A Kind of Low-cost Non-intrusive Autonomous Fault Emulation System

Qiang Zhang (Corresponding author)
Institute of Precision Guidance and Control, Northwestern Polytechnical University
PO box 250, Xi'an 710072, China
E-mail: iamadog3333@163.com

Jun Zhou
Institute of Precision Guidance and Control, Northwestern Polytechnical University
PO box 250, Xi'an 710072, China
E-mail: zhoujun@nwpu.edu.cn

Xiaozhou Yu
Institute of Precision Guidance and Control, Northwestern Polytechnical University
PO box 250, Xi'an 710072, China
E-mail: yuxiaozhou@nwpu.edu.cn

The research is financed by PhD Research Fund from Chinese Ministry of Education. No. 20070699004

Abstract

SRAM-Filed Programmable Gate Arrays (FPGA) have become one of the most important carriers of digital electronic system because of its many inborn advantages. However, as manufacture of Integrated Circuit evolves towards Very Deep Sub-Micron technology, FPGA designers must be careful of circuit's Single Event Upset (SEU) susceptibility when used in hostile environment, such as avionics and space applications where reliability is vital. We proposed a SEU-fault emulation platform to evaluate circuit's SEU mitigation performance. The platform does not need any external circuit or micro controller to manage fault emulation process compared with existing approach. Source codes of Circuit Under Test (CUT) do not need to be modified or intruded with any component. It is a non-intrusive testing. Communication between host-computer and emulation board is minimized to accelerate fault injection speed. Experimental result shows that a single fault injecting (including Multi-Bits-Upset) only costs 29us. A circuit state reloading technology is exploited to increase emulation efficiency. Moreover, in the field of evolvable hardware, genetic operations can be reconfigured and its fitness can be evaluated on-line using the proposed fast dynamic reconfiguration method, which is useful for implementing self-repair and self-evolutionary hardware.

Keywords: Electronic Reliability, Fault Injection Emulation, FPGA, SEU

1. Introduction

With the advantages of powerful performance, flexibility and low-cost, SRAM-based FPGAs have become important electronic system carriers. Thanks to reprogrammable FPGAs, electronic systems can be reconfigured easily. However, as Integrated Circuit manufacture evolves towards 40nm technology, ICs are more susceptible to space or artificial radioactive impact. These failures are modeled as Single Event Upset which means one bit in circuit flipped. Unfortunately, SRAM-based FPGAs are more prone to SEU than Application Specified Integrated Circuits (ASICs). Designers must handle radiation-induced SEUs carefully in cases such as avionics and space electronic application where circuit faces spiteful environment and reliability is vital. (Y. Bentoutou, 2009, p. 1843-1845).

In electronic systems that demand high reliability, one of the most crucial problems is to evaluate design circuit's SEU susceptibility, fault-tolerant ability (M. Jallouli, 2009, p. 2549-2557) and predict its failure probability (C. Lopez-Ongil, 2007, p. 946-950). Circuit's SEU susceptibility evaluation is necessary and it brings the following

benefits. Firstly, through fault injection emulation we can find frail area and introduce harden technologies on the area. Secondly, we can confirm design circuit's functionality through evaluation and find out whether the technologies used to harden circuit are efficient. For these purposes, we need a tool to evaluate design circuit's SEU susceptibility fast and effectively. The tool could assist designers in improving design circuit's dependability, accelerating design process and analyzing circuit in-depth and intensively ensuring design's reliability.

Fault injection technology was widely accepted in the analysis of IC's SEU susceptibility. There are several approaches proposed. One classic fault injection way is to expose chip under artificial radioactive environment. FPGA was exposed to radiation by A. Ceschia (2004, p. 328-328) so as to investigate SEU susceptibility of configure memory cell. This approach creates an artificial radioactive environment almost the same as the real application circumstance, but expensive radioactive equipment and a chip with implemented design circuit are required. So, it is generally used to confirm hardened circuit design at the final step but not at early design phase. Some other approaches, such as laser fault injection and electromagnetic interference have the same problem (C. Lopez-Ongil, 2007, p. 252-261).

For evaluating SEU susceptibility before circuit manufacture, E. Jenn (1994, p. 66–75) and V. Sieh (1997, p. 32–36) exploited HDL simulator to perform a traditional software-based fault injection. Designers can scrutinize circuit state of every wire and every flip-flop with this method. Designers are required to inject fault by modifying HDL model. The modification should simulate all possible SEU faults but not change the behavior of original circuit. That is not an easy work. Nowadays a single FPGA contains millions of gates. If you want to get an overall analysis of SEU susceptibility by injecting numerous faults, software-based simulation approach is very time-consuming.

FPGA-based fault injection emulation has been used to accelerate fault injection experiments recently. FPGAs can be used for large scale circuit fault injection emulation since a single FPGA contains millions of logic gates. FPGA-based emulation includes the following steps: generate faults, inject faults, monitor the response of Circuit Under Test, classify faults and collect experimental results.

FPGA-based emulation has two methods for fault injection.

1) Modify Circuit Under Test and add additional logic as a tool to change the value of memory cells which are needed to inject fault. The process of circuit modification is named instrumentation and the additional added component is instrument (Mohammad, 2008, p. 143-149).

2) Reconfigure FPGA partially or as a whole to change the Circuit Under Test to a fault state (M. Lanuzza, 2009, p. 74-84). Some register value represents a certain state. SEU faults are emulated by changing the register value and monitoring circuit's response.

Intensive communication, however, is needed between emulation board and host computer during emulation process in both the two methods (A. Ejlall, 2008, p. 319-328). Host computer manages injection process and evaluate every fault. The intensive communication between host computer and emulation board brings a bottleneck in performance. These FPAG-based emulations could not make full use of its speed advantage. P. Civera (2001, p. 9-13) exploited the first method that additional combinational and sequential circuit were used to implement fault injection. This method performs relatively fast. Extra chip area is required because additional flip-flops are used at every place where fault injection will perform. It works in an intrusive way that the original circuit codes are modified. Transient fault injection was supported in (L. Antoni, 2002, p. 245-253). This scenario was based on a FPGA reconfiguration for every fault. It resulted as a time-consuming method. P. Kenterlis (2006, p.235-241) proposed an automated low cost hardware/software platform for performing fault emulation experiments targeting SEUs in the configuration bits of FPGA devices. A method for reducing the fault list by removing the faults on unused LUT bit positions was presented. Experiment result showed that FPGA-based emulation performs at least two orders of magnitude faster than simulation. XHWIF interface port and external SelectMAP configuration port were required to perform fault injection in this platform. Our approach, however, does not need external controller to manage SelectMAP port or XHWIF port (M. Berg, 2008, p. 2259-2266). It is a succinct platform that all components are in side a single FPGA to run high performance fault injection emulation.

This study introduces a low cost, non-instructive, fully automated platform for efficiently fault injection emulation. FPGA and its Internal Configuration Access Port (ICAP) technology are exploited to implement the platform. It is characterized with the following features.

1) Fast and efficient fault emulation. Functions of fault generation, fault injection, circuit's response

monitoring and fault classification are all performed in emulation board. Experiment results are sent to host computer only at the end of emulation. Communication between host computer and emulation board decreases significantly, which greatly accelerates the emulation speed.

2) Low cost platform. Emulation controller and fault injection management are implemented in a FPGA. No extra Micro-processor and its concomitant circuit are needed.

3) It is a non-destructive and non-intrusive evaluation for Circuit Under Test. There is no need to modify the original codes of Circuit Under Test and that leads to a straightforward emulation. In intrusive method, designers must be careful to avoid the added fault injection logic code changing the behavior of the Circuit Under Test. That may leads to a deviated evaluation.

2. Architecture and Fault Model of FPGA

In this section, we will describe the general architecture model of SRAM-FPGA and have an overview of its internal resources. Based on this model we will discuss how SEU that occurs in FPGA configuration memory affects routing and logic resource.

2.1 FPGA Architecture

Configurable Logic Blocks (CLBs) are main resources for implementing sequential circuit and combinational logics. Different functional designs can be implemented by selectively interconnecting CLBs which contain configurable switch matrix, logic function generator (Look-Up Table LUT), arithmetic logic, carry logic and memory logic. Besides sequential and combinational logic, CLB also can be configured as distributed RAM and ROM. FPGA's floor layout is made up of CLB matrix with many IO Blocks locating around.

Routing resources connect all components in FPGA. They are divided into four categories according to their technology, length, width and located area. The first is global routing resource used for global clock and global set/reset signal. The second is long-line resource used for high speed signal between banks and second global clock. The third is short-line used for basic logic element interconnecting or routing. The last is distributed routing resource used for dedicated clock, reset and control signal. SRAM-based programmable technology employs static RAM cells to control switches or multiplexers and uses Programmable Interconnect Points (PIPs) to interconnect or cut wires.

2.2 SEU Fault Model

Effect of SEU on SRAM-FPGA has been explored by radioactive exposure. P. Bernardi (2004, p. 115-120) analyzed exposure experimental result together with the meaning of every bit in FPGA's configuration. SEU was found to destroy configuration both in logic block and switch block. One configuration bit may control two or more routing resources in some parts of FPGA. Therefore, when a SEU occurs it may change two or more routing sections, which leads to multiple errors. Two kinds of fault location should be considered when handling SEUs in SRAM-FPGA.

(1) Bit-flips exist in Flip-Flops.

(2) Upsets exist on configuration memory that control routing, combinational and sequential logic (LUTs, multiplexers, initial bit in Flip-Flops) (P. Kenterlis, 2006, p.235-241).

It may upset the earlier stored value when a radioactive particle bombs a Flip-Flop or its input control signal. If the upset propagates to other components, it may damage the system. For instance, when a state register of a finite state machine upsets, the error will propagate and the work of the whole system will be disarranged.

Besides, if a SEU exist in a configuration SRAM cell it will change the function of the configured circuit permanently. Hitting of a LUT configuration bits by a radioactive particle will change the output of function generator into another logic gate (or gate combination). See figure 1.

A SEU changes an OR gate into a XOR gate, as the description of figure 1. Such fault models are gate-level models. But it is not realistic to consider all the SEUs occur in FPGA to be structural faults, such as stuck-at and transition faults. These memory cells that control configuration of CLBs are independent of those control routing resources in SRAM-FPGA. As SEUs are the dominant faults in SRAM-FPGA, attempt to model all LUT faults as equivalent gates and inject stuck-at faults will not produce the real faulty effect. So the existing fault simulators using structural fault models is not suitable for simulating SEUs effect in SRAM-FPGA (P. Kenterlis, 2006, p.235-241).

3. Implementation of the Emulation Platform

3.1 Architecture of the Platform

The architecture of our proposed fault injection emulation platform is presented in figure 2. The emulation board contains a FPGA, a local RAM and a communication interface between the board and host computer. The FPGA acts as a central component of the fault emulation system. Besides a Circuit Under Test (CUT), the FPGA also contains a test vectors input module, a ICAP-based fault injection module, a fault classification module and a controller managing emulation process.

(1) Fault emulation controller: It manages the whole process of fault injection and emulation, including initializing circuit, loading input test vectors, faults injection, comparing expected output with actual response of faulty circuit. The emulation controller is implemented by a Microblaze soft-core processor.

(2) Circuit Under Test (CUT): It is the circuit to be evaluated. In our project, the CUT is a totally independent component and it can be any designed digital circuit. The CUT could be a simple gate combinational circuit or a complex system including multi-cores, IP cores and memory. Both the CUT and the fault emulation system are implemented in the top module of FPGA. A tool named floorplan is used to place the CUT module and emulation system module at assigned locations separately. Because fault injection execution is location-based, locations of the two modules must not overlap each other.

(3) ICAP: ICAP is a novel Internal Configuration Access Port integrated in Xilinx FPGA. Run-time partial reconfiguration can be performed through the internal interface without assistance of any other external circuit. It is different from the traditional reconfiguration method using external controller and operating through SelectMAP port. In the proposed platform, for purpose of fault injection, all configuration frames related to CUT are readable and writeable through ICAP.

(4) A local RAM is used to store emulation results during experiment. All the collected data is sent back to host computer for further analysis when emulation ends. By doing this, the fault emulation process is significantly accelerated because there is no need for communication between emulation board and host computer during emulation.

3.2 Fault Injection Emulation Flow

Two independent modules (a CUT and a fault emulation system) are contained in FPGA top module. The two modules are implemented into a single bit stream file. Once FPGA is configured and initialized, we can start fault injection flow, as figure 3 describes.

Fault injection emulation flow includes the following steps:

(1) Initialize the fault injecting emulation platform: including board power up, configure FPGA, boot fault emulation controller and initialize its memory, initialize control program. If the CUT contains a processor or memory, initialize them.

(2) The CUT performs a golden run (a fault free emulation). Response of the golden CUT is stored in fault classification module as a standard reference for sorting each injected fault.

(3) Generate location and time for current fault injection. There are two modes for generating fault injection place and clock number. One mode performs carpet fault injecting into every configuration bit at every clock time from the first to the last. The other performs fault injection at locations and clocks randomly generated, until the execution times are statistically enough. After that, a configuration frame is read back from the assigned location. One bit or multi-bits of the data frame are setup to simulate a SEU or Multi-bits Upset SEU. Then reset the CUT.

(4) The CUT pause when the assigned fault injection clock arrives. Faulty configuration frame is injected in to the assigned location through ICAP immediately. Resume operation of the CUT.

(5) Monitor response of the CUT and compare it with the stored golden run response. Once a disagreement is found, a fault injection error is recorded.

(6) All emulation result data is sent back to host computer for further analysis when the experiment ends.

4. Fault Emulation Results and Analysis

The fault injection emulation system is implemented in a single device. There are some details that need to be noticed when the system is under design. When commercial EDA tools are used to implement the system, designer should keep module's hierarchy during synthesis process and preserve hierarchy on sub-module during translation process because these modules will be assigned to dedicated locations by a tool named floorplan.

Fault emulation system module must not be placed at the location assigned to the CUT, otherwise the system may corrupt during injection. Actually, there may be some redundant circuits in design. So designers should not share resources during synthesis process. Table 1 lists the resources occupied in Xilinx V2P30 device after implementation.

Although the proposed fault emulation platform has powerful performance and works flexibly, it utilizes a small share of device resources. Rest resources of the device all can be used to implement the CUT. Even for a very complex large scale CUT which overflows V2P30, we can transplant the hardware and software of the platform to another FPGA with more capability easily, almost without any modification.

For an overall evaluation of circuit's SEU susceptibility, designer need inject fault into every configure point at every clock. If there are N configure points in the CUT and its work lasts L clocks, N×L faults injection is required. L clocks are needed for a single fault classification. There are totally N×L×L clocks are needed for the emulation process. For complex circuit which has large N and L, it is a quite long time (C. Lopez-Ongil, 2007, p. 252-261). Circuit state reloading strategy is exploited to reduce emulation time. Figure 4 describes the emulation process of a certain configure point. Figure 5 depicts the scenario under circuit state reloading strategy.

When you want to inject a fault into the CUT at a later state, you have to let the CUT perform a fault-free run until it gets the expected state. That seriously decreases the emulation efficiency. We put the fault-free state of the circuit at every clock stored in a memory. Then we can recovery the CUT to any state immediately using partially reconfiguration technology. Once the cost of fault-free run exceed the cost of partially reconfiguration, we just reload the expect circuit state. In figure 5, the blank area represents the time saved by using circuit state reloading strategy. It saves about 50% emulation time when it is used in long clock time CUTs.

For validating efficiency of the proposed platform, several design circuits were implemented for SEU fault injection emulation. In these experimental cases, there are VHDL modules designed by our team (UART, FIR, Counter modules) and module generated by commercial IP tool (CORDIC). The characteristics of the designed circuit are listed in table 2.

According to the responses of CUT, the injected faults are divided into three categories. Failed: CUT tells wrong answer when fault injected and it can not recover the error. Transient error: CUT tells wrong answer when fault injected but it recovers immediately. Silent: CUT works normally without the effect of fault injection during the whole work process. Fault injection emulation results of these CUTs are listed in Table 3.

As the emulation results listed above demonstrate, the proposed emulation platform is able to inject SEU faults into circuit under test efficiently. The time needed for a single One Bit SEU or Multi-Bits SEU fault injection was precisely measured. Experimental result showed that it cost only 29us/fault. Whereas instrumented circuit technique (P. Civera, 2001, p. 2210-2216) has given rates ranging from 100 us/fault and 830 us/fault for short testbenches (less than 1,000 cycles) and long fault lists (100,000 faults). The latter requires an intensive communication between the host computer and the FPGA for every fault emulated.

In this study, response of both failed faults and transient error faults are classified into error response when we calculate response error rate. From the experimental data, the response error percentages of UART, FIR, Counter and CORDIC are 9.24%, 12.62%, 8.46% and 15.60%, respectively. These response error percentages provide precise SEU susceptibility of CUTs to system designers. The designer must make a case-to-case decision to determine whether the CUT is reliable. If it satisfies the reliability request, the designer just integrate the CUT into system. If it does not, other technologies, such as circuit redundancy or scrubbing technology, must be used to harden the CUT (D. R. Blum, 2009, p. 1618-1628), (R. L. Shuler, 2009, p. 214-219).

5. Conclusion

A FPGA-based SEU fault injection emulation platform is introduced to evaluate circuit's SEU susceptibility. FPGA Internal Configuration Access Port is utilized to perform fault injection and external controller is not needed any more, which reduces the complexity and cost of the system. The emulation works in a convenient and non-intrusive way. Designers do not need to intervene original codes of Circuit Under Test. The automated emulation platform significantly decreases communication between the board and host computer. Efficiency of fault injection increased and experimental result showed that a single fault injection costs only 29us. Several circuit fault emulations were performed to demonstrate the validity of the platform. Apart from the use of evaluating digital circuit's SEU susceptibility, the proposed platform can also be applied to the field of evolvable hardware (EHW). Evolvable hardware focuses on development of automated electronic circuit design, or a system capable of adaptive alteration according to environment. EHW can be used in fields such as design automation, controllers for autonomous mobile robots, and wireless sensor network nodes (Y. Aihong, 2008, p.

436-441). Self-evolutionary hardware completes genetic operations and fitness evaluation in an on-chip processor. The on-chip processor in the proposed platform can perform these tasks and evolvable hardware is generated on the same device. Based on the fast dynamic reconfiguration technology presented in the paper, processor can execute evolvement arithmetic and configure the generated chromosomal encoding of circuits into hardware efficiently. Consequently, fitness evaluation can be performed on-line which is quite helpful for self-repair or self-adaptive hardware.

References

A. Ceschia, A. Violante, M. S. Reorda, A. Paccagnella, P. Bernardi & M. Rebaudengo. (2004). Identification and classification of single-event upsets in the configuration memory of SRAM-Based FPGAs. *Ieee Transactions on Nuclear Science*. vol. 51, no. 2, pp. 328-328.

A. Ejlall, & S. G. Miremadi. (2008). Error propagation analysis using FPGA-based SEU-fault injection. *Microelectronics Reliability.* vol. 48, no. 2, pp. 319-328.

C. Lopez-Ongil, L. Entrena, M. Garcia-Valderas, M. Portela, M. A. Aguirre & J. Tombs. (2007). A unified environment for fault injection at any design level based on emulation. *Ieee Transactions on Nuclear Science*. vol. 54, no. 4, pp. 946-950.

C. Lopez-Ongil, M. Garcia-Valderas, M. Portela-Garcia & L. Entrena. (2007). Autonomous fault emulation: A new FPGA-based acceleration system for hardness evaluation. *Ieee Transactions on Nuclear Science*. vol. 54, no. 1, pp. 252-261.

D. R. Blum, & J. G. Delgado-Frias. (2009). Delay and Energy Analysis of SEU and SET-Tolerant Pipeline Latches and Flip-Flops. *Ieee Transactions on Nuclear Science.* vol. 56, no. 3, pp. 1618-1628.

E. Jenn, J. Arlat, M. Rimen, J. Ohlsson & J. Karlsson. (1994). Fault injection into VHDL models: the MEFISTO tool. In *Proc. FTCS-24, Int. Symp. Fault Tolerant Computing*. pp. 66–75.

L. Antoni, R. Leveugle & B. Feher. (2002). Using run-time reconfiguration for fault injection in hardware prototypes. *In Proc. 17th Ieee International Symposium on Defect and Fault Tolerance in Vlsi Systems.* pp. 245-253.

M. Berg, C. Poivey, D. Petrick, D. Espinosa, A. Lesea & K. A. LaBel. (2008). Effectiveness of Internal Versus External SEU Scrubbing Mitigation Strategies in a Xilinx FPGA: Design, Test, and Analysis. *Ieee Transactions on Nuclear Science*. vol. 55, no. 4, pp. 2259-2266.

M. Jallouli, C. Diou, F. Monteiro, A. Dandache, H. Belhadaoui & O. Malasse. (2009). Evaluation of important reliability parameters using VHDL-RTL modelling and information flow approach. *Safety, Reliability and Risk Analysis: Theory, Methods and Applications*. Vols 1-4, pp. 2549-2557.

M. Lanuzza, P. Zicari, F. Frustaci, S. Perri & P. Corsonello. (2009). An Efficient and Low-Cost Design Methodology to Improve SRAM-Based FPGA Robustness in Space and Avionics Applications. *Reconfigurable Computing: Architectures, Tools and Applications - 5th International Workshop*, Arc 2009, Lecture Notes in Computer Science J. Becker, R. Woods, P. Athanas et al., eds., 2009, pp. 74-84.

Mohammad Shokrolah-Shirazi & Seyed Ghassem Miremadi. (2008). FPGA-based Fault Injection into Synthesizable Verilog HDL Models. In *Proc. The Second International Conference on Secure System Integration and Reliability Improvement*. pp. 143-149.

P. Bernardi, M. S. Reorda, L. Sterpone & M. Violante. (2004). On the evaluation of SEU sensitiveness in SRAM-based FPGAs. *In Proc. 10th Ieee International On-Line Testing Symposium.* pp. 115-120

P. Civera, L. Macchiarulo, M. Rebadengo, M. S. Reorda & M. Violante. (2001). Exploiting FPGA for accelerating fault injection experiments. *In Proc. 7th Ieee International on-Line Testing Workshop.* pp. 9-13.

P. Civera, L. Macchiarulo, M. Rebaudengo, M. S. Reorda & M. Violante. (2001). Exploiting circuit emulation for fast hardness evaluation. *Ieee Transactions on Nuclear Science*. vol. 48, no. 6, pp. 2210-2216.

P. Kenterlis, N. Kranitis, A. Paschalis, D. Gizopoulos & M. Psarakis. (2006). A low-Cost SEU Fault Emulation Platform for SRAM-Based FPGAs. *In Proc. 12th Ieee International On-Line Testing Symposium*. pp.235-241.

R. L. Shuler, B. L. Bhuva, P. M. O'Neill, J. W. Gambles & S. Rezgui. (2009). Comparison of Dual-Rail and TMR Logic Cost Effectiveness and Suitability for FPGAs With Reconfigurable SEU Tolerance. *Ieee Transactions on Nuclear Science.* vol. 56, no. 1, pp. 214-219.

V. Sieh, O. Tschäche, & F. Balbach. (1997). VERIFY: evaluation of reliability using VHDL-models with

embedded fault descriptions. *Proc. 27th Int. Symp. Fault Tolerant Computing*. pp. 32–36.

Y. Aihong, Z. Guoyin & G. Lin. (2008). A survey of dynamically and partially reconfigurable FPGA-based self-evolvable hardware. *CAAI Transactions on Intelligent Systems*. vol. 3, no. 5, pp. 436-441.

Y. Bentoutou & M. Djaifri. (2009). Observations of Single-Event Upsets and Multiple-Bit Upsets in Random Access Memories On-Board the Algerian Satellite. *2008 Ieee Nuclear Science Symposium and Medical Imaging Conference, Ieee Nuclear Science Symposium - Conference Record*. pp. 1843-1845.

Table 1. Resources Used by Emulation System

	Slices	Flip Flops	4- LUTs	MULT18X18s
Number	1536 out of 13696	1776 out of 27392	2436 out of 27392	17 out of 136
Percentage	11%	6%	8%	2%

Table 2. Characteristics of the Designed Circuit

Circuit	Input	Output	Slice	Flip-Flop	4-LUT
UART	2	1	100	121	173
FIR	9	8	17	23	16
Counter	1	4	20	31	11
CORDIC	50	33	699	1,176	1,270

Table 3. Circuit Emulation Results

Circuit	Injected faults	Failed	Transient Errors	Silent
UART	90,376	3,243	5,104	82,029
FIR	13,904	1,264	1,376	11,273
Counter	20,856	965	799	19,092
CORDIC	611,776	35,724	59,714	516,338

Figures:

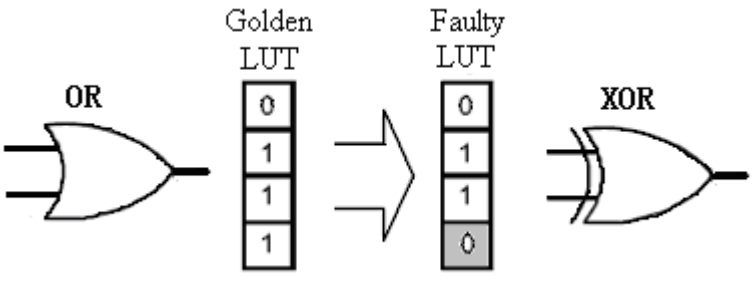

Figure 1. SEU Effect on Configuration Bits

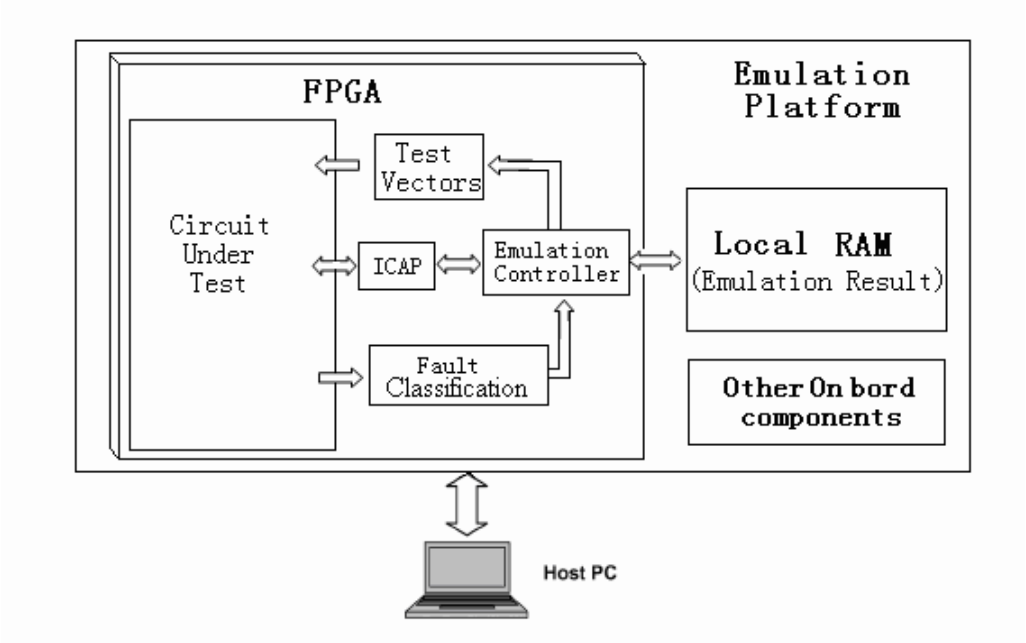

Figure 2. Architecture of Fault Emulation Platform

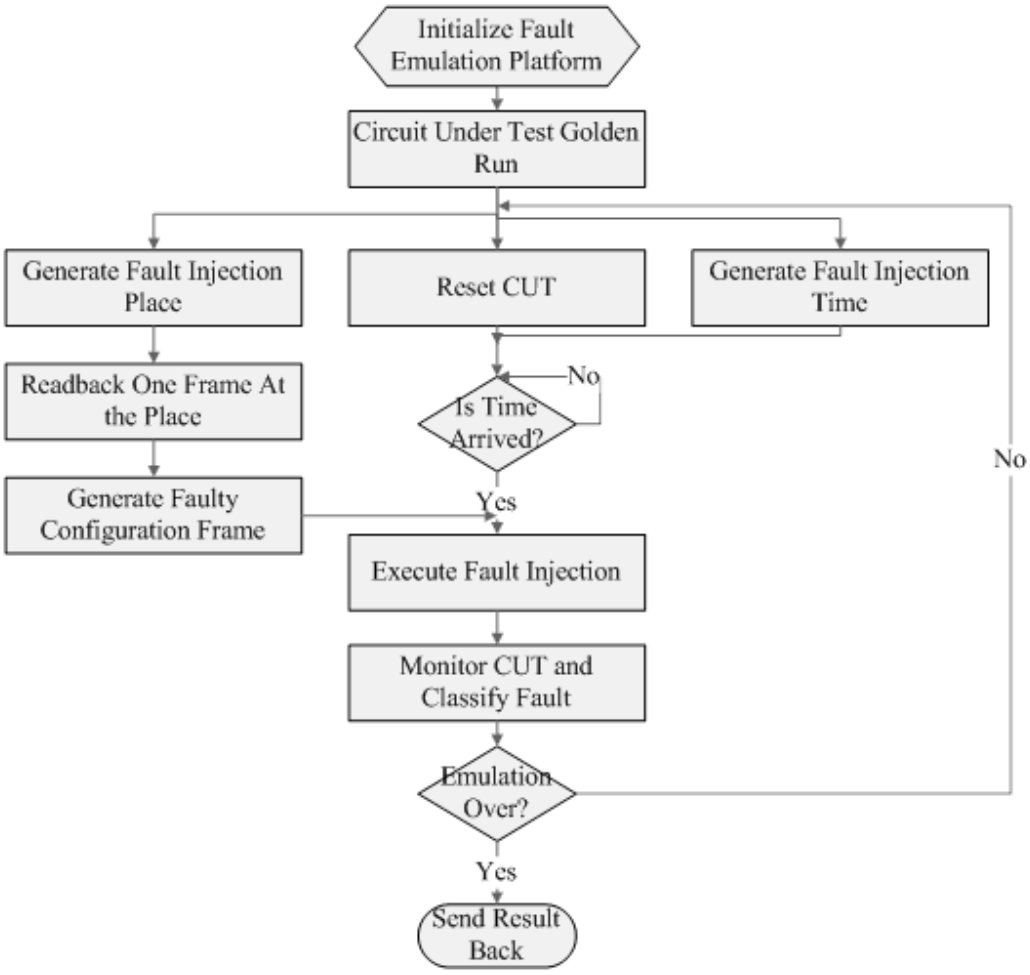

Figure 3. Fault Injection Emulation Flow

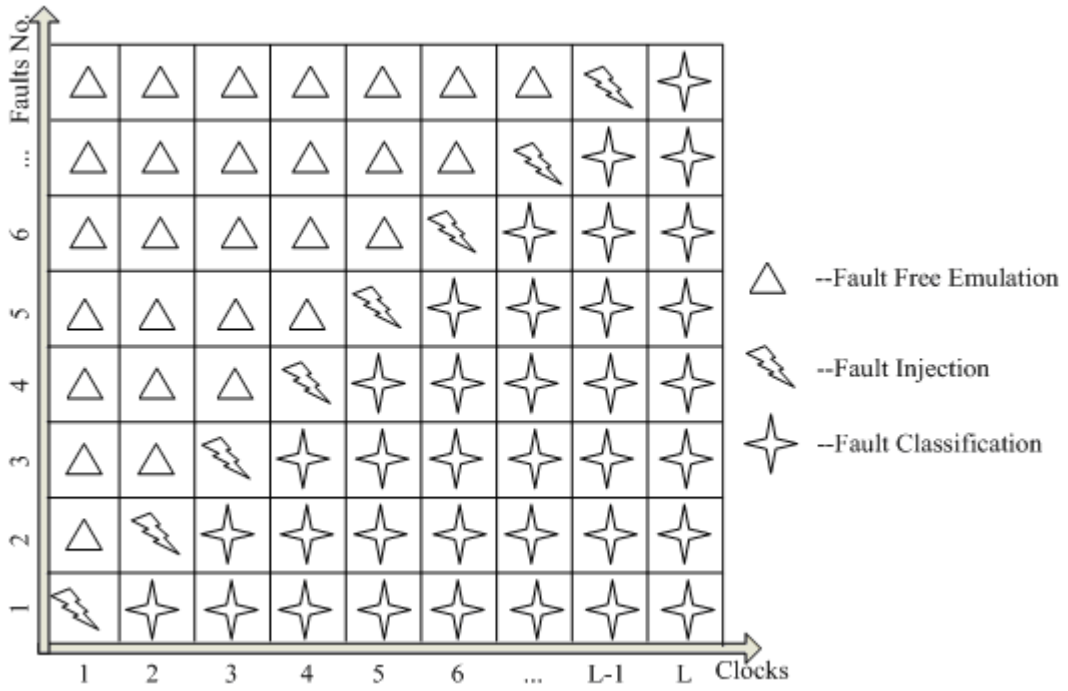

Figure 4. General Emulation Process of a Configure Point

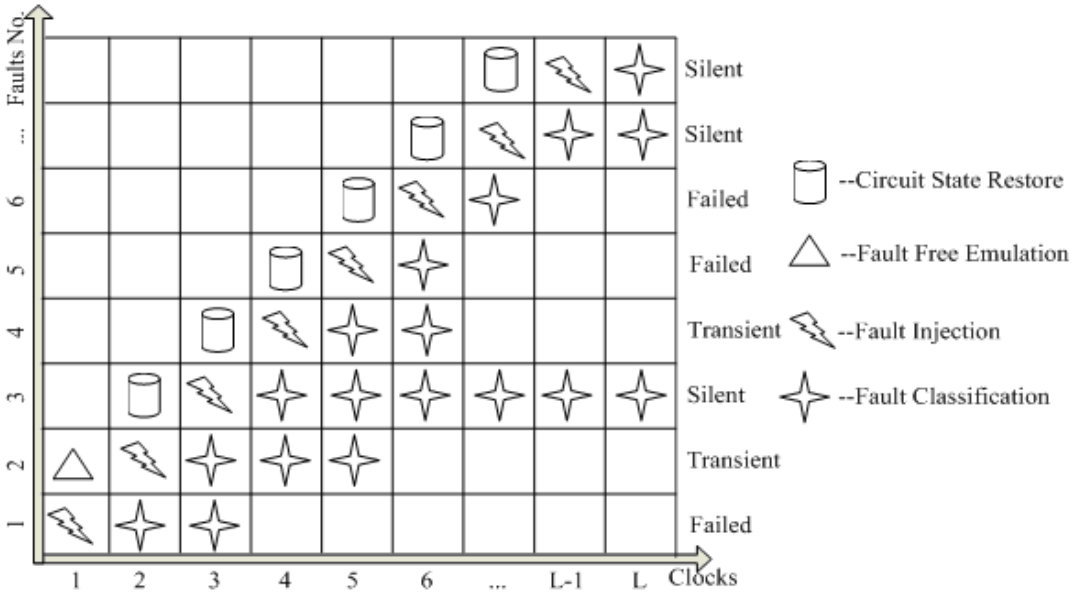

Figure 5. Emulation Process in Circuit State Reloading Scenario

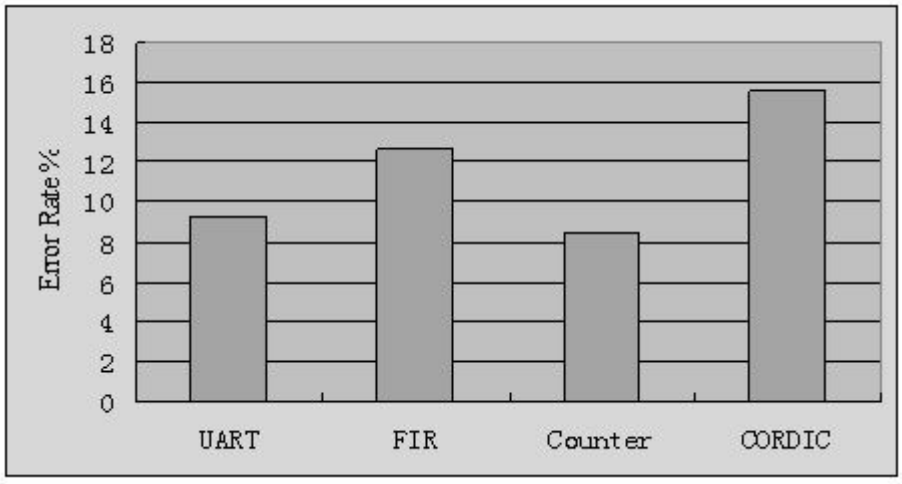

Figure 6. Response Error Rate of CUTs