Nuclear Inst. and Methods in Physics Research, A 966 (2020) 163866

Contents lists available at ScienceDirect

Nuclear Inst. and Methods in Physics Research, A

journal homepage: www.elsevier.com/locate/nima

Analyzing single event upset on Kintex-7 Field-Programmable-Gate-Array with random fault injection method

Zibo Wang [a], Wei Chen [a,b], Zhibin Yao [b], Fengqi Zhang [b], Yinhong Luo [b,*], Xiaobin Tang [a,*], Cong Peng [a], Lili Ding [b], Xiaoqiang Guo [b]

[a] Department of Nuclear Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210106, China
[b] State Key Laboratory of Intense Pulsed Radiation Simulation and Effect, Northwest Institute of Nuclear Technology, Xi'an 710024, China

ARTICLE INFO

Keywords:
FPGA
Fault injection
Random injection
Triple module redundancy

ABSTRACT

In this paper, the bitstream of 28 nm field-programmable-gate-array was resolved. The relationship between the frame address and the resource was obtained. The fault injection platform was designed based on the information of the bitstream which obtained by partial reconfiguration. With this fault injection platform, the equivalence of the global fault and random fault injections was verified. Also, the sensitivities of different circuits were tested by random fault injection. The reinforcement effect of the triple module redundancy for sensitive resources in 28 nm FPGA was also be tested.

1. Introduction

FPGA, which can be applied to aerospace engineering due to the high performance and short development cycle, has a unique structure and reprogrammable flexibility. However, the degradation of the performance or even the functional failure might occur in FPGA induced by single-event effect (SEE), which caused by large number of high-energy charged particles in the space environment. The SRAM-based FPGA can be affected by the single-event upset (SEU) easily, and then the information of the user circuit may be changed, which can lead to functioning interruption or even device damage [1]. Therefore, investigating the sensitivity of the FPGA and taking corresponding reinforcement measures is crucial for the application of SRAM-based FPGA in the space environment.

As the feature size of reduces, the resources and performance of FPGAs improve significantly whereas the power consumption is also superior. However, due to the reduction of feature size, FPGA becomes more sensitive to SEU, and the MBU sensitivity also gets increased [2]. At present, three primary methods are used to study the SEE of FPGAs: (1) using the accelerators to simulate the SEE of FPGA in aerospace; (2) analyzing the SEE of FPGA by fault injection; and (3) calculating with mathematical mode to investigate the SEE of FPGA. The most accurate results can be obtained using the accelerator, but it is limited by the long experimental period and high experimental cost. Utilizing mathematical analysis and calculation for SEE in FPGA are more complicated than the two other methods. Fault injection considers cost and accuracy, which has become an indispensable and effective method to study the SEU of FPGA [3].

When performing the fault injection, the configuration file should be modified to simulate the change of the information in FPGA caused by SEU. Corresponding to each type of FPGA, the number of frames in the configuration file is different. Generally, the specific design cannot occupy all the programmable resources in FPGA, and the un-configured resources have no impact on the function of the system. Additionally, performing the global fault injection is the waste of time, and the partial injection that injects the error into the specific resources can improve the efficiency significantly. At present, most applications of the fault injection are partial fault injection. The fault injection platforms can be divided into two types according to different injection methods. That is, internal and external injections [4]. The internal injection is mainly performed using the internal configuration resources of FPGA; this method requires less hardware, and faster speed. For Xilinx FPGA, the internal injection can be realized through ICAP (Internal Configuration Access Port). L. Sterpone was the first one using ICAP to complete a new fault injection platform [5]. Subsequently, internal injection is used by various researchers to accelerate the fault injection [6–8]. However, there is a possibility that the internal injection injects fault into itself and induce the unpredicted results. Compared with internal injection, the external injection results are more objective and accurate. External injection completes the fault injection through a control board to change the bitstream of FPGA through an external interface. In previous study [9], the method of fault injection for the flip flop in 7 series FPGA has been proposed, but it has certain limitations in the research on FPGA internal resources.

In investigating the SEU of FPGA, FPGA can be subdivided into physical and application layers [10]. Typically, research on the physical

* Corresponding authors.
E-mail addresses: luoyinhong@nint.ac.cn (Y. Luo), tangxiaobin@nuaa.edu.cn (X. Tang).

https://doi.org/10.1016/j.nima.2020.163866
Received 10 December 2019; Received in revised form 26 February 2020; Accepted 28 March 2020
Available online 2 April 2020

layer pays attention to the sensitivity of FPGA itself. In FPGA, sensitivity is considered as the configuration memory cross section, which is also named static cross section. Static cross section is independent of the designed circuit, which only depends on the architecture of FPGA. Whereas the application layer indicates the functional circuit that is implemented by FPGA. The SEE sensitivity of the functional circuit relies not only on the device but also on the functional circuit. Eq. (1) can be used to calculate the dynamic cross section of a functional circuit [11].

$$\sigma_{dynamic} = \frac{E}{F}, \tag{1}$$

where E is the number of failures of the functional circuit and F is the flux of incident particles per unit area. For the configuration circuit in FPGA, generally only a part of the configuration bits affects the function of the logic circuit, and these configuration bits can be defined as "sensitive bits" [12]. In order to express the influence of the "sensitive bits", the sensitive factor ε is defined, which can be calculated as follows Eq. (2):

$$\varepsilon = \frac{\sigma_{dynamic}}{\sigma_{static}} = \frac{N_{sensitive}}{N_{config}}, \tag{2}$$

where N_{config} is the total number of bits of the bitstream in FPGA under test, and $N_{sensitive}$ is the number of sensitive bits. The sensitivity factor is related to the sensitivity of the system, that is, the larger the sensitivity factor is, the higher the system sensitivity will be. The static cross section can only be obtained by irradiation experiments. With static cross section and sensitivity factor, the dynamic cross section can be calculated. Differently designed circuits have different sensitivity factors, and the experiment cost is too high to investigate the dynamic cross section of each user circuit. Therefore, the fault injection method can be used to obtain the sensitivity factors of different design circuits, and then the dynamic cross section can be calculated. Previously, the most common and effective injection mode is global injection. Nevertheless, with the development of FPGA, the internal resources are increased, and the corresponding bitstream has additional frames. Hence, the global injection needs extra time. To reduce the injection time, the random injection method has been gradually applied in fault injection [6,7]. Random injection is injecting a random bit into the configuration bitstream and observing the functional error of the circuit. Although random injection has been applied to the field of fault injection, whether it is equivalent to the result of global injection remains to be verified.

In this paper, the fault injection research is mainly aimed at Xilinx's 28 nm FPGA, and a fault injection platform was completed to investigate SEU in Kintex-7 FPGA. The remainder of this paper is arranged in the following manner: Section 2 describes the selected device and introduces how to resolve the bitstream. Section 3 introduces the fault injection platform. Section 4 describes the comparison of global and random injections and compares the injection results of the functional circuits with different proportions of logic resources by random injection. Whether the triple modular redundancy is effective was also tested. Section 5 summarizes the paper.

2. Device and methodology

2.1. FPGA device under test (DUT)

The Kintex-7 series FPGA can be divided into several types based on the resources, package differences, and speed grades. The types of resources in the Kintex-7 series are the same, but the performance of each type is different [13,14]. The device structure of Xilinx FPGAs is cyclical, and the corresponding configuration files are regular. The bitstream is composed of frame which is the smallest unit. The frame in the configurable file in 7 series FPGA consists of 101 32-bit bytes, and different types of Kintex-7 FPGA have different frames in their bitstream. In this paper, XC7K70T FPGA is selected because it has the minimum resources of the Kintex-7 series. It is convenient to obtain and extract information from the bitstream. Fig. 1 shows the logical structure of XC7K70T in Vivado.

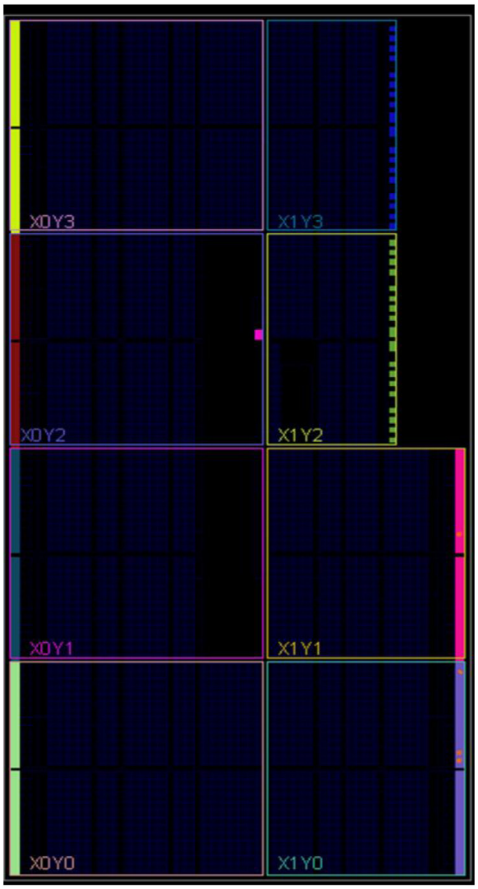

Fig. 1. Logical layout viewed through Vivado of Kintex-7 XC7K70T.

Fig. 2. Illustration of the irradiation board and the test system hardware.

The irradiation board used in this experiment includes one DUT and one golden device. The two FPGAs are configured with the same user circuit and operate at the same working voltage and clock frequency to compare their bitstream and output. The test system hardware is shown in Fig. 2.

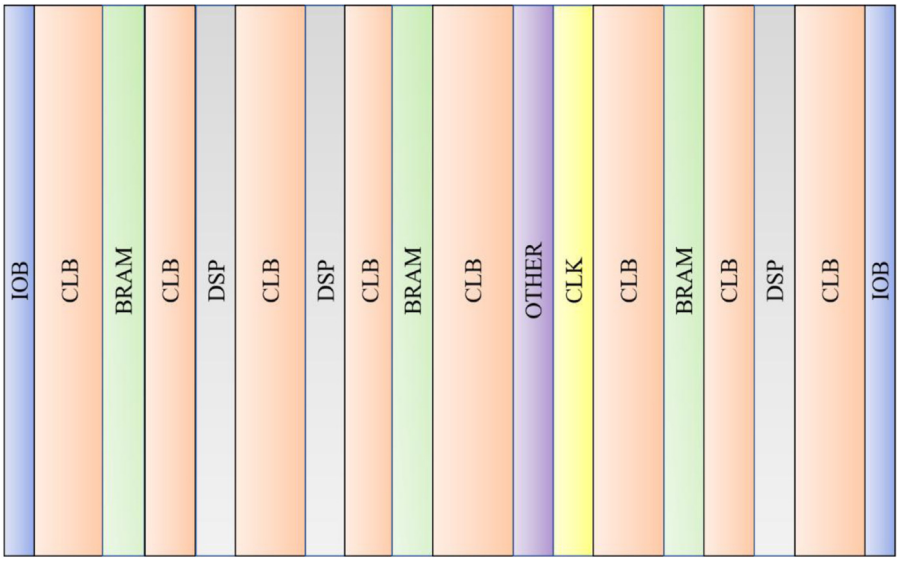

Fig. 3. Resources columns arranged in XC7K70T.

Table 1
Frame address register description [15].

Address type	Bit index	Description
Block type	[25:23]	Valid block types are CLB, I/O, CLK (000), block RAM content (001), and CFG_CLB (010)
Top/Bottom bit	22	Select between top-half rows (0) and bottom-half rows (1)
Row address	[21:17]	Selects the current row
Column address	[16:7]	Selects a major column, such as a column of CLBs
Minor address	[6:0]	Selects a frame within a major column.

2.2. XC7K70T bitstream decoding

Two points should be considered if SEU was injected into FPGA by external injection methods. One is the configuration resource used in the design circuits, and the other is the relationship between the number of the frame and the frame address in the bitstream. In the 7 series FPGAs produced by Xilinx, the programmable resources mainly include CLB, BRAM, DSP and IOB, and the resources occupied in different design circuits are also various. The frame address in the bitstream that corresponds to the resource used in the design should be modified to complete the fault injection. In this paper, partial reconfiguration has been used to investigate the relationship between the configuration resources of the XC7K70T and the frame address of the bitstream with Vivado. And these results can be applied to all 7 series FPGA.

In FPGA, each frame has its corresponding frame address. The description of the frame addresses were shown in Table 1. The bit of a specific frame can be found in the bitstream by the frame address. The interpretation of the frame address information is useful for studying the SEU of 28 nm FPGAs. Fig. 3 displays the resource arrangement of a row in the XC7K70T.

The corresponding reconfigurable modules are placed when designing the circuit. In the reconfigurable modules, an appropriate size area should be selected manually to configure this part of the functional circuit into this area. This area must also contain more resources than necessary to implement the functional circuit. It should be noted that when using Vivado for partial reconfiguration, at least two columns of resources should be selected. Therefore, the functional circuit should be designed with as few resources as possible to accurately determine the correspondence between frame address and the resources. Meanwhile, the same logic circuit is configured in the reconfigurable modules; accordingly, the contents of the reconfigurable bitstream in these reconfigurable modules are identical, which dramatically facilitates the positioning of the contents of the reconfigurable bitstream in the integrated configuration bitstreams. Another point that should be noted is that in 7 series FPGAs, only CLB, BRAM, and DSP can perform the partial reconfiguration. IOB or other resources cannot be selected, thereby increasing the difficulty of resolving the bitstream. Fig. 4 shows the flow chart of resolving the bitstream.

Table 2
The number of frames in different columns in 7 series FPGA.

Block type	Frames per column
CLB	36
BRAM	28
DSP	28
CLK	30
IOB	36
GTX channel	32
BRAM content	128

After the bitstreams are generated, we search for the contents in the integrated bitstream, which is the same as the partial reconfiguration bitstream, and the position of the partial reconfiguration bitstream in the integrated bitstream can be concluded. With the command information of the bitstream, the initial frame address and the total number of bits in the partial configuration bitstream can also be confirmed. Given that two adjacent columns must be selected in the partial reconfiguration, the number of frames per column resource can be obtained easily. For example, when two-column resources are both CLBs, the number of frames corresponding to the CLB column is identical, and then the number of frames in each column of CLBs can be concluded. Moreover, the BRAM or DSP columns must be placed adjacent to CLB columns to form two columns. The number of frames contained in CLB per column is known. Therefore, the number of frames in each BRAM or DSP column can be concluded easily. The correspondence between the configurable resources and the bitstreams, and the relationship between the resource locations and frame addresses can be obtained by partially reconfiguring each row and each column.

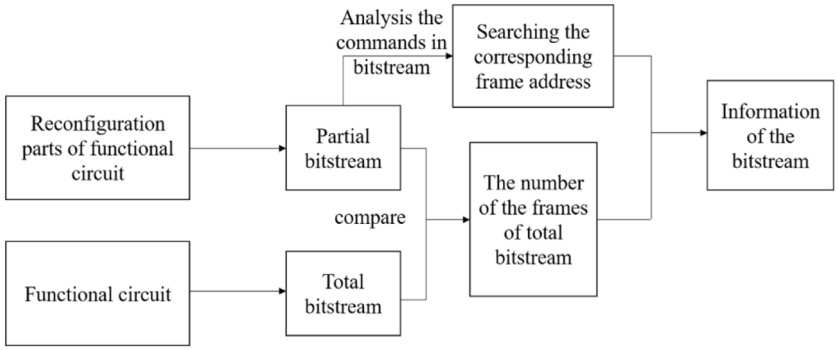

Fig. 4. The flow chart of resolving the information from the bitstream.

Table 3
Several commands in 7 series FPGA bitstream and their corresponding meanings.

Configuration data word (Hex)	Description
AA995566	Sync word
30008001	Write CMD register
30018001	Write IDCODE register
3000C001	Write MASK register
3000A001	Write CTL0 register
30002001	Write FAR register
30004000	Write FDRI register

The number of frames corresponding to each column of the resources in the FPGA is listed in Table 2. Given that 7 series FPGAs have the same logical architecture, this conclusion can also be applied to the other 7 series FPGAs. According to the layout information of FPGA in Vivado and the number of frames corresponding to each column, the relationship between the internal resources of FPGA and the configuration bitstream can be obtained through sequential calculation. The specific meaning of the commands in the configuration file should also be understood. Table 3 lists several critical commands and their corresponding definitions [15]. After realizing the definition of the corresponding commands and obtaining the relevant relationship between the frame address and resources, we can design the fault injection platform of the FPGA to investigate the SEU of 28 nm FPGA.

3. Proposed platform and methods

3.1. Fault injection platform

The proposed platform in this study can be divided into hardware and software systems. The hardware system consisting of the DUT, golden device, the system used to control output and comparison and the power supply system; the software system includes the upper computer control interface designed by LabVIEW and the program in the hardware system. Data input and readback are performed through the FPGA's SelectMAP mode. The partial reconfiguration of the 7 series FPGA must select at least two columns. However, we can inject a single bit error into FPGA after resolving the bitstream, which can reduce the injection time significantly while ensuring the accuracy.

The most important part of accomplishing the proposed fault injection platform is the frame address generator. Frame address generator can convert the frame number to the frame address, and the fault can be injected into the device precisely. Resources used in the circuit and the corresponding logical address in FPGA can be obtained in Vivado, which can help the perform fault injection to be targeted.

In the fault injection platform, three injection modes, scan, single bit and file injections, can be selected. The scan injection refers to the bit-by-bit injection which ranges from the initial frame to the end frame we set, and the injection bit is repaired before the next bit is injected. When the initial frame we set is the first frame in the bitstream and the end is the last frame in the bitstream, scan injection can be considered as the global injection. The sensitivity factor calculated from the results of the scan injection is the most accurate, however, more time is needed. Single bit injection refers to the injection of a specific bit in a specific frame. File injection provides a flexible injection mode. With file injection, the files that are full of the pseudorandom numbers can be used to complete the simulation of random injection. The random injection speed is much faster than the global injection, which can save more time but still remain the accuracy. Therefore, the random injection can be a better choice for fault injection of FPGA. The flow diagram of random injection is illustrated in Fig. 5.

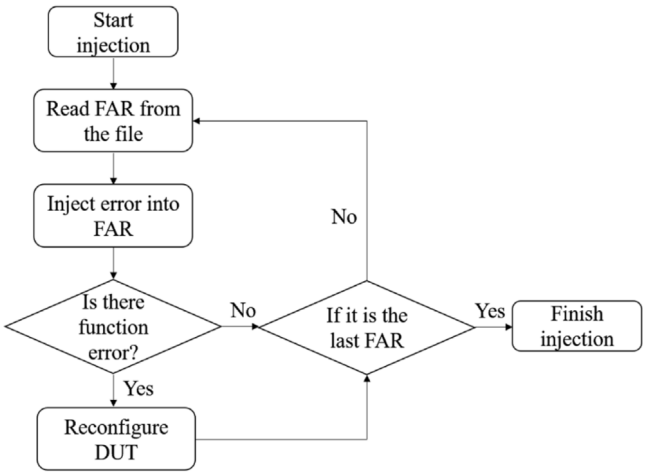

Fig. 5. Flow diagram of random injection.

There are several errors which may not be repaired during the injection process, inducing continuous errors of the functional circuit. The continuous errors will be remarkable obstacles for the subsequent injections, which may make the results unreliable. Therefore, the entire DUT should be reconfigured after observing the specific number of the functional errors. In the fault injection platform, we can design the maximum number of persistent errors. When the persistent errors reach the set maximum number, the entire device can be reconfigured. And in this paper, the number of the persistent errors was set as 1 to improve the accuracy.

3.2. Triple module redundancy validation

Triple module redundancy (TMR) is one of the most common measures used for reinforcement to mitigate the SEE of FPGAs, which are implemented by copying the same circuit into three parts and then using a voter to select the output. In this way, even if one of the circuits generates the functional failures due to the SEE, the output of the entire

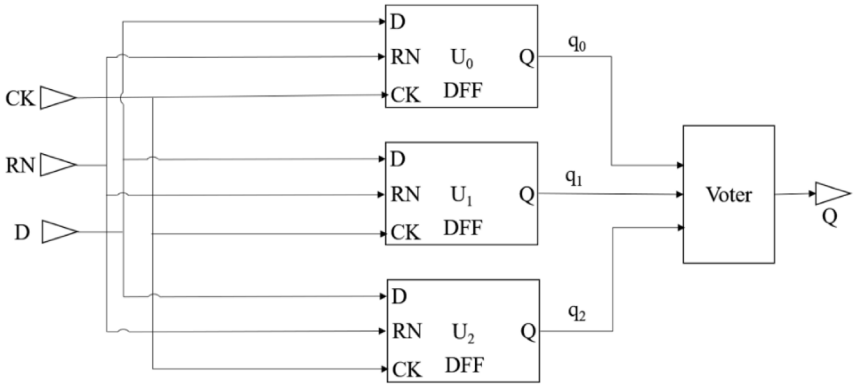

Fig. 6. The scheme of TMR.

system remains regular, with the conditions of the two other circuits operating normally. The scheme for TMR is shown in Fig. 6.

The resources required to perform TMR on the design circuit are at least three times of the necessary resources in the original design. This is an obvious waste of FPGA resources, furthermore, the additional resources may increase the sensitivity. Research [16] shows that targeted reinforcement of sensitive resources of FPGA may be useful to mitigate SEE. However, whether the partial TMR for the sensitivity resource is still valid in 28 nm FPGA must be tested. The proposed fault injection platform can also be used to verify the effect of partial TMR used for the sensitive resource or circuit.

Global injection takes a lot of time. Therefore, before the partial TMR verification, the equivalence between random and global injections can be verified. Subsequently, the random injection will be used to verify the effect of partial TMR and test the sensitivity of different circuits at the same time.

4. Experimental results

4.1. Verify the equivalence of global and random injections

Verifying the equivalence between the global and random injections is based on the injection of the same functional circuit. The circuit that is designed to test the equivalence uses BRAM and LUT to form the memory. Information on the two-part memory that is composed of BRAM and two-part memory composed of LUT will be compared in real-time. When the information on the two-part memory composed of the same resource is identical, the output result of the circuit is 0; otherwise, the output is 1. The 15% BRAM, 15% LUT, and 4% flip-flop are occupied in this verified circuit. The time of random injections is set to 10, and the number of the injected bits is 10,000, which can avoid experiment contingency. The results of the random injection are shown in Table 4. In the global injection, 24,071,936 bits are injected, and 9320 functional errors occur in the test. According to the meaning of the sensitive factor, the sensitive factor of random injection can be defined as the ratio of the number of functional errors to the number of bits. As a result, the sensitive factor can be calculated using Eq. (2). Global injection takes approximately four days to a week, whereas a random injection of 10,000 bits only requires about 1–2 h. Random injection saves more time than global injection. The sensitive factors of global and random injections are illustrated in Fig. 7. The figure shows the values of the sensitive factors calculated from the two injection methods are close. Therefore, global injection can be replaced with random injection.

4.2. Calculate the sensitive factors of different circuits

The sensitive factors obtained from global and the random injections in the same circuit are approximately identical. To save time,

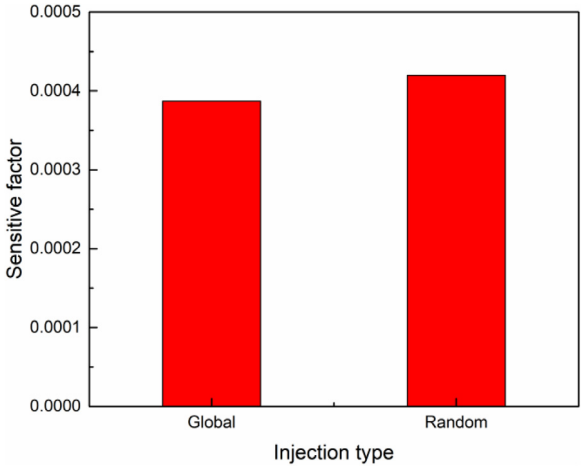

Fig. 7. Sensitive factors of global and random injection experiments.

Table 4
The number of functional errors occurred in random injection experiment.

Injection times	Random injection errors
1	4
2	4
3	6
4	3
5	2
6	4
7	5
8	5
9	3
10	6
Average	4.2

Table 5
The resource utilization of three benchmark circuits.

Resources type	Test_01	Test_02	Test_03
BRAM percentage	15%	30%	96%
DSP percentage	–	–	100%
LUT percentage	15%	30%	28%
Flip-flop percentage	4%	30%	23%
IO percentage	13%	13%	13%

we investigate the sensitivity of three different circuits under random injection. The typical resources are used in the three benchmark circuits, of which the difference is in the resource utilization. The circuits reflect the most realistic designs in FPGA. Three benchmark circuits are represented by Test_01 to Test_03 and their resource utilization are presented in Table 5.

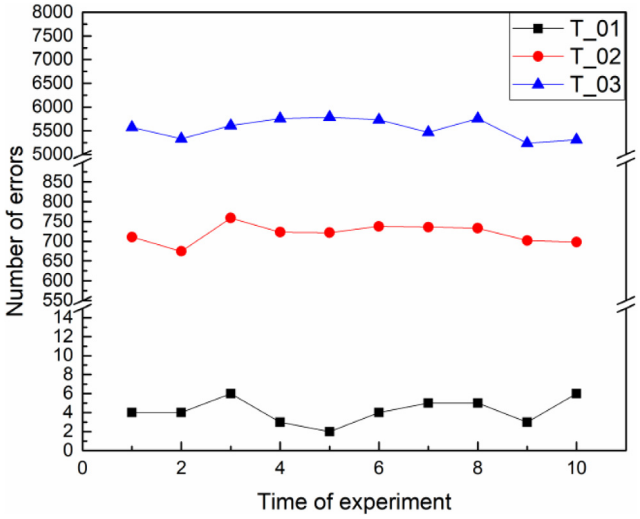

Fig. 8. Number of functional errors in three benchmark circuits obtained by random injection method.

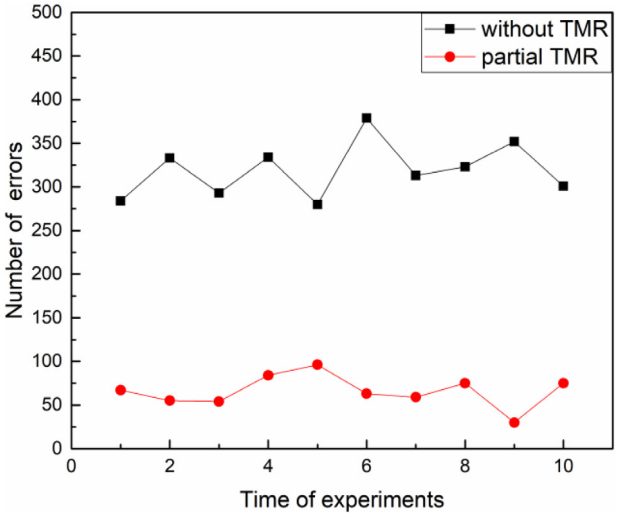

Fig. 9. The number of functional errors in TMR test circuits.

Table 6
Resource occupancy ratios of without TMR circuit and partial TMR circuit.

Resources type	Without TMR	Partial TMR
BRAM percentage	15%	15%
LUT percentage	15%	15%
Flip-flop percentage	20%	59%
IO percentage	13%	13%

The experiment performed 10 random injections on each test circuit, and 10,000 bits are injected into the bitstream randomly each time. The bitstream is immediately repaired after one bit is injected. Once the function error occurs, DUT is immediately reconfigured. The results of the random injection for three test circuits are illustrated in Fig. 8.

The sensitivity factor corresponding to each functional circuit can be calculated by the average number of functional errors of each functional circuit, wherein the sensitive factor of Test_01 is 0.00042, Test_02 is 0.072, and Test_03 is 0.55586. The more resource that the circuit uses, the larger the sensitive factor is because the design circuit cannot use all the resources in FPGA. Therefore, when performing the error injection, the fewer resources used in the circuit, the larger the proportion of the configuration bits that are not related to the circuit, and the lower sensitivity of the circuit at the same time.

4.3. 28 nm FPGA TMR tests

When injecting errors into different circuits, flip-flops are sensitive and prone to generating functional errors. As a result, the circuit without TMR and the circuit with the flip-flops partial TMR are randomly injected to compare their number of functional errors. The resources used in these circuits are shown in Table 6.

Compared with the circuit without TMR, the proportion of flip-flops used in the partial TMR has increased by approximately twofold; and the ratio of the other resources used in partial TMR is invariable. TMR experiments are also performed with random injection, and the injection results are illustrated in Fig. 9.

The number of the functional errors occurring in the circuit with TMR is reduced significantly after using partial TMR to reinforce the flip-flops, which indicating that the effect of the partial TMR is excellent. The experiment results reveal that partial TMR has an improved impact on SEU mitigation in FPGA. Meanwhile, partial TMR requires few resources, that can improve the efficiency of FPGA. Therefore, detecting the sensitive resources of the functional circuit by fault injection and reinforcing them can achieve a relative balance between the proportion of resources used and reinforcement effects.

5. Conclusions

In this paper, the bitstream of the XC7K70T was resolved, the meaning of the commands in the bitstream was translated, and correspondence between frames and resources was suggested. The bitstream information that we have obtained can also be used to other 28 nm FPGAs. Subsequently, the frame address generator is designed to complete the fault injection platform. The equivalence of global and random injections was verified using this platform.

Furthermore, this fault injection platform was used to test the sensitive factor of different circuits designed in 28 nm FPGA. The effectiveness of partial TMR for sensitive resources is also detected. The results show that partial TMR may be an excellent choice to mitigate the SEU of FPGA.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Zibo Wang: Data curation, Writing - original draft. **Wei Chen:** Supervision. **Zhibin Yao:** Software, Conceptualization. **Fengqi Zhang:** Investigation. **Yinhong Luo:** Formal analysis. **Xiaobin Tang:** Writing - review & editing. **Cong Peng:** Investigation. **Lili Ding:** Methodology. **Xiaoqiang Guo:** Visualization.

Acknowledgment

This work was supported by the Major Program of the National Natural Science Foundation of China (Grant Nos. 11690043, 11690040).

References

[1] M. Ceschia, M. Violante, M.S. Reorda, A. Paccagnella, P. Bernardi, M. Rebaudengo, A. Candelori, Identification and classification of single-event upsets in the configuration memory of SRAM-based FPGAs, IEEE Trans. Nucl. Sci. 50 (2003) 2088–2094.

[2] H. Quinn, K. Morgan, P. Graham, J. Krone, M. Caffrey, A review of Xilinx FPGA architectural reliability concerns from Virtex to Virtex-5, in: 2007 9th European Conference on Radiation and its Effects on Components and Systems, IEEE, Deauville, France, 2007, pp. 1–8.
[3] L. Sterpone, M. Violante, A new analytical approach to estimate the effects of SEUs in TMR architectures implemented through SRAM-based FPGAs, IEEE Trans. Nucl. Sci. 52 (2005) 2217–2223.
[4] U. Kretzschmar, A. Astarloa, J. Jiménez, M. Garay, J. Del Ser, Compact and fast fault injection system for robustness measurements on SRAM-based FPGAs, IEEE Trans. Ind. Electron. 61 (2014) 2493–2503.
[5] L. Sterpone, M. Violante, A new partial reconfiguration-based fault-injection system to evaluate SEU effects in SRAM-based FPGAs, IEEE Trans. Nucl. Sci. 54 (2007) 965–970.
[6] R. Zhang, L. Xiao, J. Li, X. Cao, C. Qi, J. Li, M. Wang, A fast fault injection platform of multiple SEUs for SRAM-based FPGAs, elsevier microelectron, Reliab. 82 (2018) 147–152.
[7] S. Di Carlo, P. Prinetto, D. Rolfo, P. Trotta, A fault injection methodology and infrastructure for fast single event upsets emulation on Xilinx SRAM-based FPGAs, in: 2014 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), IEEE, Amsterdam, Netherlands, 2014, pp. 159–164.
[8] J. Tarrillo, J. Tonfat, L. Tambara, R. Reis, Multiple fault injection platform for SRAM-based FPGA based on ground-level radiation experiments, in: 2015 16th Latin-American Test Symposium (LATS), IEEE, Puerto Vallarta, Mexico, 2015, pp. 1–6.
[9] A. Ullah, P. Reviriego, J.A. Maestro, An efficient methodology for on-chip SEU injection in flip-flops for Xilinx FPGAs, IEEE Trans. Nucl. Sci. 65 (2018) 989–996.
[10] M. Violante, L. Sterpone, M. Ceschia, D. Bortolato, P. Bernardi, M.S. Reorda, A. Paccagnella, Simulation-based analysis of SEU effects in SRAM-based FPGAs, IEEE Trans. Nucl. Sci. 51 (2004) 3354–3359.
[11] H. Quinn, M. Wirthlin, Validation techniques for fault emulation of SRAM-based FPGAs, IEEE Trans. Nucl. Sci. 62 (2015) 1487–1500.
[12] Z.M. Wang, Z.B. Yao, H.X. Guo, M. Lv, Bitstream decoding and SEU-induced failure analysis in SRAM-based FPGAs, Sci. China Inf. Sci. 55 (2012) 971–982.
[13] Xilinx, DS182_Kintex_7_Data_Sheet, 2019, available at https://china.xilinx.com/support/documentation/data_sheets/ds182_Kintex_7_Data_Sheet.pdf (accessed on November 21st, 2019).
[14] Xilinx, DS180_7Series_Overview, 2019, available at https://china.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf (accessed on November 21st, 2019).
[15] Xilinx, UG470_7Series_Config, 2019, available at https://china.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf (accessed on November 21st, 2019).
[16] B. Pratt, M. Caffrey, J.F. Carroll, P. Graham, K. Morgan, M. Wirthlin, Fine-grain SEU mitigation for FPGAs using partial TMR, IEEE Trans. Nucl. Sci. 55 (2008) 2274–2280.