International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 7, No.2 (Mar-2018)

http://dx.doi.org/10.12785/ijcds/070201

Transient and Permanent Adaptive Fault Classifier for FPGA Applications in the Space

Radwa M. Tawfeek[1], Yousra Alkabani[2], Mohamed G. Egila[3] and I. M. Hafez[2]

[1] *Benha Faculty of Engineering, Benha University, Benha, Egypt*
[2] *Faculty of Engineering, Ain Shams University, Cairo, Egypt*
[3] *Electronics Research Institute, Cairo, Egypt*

Received 3 Aug. 2017, Revised 17 Oct. 2017, Accepted 28 Nov. 2017, Published 1 Mar. 2018

Abstract: Field Programmable Gate Array devices (FPGAs) are used in many applications. FPGAs are subjected to faults especially in the space where they are subjected to radiation. Faults in FPGAs may be recoverable (transient) or non-recoverable (permanent). Recoverable faults can be resolved by reconfiguring the system, on the other hand non-recoverable faults, require the relocation of the logic to a non-faulty area. This paper proposes an adaptive fault classifier that can be used to differentiate fault types. The classification guides the system to use the suitable recovery strategy. Experimental results show the operation of the classifier and adaptation layers, the proposed classifier layer is smaller in area compared to previous work. The adaptation layer works satisfactory to cope with environment changes. Both of the layers work independently of the design on any other layer.

Keywords: Fault- tolerance, Fault types, Adaptive Fault Classifier, FPGA faults

1. INTRODUCTION

There is a great interest in using FPGAs in systems intended to operate in space. This interest in FPGAs has increased due to the advantages of programmability, flexibility, and high-performance capabilities. Radiation in space environment causes many faults. Faults are classified as permanent and temporary. Temporary faults may be transient or intermittent. Transient faults are temporary disturbances caused by environmental conditions such as electromagnetic interference, injection of neutrons and alpha particles, power supply, electrostatic discharge and interconnect noises. These faults are called soft errors since they do not cause any permanent damage. Temporary faults are measured by probability of error occurrence known as Soft Error Rate (SER). Single Event Effects (SEE) is an example of a transient fault. SEE are produced when extra currents flow through a semiconductor device. Based upon the type of radiations causing the effect, SEE can be categorized as Single Event Upsets (SEU) and Single Event Transients (SET). These effects cause errors in the logic function of the FPGA, and may remain until the configuration memory is refreshed.

On the other hand, intermittent faults are those faults which occur because of the presence of any unstable or marginal hardware. They are usually activated by higher temperature or voltage. If the effect of these temporary faults continues for a long duration, it may lead to permanent faults. The manufacturing faults or the physical defects are known as the permanent faults. The permanent failures occur either during the manufacturing process due to small manufacturing variations that are not detected during the production testing, or during the operation. These faults become effective during the lifetime operation of the device, and cause the aging of the device affecting the operation of the device. Phenomena such as Electro-Migration (EM) and Time Dependent Dielectric Breakdown (TDDB), are the main causes for permanent faults. These phenomena occur due to reasons such as increased gate field strength, higher current density, smaller feature size, thinner gate dielectrics, and increasing variability [1] [2] . Fault Causes and classification are summarized in Figure 1.

E-mail address: radwa.tawfeek@bhit.bu.edu.eg, yousra.alkabani@eng.asu.edu.eg, mohamed.gamal@eri.sci.eg ,ismail_hafez@eng.asu.edu.eg

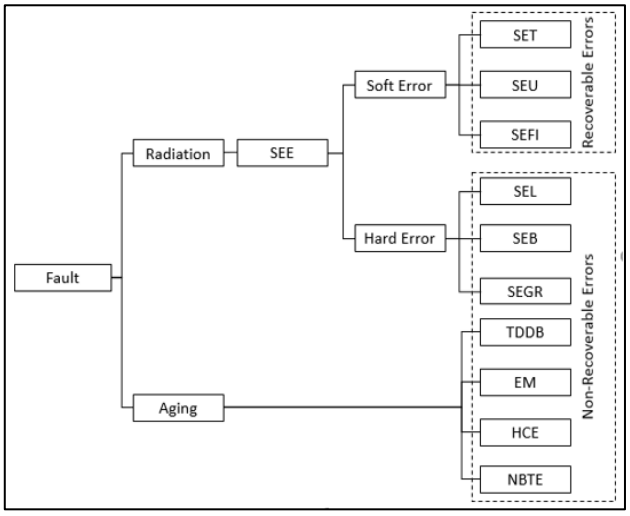

Figure 1. Fault Causes and Classification.

SRAM-based FPGAs encompass a configuration memory layer which stores the configuration (bitstream) of the FPGA in SRAM memory cells that defines the functionality performed by the FPGA, and a user logic layer where the actual circuit design is implemented and the application data being processed are stored [3] [4] . If a particle strikes the FPGA, it may affect memory resources that include the configuration memory and the user application data. Upsets are faults that may cause a failure.

The system is fortunately recovered from such failures by updating the memory cells with the correct values [4]. Fault Models in FPGA may be:

- Fault in Configuration memory: A SEU in the configuration memory can change the logic implemented on the FPGA and hence alters the function and goals of the circuit.
- A fault in a user flip-flop may cause a failure in its stored value that is used by subsequent circuitry. The failure can be measured at the output if it is propagated through the system, although it is often a transient failure. If the failure is trapped in a feedback loop the logic must be reset to an initial state.

Many researches [5] [6] [7] [8] [9] [10] [11] are concerned in the detection, localizing and recovering from the errors. Recovery is done usually by reconfiguring the FPGA totally or partially to correct the errors. But when the fault is non-recoverable, reconfiguration will do nothing since part of the FPGA is physically damaged and will cause errors again, in this case relocation of the circuit's logic is the only solution. Permanent faults in an FPGA can be repaired if there are enough fault-free elements on the FPGA so that designs can avoid using faulty elements. Few researches [12] [13] [14] are concerned with the classification of the faults types and whether they are recoverable or not.

Some of the researches used simple algorithm by considering the first fault as a transient fault, and the second consecutive to be permanent. Others used more sophisticated algorithms that measure the Mean Time Between Failures (MTBF) for every fault type and compare the MTBF for the occurred fault in the application to determine its type [12-16]. All the previous work use fixed data in their experiments and do not cope with the environmental changes.

In this research, we develop an algorithm to distinguish between fault types whether they are permanent or transient depending on number of consecutive faults that occur in the same area. The system adapts itself to any changes in the surrounding environment by changing the value of this number. The paper is structured as follows: Section II represents the related work, Section III describes the problem definition and motivation of the research, Section IV explains the proposed algorithm, Section V explores the experimental results and comparison with related work, and finally Section VI concludes the work.

2. RELATED WORK

There has been recently a large bulk of research in the field of tolerating and preventing faults in FPGA. Due to the advance in the technology, this topic has a lot of challenges. Researchers study different mitigation techniques searching for the best techniques for fault tolerance that are either application dependent or not. Most of these works are concerned with the detection, localization and recovery from errors. Few of them focus on classifying the fault to know whether it is recoverable or not. A recoverable fault can be fixed using reconfiguration, whereas, a non-recoverable fault requires relocation instead of just reconfiguration.

Yang studied the permanent effect of transient faults in asynchronous machines design [15]. For transient faults Wegrzyn and Sosnowski investigated faults in the configuration memory. They tracked the fault in multilevel, and developed a fault injector simulator for this purpose [16]. Dumitriu et al. developed a method that tolerates both transient and permanent faults using relocation. This approach limits the mitigation time to a single known quantity (the relocation time) and allows for the mitigation of both transient and permanent faults via the same process flow [1].

For classifying transient and permanent errors, some researches Yu and McCluskey, Bezerra et al. assumed that the first fault in an area is considered as recoverable fault, but if another fault in the same area occurred it will be considered as permanent [12] [17]. If the second fault is recoverable, this assumption can waste resources, since permanent fault will require allocating the circuit to another part in the FPGA, and consider the original part as faulty and never uses it again.

Pontarelli et al. calculate the time between any two consecutive faults, and compare this time with the MTBF. If the time calculated is less than the MTBF, it is considered as permanent fault; otherwise it is considered as transient fault [18]. Bolchini and Sandionigi, and Morgan illustrated that each type of fault (recoverable, not recoverable due to Total Ionizing Dose (TID), TDDB or EM has its known MTBF. When a fault is detected in an area, its MTBF is calculated and compared to the pre-computed MTBF for each fault type and classified the fault according to this comparison [19] [13].

Bolchini et al. developed an algorithm to find a number K, where K is the number of consecutive faults in the same area to be considered as transient before classifying it as permanent fault instead of computing the MTBF. Determining the value of K depends on the different MTBF for the different types of faults. K must not be too small because this will cause wrong classification of most faults as non-recoverable. Also, K must not be too large, as this will lead to late recognition of non-recoverable faults [14] [20].

3. PROBLEM DEFINITION

Most of fault-tolerant research focuses on recovering the faults by reconfiguration, but this may lead to useless reconfigurations if the hardware itself is defected. On the other hand, those who recover both transient and permanent faults by relocation have the advantage of constant time, and same detection and recovery steps. However, they waste the FPGA resources, since each relocation uses different cells/areas of the FPGA although the current area may not be damaged. Since the size of the FPGA is constant, then the number of relocation times and hence the number of tolerated permanent faults are limited according to the following scenario:

If the FPGA size is considered as S, and the application circuit is divided into N submodules each with area A_{module}, and a fault detection technique for each module with area $A_{detection}$, then the total area of the application is given by equation 1.

$$A_{total} = N \times (A_{module} + A_{detection}) \quad (1)$$

And, the free FPGA area SR that can be used as spares for relocation is

$$S_R = S - A_{total} \quad (2)$$

Hence the number of relocation times (tolerated faults) F is limited by the equation

$$F = \frac{S_R}{A_{module} + A_{detection}} \quad (3)$$

Also, the number of permanent faults that can be tolerated (F) is limited by the configuration memory size, since each relocatable module requires different bitstream, all these bitstreams are stored in the configuration memory. So, F is limited by the equation:

$$F = \frac{Configuration\ memory\ size}{\sum All\ bitstream\ files} \quad (4)$$

So, F must be the minimum of equation 3, 4.

When distinguishing between transient and permanent faults, the number of tolerated faults can be increased, where transient fault does not require relocation and can be recovered by reconfiguration. This scenario can be illustrated in Figure 2.

We expect the proposed algorithm to have the following features:

1. Deal with transient fault in the user-logic (application) registers.
2. Distinguish between transient and permanent errors in the logic configuration.
3. Adapt to the changes in the environment (i.e. amount of radiation).
4. The classifier is independent of the detection, or recovery techniques.

In order to fulfil these points, our work is based on the separation of the system component into independent layers. Multi-layered architecture is used to separate the processing of the fault tolerance algorithms from functional ones, and the classification step from the detection step, in addition to separating the adaptation mechanisms from the rest of the system.

4. THE PROPOSED METHOD

The conditions of missions to the solar system and deep space are unknown. Therefore, systems that go into spacecraft outside the earth are designed for the worst case. In this paper, we propose a structure of an adaptive fault classifier system to cope with changes of the environment in a meta-layer manner. The layers architecture simplifies the development of complex design. Each layer is responsible for a certain rule in the system. The main challenge is to design a well-structured, and a clean architecture where the layers are independent of each other, and to find a good interface between the layers. In this work, we propose the architecture of a fault tolerant adaptive system in multi- layers (4-layers) as shown in Figure 3. The relation between layers is shown in Figure 4.

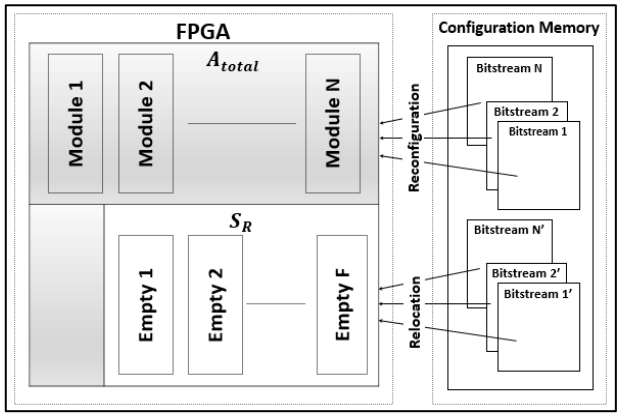

Figure 2. FPGA Model.

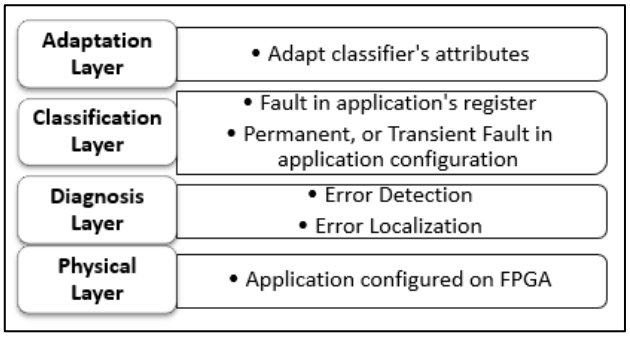

Figure 3. Proposed Multi-Layer Architecture

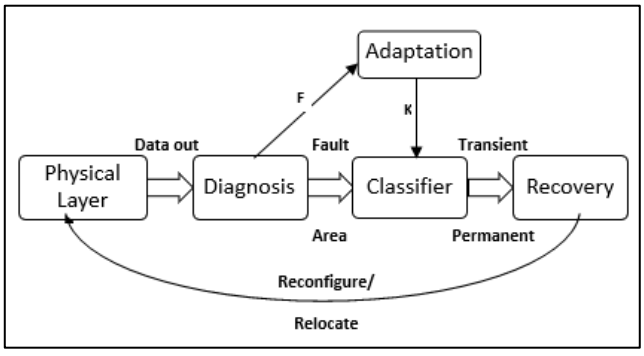

Figure 4. Interface between Layers.

The main layers are:

1. Physical Layer: It is the application configured on the FPGA, and monitored to assure the reliability of its operation. The application is described using a hardware description language, and then it is synthesized. The synthesized file is then translated, mapped then placed and routed. Finally, a bitstream file is created and loaded on the FPGA. This layer varies in size and operation depending on the application itself.
2. Diagnosis Layer: It is the first step in the fault-tolerance architecture where the operation of the application is monitored. Diagnosis is divided into two steps; first a detection technique (TMR, DMR, self-checking or other detection techniques) is used to detect if any fault has occurred, and then the localization of that fault is necessary to find where the fault exists (localization level varies from high grained where fault is localized in a module, to fine grained where localization can determine the faulty CLB). The output of this layer is a pair value <Error detected, Area>
3. Classification layer: The fault must be classified before its recovery. There are 3 main types of faults; transient fault in the application registers, transient fault in the application configuration and permanent fault in the FPGA fabric. Each type of fault is recovered in a different way. Transient faults in the application registers are simply recovered by resetting the operation. Transient faults in the configuration are recovered by reconfiguring the FPGA. Permanent faults are recovered by relocating the faulty module to a different area on the FPGA. The classifier receives the pair of values from the detection layer and the number K that is used to classify the detected fault, and then outputs the type of the fault to the recovery layer. Depending on the number of consequent faults in the same area (K) the fault will be classified.
4. Adaptation Layer: This is the layer that counts the total number of faults detected in a certain period. According to the total number of faults in a certain period the adaptation layer sends a new value of K (the number of faults that will be treated as transient before a permanent fault decision is taken) to the classification layer.

In this paper, we study the classification and adaptation layer as shown in the following subsections.

A. The Classification Layer:

This layer adopted the classifier algorithm in [14] (Classifier 1). The algorithm of the proposed classifier is shown in Alg.1. Classifier 1 receives error signals from the detector that adopts Two-Rail Code (TRC), and then the classifier compares the signals to determine whether there is a fault, and the faulty area number. The faulty area number is compared with the last faulty area (Alg.1 line3), if it is not the same, the fault is considered as transient and reconfiguration action is required in order to recover it. If the faulty area was the same as the last detected faulty area (Alg.1 line 6) then a counter is incremented (Alg.1 line 7). When the counter value reaches a pre-specified threshold (known as K); the fault is considered as permanent and a relocation action is required to recover it (Alg.1 line 11).

Alg. 1: Classifier Algorithm

Inputs: Error_Detected, Faulty_Area, K
Outputs: Error_Type

```
1  Begin
2    If Error_Detected =1 then
3      If Faulty_Area <> Last Area then
4        Error_Type transient in application register, reset is
5  required
6        Last_Area = Faulty_Area
7      Else
8        Error_Count ++
9        If Error_Count < K then
10         Error type is transient, reconfiguration required
11       Else
12         Error type is permanent, relocation is required
13       End if
14     End if
15   End if
   End
```

Classifier 1 depends on the detection technique where it receives the signal from TRC, but the new design isolates the detection and localization of fault in the diagnosis layer, and the classifier gets only a signal that there is a fault and the number of the faulty area, and hence the proposed classifier can be inserted into any fault tolerant system with any detection technique. Moreover Classifier 1 is designed with a pre-calculated K value depending on some parameters such as the required level of reliability, MTBF, and error latency. Since in the space MTBF is not constant and depends on the orbit, solar condition and other conditions, a constant K may result in different levels of reliability. In the proposed classifier, the Adaptation layer determines the value of K based on the rate of detected faults. Also, the proposed classifier considers the first fault in any area as a fault in the application registers not in its configuration, and only resets the application registers (Alg.1 line 4).

B. The Adaptation Layer:

The Adaptation layer adjusts the number of errors that will be considered as transient faults before being classified as permanent (designed as K). This number depends on the total number of faults for a certain period. For this purpose, low_threshold, and high_threshold values are selected. The low_threshold value is chosen for low rate of faults; whereas the high_threshold value is chosen for high fault rates. The total number of faults is compared to the threshold values to decide the value of K.

A complete study of the error rate in different orbits, and the effect of static and dynamic cross section on the error rate and the MTBF for both transient (non-persistent) and permanent (persistent) faults are illustrated in [19] [21] . It is proven that the dynamic cross section area (the area that is sensitive to faults) for permanent fault is very small. The error rate can be calculated using the following relations:

$$\lambda_{transient} = \frac{Dynamic\ cross\ section}{Static\ Cross\ Section} \times \lambda_{SEU} \quad (5)$$

$$\lambda_{Permanent} = \frac{Permanent\ cross\ section}{Static\ Cross\ Section} \times \lambda_{SEU} \quad (6)$$

$$MTBF = \frac{1}{\lambda} \quad (7)$$

The classifier can differentiate the faults in a single area at a time. The value of K must be determined carefully so that a fault can be classified and recovered before the occurrence of the next fault to avoid accumulation of faults. As shown in Figure 5, the MTBF is the time between the occurrences of two consecutive faults. If a permanent fault occurred between any two consecutive transient faults it must be detected and recovered before the occurrence of the second transient fault. Otherwise there will be fault accumulation, and it will be difficult to correct multiple faults in different areas. Thus, when the error rate increases, the MTBF is decreased and hence K must be decreased so that the permanent error latency (i.e. (K-1) * error latency) is smaller than the transient MTBF. Figure 6 shows the flow chart of the adaptation layer algorithm.

Decreasing and increasing K must be in a suitable range. A large K may cause fault classification of permanent faults to be transient faults for a longer time. A small K may cause classification of transient fault to be permanent fault although there is no physical destruction. Also, K is limited by the number of different areas (N) that constitute the application layer. As N increases K must be decreased, to avoid accumulation of faults in different areas before recovery from previous permanent fault.

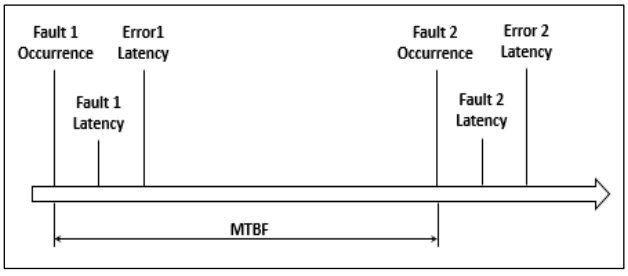

Figure 5. Mean Time Between Failure.

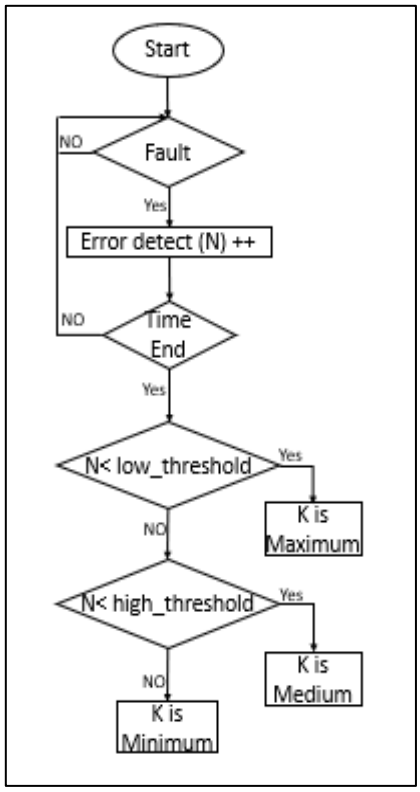

Figure 6. Flowchart of the Adaptation Layer Algorithm

5. EXPERIMENTAL RESULTS

The proposed adaptive classification algorithm has been implemented using VHDL language and XILINX Virtex 6. When the classifier is enabled it monitors the fault detection signal. The classifier works using the proposed algorithm Alg. 1until the type of fault is detected. In the same time, the adapter follows the proposed chart shown in Figure 6, and the number of K errors is updated every certain specified period. The results of the classifier are compared with related work in the following subsection.

A. Classifier Only

The classifier proposed in Alg. 1 (Classifier2) is implemented and its operation is verified. Comparing it with the previous work, both [12] [17] classify the fault to be permanent in its second occurrence. Comparing to our classifier (Classifier2) it will match only if K is chosen to be 2. As shown above, this assumption will increase the probability to classify transient fault to be a permanent one, hence the use of a bitstream file to relocate the application module where reconfiguration of it is enough.

Classifier1 in [14] is re-implemented for comparison on the same platform. In Classifier1 they localize the detected error in the classifier itself, but Classifier2 localizes the fault in the Diagnosis layer. Thus, as the number of areas increases Classifier2 utilizes smaller area than that of Classifier1 as shown in Table 1 and Figure 7. Also, Classifier2 is relatively having a smaller range in size variation with changing the number of areas. Classifier1 is dependent on the detection technique that the self-checker is using; two-rail coding, but since detection and localization is separated in the diagnose layer our classifier is not dependent on the detection technique. This classifier can be added to any known fault tolerant system that uses different types of fault detection.

To compare the functionality of classifier1 without detection and localization of fault, the error detection module, and the faulty era module are isolated from the classifier. As can be shown in Figure 7 and Table 1, as the number of areas increase Classifier2 uses less number of slices, when the number of areas decrease classifier1 is better in utilization.

In [13] they did not provide any data about the hardware utilization. But as they store the faults that occurred in each faulty area and compare the MTBF for the detected fault in the dominant area to distinguish the type of the fault, this will affect the hardware utilization and as the number of areas increases the hardware utilization is increased. As we adopted the classification according to a fixed number K as that of Classifier1, as stated in [14] this classification algorithm will assure classification of the fault in a fixed number of observations, where [13] requires different high number of observations.

TABLE 1.COMPARISON WITH RELATED WORK

	K=2, N=15		K=3, N=10		K=4, N=5		K=5, N=3	
	Slices	*Ff*	*Slices*	*Ff*	*Slices*	*Ff*	*Slices*	*Ff*
Classifier2	24	14	21	14	19	14	19	12
Classifier1	72	48	54	34	31	19	23	14
Classifier1 without Detection Module	33	19	27	14	20	10	14	8

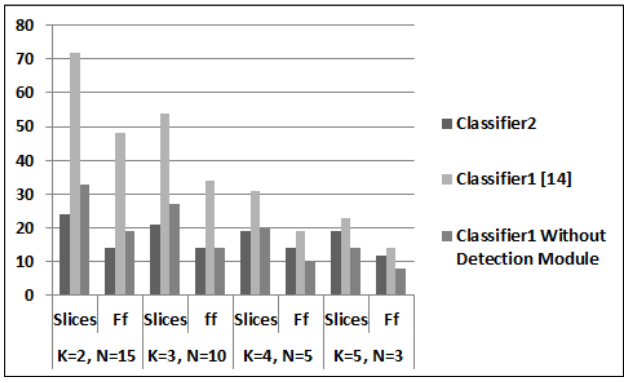

Figure 7. Comparison with Related Work.

B. Classifier and adaptation layer

The Adaptation layer depends on calculating the error rate that occurs in the application to increase or decrease the value of K. Error rate can be measured by using SEU sensors, or by calculating the number of detected faults in a certain period. The threshold values on which the K value changes, is chosen according to the SEU fault rate in the space. SEU rate depends on the environment (orbit in the space and the solar condition) as shown in Figure 8, the FPGA device parameters (Virtex II, Virtex4, Virtex 4Q, …etc.) as shown in Figure 9 and on the cross section of the application programmed on the FPGA as shown by equations 5, 6 .

The Algorithm in Figure 6 is implemented using VHDL to evaluate its performance. For the purpose of testing, we study the rate in LEO orbit. As shown in Figure 8, in LEO orbit for different FPGAs the SEU rate per day is in the range less than 10 SEU/Day, except for the trapped proton areas where it is in the rate of hundred. The low threshold is selected to be 2 SEUs/day, whereas the high threshold is selected to be 8 SEUs/day. K values are selected to be 3 faults for high error rates, 5 faults for medium error rates, and 7 for low error rates. Many researches are concerned with calculating and estimating the SEU rates for different orbits and FPGA devices [13], [22] [23]. The user can edit the values of the thresholds and value of K according to the required level of reliability he/she wants.

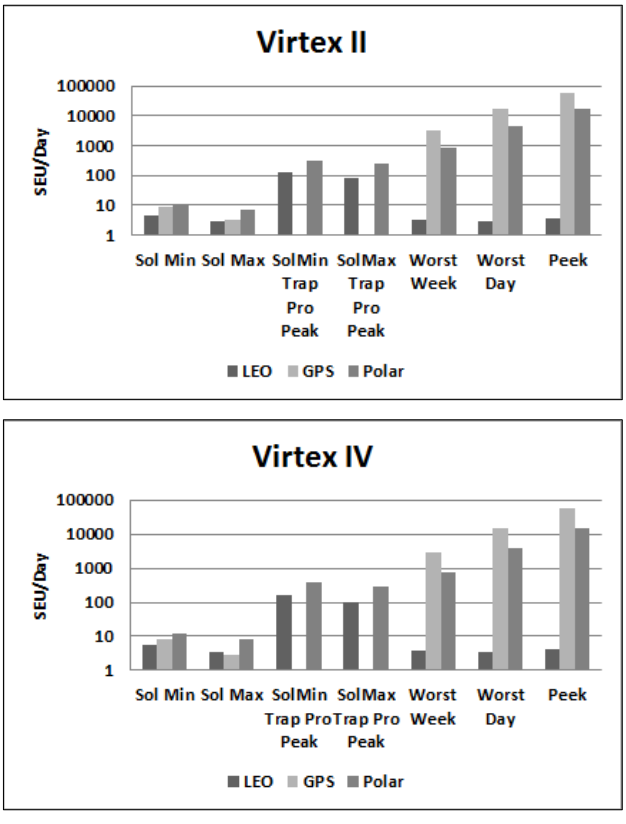

Figure 8. SEU Rates in Different Orbits and Conditions.

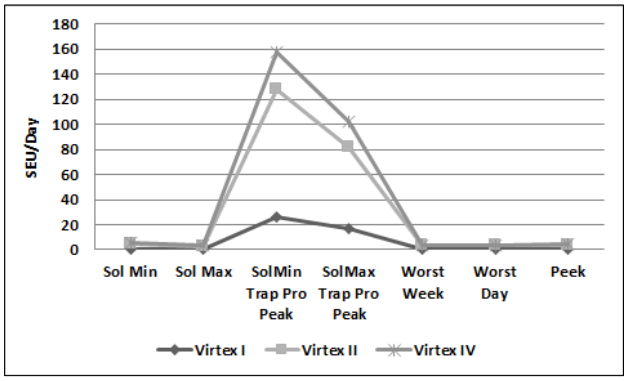

Figure 9. SEU Rate for Different FPGA Devices in LEO.

Using our algorithm, as the total number of faults increases regardless of the faulty area, the number of faults that will be considered as transient before classifying them as permanent decreases and vice versa to ensure that any permanent fault can be repaired before transient fault accumulation.

The adapter operation is illustrated in Figure 10. A counter is used to save the number of detected faults in a specified period of time (*t*), no. of detected fault per time t is known as the fault rate in this period. The results are divided into 5 areas of observation:

- Area A: It is assumed that the operation begins in a high radiation environment and K is set initially to be 3 as shown in area A in the figure.
- Area B: The error rate in area A is found to be 5, according to the flowchart in Figure 6, K is changed to the next value to be medium as shown in area B.
- Area C: Error rate in B is smaller than the low_threshold and K is changed to be maximum in area C.
- The process of calculating the error rate, comparing to the specified threshold values and changing K continues in the same manner in areas D and E.

The utilization of both the Adaptation and Classification layers using different FPGA devices is shown in Table 2. For more reliable operation the classifier itself must be hardened against faults. Hardening the classifier may be done by Triple Modular Redundancy (TMR) or with Double Modular Redundancy (DMR), for best reliability it is preferable to program each replica in a different FPGA chip.

TABLE 2. UTILIZATION OF THE PROPOSED ALGORITHM

Adaptation + Classification	Slices	FF
XC6VLX240T	45	22
XC5VFX130T	54	22
XC4VFX60	60	22

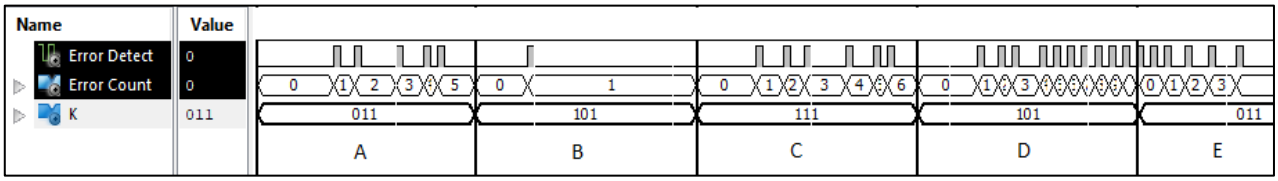

Figure 10. The Adaptation Layer Operation Result.

6. CONCLUSIONS

Classifying the faults is important to avoid useless reconfiguration of the FPGA and wasting its resources. We presented a novel approach to design an adaptive classifier in a meta-layer manner that copes with changes in the radiation effects according to the detected fault rates. The classifier is implemented using VHDL and Xilinx Virtex 6 (ML 605). Experimental results of the classifier compared to the related work show that as the number of areas increases our classifier is better in utilization (24 slices) than classifier1 (33 slices). Whereas the number of areas decreases classifier1 is better in utilization (14 slices) than ours (19 slices). Our classifier adds the ability to adapt with environmental changes which is not in the related work. Experimental results of the classifier in addition to the adaptation layer show the change in the number of faults that will be treated as transient faults before classifying it as permanent, whenever the number of detected faults exceeds predefined threshold values.

Future work includes the design of both the Diagnosis and Recovery layers, and fault injection to compute the achieved reliability of the proposed system.

REFERENCES

[1] V. Dumitriu, L. Kirischian and V. Kirischian, "Run-Time Recovery Mechanism for Transient and Permanent Hardware Faults Based on Distributed, Self-organized Dynamic Partially Reconfigurable Systems," *IEEE Transactions on Computers*, vol. 65, no. 9, pp. 2835-2847, 08 September 2016.

[2] . M. White and J. Bernstein, "Microelectronics Reliability: Physics-of-Failure Based Modeling and Lifetime Evaluation," JPL Publication 08-5 2/08, California, 2008.

[3] C. Sandionigi and C. Bolchini, "A Reliability-aware Design Methodology for Embedded Systems on Multi-FPGA Platforms," Milano, 2011.

[4] F. Siegle, T. Vladimirova, J. Ilstad and O. Emam, "Mitigation of Radiation Effects in SRAM-based FPGAs for Space Applications," *ACM Computing Surveys (CSUR)*, vol. 47, no. 2, pp. 1-34, January 2015.

[5] T. Daphne and T. Latha, "A Novel Adaptive Technique toMitigate Radiation Effects on FPGAS," *International Journal of Science and Research (IJSR)*, vol. 6, no. 2, pp. 321-325, February 2017.

[6] L. Pereira-Santos, G. L. Nazar and L. Carro, "Exploring Redundancy Granularities to Repair Real-Time FPGA-Based Systems," *Microprocessors and Microsystems*, vol. 51, pp. 264-274, 2017.

[7] S. C. Anjankar, M. T. Kolte, A. Pund, P. Kolte, A. Kumar, P. Mankar and K. Ambhore, "FPGA Based Multiple Fault Tolerant and Recoverable Technique Using Triple Modular Redundancy (FRTMR)," in *7th International Conference on Communication, Computing and Virtualization*, 2016.

[8] D. Shinghal and D. Chandra, "Design and Analysis of a Fault Tolerant Microprocessor Based on Triple Modular Redundancy Using VHDL," *International Journal of Advances in Engineering & Technology (IJAET)*, vol. 1, no. 1, pp. 21-27, March 2011.

[9] Z. Zhao, D. Agiakatsikas, N. T. H. Nguyen, E. Cetin and O. Diessel, "Fine-grained Module-based Error Recovery in FPGA-Based TMR Systems," in *Field-Programmable Technology (FPT)*, Xi'an China, 2016.

[10] A. Jacobs, G. Cieslewski, A. D. George and A. Gordan-Ross, "Reconfigurable Fault Tolerance: A Comprehensive Framework for Reliable and Adaptive FPGA-Based Space Computing," *ACM Transactions on Reconfigurable Technology and Systems,* vol. 5, no. 4, pp. 21-30, December 2012.

[11] A. Lifa, P. Eles and Z. Peng, "Performance Optimization of Error Detection Based on Speculative Reconfiguration," in *48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, New York, USA, 2011.

[12] S.-Y. Yu and E. J. McCluskey, "Permanent Fault Repair for FPGAs with Limited Redundant Area," in *Proceedings 2001 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, San Francisco, 2001.

[13] C. Bolchini and C. Sandionigi, "Fault Classification for SRAM-Based FPGAs in the Space Environment for Fault Mitigation," *IEEE Embedded Systems Letters*, vol. 2, no. 4, pp. 107-110, December 2010.

[14] C. Bolchini, C. Sandionigi and L. Fossati, "A Reliable Fault Classifier for Dependable Systems on SRAM-based FPGAs," in *Proceedings of the 17th IEEE International On-Line Testing Symposium (IOLTS'11)*, Athens, 2011.

[15] J.-M. Yang, "Tolerating Permanent State Transition Faults in Asynchronous Sequential Machines," *Journal of Computer Science and Technology*, vol. 31, no. 5, pp. 1028-1037, September 2016.

[16] M. Węgrzyn and J. Sosnowski, "Tracing Fault Effects in FPGA Systems," *International Journal of Electronics and Telecommunications*, vol. 60, no. 1, pp. 92-97, March 2014.

[17] E. A. Bezerra, F. Vargas and M. P. Gough, "Improving Reconfigurable Systems Reliability by Combining Periodical Test and Redundancy Techniques: A Case Study," *Journal of Electronic Testing*, vol. 17, no. 2, pp. 163-174, April 2001.

[18] S. Pontarelli, M. Ottavi, V. Vankamamidi, G. C. Cardarilli, F. Lombardi and A. Salsano, "Analysis and Evaluations of Reliability of Reconfigurable FPGAs," *Journal of Electronic Testing*, vol. 24, no. 1-3, pp. 105-116, June 2008.

[19] K. S. Morgan, "SEU-Induced Persistent Error Propagation in FPGAs," Provo, USA, 2006.

[20] C. Bolchini and C. Sandionigi, "Design of Hardened Embedded Systems on Multi-FPGA Platforms," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 20, no. 1, pp. 1-26, November 2014.

[21] J. Engel, K. S. Morgan, M. J. Wirthlin and P. S. Graham, "Predicting On-Orbit Static Single Event Upset Rates in Xilinx Virtex FPGAs," Los Alamos National Laboratory, Provo, USA, 2006.

[22] I. A. Troxel and A. D. George, "Adaptable and Autonomic Management System for Dependable Aerospace Computing," in *2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, Indianapolis, USA, 2006.

[23] S. Yousuf, A. Jacobs and A. Gordon-Ross, "Partially Reconfigurable System-on-Chips for Adaptive Fault Tolerance," in *International Conference on Field-Programmable Technology*, New Delhi, India, 2011.

Radwa M. Tawfeek Radwa Mohammed Tawfeek was born in 1978. She graduated in 2000 from Benha High Institute of Technology with a B.Sc. in computer engineering. She received her M.Sc. degree in 2007 from the same institute. Now, she is assistant lecturer at Benha Faculty of Engineering- Benha University, Egypt. She is a PH.D. candidate at Ain Shams Faculty of Engineering- Ain Shams University, Cairo, Egypt working on fault-tolerant reconfigurable systems.

Mohamed G. Egila Mohamed Gamal El-Deen Egila received the Bachelor degree and Master degree in Electronics and Communications from Cairo University, Egypt, in 2003 and 2008 respectively. He worked as a Researcher Assistant in Microelectronics Department, Electronics Research Institute, Cairo, Egypt, from 2004 to 2008, and as an Assistant Researcher in the Microelectronics Department, Electronics Research Institute from 2008 till 2016. He works now as a Researcher in the Microelectronics Department, Electronics Research Institute from 2016 till now. His research interests include medical signal processing, microprocessor and DSP-based medical instrumentation.

Yousra Alkabani holds B.Sc. and M.Sc.degrees in computer and systems engineering from Ain Shams University in 2003 and 2006, respectively. She received a Ph.D. in Computer Science from Rice University in December 2010. She is an Assistant Professor of Computer and Systems Engineering at Ain Shams University since May 2011 and a visiting Assistant Professor of Computer Science and Engineering at the American University in Cairo (AUC) since 2013. Her research interests include hardware security, low power design, and embedded systems.

I.M. Hafez Ismail Mohamed Hafez was born in 1961 Cairo, Egypt. He graduated in 1983 from Faculty of Engineering-Ain Shams University with a B.Sc. in Electronic and Communications. He received his PH.D. in 1990. He got his professorship in 2003. He was the head of Electronic and Engineering Department in the same faculty. Currently he is the Vice Dean for Community Services and Environmental Affairs of the Faculty of Engineering, Ain Shams University. The main scope of his research interest is analog, digital and mixed signal systems.