x = i) \times R(k,i)] \\ &= 1 + \sum_{i=1}^{k} \{P(x = i) \times \sum_{h=1}^{H} \sum_{m=1}^{M_h} \{[1 - \frac{C_{k-n_{m_h}}^{i}}{C_{k}^{i}}] * \text{group}\}\} \end{aligned} \tag{5}$$

The speedup is shown:

$$\text{speedup} = \frac{k}{\text{sum}} \tag{6}$$

The four grouping methods can obtain four speedups with the same number of the first layer faults. Based on the speedup, the optimal number of the first layer faults and the optimal grouping method can be selected. It is worth stating that the error probability also determines the upper limit for the number of the first layer faults. The error probability times the optimal number of the first layer faults should be less than one, because the case without error faults requires the least number of workload executions.

When error probability = 0, it indicates there are no error faults. To ensure the fastest speed, it is necessary to double the number of the first layer faults this time as the number of the first layer faults in the next multi-point fault injection process. When error probability = 0, the more the number of the first layer faults and the more faults can be verified in one workload execution.

2.4. Add Checkpoints to Optimize the Grouping Method

After obtaining the optimal number of the first layer faults and the optimal grouping method in Section 2.3, this section proposes a method of adding checkpoints to optimize the grouping method. By judging whether the workload execution results in errors at all checkpoints, the earliest checkpoint that detects errors when the workload execution results in errors is called the error checkpoint. If one workload execution results in errors, it Is proved that the injection time of error faults is before the error checkpoint. All faults injected after the error checkpoint need to be injected into one workload execution to determine whether these faults contain error faults because of the propagation characteristics of SEE. For the faults injected after the error checkpoint, it requires the same process for determining other error checkpoints until the workload execution does not result in errors. The process of adding checkpoints is shown in Figure 5.

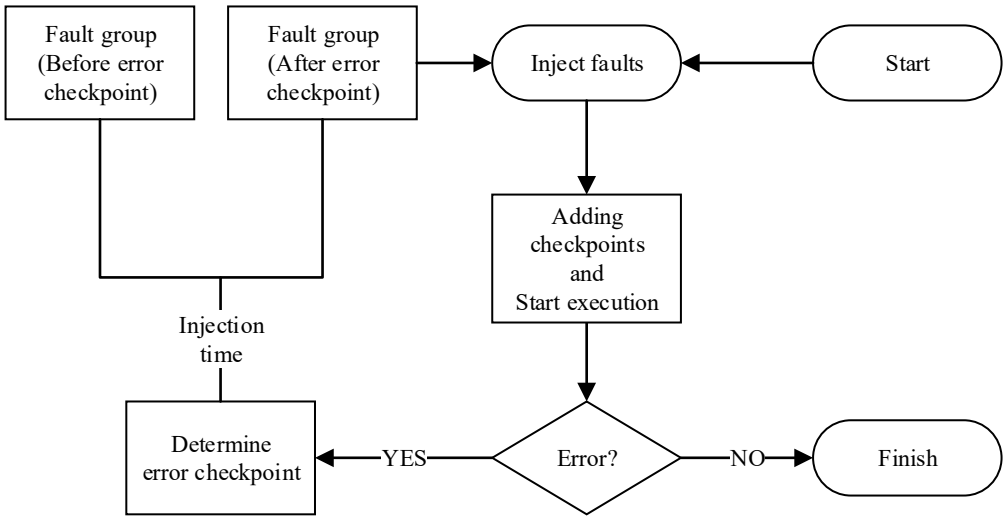

Figure 5. The process of determining error checkpoints.

The faults are divided into multiple parts by error checkpoints, and the number of faults contained in each part can be obtained through the grouping method based on checkpoints. It is equal to obtaining the number of the first layer faults in each part, so that each part can search for error faults using the optimal grouping method mentioned in Section 2.3, and the number of error checkpoints depends on the number of error faults.

The process of one multi-point fault injection process with the addition of checkpoints is shown in Figure 6. Adding checkpoints is performed to find the injection time range of error faults, and the number of workload executions required can be reduced. Results show that the method with the addition of checkpoints is faster than the multi-point injection method without the addition of checkpoints.

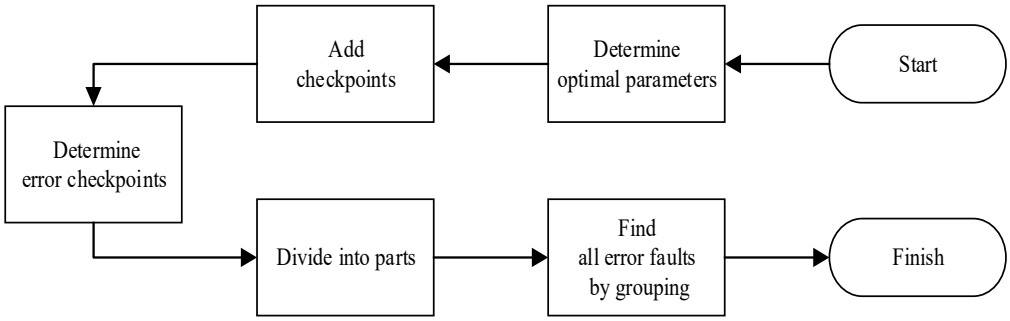

Figure 6. The process of one multi-point fault injection process with addition of checkpoints.

2.5. Overall Simulation Flow

The overall simulation flow is shown in Figure 7. We can estimate the error probability and choose an appropriate initial value to satisfy the error probability times the optimal number of the first layer faults that should be less than one. So, the initial number of the first layer faults in the first multi-point fault injection process is chosen as 10 in this paper. The simulation flow ends until the number of faults that have been verified is large enough to calculate the error probability of the circuits.

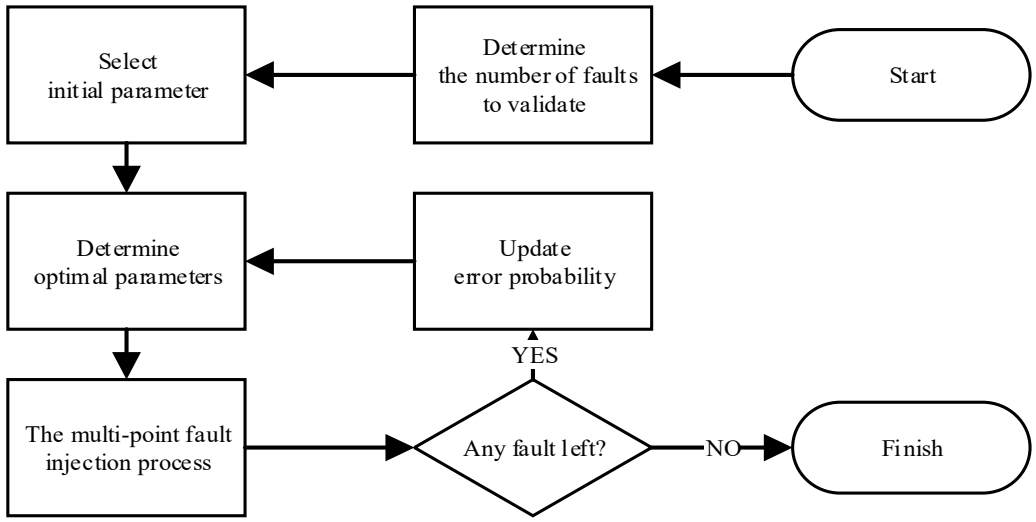

Figure 7. The overall simulation flow.

3. Results and Analysis

3.1. Simulation Setup

To validate the proposed method, the SEE simulation based on the Leon2 system is carried out in this section. Leon2 processor is a 32-bit processor developed by the European Space Agency (ESA). Its code is composed of synthesizable VHDL code. The Leon2 system has one Leon2 processor and three SRAMs. The main functional modules of the Leon2 processor include an integer unit, register-file, cache, and memory controller module. Figure 8 shows the structure of the Leon2 system.

The current source model selected in this paper is based on the previous work and it is added to the drain of NMOSFET [28]. When faults are injected into the digital circuits, the following three conditions may occur in the digital circuits [29]: a. direct errors: the output data of the workload execution is wrong; b. potential errors: the output data of the workload execution is correct, but the operating state of the workload execution is wrong; c. invalid errors: errors occur but recover quickly.

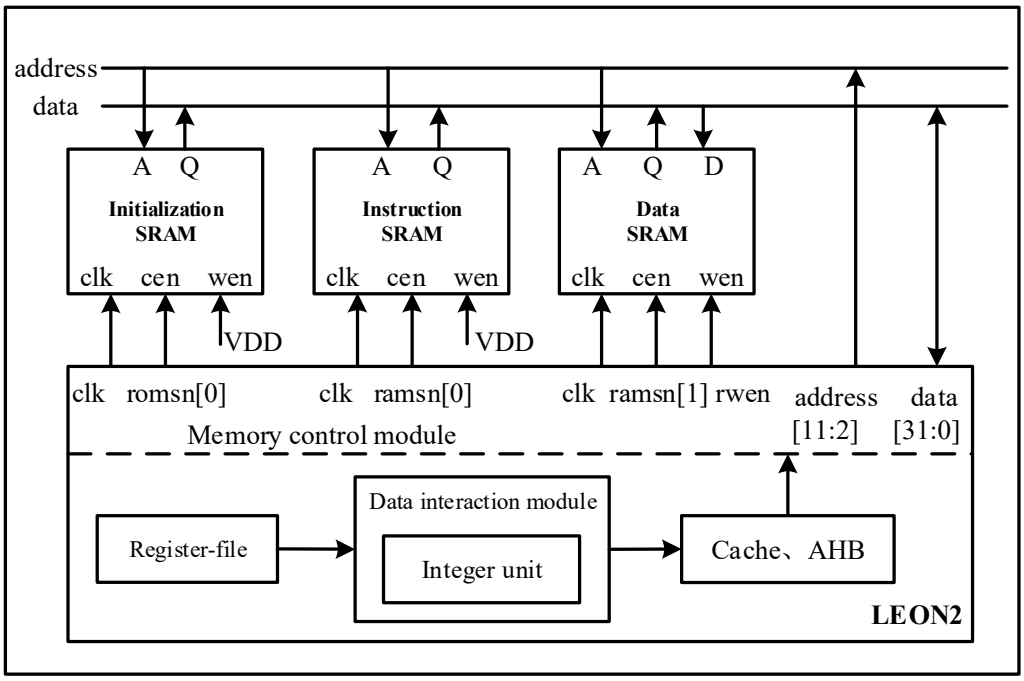

Figure 8. The structure of the Leon2 system.

This paper only considers direct errors and potential errors. When the workload execution is finished, if the information stored in the data SRAM is inconsistent, it is proved that direct errors occurred. If the five-stage pipeline registers state as inconsistent, it is proved that potential errors occurred. Either direct errors or potential errors are considered to indicate the workload execution results in errors.

The time interval between the two checkpoints chosen in this paper is the time for the aerospace integrated circuit system to complete one process from reading data to writing data, which is to ensure that the errors can be detected. The test program adopted in this paper is the matrix multiplication program. By analyzing the test program, the Leon2 system takes 28 clock cycles to complete one process from reading data to writing data. The Leon2 processor clock frequency is 100Mhz, so the time interval between two checkpoints is selected as 300 ns to round. Parameters related to one workload execution are shown in Table 1.

Table 1. Relevant simulation parameters.

Setup	Parameters	Description
Time interval	300 ns	Analyze the test program
Injection time	8700–15,000 ns	From the end of the system initialization
LET	100 MeV·cm^2/mg	Make sure errors can be detected

From 0 to 8700 ns, it is the period of system initialization. No faults are injected in this period. The modules verified in this paper include integer unit, register-file, cache, and memory controller module.

Table 2 shows the areas of the different modules obtained through the Design compile software. Considering that the register-file has a larger circuit area than other modules and makes the simulation results statistically significant [19], the other modules inject 10,000 faults each and the register-file module injects 100,000 faults.

Table 2. Areas of different modules.

Fault Injection Module	Area/μm^2
Integer unit	53,148.99
Cache	21,611.30
Memory control module	12,165.27
Register-file	421,410.11

3.2. Result and Analysis

This paper ensures that the time and location of the fault injection are consistent when comparing the speed improvement with and without checkpoints. Table 3 shows the error probability of different modules and the speed improvements with and without checkpoints, which is compared with the traditional fault injection method. The results show that the proposed method can achieve a speedup of 60.69× at most and the method of adding checkpoints can further improve the simulation efficiency.

Table 3. Speedup of different injection modules with and without checkpoints.

Fault Injection Module	Error Probability	Speedup without Checkpoints	Speedup with Checkpoints
Integer unit	2.85%	3.88×	4.17×
Cache	0.82%	9.81×	10.30×
Memory control module	1.92%	5.16×	5.34×
Register-file	0.09%	57.56×	60.69×

While the traditional fault injection method is equivalent to verifying one fault by one workload execution, the proposed method equals reducing the required number of workload executions by the grouping method in Section 2.3. In the initial multi-point fault injection processes, the number of verified faults is small, and the error probability is unstable. After the error probability is stable, the number of the first layer faults and the grouping method will not change until the end of the simulation process. Table 4 shows the optimal number of the first layer faults, the optimal grouping method for different injection modules, and the speedup by theoretical analysis after the error probability is stable.

Table 4. The optimal number and grouping method of different injection modules and corresponding speedup by theoretical analysis.

Fault Injection Module	The Optimal Number of Faults	The Optimal Grouping Method	Speedup by Theoretical Analysis
Integer unit	27	3	3.90×
Cache	81	3	9.92×
Memory control module	27	3	5.21×
Register-file	729	3	61.43×

As can be seen from Tables 3 and 4, the optimal number of the first layer faults times error probability is less than 1 and the optimal grouping method of the four modules is tripartition. The speedup obtained by simulation results is smaller than that obtained by theoretical analysis. Firstly, the error probability is unstable in the initial multi-point fault injection processes, which means speedup is unstable. Secondly, it takes some time to calculate the optimal number of the first layer faults and the grouping method before starting a multi-point fault injection process. Injecting multiple faults also increases the time to execute one workload execution compared to injecting one fault. However, the calculation time and additional time added have little impact compared to the time of executing one workload execution. Lastly, the

higher the number of faults injected in one workload, the higher the probability of resulting in additional errors. It indicates more workload executions are needed to exclude additional errors. The more faults injected in one workload execution, the more the speedup decreases compared with the theoretical analysis.

Table 3 also shows that the simulation with addition of checkpoints is more efficient than without checkpoints because error checkpoints reduce the search range by dividing the fault into several parts according to the injection time. Table 5 shows the number of faults that need to be searched under different numbers of error faults with and without addition of checkpoints after the error probability is stable.

Table 5. The number of faults that need to be searched under different numbers of error faults with and without addition of checkpoints after the error probability is stable.

Injection Module	Error Faults = 1		Error Faults = 2		Error Faults $\geq$ 3	
	With Checkpoints	Without Checkpoints	With Checkpoints	Without Checkpoints	With Checkpoints	Without Checkpoints
Integer unit	11.81	27	14.49	27	16.71	27
Cache	29.88	81	44.65	81	57.31	81
Memory control module	14.63	27	17.24	27	19.28	27
Register-file	234.25	729	348.45	729	443.33	729

Adding multiple checkpoints does increase the time of executing one workload execution, but the method proposed in this paper only adds multiple checkpoints when determining error checkpoints. After grouping based on error checkpoints, there is no need to add checkpoints. In addition, determining one error checkpoint means that an additional workload execution is needed to determine whether the faults injected after the error checkpoint contain error faults, but the increase in the number of simulations is less than the decrease in the number of simulations compared to the decrease in the search range. The results in Table 3 show that the increased time in addition of checkpoints is less than the decreased time.

The speed improvement in the method with checkpoints is not significantly increased compared to the method without checkpoints because the number of the first layer faults is selected according to error probability. Therefore, the promotion after adding checkpoints depends on the probability that the error fault = 0 occurs. Table 6 shows the probabilities of different numbers of error faults.

Table 6. The probabilities of different numbers of error faults.

Injection Module	The Probability of Error Fault = 0	The Probability of Error Fault = 1	The Probability of Error Fault = 2	The Probability of Error Fault $\geq$ 3
Integer unit	45.13%	39.82%	9.73%	5.32%
Cache	51.30%	33.91%	12.17%	2.62%
Memory control module	60.55%	29.05%	8.26%	2.14%
Register-file	53.15%	28.83%	13.51%	4.51%

Table 3 shows that speedup depends on error probability and the lower the error probability, the greater the speed improvement. The integer unit has the highest error probability than other modules because the five-stage pipeline registers state of the Leon2 processor is mainly determined by the integer unit. The register-file module has the lowest error probability; it is related to the number of registers used by the test program.

In this study, we evaluated the Leon2 system by running the matrix multiplication program. The method proposed in this paper can be applied to other processors and programs. With the continuous progress in aerospace engineering, the error probability must be smaller and smaller, so the method in this paper has a strong application prospect.

4. Discussion

Although the mixture simulation method is adopted in this paper, the method proposed in this paper is suitable for any hierarchy, and it can also be used to improve the speed of system-level simulation in cases where the time cost is more concerned. In this paper, 21 checkpoints are added to one workload execution to determine error checkpoints. This is because the mixture simulation takes several minutes to complete one mixture workload execution. Adding 21 checkpoints will not increase the time particularly. However, the number of checkpoints needs to be reduced if the method is applied to system-level simulation because the system-level simulation time is shorter than the mixture simulation method. Adding too many checkpoints may reduce simulation efficiency.

The time interval between the two checkpoints proposed in this paper is determined by the time the processor completes one process from reading data to writing data, related to the test program. A more general method to select the time interval will be studied in the future. This method will be applied to multiple-bit upset (MBU) in future work.

5. Conclusions

Simulating fault injection using computer simulation techniques is an effective way to reflect the SEE in aerospace integrated circuits. This paper proposes a multi-point fault injection method to achieve a good acceleration effect. Further, this method does not make the accuracy decline. Simulation results based on the Leon2 system validated the correctness of the proposed approach and showed that the proposed method could achieve a speedup of 60.69× at most.

Author Contributions: Data curation, X.Z.; funding acquisition, D.C. and C.X.; methodology, X.Z.; Project administration, Y.L. and Y.Y.; writing—original draft, X.Z. and C.X.; writing—review and editing, C.X. and X.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Equipment Pre-research Project of China under Grant 80924020307, by the National Natural Science Foundation of China under Grant 62004146, by the China Postdoctoral Science Foundation funded project under Grant 2021M692498, by the Fundamental Research Funds for the Central Universities under Grant XJSJ23106, and by Science and Technology Projects in Guangzhou under Grant SL2022A04J00095.

Data Availability Statement: Not applicable.

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

References

1. Schwank, J.R.; Shaneyfelt, M.R.; Fleetwood, D.M.; Felix, J.A.; Dodd, P.E.; Paillet, P.; Ferlet-Cavrois, V. Radiation Effects in MOS Oxides. *IEEE Trans. Nucl. Sci.* **2008**, *55*, 1833–1853. [CrossRef]
2. Wang, Q.; Liu, H.; Wang, S.; Chen, S. TCAD Simulation of Single-Event-Transient Effects in L-Shaped Channel Tunneling Field-Effect Transistors. *IEEE Trans. Nucl. Sci.* **2018**, *65*, 2250–2259. [CrossRef]
3. Uemura, T.; Lee, S.; Monga, U.; Choi, J.; Lee, S.; Pae, S. Technology Scaling Trend of Soft Error Rate in Flip-Flops in 1 × nm Bulk FinFET Technology. *IEEE Trans. Nucl. Sci.* **2018**, *65*, 1255–1263. [CrossRef]
4. Bennett, W.G.; Schrimpf, R.D.; Hooten, N.C.; Reed, R.A.; Kauppila, J.S.; Weller, R.A.; Warren, K.M.; Mendenhall, M.H. Efficient Method for Estimating the Characteristics of Radiation-Induced Current Transients. *IEEE Trans. Nucl. Sci.* **2012**, *59*, 2704–2709. [CrossRef]
5. Wu, Z.; Chen, S. Heavy Ion, Proton, and Neutron Charge Deposition Analyses in Several Semiconductor Materials. *IEEE Trans. Nucl. Sci.* **2018**, *65*, 1791–1799. [CrossRef]
6. Lu, Z.; Zhou, W. Analysis of Single Event Effect on SRAM Based on TCAD Simulation. In Proceedings of the 2015 8th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 12–13 December 2015; pp. 588–591.
7. Borisov, A.Y.; Novikov, A.A.; Petrun, N.L.; Nefedova, A.A.; Chukov, G.V. The Optimal Estimation of Single Event Effects Sensitivity Parameters Using a Focused Laser Source and Heavy Ion Cyclotron on Example of a Library of Analog IP Units. In Proceedings of the 2021 International Siberian Conference on Control and Communications (SIBCON), Kazan, Russia, 13–15 May 2021; pp. 1–5.

8. Gasiot, G.; Abouzeid, F.; Malherbe, V.; Damiens, J.; Monsieur, F.; De Boissac, C.L.M.; Soussan, D.; Lorquet, V.; Thery, T.; Roche, P. New Single Event Transient phenomenon in 28FDSOI and its impact on hardening. In Proceedings of the 2019 19th European Conference on Radiation and Its Effects on Components and Systems (RADECS), Montpellier, France, 16–20 September 2019; pp. 1–5.
9. Truyen, D.; Montagner, L. New Approach of Single Event Latchup Modeling Based on TCAD Simulations and Design of Experiment Analysis. In Proceedings of the 2019 19th European Conference on Radiation and Its Effects on Components and Systems (RADECS), Montpellier, France, 16–20 September 2019; pp. 1–4.
10. Fleetwood, Z.E.; Lourenco, N.E.; Ildefonso, A.; Warner, J.H.; Wachter, M.T.; Hales, J.M.; Tzintzarov, G.N.; Roche, N.J.-H.; Khachatrian, A.; Buchner, S.P.; et al. Using TCAD Modeling to Compare Heavy-Ion and Laser-Induced Single Event Transients in SiGe HBTs. *IEEE Trans. Nucl. Sci.* **2017**, *64*, 398–405. [CrossRef]
11. Maiti, C.K.; Dash, T.P. Simulation of single event upset in power MOSFETs. In Proceedings of the 2017 Devices for Integrated Circuit (DevIC), Kalyani, India, 23–24 March 2017.
12. Ding, L.; Chen, W.; Wang, T.; Chen, R.; Luo, Y.; Zhang, F.; Pan, X.; Sun, H.; Chen, L. Modeling the Dependence of Single-Event Transients on Strike Location for Circuit-Level Simulation. *IEEE Trans. Nucl. Sci.* **2019**, *66*, 866–874. [CrossRef]
13. Shambhulingaiah, S.; Lieb, C.; Clark, L.T. Circuit Simulation-Based Validation of Flip-Flop Robustness to Multiple Node Charge Collection. *IEEE Trans. Nucl. Sci.* **2015**, *62*, 1577–1588. [CrossRef]
14. Andrianjohany, N.; Pourrouquet, P.; Coulié, K.; Chatry, N.; Rahajandraibe, W.; Standarovski, D.; Rolland, G.; Ecoffet, R. Prediction methodology of single event effect sensitivity and application on SRAM device. In Proceedings of the 2016 16th European Conference on Radiation and Its Effects on Components and Systems (RADECS), Bremen, Germany, 19–23 September 2016; pp. 1–7.
15. Mansour, W.; Velazco, R. SEU fault-injection in VHDL-based processors: A case study. In Proceedings of the 2012 13th Latin American Test Workshop (LATW), Quito, Ecuador, 10–13 April 2012; pp. 1–5.
16. Cannon, M.; Rodrigues, A.; Black, D.; Black, J.; Bustamante, L.; Breeding, M.; Feinberg, B.; Skoufis, M.; Quinn, H.; Clark, L.T.; et al. Multiscale System Modeling of Single-Event-Induced Faults in Advanced Node Processors. *IEEE Trans. Nucl. Sci.* **2021**, *68*, 980–990. [CrossRef]
17. Champon, R.; Beroulle, V.; Papadimitriou, A.; Hély, D.; Genévrier, G.; Cézilly, F. Comparison of RTL fault models for the robustness evaluation of aerospace FPGA devices. In Proceedings of the 2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS), Sant Feliu de Guixols, Spain, 4–6 July 2016; pp. 23–24.
18. Li, Z.B.; Liu, Y.; Xu, C.Q.; Song, L.T.; Wu, H.P. SRAM Single Event Effect Simulation Method Based on Random Injection Mechanism. In Proceedings of the 2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT), Qingdao, China, 31 October–3 November 2018; pp. 1–3.
19. Leveugle, R.; Calvez, A.; Maistri, P.; Vanhauwaert, P. Statistical fault injection: Quantified error and confidence. In Proceedings of the 2009 Design, Automation & Test in Europe Conference & Exhibition, Nice, France, 20–24 April 2009; pp. 502–506.
20. Reick, K.; Sanda, P.N.; Swaney, S.; Kellington, J.W.; Mack, M.; Floyd, M.; Henderson, D. Fault-Tolerant Design of the IBM Power6 Microprocessor. *IEEE Micro* **2008**, *28*, 30–38. [CrossRef]
21. Cher, C.Y.; Muller, K.P.; Haring, R.A.; Satterfield, D.L.; Musta, T.E.; Gooding, T.M.; Davis, K.D.; Dombrowa, M.B.; Kopcsay, G.V.; Senger, R.M.; et al. Soft error resiliency characterization and improvement on IBM BlueGene/Q processor using accelerated proton irradiation. In Proceedings of the 2014 International Test Conference, Seattle, WA, USA, 20–23 October 2014; pp. 1–6.
22. Meng, X.; Tan, Q.; Shao, Z.; Zhang, N.; Xu, J.; Zhang, H. SEInjector: A Dynamic Fault Injection Tool for Soft Errors on X86. In Proceedings of the 2017 International Conference on Computer Systems, Electronics, and Control (ICCSEC), Dalian, China, 25–27 December 2017; pp. 1492–1495.
23. Sangchoolie, B.; Johansson, R.; Karlsson, J. Light-Weight Techniques for Improving the Controllability and Efficiency of ISA-Level Fault Injection Tools. In Proceedings of the 2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC), Christchurch, New Zealand, 22–25 January 2017; pp. 68–77.
24. Pusz, O.; Kiechle, D.; Dietrich, C.; Lohmann, D. Program-Structure-Guided Approximation of Large Fault Spaces. In Proceedings of the 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC), Kyoto, Japan, 1–3 December 2019; pp. 138–13809.
25. Dietrich, C.; Schmider, A.; Pusz, O.; Vaya, G.P.; Lohmann, D. Cross-Layer Fault-Space Pruning for Hardware-Assisted Fault Injection. In Proceedings of the 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 24–28 June 2018; pp. 1–6.
26. Ubar, R.; Jürimägi, L.; Orasson, E.; Raik, J. Scalable algorithm for structural fault collapsing in digital circuits. In Proceedings of the 2015 IFIP/IEEE International Conference on Very Large-Scale Integration (VLSI-SoC), Daejeon, South Korea, 5–7 October 2015; pp. 171–176.
27. Hari, S.K.S.; Adve, S.V.; Naeimi, H.; Ramachandran, P. Relyzer: Application Resiliency Analyzer for Transient Faults. *IEEE Micro* **2013**, *33*, 58–66. [CrossRef]

28. Chen, Q.; Liu, Y.; Wu, Z.; Liao, J. A Single Event Effect Simulation Method for RISC-V Processor. In Proceedings of the 2021 IEEE 15th International Conference on Anti-counterfeiting, Security, and Identification (ASID), Xiamen, China, 29–30 December 2021; pp. 106–110.
29. Valderas, M.G.; Peronnard, P.; Ongil, C.L.; Ecoffet, R.; Bezerra, F.; Velazco, R. Two Complementary Approaches for Studying the Effects of SEUs on Digital Processors. *IEEE Trans. Nucl. Sci.* **2007**, *54*, 924–928. [CrossRef]