IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 10, Issue 3, March 2022

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.165

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165

|| Volume 10, Issue 3, March 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1003006 |

Configuration of SRAM-Based FPGAs (A New Fault Injection Approach to Study the Impact of Bitflips)

Prof. Dr.Kamal .M. Alaskar

Professor, Department of Computer Applications, Bharati Vidyapeeth (Deemed to be University) Institute of Management, Kolhapur, India

ABSTRACT: A new method for injecting faults in the configuration bits of SRAM-based FPGAs is proposed. The main advantages over previous methods are its ability to simultaneously inject several faults or bit-flips in the FPGA by "pipelining" the fault injection process. The design to be tested is divided into modules. The first step in the fault injection technique would be inserting one fault in each of the modules and observing the potential misbehavior of these modules. In thesecond step the effects on the whole system of the misbehavior of the module are independently evaluated. Using this technique makes possible to inject several faults when reconfiguring the FPGA with the faulty bitstream, while other techniques were able to insert only one fault on each reconfiguration. Thus the speed in which faults are injected is significantly increased and the time needed to conduct the experiment is shortened. A simulation is described to validate the new fault injection process.

KEYWORDS: FPGA, Fault injection techniques, SEU, and fault tolerance.

I. INTRODUCTION

Although FPGAs are becoming increasingly popularand thus potential candidates to a large scope of applications, they are still considered disqualified to be used in critical applications like aeronautics and space instruments. The SRAM-based FPGAs suffer from their vulnerability to the effect of radiation. Particularly, the so called Single Even Upset (SEU) phenomenon may result in the modification (bit flip) in the configuration bits that control the routing, logic behavior and other critical aspects of the FPGA designs potentially leading to a drastic misbehavior of the FPGA. Radiation hardened FPGAs have been introduced by manufacturers in order to overcome this vulnerability. Such FPGAs are significantly moreexpensive than the commercial ones.

An alternative solution is the introduction of design redundancy techniques, mainly Triple Modularly Redundancy (TMR) techniques, in commercial FPGAs. In order to evaluate the efficiency of such techniques, fault injection experiments must be conducted. These experiments range from exposure to radiation or other disturbances, software simulation and fault injection by reconfiguring the FPGA.

To evaluate the performance and the fault tolerance of a design meant to be implemented on an FPGA,emulation of the faults is practically useless since the emulation will only be able to test the faults that occur on the design level rather than the hardware level. To overcome this, an FPGA can be exposed to radiation to inject faults at the hardware level of the design; however such experiments are time consuming andlack the ability to control the exact amount and location of the faults, which is very crucial in such experiments. An alternative solution is to inject faults in the FPGA at the hardware level by inserting bitflips in the configuration bit-stream. However such techniques require constant reconfiguration of the FPGA to inject the new fault. This creates a performance bottleneck for these techniques compared to emulation techniques which required no reconfiguration during the fault injection process.

In this paper a fault injection method is proposed in order to overcome the performance bottleneck suffered by other hardware fault injection methods. This is done by dividing the fault injection processinto two steps: hardware fault injection by reconfiguration (similar to typical hardware fault injection methods) and emulation. The introduction of the emulation step is the key in removing theperformance bottleneck and making hardware fault injection method as fast as emulation methods. However since the fault injection step that requires reconfiguration is still there, the bottleneck would not removed. To overcome this, multiple faults will be injected instead of a single fault in each reconfiguration cycle. This will drastically reduce the number of reconfigurations to be done and hence overcomes the

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165

|| Volume 10, Issue 3, March 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1003006 |

reconfiguration bottleneck. Simulation isdone to prove that the unique design of this proposed fault injection method is able to remove the reconfiguration bottleneck by injecting multiple faults at each reconfiguration and yet still be able to individually observe the effect on the entire system of each of the faults.
The paper is organized as follows: in section II are described the types and nature of the SEUs that may occur in the FPGA. In section III are summarized previous fault injection methods. In section IV is presented in detail the proposed fault injection method. Sections V and VI present the simulation experiment and section VII concludes the paper.

II. THE RESEARCH METHOD

A Single Event Transient (SET) is a pulse of current generated by the impact of energetic particles hitting sensitive nodes of micro-electronic devices. SETs can alter, directly or indirectly, the content of a memorycell, phenomenon called SEU. Direct alteration occurs when the induced charge hits the memory unit itself. Indirect alteration occurs when an altered output of an erroneous combinational circuit is captured by amemory unit (Flip-Flop, Register, SRAM). For a sequential circuit, a transient error occurring in the combinational part of the circuit will have no effect if this error faded away before the arrival of the clock edge. Complex integrated circuits (microprocessors, semiconductor memories and FPGAs, …) may have significant number of memory cells thus being potentially sensitive to SEUs. At the application level, the error caused by an SEU in device outputs or operations is called soft error.
There are two kinds of bits in the SRAM-based FPGAs: the user bits and the configuration bits. User bits are the bits programmed by the user mainly asmemory elements. Configuration bits are the bits used by the FPGA to implement the routing of the circuit in addition to the combinational blocks. The number of SRAM configuration cells is more than 98% of all memory elements inside an FPGA [2]. Hence most SEUs will potentially occur in the configuration bits of the FPGA.

Though SEUs might occur in the user bits as well as in the configuration bits of an FPGA, those that occurat the user level are more predictable and most of them are non-persistent errors, though some might turn out tobe persistent [7]. This happens when an SEU occurs in circuit structures that contain feedback and storeinternal state. The feedback structures “trap” the incorrect state and store this erroneous state until appropriate reset measures are taken. For example, a bitflip in the current state of a design might permanently alter its state and its corresponding outputs making mandatory a reset to recover [7]. Non-persistent errors can be viewed as transient or temporarily errors.
SEUs occurring in the configuration bits are virtually unrecoverable or permanent until a reconfiguration of the FPGA either partially or globally are made [2]. Nevertheless, an SEU in the configuration bits might lead to a persistent error that needs a system reset just like the aforementioned case of an SEU in the user bits. It is possible that one configuration bit might be in control of two or more logic or routing resources. Hence, at the higher levelan SEU in the configuration memory of the FPGA might become an MEU (Multiple Events Upset) by altering different resources [5]. However there is alsoa great chance that an SEU in the configuration bits might not produce any error at all, as some of these configuration bits have the ‘don’t care’ status withrespect to the configuration [2]. SEUs alter both the configuration of logic blocks as well as switching blocks.

III. THE REFLECTIVE PROCESS

Fault injection is necessary for evaluating the behaviorof a system when a fault occurs. Fault injection is veryimportant for evaluating fault tolerant designs and drawing conclusions about both the efficiency of fault tolerance techniques and the weakness of hardeneddevices. Fault injection in general can be made at any level, whether software, i.e., altering inputs and the program running on the design, or hardware where actual faults are injected in the design either by the useof radiation instruments or the use of emulation where the design is altered and retested using FPGAs [6]. However, software fault injections are not sufficient, since they do not mimic the real occurrence of faultsin the design: A fault injection at the design levelshould be made and this can be done by altering the design and emulating the effects of such alteration.

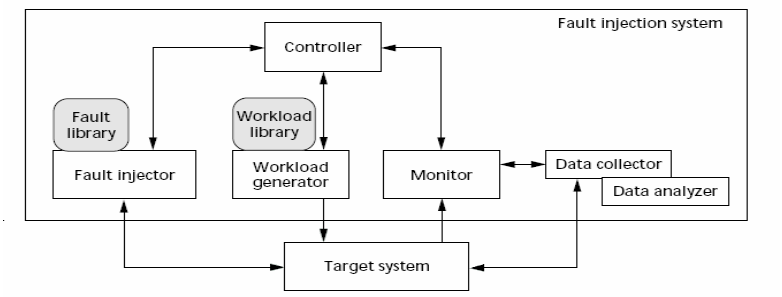


Figure 1. Basic components of a fault injection system.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165

|| Volume 10, Issue 3, March 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1003006 |

For designs meant to be implemented on FPGAs, even fault injection at the design level is not enough. FPGAs contain configuration bits that can alter the routing, combinational circuitry and other important characteristics of the circuit [2, 9, 12]. Hence, a fault injection at the design level is not enough. Bit flips should be inserted in the configuration bits of the FPGA and this is only applicable by either reconfiguring the FPGA or inducing real SEUs byexposing the design to radiation fluxes. Any fault injection system shall have the basic design presented in Figure 1.

Fault Injection Through RadiationExposure

Fault injection through radiation exposures may be favored by engineers for the ability to mimic the physical phenomenon that induces the studied faults[6]. However, this technique suffers from several disadvantages. First there is no ability to control the exact location at which a fault can be injected neither the amount of faults injected can be controlled properly. Moreover, this technique is found to be much time consuming and expensive [6]. Nevertheless, such techniques are necessary for the assessment of critical designs and fault tolerance techniques.

Fault Injection by Hardware Techniques inFPGAs

Though the principles remain the same, the setups and algorithms might differ as the technology provides better solutions. The basic steps and modules which arepresent in FPGA SEUs injection can be explained in spite of the different techniques and technologies present. The injection process can be divided into three different phases [1]:

1. Fault injection of SEUs in the configuration bitstream of the FPGA and generating faulty bit-stream files. The fault injector should read a correct configuration bitstream of the FPGA and inject a bit flip in the stream at the desired bit in the bitstream. The fault list generator is usually a software tool run in the PC prior to the beginning of the fault injection process. The fault injector however might be a software module implemented on the PC as well as ahardware model implemented on an FPGA or anembedded microprocessor.
2. Performing experiments in charge of the programming of the FPGA and providing input stimuli. This includes programming the FPGA with the faulty configuration bitstream or the correct bitstream for reset purpose. This phase can be implemented by software running on a PC communicating with special modules of the FPGA incharge of reconfiguring the design.
3. Results analysis of the previous phases. The results of the faulty emulation are compared to the golden run and the fault's effects are categorized and archived.

Recent FPGAs allow partial reconfiguration of the circuit which increases by several orders of magnitude the rate of fault injection [11].

IV. PROPOSED FAULT INJECTION METHOD

A new fault injection method was studied in order to speed up the fault injection and simulation process. Theidea is to insert many SEUs at a time in the design at each reconfiguration cycle. Previous method insertedone fault at a time in order to be able to study each fault by itself. However, the newly proposed method can insert multiple faults at a time and in addition it can study each fault impact by itself. This was achievable by dividing the fault injection process into two stages. In the first stage multiple faults are injected and in the second stage, each of these faults is studied by itself. However many problems, restrictions and overhead may be identified for this technique. The purpose of this section is to explain the details of this technique and identify the problems and restriction it faces. On the light of these problems, a decision should be made whether to proceed with the development of this technique or work on finding a more efficient one.

The Method

The method begins with the division of the designsinto sub-modules that have their own inputs and outputs vectors. A golden run is made and each module inputs and outputs are recorded. These modules are then all implemented on the FPGA. However each module will not be connected to therest of the design but to an input bitstream recorded from the previous golden run of the design. Thus,though all modules are implemented on the FPGA, they are actually totally disconnected, each modulebeing tested independently of the others. A fault is simultaneously injected in each of the models implemented on the FPGA. The output of each module is recorded and compared to the golden run module. Error(s) detected in the output stream of each module are studied independently on another FPGA by injecting it as an output bitstream of the corresponding module and seeing the faulty bitstream impact on the whole system. Though one fault is analyzed at a time, the latter step, the fault injection i.e., the bitstream injection, does not require reconfiguration of the circuit allowing thus the fault being injected rapidly.

Design Modulation

First the circuit is studied and divided into modules with a limited number of inputs and outputs. The modules should however be chosen with minimumfeedbacks of their output through preceding circuitry. Feedback can reduce the accuracy of the simulation, since the inputs provided to the module do not depend on the output of the modules being tested; rather it is arecorded bitstream from the previous golden run. The presence of a close feedback to the module may force the fault injected in the module to affect the inputs of the modules themselves. As an example, in a simple pipeline

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165

|| Volume 10, Issue 3, March 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1003006 |

processor, we can have the pipeline, the memory and the control unit each resembling a module. The pipeline modulation should be avoided asthere might be a lot of feedback among the pipeline stages. It should be noted that different instants of the same module can be implemented on the FPGA if the FPGA's resources allowed it.

The Golden Run

After the modules have been chosen, the whole design should be implemented on the FPGA and a golden runis made where all the outputs of the modules are recorded and stored either on a RAM or on a PC to be used for further comparison. Note that in the golden runthe modules are connected between them as required bythe design. The golden run consists in recording the outputs of each of the modules during the normal operation of the circuit.

In a way we'll be actually building several circuits ofthe whole system and each will have one faulty module.However, we used one "Golden" circuit as a common circuit to be used by all the faulty modules. Thus by having a Golden circuit and a faulty copy of each of themodules we're creating the illusion that we have a complete circuit for each of the faulty modules. In essence this is what allowed the faulty module to behave as if it is the only faulty module in the whole system.

It is noted also that several copies of the same module can be implemented and a fault injected in each. This will also reduce the number of times a fault would be injected in a module. In a way we would beas if we are running the same experiment on different FPGAs or different circuits and injecting a different fault in the same module across the circuits. However the Golden circuit will actually play the role of "the restof the design" for all the implemented modules and their copies.

Impact on the System

The same modules are now implemented on the FPGA. However the modules will here be totally disconnected.The input of each of the modules will be the bitstream recorded in the "golden run". In the golden run theinputs of the modules are monitored and recorded. The bitstream of the inputs recorded in the golden run will be provided as an input to the module in the fault injection phase. The need for this bitstream is the fact that each module in the fault injection process is not connected to any of the other modules, hence the needto provide its inputs from the golden run. Before running the bitstream, one SEU is injected in the configuration bits of the each of the module, i.e., the configuration bits of the FPGA blocks and components concerning these modules. The total number of SEUs injected in one reconfiguration will be equal to the number of modules in the design. The design is thenrun and the outputs of the modules recorded and compared with the golden run of these modules. After recording the outputs, a reconfiguration of the FPGA is done where different SEUs are inserted in the configuration bits of the modules and the previously injected SEUs being removed. These steps are further repeated until all SEUs have been injected.

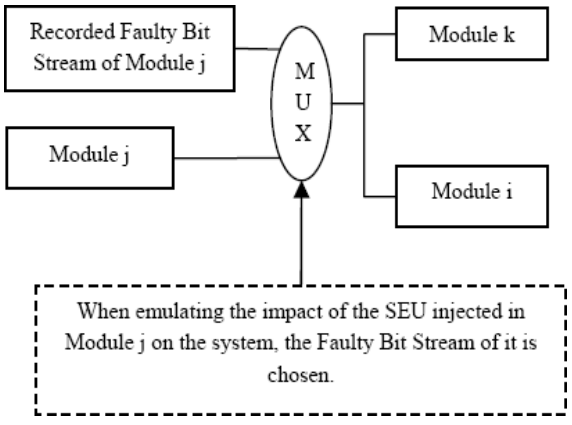


Figure 2. Activating the faulty behavior of a module.

This step can be described as the opposite of the first step. After recording the behavior of a faulty module connected to a "golden" circuit i.e. as if it isthe only module having faults, we are now observing the behavior of the whole system when this modulehas a faulty behavior. However, this second step is done without the need of reconfiguration and this is what the new method is all about: overcoming theperformance bottleneck of the reconfiguration process required for each injected fault. Multiple faults areinjected at each reconfiguration and then the effect on the system of each of these faults' is independently observed without the need of reconfiguration. This will allow the whole experiment to run at a higher speed that resembles that of an emulation process that requires no reconfiguration.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165

|| Volume 10, Issue 3, March 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1003006 |

Limits on SEUs Injection Location

Since the design is divided into modules in which one SEU is injected in each of them, it is undesirable that an injected SEU on one of the modules affects another module. This could happen when an SEU is inserted in the routing configuration bits of the configurable logic blocks (CLBs) of a module which are adjacent to CLBs of other module. Such SEU may cause a connection between the two modules, and the injected SEU effect on both of the modules will be invalid. To avoid this, such SEUs are not inserted in this method; rather they are simulated in the original RTR (Run Time Reconfiguration) fault injection method. However the percentage of such SEUs may be negligible and the overall performance will still be an improvement.

4. The New Fault Injection ExperimentDesign

A design has been made to check the validity of the proposed fault injection method. The new method is tested on a simple pipelined adder. The experimentdesign is divided into two parts:

1. The Multiple Fault Injection
2. The System Emulation of Individual Faults

The Pipelined Adder

The adder is made of 8 pipelined stages with each stageadding 3 bits: the carry from the previous stage and onebit from each of the numbers to be added. The adder is able to add two 8-bit numbers at each clock cycle.

A & B represents the 8-bit numbers to be added. F is the forwarded result of previous additions of previous bits of A & B. A_{out} & B_{out} are the remaining bits of A & B to be added in the preceding module. R is the result of the current addition along with the previous additions of A & B. At the end of the pipeline the addition result will be output in R and the carry will be the C_{out} signal. The modules are implemented using Verilog HDL and they are all the same codewise.

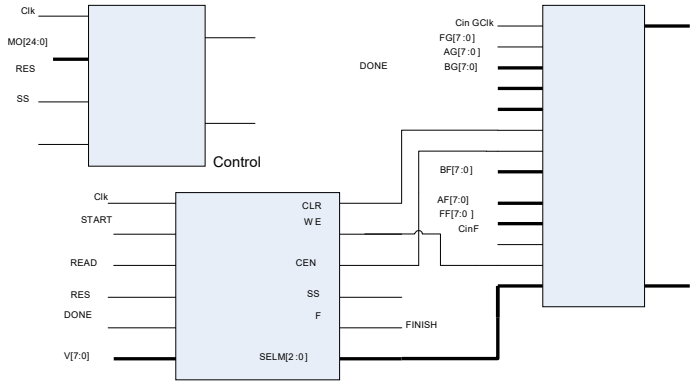
The Multiple Fault Injection Design (MFI)

The MFI design is shown in Figure 3. The three modules of the design are the fault injection circuit for testing, shown as TEST_SCHEMATIC in Figure 3, a control unit to manage the fault injection process shownas CONTROL in Figure 3 and a transmission unit used to serially transmit the data obtained from the faultinjection process to be archived and used later in the system emulation process.

The fault injection circuit shown in Figure 3, called the TEST_SCHEMATIC block, includes the typical pipelined adder, and 8 faulty modules each representing one of the pipelined adder modules. The outputs from the modules of the typical fault free pipeline are used as inputs for each of the faulty modules implemented in the design. The memory units have a 25-bit width words and have a depth of 100 in this simulation. The 25-bit width is chosen because the outputs of the module are 25bits all together. The depth of 100 was enough, as for the simulation purpose there is no need to run the design for more than 100 clock cycles. An eight bit counter is connected to all the memory units to manage the address increment at each clock cycle.

5.3. Fault Injection Circuit

Transmit



| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165

|| Volume 10, Issue 3, March 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1003006 |

Control Unit

The control unit was designed using verilog HDL. The control unit is responsible for managing the fault injection process and communicating with the transmission unit. Selecting the memory to be read, enabling memory write, managing the counter and taking commands from the user. The design is based on a state machine coded with Verilog HDL. The statemachine contains four states:

1. A default state where reset signals are asserted which waits for the "inject" signal indicating the initialization of the fault injection process.
2. A fault injection state. In this state the reset signals are released, the counter in the fault injection circuitis enabled and memory write is enabled. Once the signal "inject" is released, the counter value is saved in the state machine the counter being disabled and reset. The state machine immediately switches to the next state i.e. the third one.
3. The state waits for the signal "read" to start reading the memory contents of the memory elements in the fault injection circuit. Once the read signal is asserted the state machine switches to the next state.
4. This compounded state is actually made of two states. The state's purpose is to read the memory contents of the fault injection circuit.

5.3. Fault Injection Circuit

The fault injection circuit shown in Figure 3, called the TEST_SCHEMATIC block, includes the typical pipelined adder, and 8 faulty modules each representingone of the pipelined adder modules. The outputs from the modules of the typical fault free pipeline are used asinputs for each of the faulty modules implemented inthe design.

The memory units have a 25-bit width words and have a depth of 100 in this simulation. The 25-bit widthis chosen because the outputs of the module are 25bits all together. The depth of 100 was enough, as for themanages the fault emulation process and communicates with the data transmitter, which is responsible of serially sending the data using RS-232 protocol. The use of the RS-232 protocol was motivated by the fact that the goal of the experiment isto assess the validity of the method rather than its performance.

The control unit or SysCon is also based on a state machine coded with verilog HDL. The Transmit unitis the same than the one used in the multiple fault injection system design. The System module however is what defines this design. The system emulation circuit is implemented in this module.

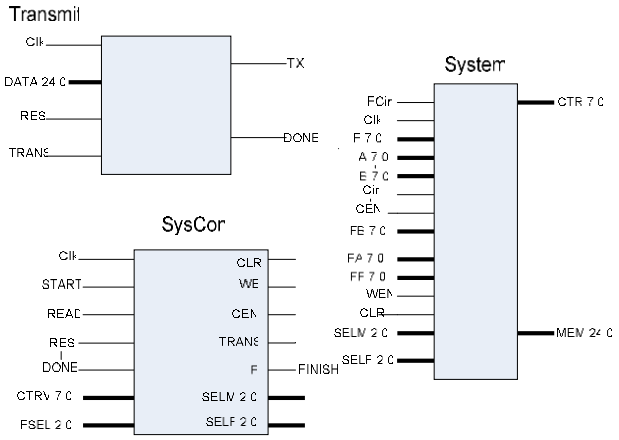


Figure 4. Design components of the emulation system.

Fault Emulation Circuitry

The emulation circuit is composed of a modified version of the pipelined adder design. The inputs of all the modules are connected to a multiplexer that chooseswhether to input the normal outputs of the previous module or the outputs of the faulty module obtained from the fault injection process.

Control Unit

A control unit is used to manage the fault emulation process. The unit resembles a state machine that was implemented using verilog HDL. The state machinecontains four states:

1. The default state where reset signals are asserted which waits for the "emulate" signal which signals the initialization of the fault emulation process.
2. The fault emulation state. In this state the reset signals are released, the counter in the fault injection circuit is

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165

|| Volume 10, Issue 3, March 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1003006 |

enabled and memory write is enabled.

3. The state waits for the signal "read" to start reading the contents of the memory elements in the fault injection circuit. When all memory elements are read, the counter which controls the addressing of the memory units is disabled. When the "read" signal is released the state machine jumps to the next state.

4. This state is similar to state four in the control unit described in § 5.4.

Typical Fault Injection Method Design

A design of a typical fault injection method on the pipelined adder was also created for the purpose of comparing the circuit behavior with that of the new method. The design resembled a normal pipelined adder with its circuit behavior recorded in memory elements (RAMs).

Fault Injection Simulation

To assess the new fault injection method's validity a fault injection simulation was done, the results being compared to the results of a typical fault injection method where only one fault is injected at a time. The simulation was done using the Xilinx ISE Simulator.

The Fault Injection Process

The fault injection process consists in injecting one fault in each of the pipelined adder modules in the module Test_Schematic. The fault injection was done by modifying the HDL codes implementing each of the simulator modules of the pipelined adder. The pipelined adder modules and the faulty modules share the same HDL code, however different source files are made for the pipelined adder and for the modules where the fault were to be injection. This would allow injecting faults in the modules by modifying the source files without affecting the "golden" modules of the pipelined adder which, for the purpose of the experiment, should have fault free performance.

This method of fault injection, i.e., the fault injection and the HDL level, has a performance shortage. In real fault injection experiments the faults are injected by modifying the configuration bitstream of the FPGA on the run between each fault injection. This would have required knowledge of the FPGA configuration bitstream to make sure the faults are injected in their corresponding locations. However since the purpose of this experiment is to assess the validity of the method rather than its performance, the HDL level fault injection was used.

The outputs of the modules are stored in the memory elements of the circuit. The control and transmitter modules manage the memory reading process. One of the memory elements corresponding to one of the modules is selected to have its outputs transmitted to the emulation circuit.

Emulation Process

In the emulation circuit the transmitted outputs of the faulty modules are provided to the circuit by selecting these stored outputs rather than the corresponding module's output of the pipelined adder. The output of each of the modules is saved in its corresponding memory element. The memory elements of the corresponding module have a 25-bit width word. Each word represents the outputs of the module during one clock cycle. The 25-bit width word is injected in the inputs of all the multiplexers of the emulation circuit, since only one fault will be emulated at a time. Each multiplexer has a select input to choose between the output of the previous module or the faulty word connected to it.

A typical fault injection method is simulated to compare the results with those obtained from the new fault injection method. The design is composed of a simple pipelined adder with a memory unit attached to the outputs of each of the modules. For a fault to be injected, one of the modules of the pipelined adder will have its HDL source file modified. The simulation is then run and the results will be read from the memory elements of the circuit.

Simulation Results

A waveform vector file was created for each simulation. Commands units were asserted and released at the appropriate times as for the inputs.

Several faults have been injected at different modules of the design. The system behavior has been observed and compared to the system behavior of a typical fault injection method. Comparing these outputs it can be seen that the new method was able to mimic the behavior of a typical fault injection method though multiple faults were injected at the same time.

Another module's stored outputs were observed in the same experiment. Again, the results confirmed that the observation is the same than the one of a typical fault injection process though multiple faults were simultaneously injected.

Other faults were further injected in the same modules and the same results were observed again in typical fault injection method as well as in the new method. Moreover, other modules were chosen as targets for these faults and again the circuit behavior matched that of the typical method. This further assured that the new method is able to observe the behavior of a design for each fault injected as if it was rather injected individually.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165

|| Volume 10, Issue 3, March 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1003006 |

V. CONCLUSIONS

While state-of-the-art FPGA fault injection methods suffer from a performance bottleneck because of the necessity to reconfigure the FPGA with the faulty bit- stream each time a fault needs to be injected, the proposed method eliminates this bottleneck by injecting multiple faults at a time. However to enable the observation of each fault's effect on the whole system asecond emulation step was added to the fault injection process. The emulation steps do not require any reconfiguration and thus the reconfiguration bottleneck is eliminated.

An experiment was made where a typical fault injection system was designed along with the new fault injection system. Faults were injected and simulation results showed that the new method's observation is identical to that of typical method, yet multiple faults were being injected simultaneously and emulated whichremoves the reconfiguration bottleneck. The analysis of the new method shows that it will clearly have advantages over other methods. The design modulation step that was introduced is easily feasible as only few modules are needed to achieve the performance advantage. Moreover most fault tolerance designs are already modulated to apply TMR and better fault detection mechanism. This can make the modulation of the design a lot easier.

Furthermore, the need to observe the outputs of all the modules where faults are injected will not introduce any communication bottleneck as all fault injection methods' purpose is to observe the behavior of the system and its internal signal and components including what would be its modules.

REFERENCES

1. Asadi G., Miremad S., Zarandi H., and Ejlali A., "Fault Injection into SRAM-Based FPGAs for the Analysis of SEU Effects," *in Proceedings of IEEE International Conference on Field- Programmable Technology*, pp. 428-430, 2003.
2. Asadi G. and Tahoori M., "Soft Error Rate Estimation and Mitigation for SRAM-Based FPGAs," *in Proceedings ACM/SIGDA 13th International Symposium on Field- Programmable Gate Arrays*, pp. 149-160, 2005.
3. Bellato M., Bernardi P., Bortolato D., Candelori A., Ceschia M., Paccagnella A., Rebaudengo M., Reorda S., Violante M., and Zambolin P.,"Evaluating the Effects of SEUs Affecting the Configuration Memory of an SRAM-based FPGA," *in Proceedings of the Design, Automation, and Test in Europe*, vol. 1, pp. 10584-10584, 2004.
4. Bernardi P., Reorda S., Sterpone L., and Violante M., "On the Evaluation of SEU Sensitiveness in SRAM-Based FPGAs," *in Proceedings of the 10th International On-LineTest Symposium*, pp. 115-120, 2004.
5. Ceschia M., Violante M., Reorda S., Paccagnella A., Bernardi P., Rebaudengo M., Bortolato D., Bellato M., Zambolin P., and Candelori A., "Identification and Classification of Single- Event Upsets in the Configuration Memory of SRAM-Based FPGAs," *Computer Journal of IEEE Transactions on Nuclear Science*, vol. 50, no. 6, pp. 55-59, 2003.
6. Hsueh C., Tsai K., and Iyer K., "Fault Injection Techniques and Tools," *Computer Journal of*
7. *IEEE Computer Society Press*, vol. 30, no. 4, pp.
8. 75-82, 2004.
9. Morgan K., Caffrey M., Graham P., Johnson E., Pratt B., and Wirthlin M., "SEU-Induced Persistent Error Propagation in FPGAs," *Computer Journal of IEEE Transactions onNuclear Science*, vol. 52, no. 6, pp. 2438-2445, 2005.
10. Reorda S., Sterpone L., and Violante M., "Multiple Errors Produced by Single Upsets in FPGA Configuration Memory: A Possible Solution," *in Proceedings of Europe*, pp. 136-141,2005.
11. Samudrala K., Ramos J., and Katkoori S., "Selective Triple Modular Redundancy Based Single-Event Upset Tolerant Synthesis for FPGAs," *Computer Journal of IEEE Transactions on Nuclear Science*, vol. 51, no. 5, pp. 2957-2966,2004.
12. Sterpone L. and Violante M., "A New Partial Reconfiguration -Based Fault-Injection System to Evaluate SEU Effects in SRAM-Based FPGAs," *Computer Journal of IEEE Transactions on Nuclear Science*, vol. 54, no. 4, pp. 965-970, 2007.
13. Sterpone L., Violante M., and Rezgui S., "An Analysis Based on Fault Injection of Hardening Techniques for SRAM-Based FPGAs," *Computer Journal of IEEE Transactions on Nuclear Science*, vol. 53, no. 4, pp. 2054-2059, 2006.
14. Velazco R., Fouillat P., and Reis R., *Radiation Effects on Embedded Systems*, Springer, 2007.

INNO SPACE
SJIF Scientific Journal Impact Factor
Impact Factor: 7.542

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

Scan to save the contact details