Chapter 10
Multiple Fault Injection Platform for SRAM-Based FPGA Based on Ground-Level Radiation Experiments

Jorge Tonfat, Jimmy Tarrillo, Lucas Tambara, Fernanda Lima Kastensmidt, and Ricardo Reis

Abstract Fault injection by emulation is a well-known method to analyze the reliability of a circuit. SRAM-based FPGAs provide the hardware infrastructure to implement fault injectors taking advantage of dynamic partial reconfiguration. This chapter presents the details of a Multiple Fault Injection Platform and the analysis of the configuration memory upsets of the FPGA. Results of fault injection campaigns are presented and compared with accelerated ground-level radiation experiments.

10.1 Introduction

Field-Programmable Gate Arrays (FPGAs) nowadays are not only used for ASIC prototyping but also to replace them in some ground-level and space applications. SRAM-based FPGAs take advantage of the latest semiconductor fabrication processes, allowing high-density logic integration. This scenario allows them to achieve expected performance levels in a variety of applications. Moreover, the reconfigurability feature of SRAM-based FPGAs allows the same device to perform multiple functionalities during its lifetime.

These characteristics make SRAM-based FPGAs attractive to critical applications. But since configuration bits are stored into volatile SRAM cells, radiation effects can generate single or multiple bit-flips in the configuration memory. Such single event upsets (SEUs) or multiple bit upsets (MBUs) can induce functional errors in the implemented design. In order to tolerate these faults, many techniques were proposed in the literature. However, it is necessary to validate the efficiency of these techniques closest to the real effect as possible, but also considering the controllability, observability and cost.

Jimmy Tarrillo • Lucas Tambara • Ricardo Reis • J. Tonfat (✉) • F.L. Kastensmidt
Instituto de Informática, Universidade Federal do Rio Grande do Sul (UFRGS),
Porto Alegre, Brazil
e-mail: jorgetonfat@ieee.org; fglima@inf.ufrgs.br

F. Kastensmidt, P. Rech (eds.), *FPGAs and Parallel Architectures for Aerospace Applications*, DOI 10.1007/978-3-319-14352-1_10

Fault injection by emulation is an important method to predict in the early stages of the design phase the susceptibility of the design under upsets. Emulation of SEUs and MBUs by flipping the configuration bits on an FPGA is an attractive technique to evaluate the behavior of a design before it is working in radiation environments. In addition, fault injectors can take advantage of partial reconfiguration capabilities of FPGAs to reduce even more the time to inject upsets. The main goal of this approach relies on the fact that it allows fast injection campaigns, once the circuit under test (CUT) executes at the full FPGA speed and not on simulation speed.

Moreover, the amount of injected faults per unit of time (upset rate) is higher compared to radiation tests on particles accelerators because a bit-flip is directly injected in the memory cell. The control of the test is also superior compared to a radiation test, since a precise location is flipped (a known bit), which allows the user to reproduce a real radiation test.

The fault injection can be performed by an external or internal programmable port of the FPGA. The internal configuration access port (ICAP) [1] provides some advantages such as the possibility to reconfigure frame by frame without the necessity of using input/output pins. The ICAP can be controlled by the SEU controller macro [2] and an embedded soft-core as PicoBlaze; or by a specific control design developed by the user [3]. SEUs can be injected in the bitstream in random locations, sequentially (every configuration bit or configuration control register is flipped in sequential order), or user-defined.

10.2 Related Works

Other fault injection platforms are available to inject SEU in SRAM-based FPGAs as described in [4]. FLIPPER [5] that is targeted to Virtex-2 devices is one example. It uses a scheme based on a control motherboard and a DUT board. The fault injector is implemented in the mother-board FPGA and a host PC. The DUT board contains the target FPGA. The configuration memory of this FPGA is modified with partial reconfiguration using an external configuration port. In [6] the fault injector and the DUT are implemented in the same FPGA and in order to inject faults a host PC creates faulty bitstreams. FT-SHADES [7] and [8] are other examples of fault injectors but in this case they use an internal injection approach using the ICAP to inject single faults in the bitstream.

With internal fault injection [7–9], we do not need to reconfigure the entire FPGA, so the fault injection speed is increased, but a problem arises. The quality of the fault injection can be reduced by fault injection side-effects as shown in [9]. A fault injected in the configuration memory can affect the fault injector itself. So the fault injection can stop unexpectedly or even worst, the fault injector can wrongly report that a fault is injected.

In this work, we present a multiple fault injector platform able to emulate SEU and MBU in the configuration memory bits of an SRAM-based FPGA. Our goal is to replicate the effects of radiation to validate protection techniques and improve the

radiation test methodologies and test plans under accumulated multiple faults. The proposed Fault Injection Platform uses the ICAP module to flip a configuration bit, and takes the bit location from an external database bank. The bit-flip locations were taken from previous experiments in neutron radiation test from ISIS facilities [10] and also generated by a MATLAB pseudo-random generator. During the fault injection procedure, the fault injector takes the necessary actions to guarantee a correct fault injection and minimize the side-effects improving the quality of the results.

10.3 Hardware Implementation of the Multiple Fault Injection Platform

The proposed Multiple Fault Injection Platform is composed of a single SRAM-based FPGA, a flash-based external memory and a host computer. We use the Digilent Genesys prototype board containing a Xilinx Virtex-5 FPGA, part XC5VLX50T-FFG1136 and other resources. For our fault injection platform, we use the external flash memory connected to the FPGA to store the bit-flip locations. This memory stores the SEU locations database bank. A block diagram of the Multiple Fault Injection Platform is shown in Fig. 10.1.

The FPGA contains the DUT (Design Under Test) and the fault injector. It is well-known that internal injectors suffer from side-effects because an injected fault can provoke an error on the injector itself. But to mitigate these effects, the fault injector can avoid bit-flips in its configuration bits.

The fault injector is composed of an ICAP controller, a flash memory controller and a PicoBlaze 8-bit soft processor.

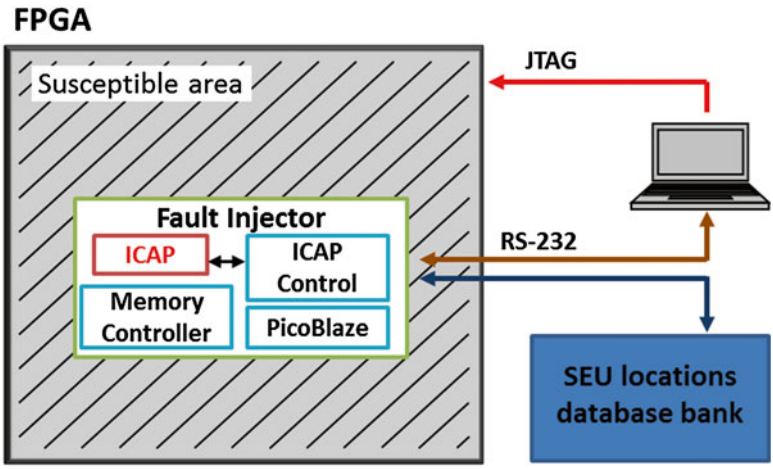

Fig. 10.1 Architecture of the Multiple Fault Injection Platform

Table 10.1 Frame address field descriptions

Field	Description
Type	Defines the type of frame. Can be a configuration frame (type 0), BRAM content (type 1) and other two types not well documented in the literature
Top/bottom	Defines the half (top or bottom) of the FPGA where the frame is located
Row	Defines the frame row. The row number increases from the middle of the FPGA
Column	Defines the frame column. A column is defined by the type of resource (ex. CLB, DSP, etc.)
Frame in column	Defines the frame position inside the column

The main function of the PicoBlaze is to control the execution of the fault injection campaign. The ICAP controller manages all the commands to read and write frames from the configuration memory using the ICAP. The ICAP is the interface that enables access to the configuration memory from an internal circuit in the FPGA. With a suitable set of commands, we can modify the configuration memory without stopping the application running in the FPGA. This method is also known as dynamic partial reconfiguration.

In order to control the ICAP, we must understand the configuration memory of the FPGA and the way to read and write in this memory.

10.3.1 Organization of Virtex-5 FPGA Configuration Memory

The FPGA can be seen as a device with two layers. One is the logic layer that includes all the user application resources such as the Configurable Logic Blocks (CLB), the Block RAMs, I/O blocks, etc. The other is the configuration layer that comprises the configuration memory and the associated access ports.

Understanding the organization of the configuration memory will allow us to know the relation between configuration bits and resources of the FPGA.

The following information is based on the Virtex-5 Configuration User Guide [1].

The FPGA configuration memory is composed of small memory segments called *configuration frames*. So a configuration frame is the smallest addressable segment of the FPGA configuration memory, and the frame size varies among FPGA families. In the case of Virtex-5, it is composed of 41 words of 32 bits (1,312 bits).

Each frame has a unique address that is related to the physical position in the FPGA floorplan. Each frame address has five fields. Each field is described in Table 10.1 and corresponds to the organization of the FPGA floorplan.

Due to this organization, frame addresses are not consecutive. A graphical description of the organization of the floorplan is shown in Fig. 10.2.

The floorplan is divided into two main regions: top and bottom. Each region is organized in rows and columns. One frame has the height of a row, and the columns are organized according to the type of resource (ex. CLB, BRAM, DSP, etc.). Each

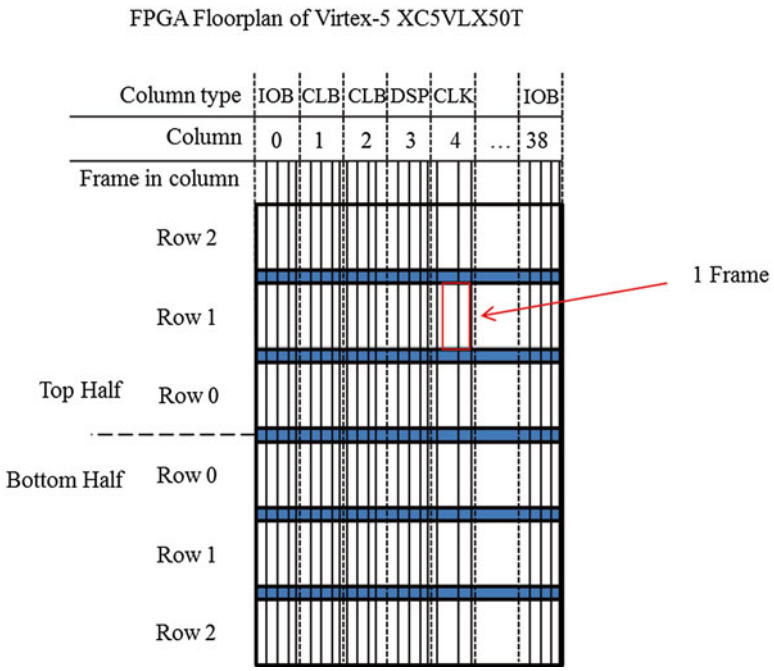

Fig. 10.2 Example of the organization of the configuration memory of a Virtex-5 FPGA

Table 10.2 Number of frames per column

Column type	Number of frames
CLB	36
DSP	28
Block RAM (configuration)	30
IOB	54
CLK	4

column contains a group of frames. The number of frames on each column depends on the type of column as shown in Table 10.2.

Depending on the device selected, some of the frames in this organization are not implemented. This case is common for IOB columns, where not all the rows of an IOB column have the corresponding frames since the IOB resources depend on the number of pins of the FPGA.

10.3.2 Methodology for a Fault Injection Campaign

With the information about the organization of the configuration memory and the specific commands sequence to read and write frames, we can flip any bit of the configuration memory thus emulating the effect of an SEU.

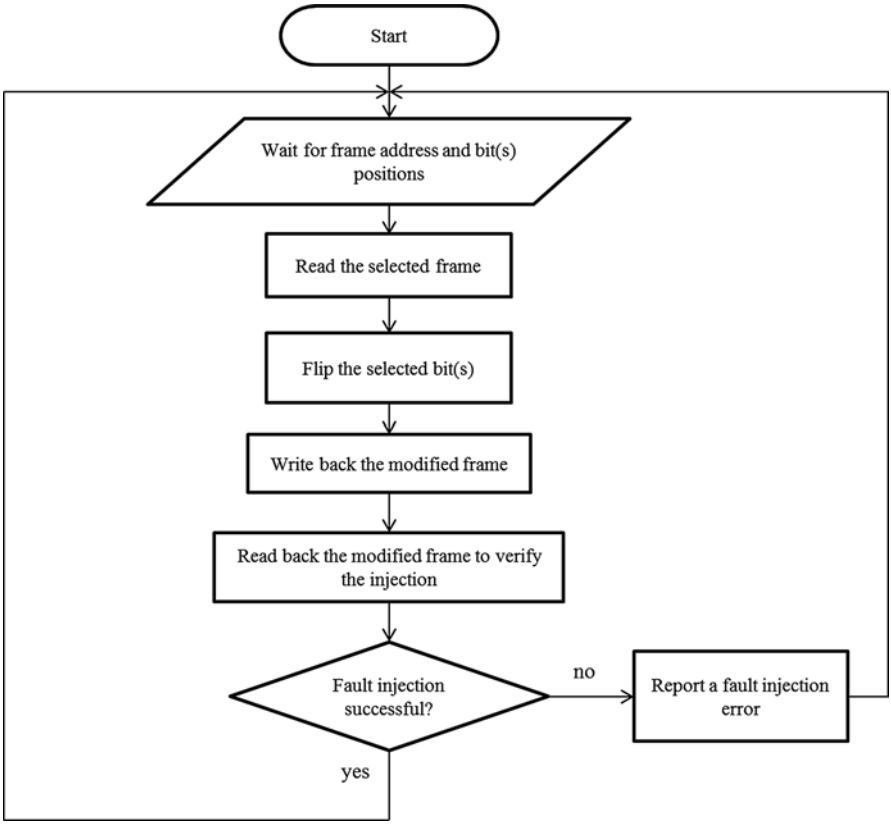

Fig. 10.3 Flow diagram of the procedure to inject one fault

Figure 10.3 shows the procedure executed by the ICAP controller to inject one fault into the configuration memory. The only information needed to flip a bit is the selected frame address and the selected bit inside this frame. This information comes from the SEU database stored in the external memory and is managed by the PicoBlaze soft processor. It is important to mention that this method can also emulate intra-frame multiple bit-flips.

Since the smallest segment of the configuration memory is a frame, the ICAP controller needs to read the entire frame and store it in a temporal buffer. Then the selected bit(s) position(s) are flipped. Finally, the modified frame is written back to the configuration memory. In order to verify the correct insertion of the fault, the frame is read back again and compared to the modified frame stored in the temporal buffer. If differences are found between them, the ICAP controller reports a fault injection error.

Most of the time injection errors are due to the inexistence of the selected frame address in the FPGA as mentioned in the previous section. This type of error injection does not interfere with our results since these missing frames cannot be flipped

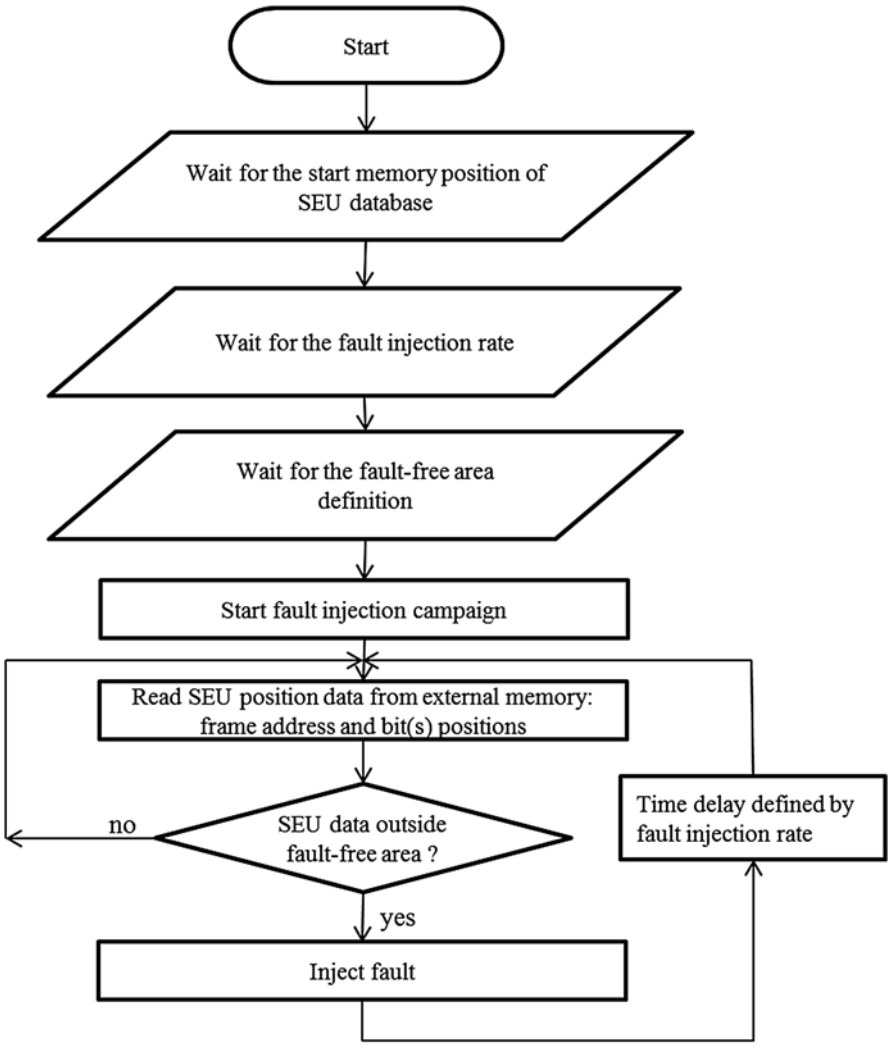

Fig. 10.4 Flow diagram of the procedure to control a fault injection campaign

by real SEUs. The ICAP controller reports failed injections to take into account this information when the fault campaign report is generated.

So a complete fault injection is completed in 310 clock cycles. With a clock frequency of 50 MHz, one injection is completed in 6.2 µs.

The PicoBlaze manages the execution of a complete fault injection campaign. The procedure is described in Fig. 10.4. The procedure starts with the definition of the parameters of the campaign. These parameters are the start memory position of the SEU database, the fault injection rate and the definition of the fault-free area.

The start memory position of the SEU database is the reference point to the PicoBlaze in order to read consecutively from this point the bit-flip data stored in the external memory. The fault injection rate defines the amount of faults injected per time unit. This parameter can be used to emulate different radiation environments.

The definition of the fault-free area is to protect the circuits that can interfere with the execution of the fault injection campaign. For instance, the fault injector area needs to be included in this protected area. This method minimizes the possibility of a functional error in the fault injector itself that is one of the side-effects of internal fault injection. Other circuits that can be included are, for example, the circuit that controls the execution of the DUT. Since a functional error in this block can generate a false functional error of the DUT, we must protect this block from bit-flips. The fault-free areas need to be in agreement with the placement constraints set during the design implementation phase.

So when the fault injection campaign starts, each SEU position read from the external memory is analyzed to determine if it is inside the fault-free area. When the bit-flip position is inside the protected area, the bit-flip is not injected, and the next SEU position is loaded. If not, the PicoBlaze commands the ICAP controller to inject the corresponding fault.

At the top level, the host PC is in charge of the execution of multiple fault injection campaigns. The procedure is shown in Fig. 10.5. The first step is to set the corresponding parameters.

The first parameter is the maximum time for a single fault injection campaign. This time is variable and depends on the DUT and the fault injection rate. This setting helps to determine when a fault injection campaign reaches an unknown state.

The start memory position of the SEU database defines the starting point of the first fault injection campaign. The subsequent campaigns will start from the last injected SEU position. In this way, we assure different SEU patterns for each fault injection campaign.

The fault injection rate and fault-free areas are also defined. These parameters can be fixed for all the fault injection campaigns or can be variable among campaigns according to the user needs.

When all parameters are set, the host PC configures the FPGA with the DUT and the fault injector module through the JTAG interface and the fault injection campaigns begins.

To recognize the end of a fault injection campaign, it is necessary a DUT end condition event. In our case, we want to test the maximum number of accumulated faults that a design can tolerate before it starts to fail. When it reaches a certain condition, the DUT sends a signal that is captured by the host computer. It also receives the information of SEU positions injected and the information when a fault injection has failed.

The fault injector was implemented into the XC5VLX50T FPGA on the Genesys Digilent board and the synthesis result is detailed in Table 10.3.

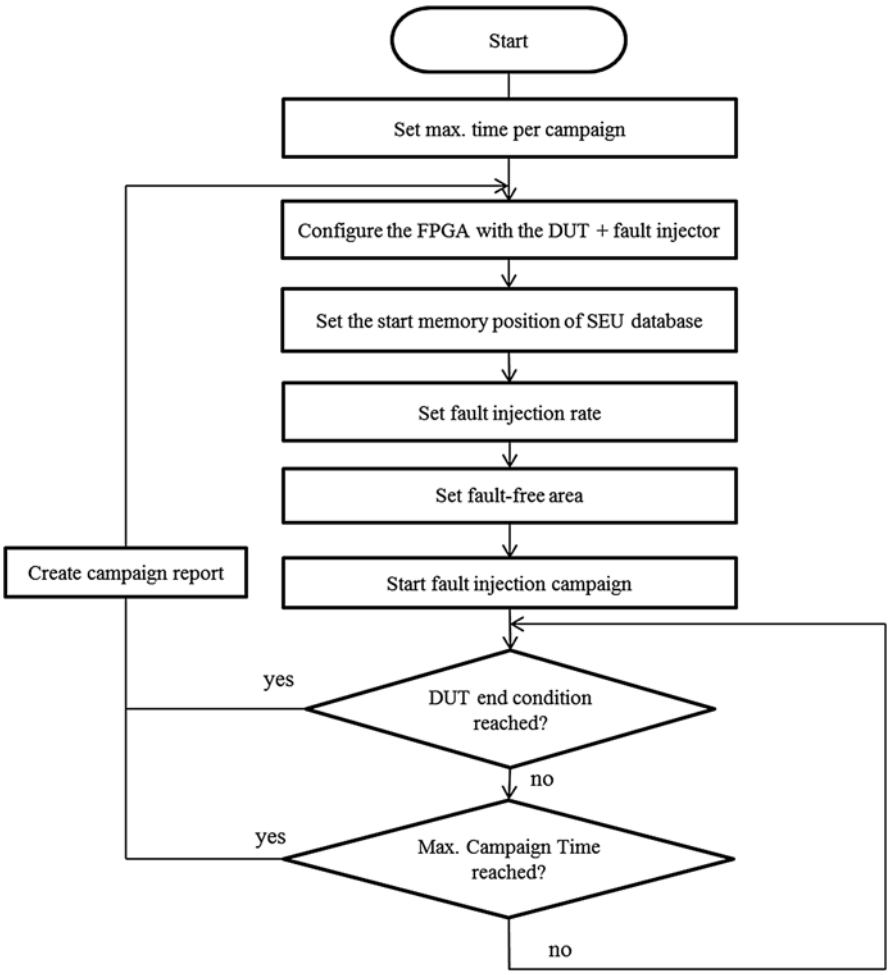

Fig. 10.5 Flow diagram of the procedure to control multiple fault injection campaigns

Table 10.3 Resource utilization of the fault injector

	LUTs	Registers	Block RAMs
PicoBlaze soft processor	147	76	1
Flash memory controller	86	68	0
ICAP controller	705	417	1
Total	938	561	2

10.4 Methodology for Capturing and Modeling Single Bit Upsets

The injected faults are modeled mainly with two different approaches:

- By using a radiation database from previous radiation experiments.
- By using a computer generated database based on a pseudo-random generator with a uniform distribution.

10.4.1 Modeling Using Data from Previous Ground-Level Radiation Experiments

The database is composed of multiple and accumulated faults in Virtex-5 FPGA. These faults were obtained from previous radiation experiments at ISIS facilities of Rutherford Appleton Laboratory (Didcot, United Kingdom).

During the tests, bit-flips in the configuration memory were detected using a readback procedure as described in Fig. 10.6. It is important to mention that this procedure logs bit-flips in the configuration memory and the content of block RAMs. So we use the mask file (generated by Xilinx tools) to filter our logs from bit-flips in block RAMs and bit-flips due to shift registers or LUT RAMs used by the DUT.

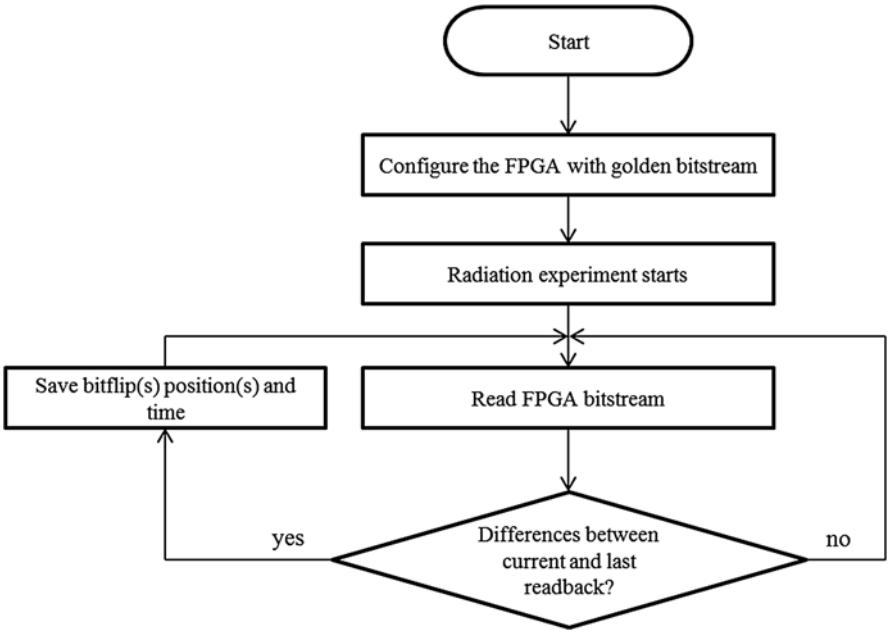

Fig. 10.6 Procedure to capture bit-flips in the configuration memory

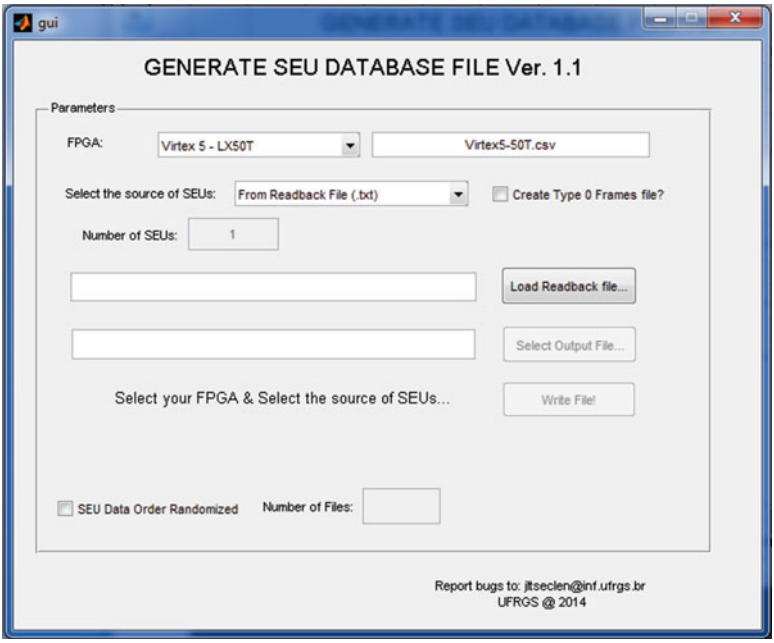

Fig. 10.7 GUI of the tool to create SEU databases

Based on our knowledge of the FPGA configuration memory and the readback bitstream, we can precisely determine the frame address and bit position of each SEU registered during the experiment. The location of the bit-flip is the information needed by the fault injector to inject a bit-flip.

We developed a software tool to automate this process. The tool takes the text reports from the radiation experiments and creates the binary file for the external flash memory automatically. Figure 10.7 shows a screenshot of the GUI of this tool.

In our previous radiation experiments, more than 2,600 SEUs were identified. This information is stored in the external flash memory. In the case of the Genesys board, it has a flash memory of 256 Mbit (organized as 16-bit by 16 Mbytes) for non-volatile storage of FPGA configuration files. We used three memory addresses to store the information of each SEU. The first two positions store the frame address and the last position store the bit position inside the frame. So, up to five million SEUs can be stored in this memory.

10.4.2 Modeling SEUs Using Computer Generated Data

Based on the analysis of the accumulated bit-flips obtained from radiation experiments at ISIS, we also generate bit-flips locations that resemble the original ones. We achieve this using MATLAB and a pseudo-random generator with a uniform

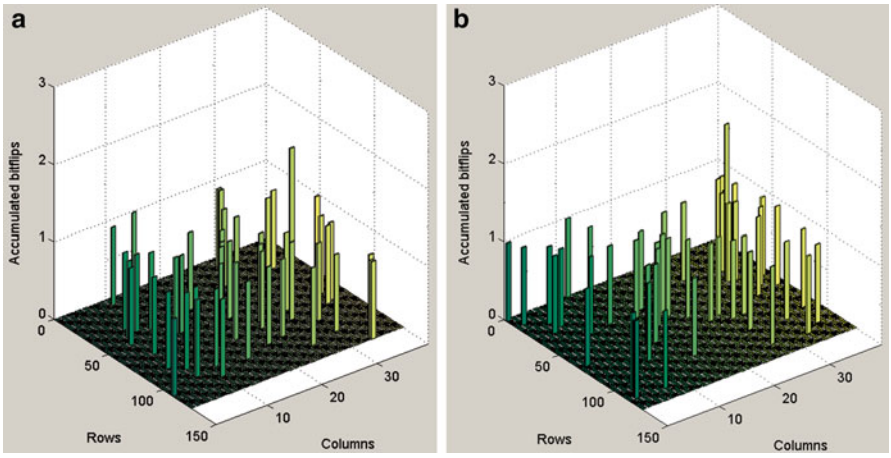

Fig. 10.8 Comparison of bit-flips from radiation experiments and MATLAB generated. (**a**) 50 ISIS bit-flips, (**b**) 50 MATLAB generated bit-flips

distribution. Figure 10.8 shows a graphical comparison between collected bit-flips and generated bit-flips. Each bar represents the number of accumulated bit-flips per resource in the FPGA (ex. 1 CLB). The color scale is only for visualization purposes. In the case of the Virtex-5 XC5VLX50T FPGA, the resources form a matrix of 120 rows by 39 columns.

The option to generate bit-flips is also included in the same tool that creates the SEU database from radiation experiments.

10.5 Fault Injection Campaign Results and Comparisons

In order to validate the fault injection platform, we have evaluated one case study design. Then we have compared the fault injection results with the neutron radiation experiments results.

This design implements an N-modular redundancy (nMR) scheme as a technique to tolerate multiple fault accumulation. The nMR is composed of n functionally identical modules, which receive the same m-bits input and deliver p-bits output to the Self-Adapted voter (SAv), Fig. 10.9 [11].

The SAv receives $n \times p$ bits from all modules and generates the fault-free p-*output*, n-error status flags (ESF), and a non-masked fault signal (NMF). In this scheme, the system allows the accumulation of defective modules, until remaining at least two modules without fault. The SAv is a majority voter, considering as population fault-free modules.

The implemented design is a 7-MR adder chain. The architecture is shown in Fig. 10.10. The criteria for selecting this design were the low logic masking of faults

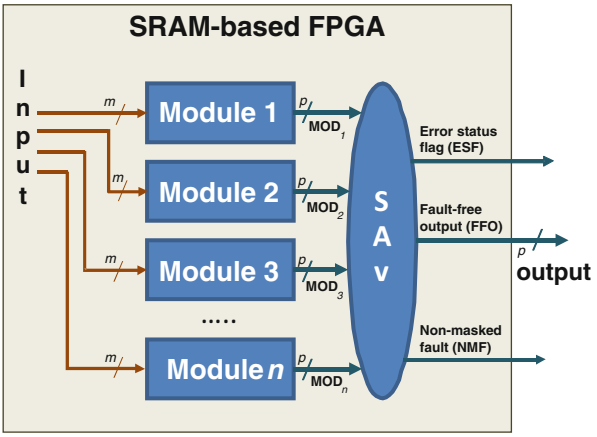

Fig. 10.9 nMR-based technique with SAv voter

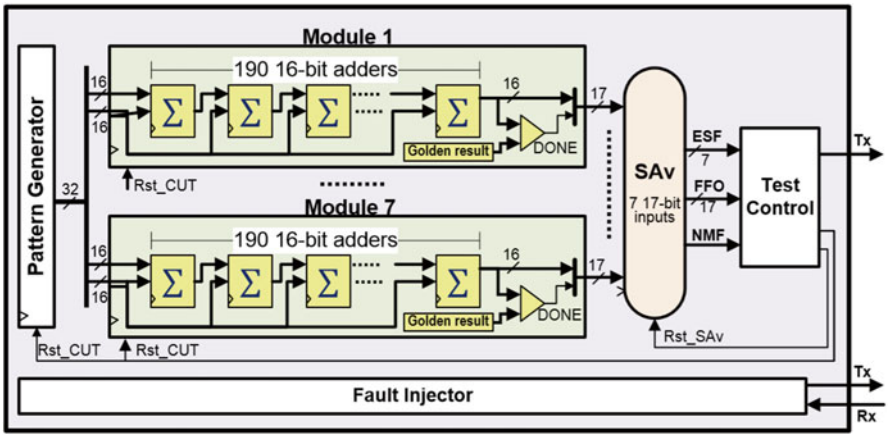

Fig. 10.10 Block diagram of the adders chain DUT and the fault injector

and the ease to scale. This design has a control module to manage the input pattern generator of the adder chains and to monitor the correct response of the 7-MR system. When a functional error is detected, the control block sends error signals to the host PC, and the fault injection campaign ends.

Figure 10.11 shows the final placement of the 7-MR adder chain and the fault injector. The areas of the fault injector and the control module are included in the fault-free area of the fault injector.

The objective of the test is to determine if the fault injector can predict the tolerance of this design under neutron radiation. So the test reports the number of accumulated faults needed to provoke the failure of each of the seven modules. The end condition of the test is when only two correct modules remain.

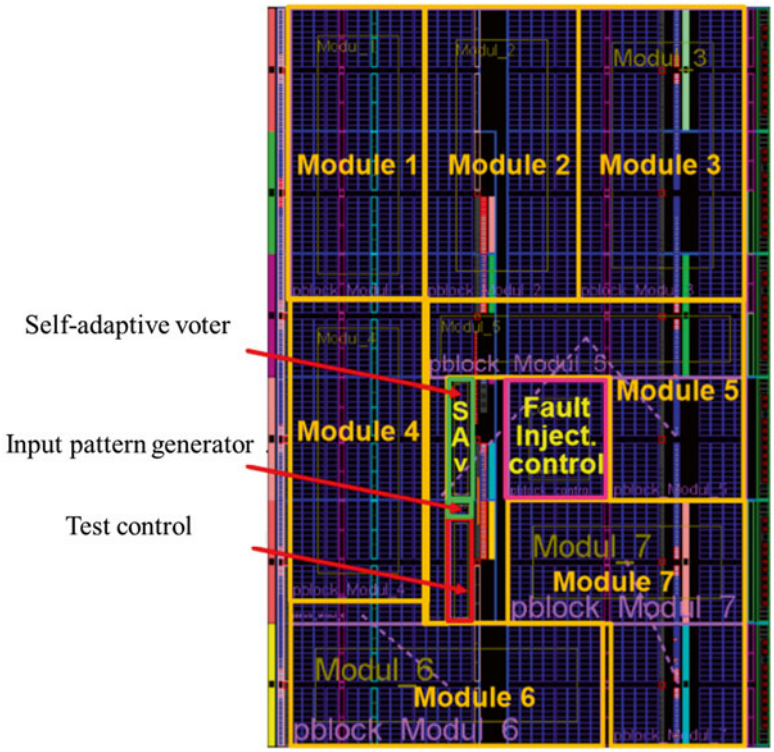

Fig. 10.11 Placement of the adders chain DUT and the fault injector

Figure 10.12 presents the results of the fault injection campaigns. We run 25 injection campaigns and it was injected an average of 98.33 faults per campaign.

Figure 10.13 shows the results from the radiation experiment. Due to beam time restrictions, we were able to run the test few times.

And Fig. 10.14 shows the comparison between the results from fault injection and radiation experiments. Both present similar average accumulated faults for each of the faulty modules count.

10.6 Conclusions

This work presents a multiple fault injection platform to evaluate accumulated SEU effects in Virtex-5 FPGA. The platform uses bit-flip positions generated by a pseudo-random generator or taken from a database composed of pre-collected real bit-flips location detected from previous neutron accelerated experiments at ISIS facilities. The flipped bits distribution of real radiation test and fault injector were shown and analyzed. Also, the effects of accumulation SEUs on a design using real

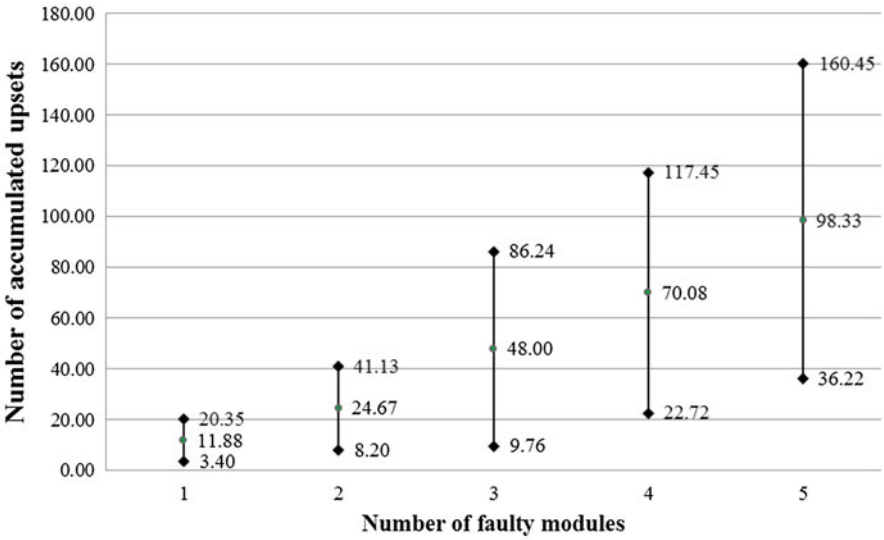

Fig. 10.12 Number of accumulated faults needed to provoke multiple faulty modules under fault injection for the adder chain case-study

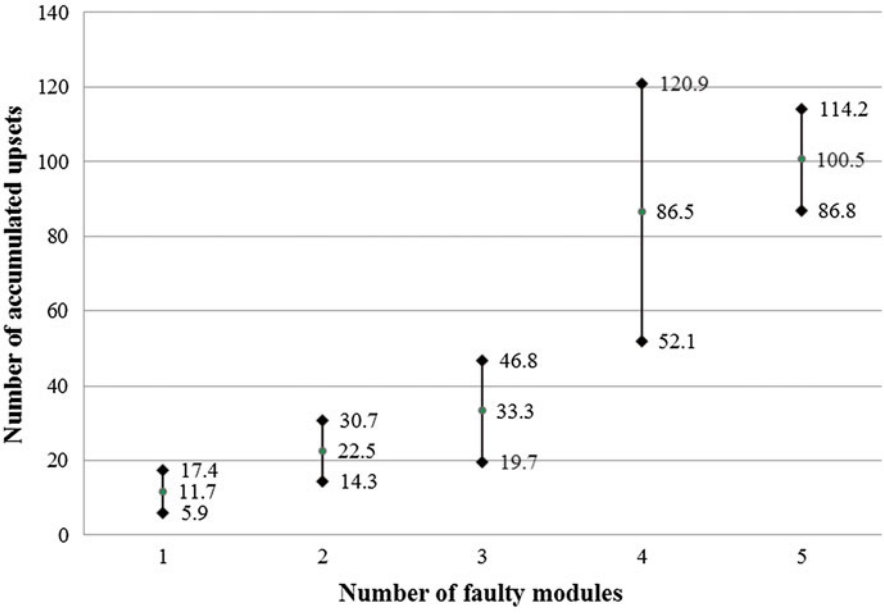

Fig. 10.13 Number of accumulated faults needed to provoke multiple faulty modules under radiation experiment for the adder chain case-study

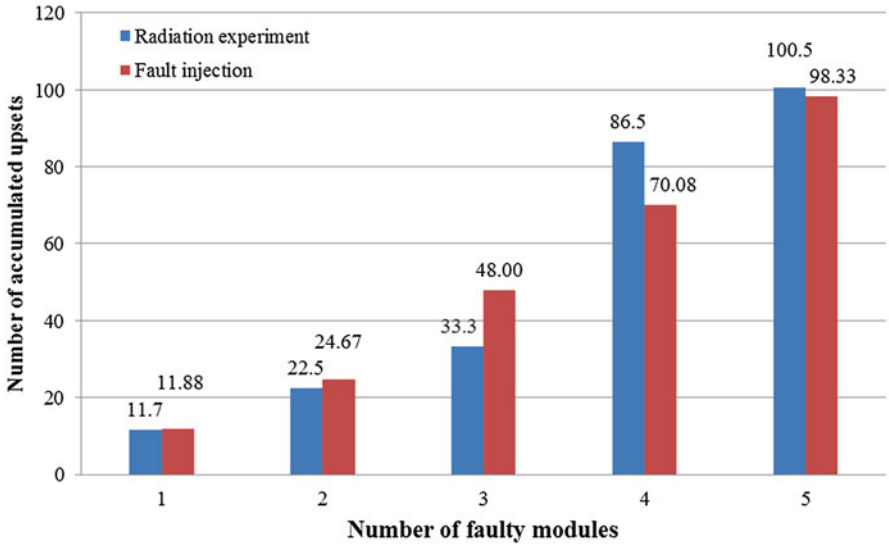

Fig. 10.14 Comparison between fault injection and radiation experiment results of adder chain case study

radiation test and fault injection were tested. Results show the real capability of the platform proposed to predict the effects of radiation in FPGA designs and mitigate successfully the side-effects related to internal fault injectors.

References

1. Xilinx, UG191 (2012) Virtex-5 FPGA configuration user guide, 19 Oct 2012
2. Chapman K (2010) SEU strategies for Virtex-5 devices. XAPP864 v2.0
3. Tarrillo J, Escobar FA, Lima Kastensmidt F, Valderrama C (2014) Dynamic partial reconfiguration manager. In: 2014 IEEE 5th Latin American symposium on circuits and systems (LASCAS), 25–28 Feb 2014, pp 1–4
4. Alexandrescu D, Sterpone L, Lopez-Ongil C (2014) Fault injection and fault tolerance methodologies for assessing device robustness and mitigating against ionizing radiation. IN: 2014 19th IEEE European test symposium (ETS), 26–30 May 2014, pp 1–6
5. Alderighi M, Casini F, Citterio M, D'Angelo S, Mancini M, Pastore S, Sechi GR, Sorrenti G (2008) Using FLIPPER to predict irradiation results for VIRTEX 2 devices. In: Radiation and its effects on components and systems (RADECS), pp 300–305
6. Sterpone L, Violante M, Rezgui S (2006) An analysis based on fault injection of hardening techniques for SRAM-based FPGAs. IEEE Trans Nucl Sci 53(4):2054–2059
7. Guzman-Miranda H, Tombs JN, Aguirre MA (2008) FT-UNSHADES-uP: a platform for the analysis and optimal hardening of embedded systems in radiation environments. In: IEEE international symposium on industrial electronics, ISIE 2008, pp 2276–2281
8. Nazar GL, Carro L (2012) Fast single-FPGA fault injection platform. In: 2012 IEEE international symposium on defect and fault tolerance in VLSI and nanotechnology systems (DFT), 3–5 Oct 2012, pp 152–157

9. Kretzschmar U, Astarloa A, Jimenez J, Garay M, Del Ser J (2014) Compact and fast fault injection system for robustness measurements on SRAM-based FPGAs. IEEE Trans Ind Electron 61(5):2493–2503
10. Violante M, Sterpone L, Manuzzato A, Gerardin S, Rech P, Bagatin M, Paccagnella A, Andreani C, Gorini G, Pietropaolo A, Cardarilli G, Pontarelli S, Frost C (2007) A new hardware/software platform and a new 1/E neutron source for soft error studies: testing FPGAs at the ISIS facility. IEEE Trans Nucl Sci 54(4):1184–1189
11. Tarrillo J, Rech P, Kastensmidt F, Valderrama C, Frost C (2013) Neutron cross-section of N-modular redundancy technique in SRAM-based FPGAs. In: 2013 14th European conference on radiation and its effects on components and systems (RADECS). IEEE, Oxford, pp 1–6