2013-f-11

LEO Single Event Upset Emulator for Validation of FPGA Based Avionics Systems

By Mohamed Mahmoud Ibrahim, Kenchi Asami, and Mengu Cho

Kyushu Institute of Technology, KitaKyushu, Japan

This paper presents a complete design and implementation of a Single Event Upset (SEU) emulation system that can be used to inject faults in Static Random Access Memory (SRAM) based Field Programmable Gate Array (FPGA). The FPGA is used to implement an avionics system for a small satellite. The fault injector emulates the expected Single Event Upset (SEU) rate as it would be in the Low Earth Orbit (LEO) of the polar orbiting satellites at inclinations close to 98° deg., and altitude of about 670 km. The emulator injects faults in the configuration bit-stream of the FPGA without stopping its operation. It makes use of the partial reconfiguration feature of today's FPGAs. This provides a facility to assess the design performance in space even if radiation testing will not be conducted before launching. Also, it simulates the expected upset rate and hence calculates the corresponding data failure rates for Triple Modular Redundancy (TMR) fault tolerant designs. The system was implemented using the Xilinx Virtex- LX50T FPGA. The FPGA suffered system failures during the fault injection test. It recovered about 50% of the failures. TMR simulation at an upset rate of 0.1 upsets (per bit per second) for a data size of 2048 bits showed that about 33% of the faults will be fully corrected.

Key Words: FPGA, SEU, Avionics Systems, TMR, Fault Tolerance

1. Introduction

Design of fault tolerant systems for space applications uses redundancy in implementation. Redundancy can be in software code, hardware units, and time of execution and data bits. The protection techniques can be used individually or concatenated. They add to the improvement of the system capability in detecting and correcting faults hence increasing its reliability, however, they also add overhead.

It is often required to assess the reliability of fault tolerant systems in operating conditions close to the environment where they will be used. Satellites are tested in electrical, thermal vacuum, mechanical and radiation conditions as close as possible to the target orbit.

Radiation testing at proton accelerators is expensive, not readily available and needs complicated setups. The purpose of radiation testing is to evaluate how the design will perform in the space radiation environment. The common tests include Single Event Upsets (SEE) and Total Ionization Dose (TID). The Single Event Upset (SEU) is part of the SEE where the logic values of the bits stored in the processor registers and memory cells are altered. This might lead to malfunctions and inappropriate operations. In SRAM-based FPGAs, where the design is stored in the internal SRAM after being loaded from the boot-up flash, bit alteration due to SEU can be severe. It might lead to changing the functioning logic and complete failure of the system.

This paper emulates and simulates the effects of SEUs as they would be found in LEO orbits at an altitude of approximately 670 km and inclination of about 98° deg. The purpose is to develop a complete LEO SEU radiation environment emulator that can be used in fault injection in SRAM-based FPGA avionic systems designs. We hope that this work would save the proton accelerator tests and provide simple and confident test techniques.

In the following sections the paper introduces the SEU fault injection concept in section 2, the SEU fault injector is presented in section 3, the emulation results and discussion are presented in section 4, and the conclusion and future work are presented in section 5.

2. SEU Fault Injection Concept

Fault injection in functioning systems is a technique used to insert deliberate faults at selected and/or random units of the design to assess its sensitivities. This technique is implemented by adding additional hardware and software to the system to handle the insertion of faults, monitoring of performance and collection of results. Figure 1, shows the architecture of a fault injection system.

The design under test is interfaced to a faults insertion unit which has access to the design units where faults are to be injected. The faults vector calculation and generation unit prepares faults vectors that match the required test objectives. The fault insertion unit can be a combination of hardware and software. It handles the overriding of the normal operation into a faulty one. For example, the fault insertion unit can be a code that reads back a previously calculated value by the normal DUT code and then overwrites with a faulty value to simulate a specific condition. The insertion can be done without stopping the main operation. In some designs it might be inevitable to interrupt the normal operation flow by suspending it and then resuming after the injection takes place. The function monitoring and control unit takes care of monitoring the operation of the DUT. It stops the DUT operation in case of noticing an emergency and provides a control path to set the DUT in specific operating modes and operation settings. The performance of the DUT is statistically analyzed to detect anomalies in normal operation as faults are injected. The feedback about how the DUT behaves while in fault injection mode is provided to the faults vector calculation and generation unit. It uses that information in generating new fault vectors. For example, the feedback statistical information might show that there is a repetitive pattern in the output when certain fault sequence is followed. The faults vector generation and calculation unit might repeat the vectors with different variations to study the statistical dependence between injected faults and output vectors. Fault monitoring and control unit also feedback the faults vector calculation and generation with information about the behavior of the DUT during the fault injection process. For example, it might be necessary to feedback the faults vector calculation and generation unit with the moments where the system completely stopped working and needed a deep reset. This information can be used in detecting the types of faults that lead to total failure.

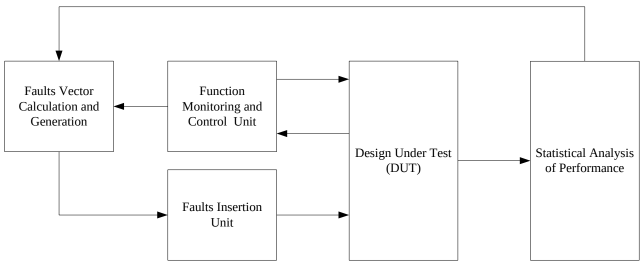

Faults Vector Calculation and Generation
Function Monitoring and Control Unit
Faults Insertion Unit
Design Under Test (DUT)
Statistical Analysis of Performance

Fig. 1. Fault injection cycle. The faults are injected to the DUT and statistical results are issued as a feedback to the injecting machine for test vectors adjustment.

The SEUs which occur in space are probabilistic. Poisson distribution is used to estimate the expected number of upsets (k) which happens in the time interval (T) with an average number of upsets (μ) according to the probability density function shown in Eq. (1) [1][2]. The exponential distribution is used to estimate the expected time between upsets (τ) with an average number of upsets in unit time interval (λ) as shown in Eq. (2). The relationship between both distributions can be set as (μ = λT).

$$P(x|\mu) = (\mu^x/x!)\, e^{-\lambda} \quad (1.)$$
$$P(\tau|\lambda) = \lambda e^{-\lambda\tau} \quad (2.)$$

The SEU rate can be estimated using the Cosmic Ray Effects on Micro Electronics (CREME) model [3]. The Space Environment Simulator web tool [4] is used to draw the estimations [4]. Figure 2, shows the SEU estimation for a 670 km with an inclination of 98° deg. The peaks in the figure are related to upsets taking place at the South Atlantic Anomaly (SAA). The upset rate estimation is based on the values of the radiation testing of the Xilinx Virtex 5 LX50 FPGA [5-8]. Fault Injection rate is estimated by using the per bit upset rate from the SPENVIS simulation shown in Figure 2. Faults are injected to the FPGA design using an IP core provided by Xilinx called the SEU controller [6].

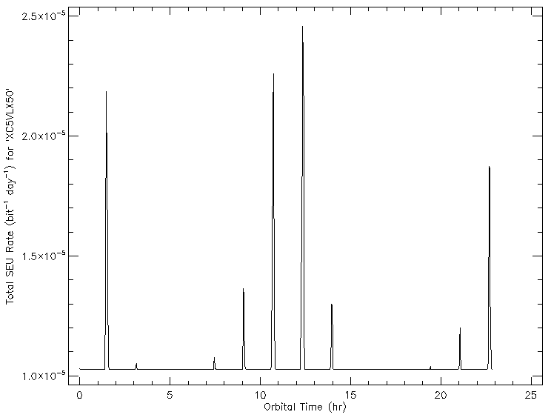


Fig. 2. SEU rate during 1 day of flight. The peaks are orbital positions corresponding to the South Atlantic Anomaly.

3. SEU Fault Injector

The fault Injector that is used during the test setup is shown in Figure 3. The injector uses an internal hardware unit that can reconfigure the FPGA bit stream, the SEU controller. It is an IP core that is provided by Xilinx which can be controlled from outside the FPGA to produce faults in the form of bit flipping in the FPGA configuration frame. The control of the SEU controller is through serial communication over the RS232 channel to send commands to it and receive response .

The fault injector system contains 3 external computers to support its function. The fault injector computer which runs MATLAB script to generate random faults based on the Poisson distribution of the SEUs in the target orbit. It generates the timing at which faults will be injected which follows the exponential distribution as described earlier.

Another computer is used for configuring the FPGA with the bit-stream which contains the hardware design. The design that is being used here consists of four cores of the Microblaze processor which runs together to form the avionics system of a small satellite. The cores exchange data with each other through the Fast Simplex Link (FSL) bus. This a peer to peer direct communication between the Microblaze processors.

The function monitoring of the processors is done through sending the processors status and results of executing a simple counter program to the UART interfaces which are monitored by an external computer to collect the results and analyze them. The system runs the simulation for number of times and it generates a new fault injection vector at each time. the fault injection vector contains the bit location that will be flipped which is a random number from (0 t0 1311) and the frame number where flipping will take place which is a random number from (1 to 8662). The faults are accumulated and their effects are watched as they are injected. At the end of the injection cycle an auto correction mode is enabled to recover the injected faults and restore the operation of the cores. The flow chart in Figure 4, shows the test flow. The detection mode is used during accumulated fault injection.

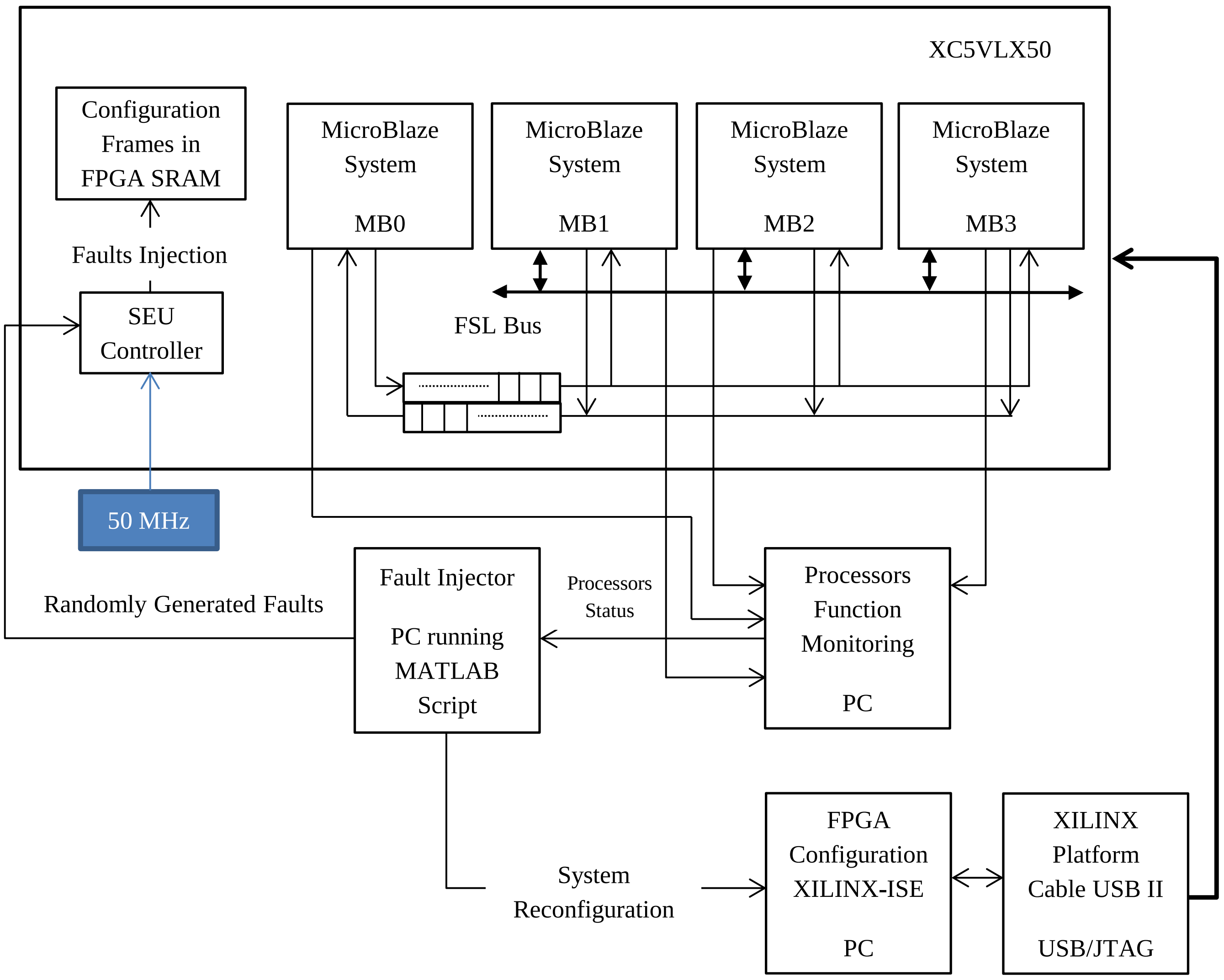
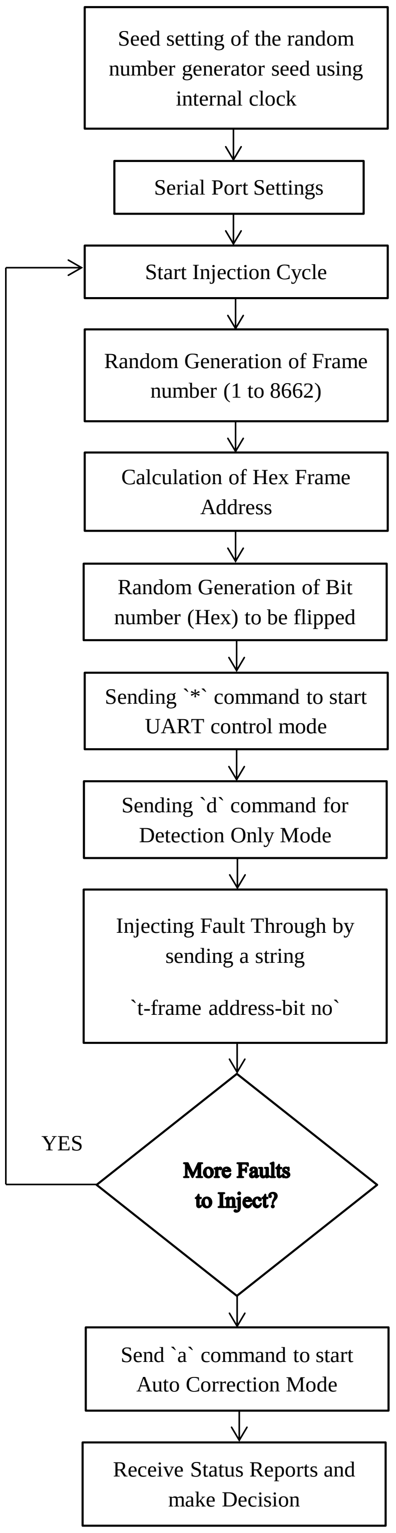


Fig. 3.   SEU Fault Injector Setup.



Seed setting of the random number generator seed using internal clock
Serial Port Settings
Start Injection Cycle
Random Generation of Frame number (1 to 8662)
Calculation of Hex Frame Address
Random Generation of Bit number (Hex) to be flipped
Sending `*` command to start UART control mode
Sending `d` command for Detection Only Mode
Injecting Fault Through by sending a string
`t-frame address-bit no`
YES
More Faults to Inject?
Send `a` command to start Auto Correction Mode
Receive Status Reports and make Decision

Fig. 4. Test flow Setup. The faults are injected using the commands over serial interface with the SEU controller. The results are collected over the serial interface with the processors cores. The auto-correction mode is enabled at the end of operation to recover all the injected faults.

4. The emulation Results and Discussion

The results of running the simulation for 10 times in two sizes of batches: 50 accumulated errors and 100 accumulated errors are shown in Figure 5.

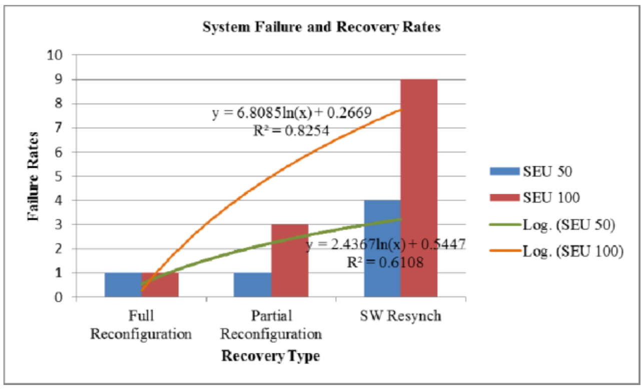


Fig. 5. Fault Injection Results.

Three types of correction can take place: the full reconfiguration, the partial reconfiguration and the Software resynchronization. The full reconfiguration is the mode where the FPGA stopped working due to fault injections. The entire bit-stream of the design should be reloaded to the internal SRAM in order to restore the correct operation. The partial reconfiguration is the mode where one or more processor stopped working but not the whole system. The system can be partially reconfigured without stopping the other processors to restore the operation. The Software resynchronization is the mode where the software of the working processors need to be resynchronized to the same operation after one or more processors stop working and then resumes again.

The results show that about 10% of the injected faults in the 50 faults batch and 10% of the 100 faults batch needed full reconfiguration. Another 10% of the faults in the 50 faults batch needed partial reconfiguration while 30% of the faults injected in the 100 faults batch needed partial reconfiguration. This means that in the 50 faults batch, only 80% of the injected faults where totally recovered through the auto-correction mode without the need for partial or full reconfiguration. In the case of the 100 faults batch, 60% of the injected faults were fully recovered with no need of any reconfiguration. The software resynchronization takes place whenever a partial reconfiguration is initiated or a processor stops operation then resumes after the auto-correction mode has been enabled.

Figure 6-8, shows the results of applying the fault injection over a packet size of 2048 bits in a TMR operation. The packet contained a random vector of data and the vector is compared between three of the operating cores after faults were injected randomly in it. The vectors are compared value by value in an TMR operation through a voter in the fourth processor. The results shown in figure 6, are the log plot for the different upset rates versus the residual failures. This is the condition where the data from the three processors is indifferent and no consensus can be found among it or it is similar but still incorrect.

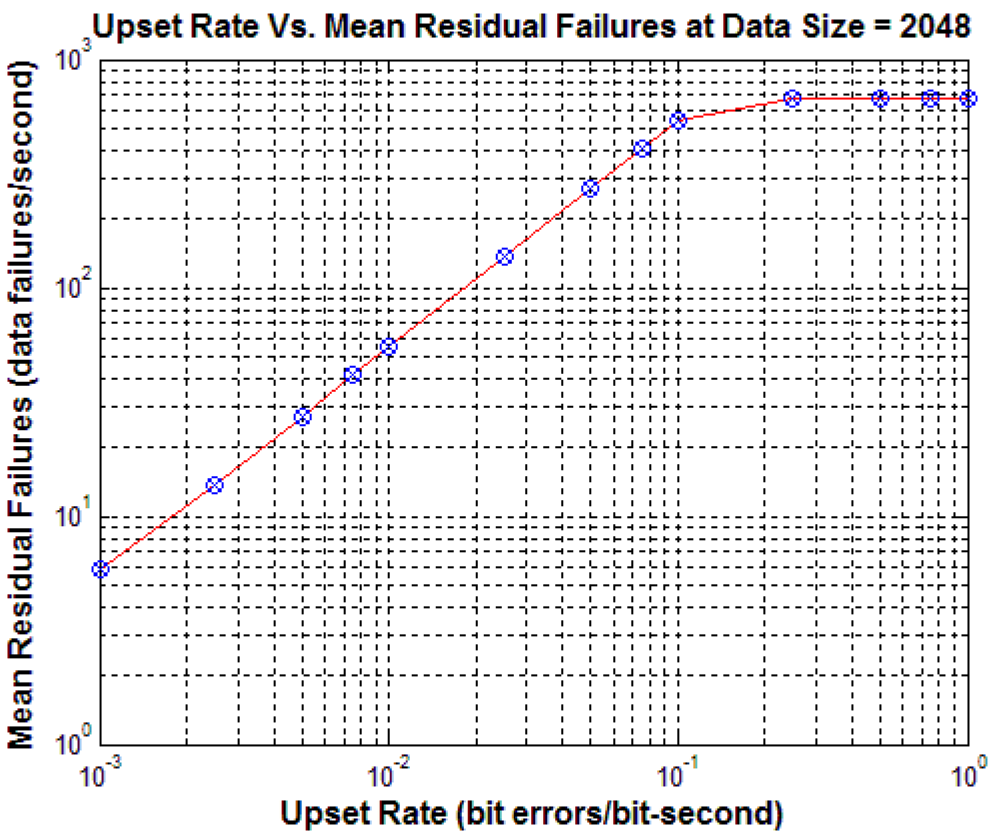


Fig. 6. Log plot of the residual failure versus the upset rate. The residual failures are the failures which still remain in the data even after being corrected or the failures that can not be corrected at all. Failures can still be remaining in the data even after correction when the faults are injected in the same bit positions in different words giving mistakenly similar data values in two or more data sets out of the three data sets to be compared.

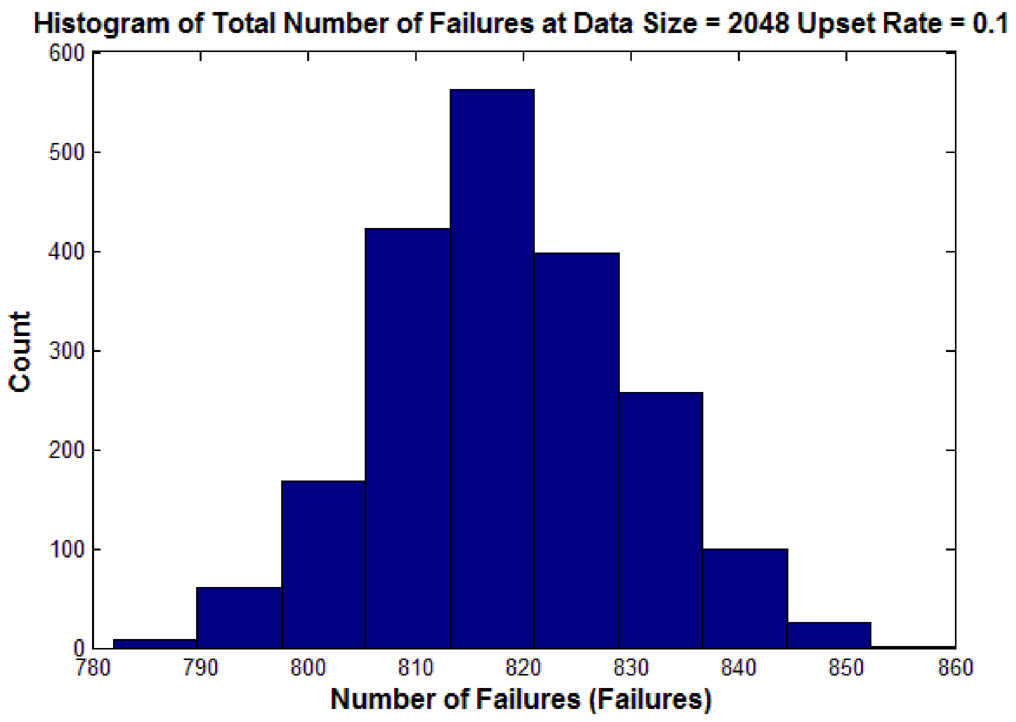


Fig. 7. Histogram of faults injection in the data vector.

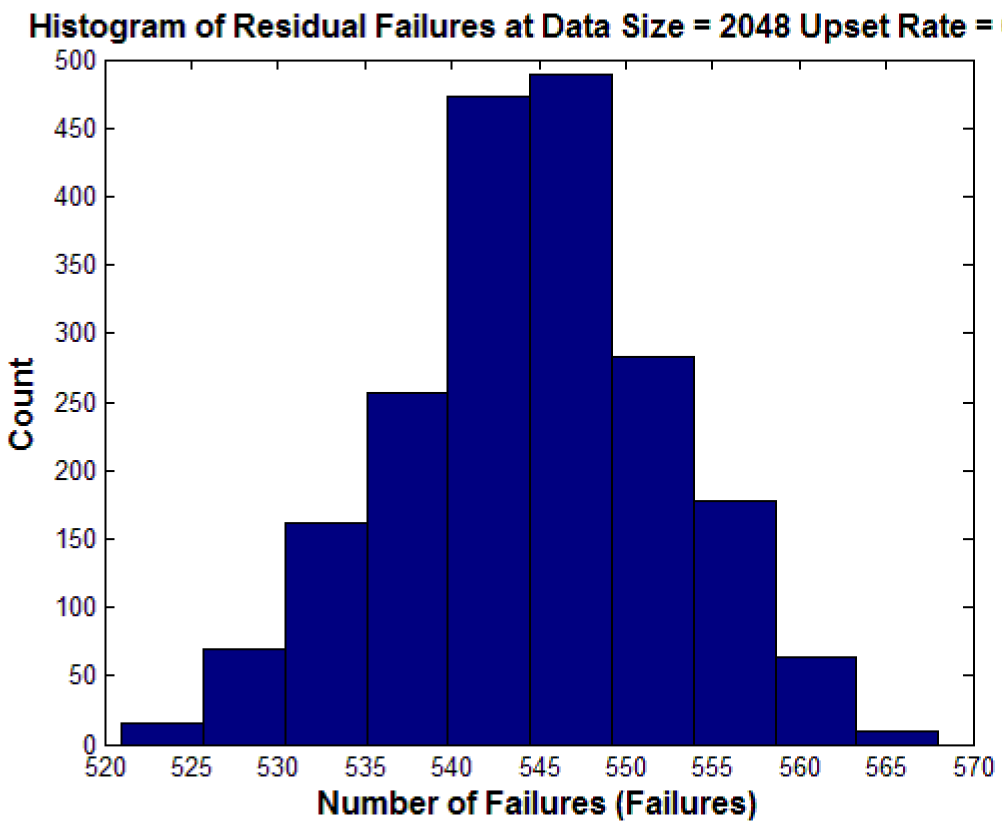


Fig. 8. Histogram of the residual failures after applying the TMR voting and correction procedure.

5. Conclusion and Future Work

This paper presented a fault injection emulator that can be used for injecting random faults in the FPGA bit-stream to simulate the effects of the space environment. About 10% of the injected faults in the hardware bit-stream needed full reconfiguration. In the case of data fault injection at an upset rate of 0.1 upsets per bits per second, more than 50% of the data will have residual failures. We recommend to study the effects of faults injection on many fault tolerant designs to asses their reliability and improve the injector by having more injection capabilities.

References

[1] Actel website: http://www.actel.com

[2] Xilinx website: http://www.xilinx.com

[3] Adams, J. H., Jr., Cosmic Ray Effects on MicroElectronics, Part IV, NRL Memorandum Report 5901, 1986.

[4] SPENVIS website: http://www.spenvis.oma.ba

[5] Quinn, H.; Morgan, K.; Graham, P.; Krone, J.; Caffrey, M.; , "Static Proton and Heavy Ion Testing of the Xilinx Virtex-5 Device," Radiation Effects Data Workshop, 2007 IEEE , vol.0, no., pp.177-184, 23-27 July 2007.

[6] Ken Chapman, "New Generation Virtex-5 SEU Controller," Xilinx, Version A.2 – s4th November 2009.

[7] Xilinx, Virtex-5 FPGA Configuration User Guide, Xilinx UG191 (v3.10), 2011.

[8] Ken Chapman, "SEU Strategies for Virtex-5 Devices," Xilinx, XAPP864, April, 2010.