SCFIT: A FPGA-based Fault Injection Technique for SEU Fault Model

Abbas Mohammadi Mojtaba Ebrahimi Alireza Ejlali Seyed Ghassem Miremadi
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
Email: {amohammadi, mojtaba_ebrahimi}@ce.sharif.edu, {ejlali,miremadi}@sharif.edu

***Abstract*—In this paper, we have proposed a fast and easy-to-develop FPGA-based fault injection technique. This technique uses the Altera FPGAs debugging facilities in order to inject SEU fault model in both flip-flops and memory units. Since this method uses the FPGAs built-in facilities, it imposes a negligible performance and area overhead on the system. The experimental results on Leon2 processor shows that the proposed technique is on average four orders of magnitude faster than a simulation-based fault injection.**

I. Introduction

By technology scaling in nowadays digital circuits, Single Event Upsets (SEUs) becomes more probable, affecting not only space applications, but also applications on the ground level [1]. Studying the effect of these faults on the behavior of safety- and mission-critical applications, is of decisive importance. In this regard, designs should be evaluated using a fast and accurate dependability assessment technique.

Some analytical methods [2], [3] and fault injection techniques [4], [5], [6], [7] have been proposed to evaluate dependability measures of digital designs. Analytical methods are so complicated and time consuming in case of complex circuits [8]. Consequently, using this kind of methods cannot be commodious in comparison with fault injection techniques for complex circuits.

Fault injection techniques can be divided into three main categories: simulation-based, physical, and FPGA-based. Simulation-based fault injection techniques are flexible and have good controllability and observability. These features make them a very effective solution to analyze the effect of the faults in the early design steps. However, these techniques require a lot of time to simulate the model of the design and cannot be used for exhaustive fault injection campaigns. Physical fault injection techniques are very fast (i.e., at system run time speed) and can be used when a prototype of the system is available. The main drawbacks of these techniques are the high cost of required facilities and their poor controllability and observability. FPGA-based fault injection combines the speed of physical-based techniques and the flexibility of simulation-based techniques [7].

Generally, FPGA-based fault injection techniques are based on either reconfiguration or instrumentation [9]. Reconfiguration-based techniques [4], [5] use partial or full reconfiguration to inject a fault in the design under evaluation (DUE). These techniques do not impose any area overhead on the system as they use resources within FPGA for fault injection purpose. However, their time overhead is considerable. The instrumentation-based techniques rely on fault injector circuits called saboteurs [9]. The saboteurs are added to each fault site of DUE in order to inject a fault whenever and wherever is needed. Saboteurs are always simple circuits that can produce the preferred fault models. Saboteur units for stuck-at fault [6], [7], bridging-AND [7], transient faults in a wire [10] and SEU faults [9], [10], [11] have been proposed in previous works. As these works reported, these techniques do not impose considerable performance and area overhead on DUE. However, in order to insert these saboteurs in DUE, some special purpose tools are needed. Development of such tools, in most cases, are so time consuming.

978-3-9810801-8-6/DATE12/

In this paper, we have proposed an instrumentation-based fault injection technique which uses two Altera debugging facilities including In-system Memory Content Editor (MCE) and In-system Sources and Probes (SAP) for fault injection purposes. Since these debugging facilities has no impact on the behavior of the DUE, we call our proposed technique, *Shadow Components-based Fault Injection Technique (SCFIT)*. Unlike previously proposed techniques, SCFIT completely relies on commercial tools to implement the saboteurs. This feature makes the development of the proposed technique much easier and faster than previous works. SCFIT has the capability of SEU injection in both flip-flops and memory units. For fault injection in flip-flops, a scan-chain is used and fault is injected in target flip-flop during scan chain circulation. For fault injection in memory units, single-port memories are converted to dual-port memories using the MCE facility. The additional port is used to write the faulty data during the fault injection time and to read memory contents whenever is needed. In SCFIT, SAP has the responsibility of establishing communication between Quartus II software in host computer and Fault Injection Board (FIB) inside FPGA. Using FPGAs built-in facilities results in higher speed-up (4 orders of magnitude) in comparison with other FPGA-based fault injection techniques.

The rest of this paper is structured as follows: Section II introduces Altera debugging facilities. Section III explains SCFIT fault injection platform. Section IV describes the experimental environment and results. Finally, Section V concludes the paper.

II. Altera Debugging Facilities

Altera provides a complete set of debugging tools for the design verification on the FPGA devices. These tools use a combination of available memory, logic, and routing resources to provide the debugging facilities for the designer. By means of these tools, observability and controllability of the design

TABLE I
MAIN TCL COMMANDS IN QUARTUS II FOR HANDLING MCE AND SAP FEATURES

Feature	Function	Description
MCE	read_from_memory (name, address)	Read a specific word of a memory and return it as its output
	write_to_memory (name, address, content)	Write a new data in the specific address of a memory
SAP	write_source (name, value)	Change the value of a source
	read_probe (name)	Return the current value of a probe

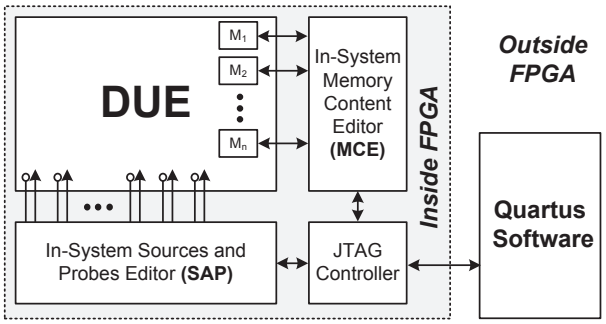

Figure 1. MCE and SAP Facilities to DUE in FPGA (Adapted from [12])

is increased in such a way that the values of the memories and the internal nodes can be read and modified during the system run-time. We have used some of these debugging facilities to propose an easy-to-develop FPGA-based fault injection platform. These facilities which named In-System Sources and Probes (SAP) and In-System Memory Content Editor (MCE) are used for controlling and observing the values of the internal nodes and the memory elements, respectively. Both MCE and SAP can be applied to any arbitrary design on the Altera FPGAs. Figure 1 shows simplified block diagram of a design inside an Altera FPGA connected to the SAP and MCE debugging facilities. As it can be seen, both SAP and MCE are controlled by a JTAG controller. The JTAG controller is the interface between Quartus Software on host computer and debugging facilities inside FPGA. The SAP and MCE facilities are supported by Quartus in both TCL scripting and GUI modes. We have used TCL scripting in our proposed fault injection technique to increase the level of automation.

MCE prepares the ability to view and update memory contents during the system operation. Using this feature, all single-port memories are converted to dual-port memories. The extra port is instrumented by the JTAG controller. Using this port, Quartus can trigger the JTAG controller to read from and write in the desired address of the memory.

The SAP facility is used to input simple virtual stimuli and to capture the current value on the target nodes. This feature provides single-cycle samples and single-cycle writes to selected logic nodes. Each SAP unit has a Source pin to input virtual stimuli and a Probe pin to sample the current value of the target node. We have used this feature to control the fault injection process parameters with TCL commands.

There are several TCL commands in Quartus II for controlling SAP and MCE facilities. We have summarized the most important commands in Table I. Complete list of Quartus TCL commands can be found in [12].

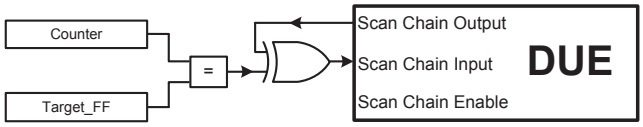

Figure 2. Fault Injection Controller for SEU Injection in Flip-flops

III. FAULT INJECTION USING MCE AND SAP DEBUGGING FACILITIES

The SCFIT technique uses different methods for fault injection in memory units and flip-flops. Fault injection in memory units can be easily done using the MCE feature of the Altera FPGAs. As mentioned in the previous section, using MCE, the content of memories can be read or written during the run-time operation. The ability to read memory contents is used to acquire the value of fault site at fault injection time. After fault injection in the original value, the faulty value is written back into the fault site using the write capability.

For fault injection in DUE flip-flops, since there is no way to observe and edit the value of the flip-flops, a scan-chain is inserted in DUE. By scan-chain insertion in DUE, faults can be injected during complete circulation of the scan-chain by inversion of the target flip-flop value. Figure 2 shows a fault injection controller designed for SEU injection in DUE flip-flops. A register called Target_FF is used to store the target flip-flip number which is selected as a fault site. While shifting the flip-flop values in the scan-chain, a counter is used to enumerate the number of clock cycles. First, this counter is loaded with total number of flip-flops. Afterwards, during the shift process, the counter is decremented by one at each clock cycle. Whenever the value of the counter and Target_FF are matched, the fault-free value of the fault site can be accessed at the scan-chain output port. By inversion of this value for one clock cycle, fault is injected at the value of target flip-flop. After fault injection, shifting the scan chain continues until the circulation process is completed. Once the counter reaches zero, the fault injection process will be completed and all flip-flops except the target flip-flop have their previous value.

In order to control the fault injection process, a general purpose FIB is designed. The schematic of this board is shown in Figure 3. There are five SAP units on FIB including Write, Run, Counter, FI_FF, and Target_FF. The value of these SAPs can be easily read and written using TCL commands.

Any change in the source of a SAP by TCL scripts causes its value immediately changes at the FIB. However during the fault injection process, it will be needed to change the value of several SAPs at the same time. The Write SAP is used to synchronize the write operation on all other SAPs. In this regard, a register is assigned to each SAP unit. At the rising edge of Write SAP, the value of all SAPs are loaded to their

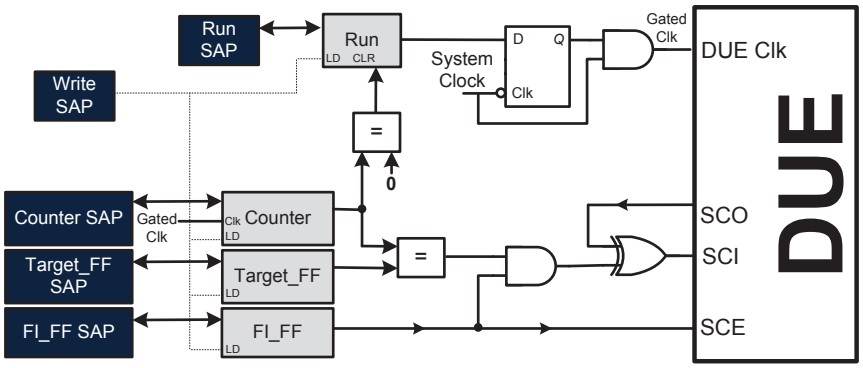

Figure 3. Fault Injection Board

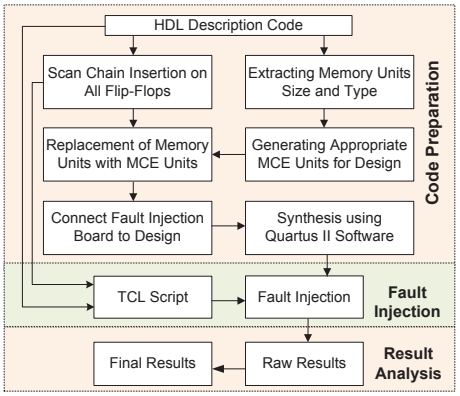

Figure 4. Fault Injection Flow

corresponding registers (see Figure 3). This means, all changes in the value of SAPs are applied to the board at the rising edge of the Write SAP. It should be noted that since the registers output is connected to the probe of their corresponding SAP unit, there is no limit in reading the value of SAPs. Any change in the value of the registers will affect the probe of their corresponding register and can be immediately read by TCL scripting commands.

Run is another control SAP that indicates the DUE status (running or halt). When Run SAP is reset, the clock of DUE is gated, thus DUE halts. This SAP is used to halt DUE during the fault injection process and whenever is needed. The Counter SAP is a countdown counter. It is used to enumerate the number of clock cycles during the fault injection process. It can be loaded with any arbitrary value whenever is needed. Afterward, when the Run SAP is set, it is decremented by one unit at each clock cycle. Whenever this counter becomes zero, it will reset the Run SAP value to halt DUE.

The FI_FF SAP is employed to trigger the scan-chain enable (SCE) signal during fault injection in flip-flops. Fault injection controller for SEU injection in DUE flip-flops discussed previously (see Figure 2). The same architecture can be seen in Figure 3. In FIB, during fault injection in flip-flops, the FI_FF SAP should be activated to enable the Scan Chain Enable signal. Also, Counter should be loaded with scan chain length and then during circulation it will decremented by one unit at each clock cycle. The value of Counter is continuously compared with Target_FF SAP to find the fault injection time. If Counter and Target_FF are matched, a SEU is injected in the value of the target flip-flop.

So far we have introduced our MCE and SAP debugging facilities. Also we introduce a general purpose fault injection board (i.e. FIB). FIB is the only part of our fault injection technique that should be implemented by the designer. Other parts of the fault injection system are automatically added to the DUE by commercial tools. Figure 4 shows the fault injection flow of the SCFIT technique. First, a scan-chain is inserted in all flip-flops of DUE using commercial tools like Synopsis DFT compiler [13]. Then, all structural information about the memory units are extracted. Using this information, new MCE memories are created by means of Quartus II Software [14] and replaced with the original memories. Now,

Algorithm 1: SEU Injection TCL Script

Inputs:
FAULT_TIME: Fault injection clock cycle
TOTAL_TIME: Total clock cycles
fault_site_type = FF { FAULTY_FF: Target flip-flop number; FFs: Total flip-flops
fault_site_type = Mem { FAULTY_MEM: Target memory name; FAULTY_ADD: Address of target word; FB: Target bit number

```
1: write_source (Write, 0)
2: write_source (Counter, FAULT_TIME)
3: write_source (Run, 1)
4: write_source (Write, 1)
5: repeat until {read_probe (Run) = 0}
6: if fault_site_type = FF then
7:     write_source (Write, 0)
8:     write_source (Counter, FFS)
9:     write_source (Target_FF, FAULTY_FF)
10:    write_source (FI_FF, 1)
11:    write_source (Run, 1)
12:    write_source (Write, 1)
13:    repeat until {read_probe (Run) = 0}
14:    write_source (Write, 0)
15:    write_source (FI_FF, 0)
16:    write_source (Write, 1)
17: elsif fault_site_type = Memory then
18:    DATA=read_from_memory( FAULTY_MEM, FAULTY_ADD)
19:    DATA=DATA Xor 2^FB
20:    write_to_memory(FAULTY_MEM, FAULTY_ADD, DATA)
21: end if
22: write_source (Write, 0)
23: write_source (Counter, TOTAL_TIME - FAULT_TIME)
24: write_source (Run, 1)
25: write_source (Write, 1)
26: repeat until {read_probe (Run) = 0}
```

DUE with implemented saboteur is ready. After connecting FIB to DUE, all design is synthesized using Quartus II Software and then programmed to the FPGA. Now FPGA-based fault injection platform is ready to work. Using TCL scripts and sending appropriate commands to MCE and SAP units faults are injected in specified units.

Algorithm 1 shows a simple TCL script for injecting SEU fault model in DUE. First part of the algorithm shows inputs. As it can be seen, this algorithm needs complete information about the fault injection time and location. For SEU injection in flip-flops, scan chain length and target flip-flop number

are needed, while for fault injection in memories, memory name and location of the target bit should be specified. Second part of the algorithm (lines 1-5) stimulates the DUE to run until the fault injection time. To do this, the value of FAULT_TIME should be provided to the Counter SAP and Run SAP should be activated. Afterwards, at each clock cycle the value of Counter is decreased by one unit. Whenever this counter becomes zero, Run SAP is deactivated and this part is terminated. Note, as mentioned earlier, before any modification at the source of SAP units, the value of the Write SAP should become zero and after that, this value should be changed to one. These commands create a rising edge at the Write register value and consequently all SAPs values are loaded to their corresponding registers. Third part of algorithm (lines 6-21) shows the fault injection process. For fault injection in flip-flops (lines 7-16), a flip-flop should be selected as target flip-flop. Also, fault injection in flip-flops must be enabled (line 10). Then, the counter should be loaded with total number of flip-flops and Run SAP is activated. By this works, scan chain is circulated and target flip-flop becomes faulty. Whenever the circulation process is finished the Counter SAP becomes zero and Run SAP is deactivated. At the end, fault injection in flip-flops should be disabled by deactivating the FI_FF SAP. For fault injection in memories (lines 18-21), firstly, the fault free value is read from the memory. Then the faulty word is created by inversion of the target bit (line 19), and finally the faulty word is written back to the memory. Last part of the algorithm (22-26) is used to stimulate the SUE to run until the end of the testbench. This part is similar to the second part, but this time the Counter SAP is loaded with total number of remaining clock cycles (line 23).

IV. A Case Study: Fault Injection on Leon2 Processor

As a case study, we have used Leon2 as a DUE in our technique. Leon2 implements a 5-stage pipeline with separate instruction and data cache buses (Harvard architecture). It also has a three-port register file with one write and two read ports. We have injected SEU fault model in its flip-flops and caches.

Some programs of MiBench [15] automotive benchmarks including Bitcount, Basicmath and Qsort, have been used as our test benches during the fault injection. In order to show the efficiency of this technique, we have compared it with a simulation-based fault injection. An Altera Stratix IV FPGA board is employed as our platform in the FPGA-based fault injections. This board is connected to a host computer with 2.8 GHz Dual-Core CPU and 4 GB RAM. The simulation-based fault injections have done with ModelSim simulator [16] on the same computer. In both fault injection methods the clock frequency of Leon2 processor was 50 MHz. Each simulation-based fault injection takes on average 3168.6 minutes to be done while the similar fault injection takes on average 0.3 minutes on the FPGA. This means that our method has a speed up equal to 10562 times.

Implementation of SCFIT, including built-in facilities and fault injection board, take less than 5% of the FPGA board resources. Additionally, in our case study, i.e. Leon2, implementation of this technique increases the critical path of the processor by less than 1%. Our investigation shows that this increase is due to adding a multiplexer in front of all flip-flops in the scan chain insertion process.

V. Conclusions

In this paper, we proposed a novel FPGA-based technique based on Altera FPGAs debugging facilities called SCFIT. Development of the SCFIT technique, except its FIB, is completely relies on commercial tools. Additionally, using the proposed technique a fault injection campaign for an arbitrary design is simplified to a TCL programming problem. These features make our technique so flexible and easy-to-develop. We have evaluated the Leon2 processor using this technique. Our SEU injection results on this processor shows that this technique is about 4 orders of magnitude faster than a simulation-based fault injection.

References

[1] P. Shivakumar, M. Kistler, S.W. Keckler, D. Burger, and L. Alvisi. Modeling the effect of technology trends on the soft error rate of combinational logic. In *Proc. of Dependable Systems and Networks (DSN)*, pages 389–398, 2002.

[2] G. Asadi and M.B. Tahoori. An analytical approach for soft error rate estimation in digital circuits. In *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2991–2994, 2005.

[3] M. Fazeli, S.G. Miremadi, H. Asadi, and S.N. Ahmadian. A fast and accurate multi-cycle soft error rate estimation approach to resilient embedded systems design. In *Proc. of Dependable Systems and Networks (DSN)*, pages 131–140, 2010.

[4] L. Antoni, R. Leveugle, and B. Feher. Using run-time reconfiguration for fault injection applications. *IEEE Transactions on Instrumentation and Measurement*, 52(5):1468–1473, 2003.

[5] L. Sterpone and M. Violante. A new partial reconfiguration-based fault-injection system to evaluate SEU effects in SRAM-based FPGAs. *IEEE Transactions on Nuclear Science (TNS)*, 54:965–970, 2007.

[6] K. Cheng, S. Huang, and W. Dai. Fault emulation: A new methodology for fault grading. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, 18:1487–1495, 1999.

[7] A. Ejlali, S.G. Miremadi, H. Zarandi, G. Asadi, and S.B. Sarmadi. A hybrid fault injection approach based on simulation and emulation co-operation. In *Proc. of Dependable Systems and Networks (DSN)*, pages 479–488, 2003.

[8] S. Hwang, J. Hong, and C. Wu. Sequential circuit fault simulation using logic emulation. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, 17:724–736, 1998.

[9] A. Ejlali and S.G. Miremadi. Error propagation analysis using FPGA-based SEU-fault injection. *Microelectronics Reliability*, 48:319–328, 2008.

[10] M. Shokrolah-Shirazi and S.G. Miremadi. Fpga-based fault injection into synthesizable verilog HDL models. In *Proc. of International Conference on Secure System Integration and Reliability Improvement.*, pages 143–149, 2008.

[11] P. Civera, L. Macchiarulo, M. Rebaudengo, M.S. Reorda, and M. Violante. Exploiting circuit emulation for fast hardness evaluation. *IEEE Transactions on Nuclear Science*, 48(6):2210–2216, 2001.

[12] Altera Corp. *Quartus II Handbook Version 11.0*. Aletra, 2011.

[13] Synopsys DFT compiler, www.synopsys.com, 2009.

[14] Altera corporation, www.altera.com.

[15] M.R. Guthaus, J.S. Ringenberg, D. Ernst, T.M. Austin, T. Mudge, and R.B. Brown. Mibench: A free, commercially representative embedded benchmark suite. In *Proc. of IEEE International Workshop on Workload Characterization, WWC-4*, pages 3–14, 2001.

[16] ModelSim SE, Mentor Graphics Corporation, www.model.com.