Computer
& Robotics

Soft Error Rate Estimation of Logic Circuits Using Recurrent Neural Networks

Rasoul Farjaminezhad [a], Saeed Safari [b,*], Amir Masood Eftekhari Moghadam [c]

[a] *Faculty of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*

[b] *College of Engineering, School of Electrical and Computer Engineering, Tehran University, Iran*

[c] *Faculty of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*

Abstract

Nano-scale technology has brought more susceptibility to soft errors for the generation of complicated and state-of-the-art devices. These soft errors are the impacts of radiation of the particles like a neutron, alpha, and ions on the surface of the circuits. To tackle the system malfunctions and provide a reliable device, studying the transient fault effects on the logic circuits can be a more significant issue. This paper presents a new approach based on Recurrent Neural Networks (RNNs) to estimate ICs' Soft Errors Rate (SER). As RNN can be deployed for signal processing and time series, we applied it to investigate transient fault effects while propagating through the combinational and sequential parts of a test chip with computing its SER by simulating and analyzing the circuit outputs. In this paper, the proposed RNN model's use to estimate the SER of ISCAS-85 standard circuits is investigated.

***Keywords**: Recurrent neural networks, Circuit modeling, Transient fault, Soft error rate.*

1.Introduction

Since technologies related to the VLSI circuits downscaled, the circuits' susceptibility to transient faults, induced by high energy particle strikes, has increased [1]. When a particle hits a transistor, a charge is disposed of which can cause a transient voltage pulse to be created at the output of the related component [1]. The pulse can result from single/multiple event transient (SET/MET) faults. The transient faults can propagate through the circuits, and if masking factors do not mask them, they are finally liable to be observed in the circuit outputs or latched in the memory elements. When a bit of the memory is flipped by the transient faults, the soft error occurs. Due to the increase in the circuit vulnerability against the soft errors [1] [2] [3], it is valuable for an oriented method to be proposed to tackle the challenge of soft error rate computation. This is beneficial particularly for complicated and very large-scale circuits. Since HSPICE is commonly devoted to analyzing circuits, it takes a long simulation time for devices with a high quantity of components [14]. Therefore, in this paper, we have presented a method that relied on RNN to model and obtained the SER of an under-investigated electronic chip. The proposed RNN can learn the electronic element's behavior, like logic gates, and predict changes of the glitches that might occur and propagate through a circuit. Moreover, the model considers triple masking effects as well as re-convergent path effects to provide an accurate estimation of SER. By making RNN models of

* Corresponding authors: Email: saeed@ut.ac.ir

the base gates and training them by the data obtained from their simulations in the HSPICE, we could predict the injected signals' output shape. Then, the SER has been computed by running a simulation based on the circuit layout and injected faults. To verify the proposed model, we calculated the SER for the small circuits like C17 and Invchain and compared the results with the HSPICE outputs. The results demonstrate 12.08x and 15.65x speed-up with the %4.49 and %5.27 penalty accuracy. Additionally, we applied the proposed RNN method to compute the SER of the ISCAS-85 benchmark circuits.

The rest of the paper is organized as follows: Section 2 provides related works. Section 3 describes the methodology of RNN deployed for analyzing transient faults. Section 4 represents some sample circuits and illustrates the results of utilizing RNN for modeling them. Moreover, this section provides a prediction of SER for the ISCAS-85 benchmark circuits. Finally, section 5 concludes the paper.

2.Literature Review

To estimate a device's vulnerability against soft errors, occasionally, two types of software-based fault injection techniques are applied [4]. In one category, software like HSPICE is deployed to set up glitches and compute the rate of the faults observed at the output of an under-test unit [4], and the other applies mathematical and probabilistic techniques to the model and approximates the rate of the soft errors [3]. Dynamic and static methods are the names given to these two strategies [5]. In the dynamic technique, some nodes of a circuit are affected by a set of transient faults; then, the circuit is simulated for a variety of input vectors to obtain the related SER. On the other hand, in the analytical process, models based on mathematical equations are provided to calculate the rate of soft errors stemming from transient faults [5]. Each of the techniques has advantages and drawbacks. For instance, in some cases of analytical estimation of SER, there is a 7x overestimation in SER analysis [3]. Moreover, most dynamic techniques have not considered all masking factors in their approaches or even demand for large input vectors and have a significant challenge in dealing with the re-convergent paths [6]. Thus, maybe they have a time overhead problem during a circuit simulation [6]. Given that the proposed RNN model is based on circuit simulations in the neural network environments, we mostly concentrate on dynamic investigations. In addition, [6] produces a framework, named MASKit, which computes SER of the circuits using signal probabilities, heuristic simulation techniques, and backward-traversing algorithm. Compared to the traditional fault injection-based methods, like Monte Carlo, the model is twice faster with the low time overhead. Moreover, it provides %96 accuracy in SER estimating. In [7], to achieve the SER, logical and flip-flop descriptions are applied. In this approach, a complicated signal is considered as several pulses that are independent. It has been done due to the challenges of re-convergent paths leading to a remarkable loss of accuracy in evaluating transient fault effects. In [8], initially, by making gate simulations for a variety of input vectors, the SER of each gate of a circuit is computed. Then, the soft error rate for circuit paths (from the gate to the memory elements) is obtained. The circuit SER is calculated by combining the SER of some paths to the nearest one. Reference [4] deployed HSPICE tools to extract transient pulse width probability distributions and compute device SER. In [9] graph theory and HSPICE simulation are applied to analyze SER of combinational circuits and memory elements.

Furthermore, the reference performs heuristic algorithms to reduce simulation time. Reference [10] has utilized random input vectors based on the simulation method to compute the soft error rate for a given circuit. In the end, there are plenty of techniques proposed in diverse references to address the SER computation challenge, which has been ignored due to paper constraints.

However, since circuit simulation tools like HSPICE demand a long simulation time to analyze SER of complicated devices [1], we proposed RNN based method for evaluating circuits and obtaining their SER. Training RNN units perform this by the transient faults' characteristics, attitudes, and behavior that occurred in the electronic component. Providing fast and accurate analysis of a circuit and considerations of other constraints like performance, area, and the supply voltage can help designers adopt powerful optimized techniques to decline overall SER. This can be achieved by applying the introduced RNN model process. In the next section, the methodology of deploying the RNN technique will be entirely detailed.

3. Methodology

Extensive usage of artificial neural networks, like RNNs, in fields of machine translation, speech recognition, natural language processing, time series prediction, and many other areas [11], gave us enough incentive to handle RNN for estimating a circuit SER. By the reference that there are numerous approaches to manipulating neural networks and machine learning in the system reliability evaluations [2], we proposed a method that exploited RNN to estimate SER of a circuit by learning interactions of the circuit elements against transient faults. To develop the idea, we generated RNN models of some basic gates like NOT, AND, NAND, NOR, etc. The network training data has been obtained by simulating circuits involving a chain of gates in the HSPICE environment. The extracted data changed to a time-discrete one for feeding to the RNN. During the learning process, the network learns how the glitches at the output will be when passing through a certain gate. Therefore, attributes of the faults like width, amplitude, and delay will be estimated. Once the glitches' attenuation was determined, it is possible to evaluate the rate of faults reaching the circuit outputs or latching in the memory elements. As a transient pulse propagates through the logic gates, it is attenuated by width and height. Thus, it is more likely for an attenuated pulse to be prohibited from propagating due to a lack of power. Therefore, if the power of a pulse is less than a certain value, it will be masked based on the electrical making effect. Besides, if the transient pulse occurs in a non-controlling input of a gate, it would still be masked by a logical masking factor. For instance, the output value of a 2-input AND gate consistently is zero when at least one of the inputs stays on zero. Even if a transient fault can pass through the mentioned masks, another masking part can be eliminated named the latching window. This factor prevents glitches to be captured in the memory elements. The cover item emerges when the fault arrives at the input of a memory component out of the clock cycle set up and hold time [5]. Fig 1 presents the timing mask effect in both masked and latched Single Event Transient (SET).

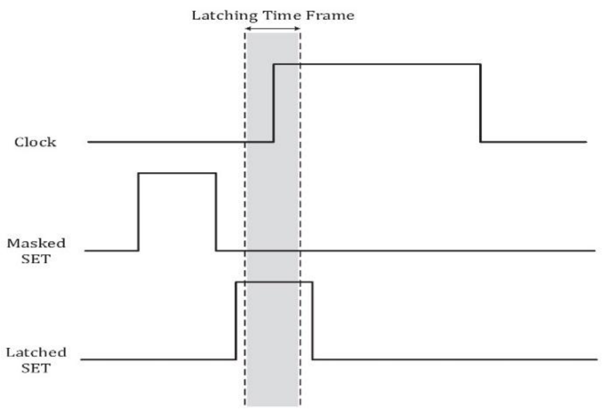

Fig 1. Timing mask effects on the SETs [5].

The proposed RNN model has been constructed by investigating the influences of high energy particle-induced transient faults in the base logic circuits. The process carried on by modeling a transient pulse with an independent current resource, formulated in Eq.) and described in [12].

$$I(t) = \frac{Q_{coll}}{\tau_{fall}-\tau_{rise}}\left\{\exp\left(\frac{t}{\tau_{fall}}\right) - \exp(\frac{t}{\tau_{rise}})\right\} \quad (1)$$

Where $\boldsymbol{Q_{coll}}, \boldsymbol{\tau_{fall}}, \boldsymbol{\tau_{rise}}$ are collected charges stemming from high energy particle strikes, rising, and falling times, respectively [12]. A voltage waveform can be generated by injecting a current pulse into the electrical component like a logic gate if the pulse has met desired magnitude and width conditions [1]. We gathered data of the gates by injecting various current pulses into the inputs of the gates and tracking their behavior during simulation in HSPICE. Once the information is obtained, they are converted to the discrete-time value feeding as datasets to the RNN, later. Indeed, data achieved from the gate's simulations compose supervised RNN training, evaluating, and testing data. Then, the network can estimate fault distributions of a device by learning the gate responses to the different voltage pulse inputs. This constructs the RNN model of a gate. Once the models of electrical components are created, it enables us to evaluate the SER. The circuit applied to obtain a gate attitude contains a

design of connecting a chain of the gate. For instance, to create the RNN model of AND gate, initially, a circuit involves a number of the gate, connected consecutively, designed, and simulated in the HSPICE for a variety of fault signals, obtained by changing parameters of Eq. .
Then, the simulation data are extracted and normalized to feed to the RNN. Finally, the RNN is trained, evaluated, and tested by the data. If the results of the testing are in an acceptable range of accuracy, then the structure of trained RNN, containing internal parameters like link weights, is stored as the AND gate RNN model. By performing the process for other logic gates, it is possible to achieve the architectures and attributes of non-linear electronic elements. Then by applying the models and creating software, based on the circuit layout, one can analyze an under-test device for a fault effect and SER estimation. We verified the models for some small circuits by computing SER and comparing the RNN outputs and HSPICE simulation results. Moreover, we developed our experiments for other complicated devices like ISCAS-85 benchmark circuits to demonstrate proposed model scalability. The structure of method and training strategies are explained as follows.

3.1. Descriptions of Proposed RNN Model

The architecture of described RNN model is depicted in Fig. 2. As the picture illustrates, the model is composed of the input layer, two hidden layers, and one output layer. Variables $X(t)$ and $Y(t)$ displays primitive waveforms of the input and output, obtained from the component simulations in HSPICE. The proposed network accepts an input vector that contains a history of previous input and output signals.

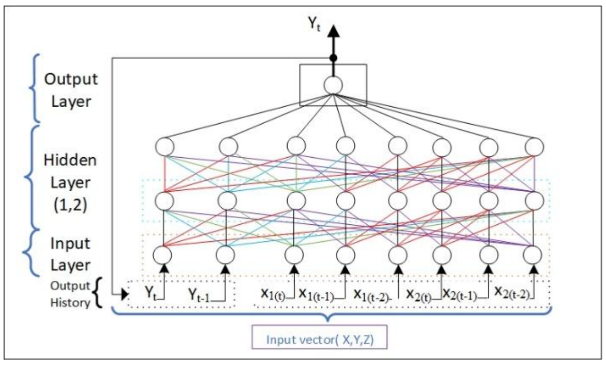

Fig. 2. Diagram of supposed RNN model

The outputs are fed back to the input by the straight connections. While t shows a device simulation time, the network output is denoted by $\hat{Y}(t)$, which is relied on $i)$ current state inputs, $X(t-1), X(t-2), \dots, X(t-p)$ where p is the steps of the input histories, $ii)$ output archive signals, $\hat{Y}(t-1), \hat{Y}(t-2), \dots, \hat{Y}(t-q)$ where q displays the buffered outputs, and $iii)$ RNN internal parameters, like weights supposed as

Table 1 descriptions and denoted by $\phi = [B^1 W^{1,2} B^2 W^{2,3} B^3]$. The N_l variable of the table shows the number of neurons applied in layer l computed by Eq.

).

$$N_l = (p+1)N_{input} + qN_{Output} \Rightarrow N_l = 3N_{input} + 2N_{Output} \tag{2}$$

Table 1
Parameters of proposed RNN model

Description	Vector
Weights between layer m, l	$w^{m,l} = [w^{m,l}(1)\ w^{m,l}(2) \dots w^{m,l}(N_m)]$
Weights of neurons	$w^{m,l}(x) = [w^{m,l}_{x,1}, \dots, w^{m,l}_{x,N_l}]$
Biases of layer l	bl= [b1l b2l… b8l]
Output shape	Based on simulation time
Number of neurons in the input layer	8
Number of neurons in the hidden layer	32

WhereN_{input}, N_{output} represent the quantity of input and output neurons. The output value of the network can be formulated by Eq. (3).

$$\hat{Y}(t) = f_{RNN}\big(\hat{Y}(t-1), \hat{Y}(t-2), \dots, \hat{Y}(t-p), X(t-1), X(t-2), \dots, X(t-q), \phi\big) \tag{3}$$

$$\boldsymbol{if\ p = q = 2} \ \Rightarrow$$
$$\hat{Y}(t) = f_{RNN}(\hat{Y}(t-1), \hat{Y}(t-2), X(t-1), X(t-\ 2), \phi)$$

Where f_{RNN} denotes the function that generates the output.

3.2. Training and Testing Phases

In this subsection, the process of training and evaluation as well as testing for the RNN, designed based on previous descriptions, will be explained. By computing error rates for all of the datasets, the network performance is evaluated. Computing the optimal value of the RNN internal parameters is the training phase's goal until achieving a minimum error value. Associated training data are presented by $(X_n(t), Y_n(t))$ as an original signal, where n is the index of n^{th} training sample. When the network error is less than 10e-4, the process of training ends. The optimization process is considered as Eq. Error! Reference source not found.).

$$\min_{\Phi} \frac{1}{2N_m} \sum_{n=1}^{N_m} \sum_{t=1}^{N_t} (Y_n(t) - \hat{Y}_n(t))^2 \tag{4}$$

Where N_m, N_t respectively represent the total number of dataset samples (n=1, 2...N_m), and a circuit simulation time like how it is executed in HSPICE. The equation related to error computation, for the proposed RNN, is formulated as Eq. Error! Reference source not found.).

$$E(k) = \frac{1}{N_t} \sum_{t=1}^{N_t} (e_t(k))^2 \tag{5}$$

Where

$$e_t(k) = Y_t(k) - \hat{Y}_t(k) \tag{6}$$

Eq. (6) gives the error of k^{th}sample of a test signal at the point *t*. To make it feasible for selecting an optimal value of training rate, we considered Adam optimizer elaborated in [13]. The iteration of each training model is supposed to be 3000. In addition, 1000 epochs are fed to the network in every repetition. As mentioned, the input vector contains a buffer of prior steps of loaded and result waveforms. The hidden layers include the activation function of the log-sigmoid, $f(x) = \frac{1}{1+\exp(-x)}$,. Moreover, the activation function of input and output layers is linear. Details of training and testing phases of some basic gates are provided in Table 2.

Table 2

Details of training and testing RNN model of the basic gates

Gate Type	Mean error of total test samples	Max Error	Number of training samples	Number of testing samples	Mean Learning Time (Hour)
AND	0.7170	2.3762	1552	388	0.45
NAND	2.4132	3.8260	1972	493	0.75
NOT	1.61790	3.8387	564	141	0.5
NOR	2.0112	2.8240	2316	579	0.79

As the table illustrates, the test datasets' mean and maximum errors are displayed in the columns 2 and 3. Whereas the error of each signal is computed by Eq. Error! Reference source not found. and Eq. **(6)**, the mean error of total testing samples are denoted in column 2, achieved through Eq. Error! Reference source not found.. Indeed, each waveform's error is calculated based on Eq. Error! Reference source not found., Eq. **(6)**, and mean squared error of point-to-point comparison of the RNN results and original signals. The circuit simulation was performed in the HSPICE environment for each chip separately. In this approach, the value of time is considered to be 500 picoseconds. As mentioned above, the process is performed for each type of the base gates, with single-cycle simulations and diverse values of pulse widths and amplitude, varying by changing $\boldsymbol{\tau_{fall}}, \boldsymbol{\tau_{rise}}$, and $\boldsymbol{Q_{coll}}$ parameters. At the end of the HSPICE simulation, we attained $\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{T}})$. It is a sequence of data corresponding to the gate $\boldsymbol{\mathcal{G}}$ with inputs $\boldsymbol{\mathcal{X}}$, outputs $\boldsymbol{\mathcal{Y}}$, and $\boldsymbol{\mathcal{T}}$as the interval duration of simulation time. By constructing the RNN models of a circuit, based on its gate-level layout, and simulating the circuit in a neural network platform, like Python, we analyzed the SER of the circuits. This is done by using a variety of faults injected into the device nodes and evaluating the portions of transient glitches captured by the memory elements.

To further explore the described model and evaluate its performance for assorted devices, in the next section, we have provided an estimation of SER for a variety of circuits. Initially, SER of some small chips like C17 and Invchain are computed; then, to examine the model's scalability, we applied designed RNN to estimate the SER of ISCAS-85 benchmark circuits.

4. Modelling Instances and Results

When the RNN models of simple components are generated, they can be exploited to estimate a circuit SER. This is performed by analyzing transient fault impacts on a circuit while passing through it. To explain the proposed model, SER of diverse circuits were computed and tested by injecting random pulses in their nodes and tracking the number of faults causing erroneous outputs or bit flips in the memory elements. At first, the SER of two small designs C17 and Invchain, presented in

Fig **3** and Fig 4, are calculated. Then, the models have been applied to estimate the SER of complicated ISCAS-85 benchmark circuits. For the first test benches, random pulses are injected into the arbitrarily selected nodes of the circuits. The faults were injected by the current pulses with the charge of Qt varying boundary of 100fc to 200fc, as well as rise and fall times with the values of 1ns. Then the circuits are investigated and examined for the error occurred which are observed in their output. Furthermore, to compute the SER of the circuits that contain sequential parts, the state of the memory elements is checked for the bit flips. The error rate is computed by the division of observed errors in the circuit output to a total number of injected faults. For each of the two mentioned small circuits, the process of fault injection was repeated 1000 times in HSPICE and RNN environments, separately. Then, the attained errors are applied to compute the SERs. The results of the simulation are compared in table 3 **Table 3** By producing some transient current pulses and injecting them into random nodes of the circuits, for different values of the inputs, the rate of the errors observed in the circuit outputs can be obtained.

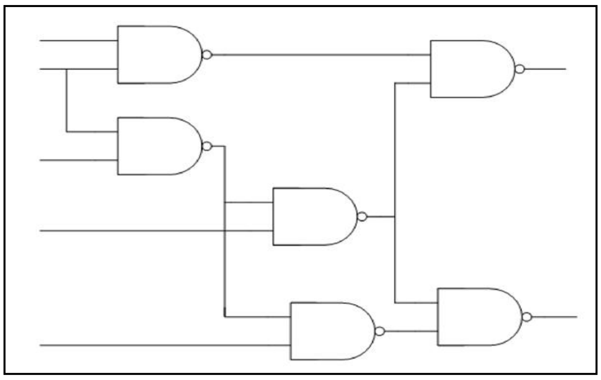
Fig 3. Schematic of C17 circuit [15].

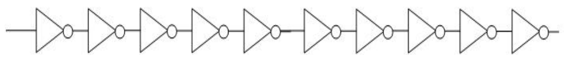
Fig 4. Block diagram of Invchain circuit

Table 3

Soft error rate comparison HSPICE

Circuit	HSPICE computed SER	RNN computed SER	Error	HSPICE time(s)	RNN time(s)	Speed-up
C17	0.1489	0.1556	%4.49	5111	423	12.08x
In chain	0.0816	0.0859	%5.27	3240	207	15.65x

As table 3 illustrates, for the listed C17 and Invchain circuits, the SER computed by RNN models are respectively 12.08 and 15.65 times faster than the HSPICE simulator. Additionally, compared to the HSPICE results, there is a %4.49 and %5.27 loss of accuracy in SER estimation by the RNN method which is negligible. Because, for the complicated circuits, the RNN can have an estimation in an acceptable time, while HSPICE takes a long duration for simulating a complex device. Therefore, applying the RNN model for the state of the art components can be remarkable. Column 5, 6 of the table displays the time for 1000 iterations. To present the RNN model scalability, we did the SER estimation process for the ISCAS-85 circuits. All the benchmarks applied were much bigger than the small chips. Hence, HSPICE could not be applied for verification of RNN results since it takes a very long simulation time. All considerations were like the small ones which are executed on Intel(R) Core (TM) i7 with 8-Gbyte RAM system. The platform utilized to run RNN programs was the PyCharm community edition 2020.1.1 x64 version. Also, Hspui H-2013.03-SP2 was the version of HSPICE applied to compute circuits SER in the HSPICE environment.

Table 4 displays details of parameters considered to design circuits in HSPICE. Additionally, Table 5 shows estimated values of SER for the ISCAS-85 benchmark circuits as well as the time is taken for 1000 iterations.

Table 4

Parameters considered in the HSPICE simulator

Parameter	Value
NMOS Transistors width	0.210U
NMOS Transistors length	0.050U
PMOS Transistors width	0.315U
PMOS Transistors length	0.050U
Vdd	0.6 v
Temperature	25^{o}c
Lmin	45nm

Table 5

SER for ISCAS-85 benchmark circuits

Circuit	SER	Time(s)
C432	0.159	1552
C499	0.51	1959
C880	1.89	3715
C1908	2.21	6213
C2670	3.62	8536
C3540	1.59	10572
C5315	7.38	12189
C6288	2.30	14378
C7552	3.93	15338

4. Conclusion

In this paper, we have proposed a new paradigm based on RNN to estimate the SER of logic circuits. Designed RNN models take the gate-level layout of a circuit and compute the SER of the circuit for given transient faults. We verified the method using HSPICE simulation for small devices with random fault injections to arbitrary nodes. To demonstrate the proposed RNN model scalability, we applied it for ISCAS-89 benchmark circuits. The results and time required to predict the SER of the circuits are provided in the paper. Consequently, as the HSPICE simulations for the complicated circuits take a long time, RNN can be an acceptable alternative method to analyzing and predicting circuits SER. Finally, since the RNN model was produced using Python and Tensor flow scripts, it can provide many optimizations, like applying GPU microprocessors, to make the method much faster and efficient than the current status.

References

[1] G. I. Paliaroutis, P. Tsoumanis, N. Evmorfopoulos, G. Dimitriou and G. I. Stamoulis, "A Placement-Aware Soft Error Rate Estimation of Combinational Circuits for Multiple Transient Faults in CMOS Technology," in 2018 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Chicago, IL, USA, 8-10 Oct. 2018.

[2] A. Balakrishnan, T. Lange, M. Glorieux, D. Alexandrescu and M. Jenihhin, "Composing Graph Theory and Deep Neural Networks to Evaluate SEU Type Soft Error Effects," in 2020 9th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, Montenegro, 8-11 June 2020.

[3] S. Mukherjee, C. Weaver, J. Emer, S. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36, San Diego, CA, USA, USA, 5-5 Dec. 2003.

[4] Y.-H. Kuo, H.-K. Peng and C. H.-P. Wen, "Accurate statistical soft error rate (SSER) analysis using a quasi-Monte Carlo framework with quality cell models," in 2010 11th International Symposium on Quality Electronic Design (ISQED), San Jose, CA, USA, 22-24 March 2010.

[5] Mohammad Eslamia, Behnam Ghavamia, Mohsen Rajib and AliMahani, "A survey on fault injection methods of digital integrated circuits," ScienceDirect, Integration, the VLSI Journal, vol. 71, pp. 154-163, 11 November 2019.

[6] M. Anglada, R. Canal, J. L. Aragón and A. González, "MASkIt: Soft error rate estimation for combinational circuits," in 2016 IEEE 34th International Conference on Computer Design (ICCD), Scottsdale, AZ, USA, 2-5 Oct. 2016.

[7] Y. Dhillon, A. Diril and A. Chatterjee, "Soft-error tolerance analysis and optimization of nanometer circuits," in Design, Automation, and Test in Europe, Munich, Germany, Germany, 7-11 March 2005.

[8] J. F. Ziegler, H. W. Curtis, H. P. Muhlfeld, C. J. Montrose, B. Chin, M. Nicewicz, C. A. Russell, W. Y. Wang, L. B. Freeman, P. Hosier, L. E. LaFave, J. L. Walsh, J. M. Orro, G. J. Unger, J. M. Ross, T. J. O'Gorman, B. Messina, T. D. Sullivan, A. J. Sykes, H. Yorke, and T. A. E, "IBM experiments in soft fail in computer electronics (1978–1994)," IBM Journal of Research and Development, vol. 40, no. 1, pp. 3 - 18, Jan. 1996.

[9] M. Zhang and N. Shanbhag, "Soft-Error-Rate-Analysis (SERA) Methodology," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 25, no. 10, pp. 2140 - 2155, Oct. 2006.

[10] P. Dodd and L. Massengill, "Basic mechanisms and modeling of single-event upset in digital microelectronics," IEEE Transactions on Nuclear Science, vol. 50, no. 3, pp. 583 - 602, June 2003.

[11] Z. Naghibi, S. A. Sadrossadat, and S. Safari, "Time-domain modeling of non-linear circuits using deep recurrent neural network technique," AEU - International Journal of Electronics and Communications, vol. 100, pp. 66-74, February 2019.

[12] G. Srinivasan, P. Murley and H. Tang, "Accurate, predictive modeling of soft error rate due to cosmic rays and chip alpha radiation," in Proceedings of 1994 IEEE International Reliability Physics Symposium, San Jose, CA, USA, USA, 11-14 April 1994.

[13] D. P. Kingma and J. L. Ba, "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION," in 3rd International Conference for Learning Representations (ICLR), San Diego, 2015.

[15] https://www.researchgate.net/figure/C17-Benchmark-Circuit_fig2_304670382.