Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 19(3): 175-179, Sep. 2021 **Regular paper**

Simulation-Based Fault Analysis for Resilient System-On-Chip Design

Chang Yeop Han, Yeong Seob Jeong, and Seung Eun Lee*, *Member, KIICE*

Department of Electronic Engineering, Seoul National University of Science and Technology, Seoul 01811, Korea

Abstract

Enhancing the reliability of the system is important for recent system-on-chip (SoC) designs. This importance has led to studies on fault diagnosis and tolerance. Fault-injection (FI) techniques are widely used to measure the fault-tolerance capabilities of resilient systems. FI techniques suffer from limitations in relation to environmental conditions and system features. Moreover, a hardware-based FI can cause permanent damage to the target system, because the actual circuit cannot be restored. Accordingly, we propose a simulation-based FI framework based on the Verilog Procedural Interface for measuring the failure rates of SoCs caused by soft errors. We execute five benchmark programs using an ARM Cortex M0 processor and inject soft errors using the proposed framework. The experiment has a 95% confidence level with a ±2.53% error, and confirms the reliability and feasibility of using proposed framework for fault analysis in SoCs.

Index Terms: Resilient design, Fault analysis, Fault injection, Soft error, System-on-chip

I. INTRODUCTION

The failure rate in system-on-chip (SoC) designs has increased along with developments in process technologies and demands for low-power designs. Efforts to reduce the failure rate are important for various applications such as electrical devices in automobiles and U-health care, in addition to fields requiring high safety and reliability, such as those for nuclear power plants, airplanes, and satellites. Therefore, designers should consider several fault-tolerant techniques to maintain the reliability of the system. When a fault occurs in the target system, it can cause an error. By considering faults of the system in the design stage, the design can be made fault-tolerant. Research activities on fault tolerance have been actively conducted over the past half century, aiming to provide the ability to complete accurate functions at all times, even when a system suffers damage. Recently, research has been conducted on diagnosing faults through machine learning [1]. In general, a fault-tolerant technique identifies system faults and corrects them using additional resources. Aiming to provide stability in system currents, research has also been conducted on controlling currents based on calculations of the inductor current [2]. Fault diagnosis resolves the verification concerns of a resilient system. Fault diagnosis confirms the status of the system and its fault-tolerance capability. The miniaturization of integrated circuits is likely to increase the frequency of errors. For this reason, studies have been conducted to analyze system vulnerabilities through microarchitecture-level fault injection (FI) [3, 4]. FI techniques have been widely used in fault diagnosis. FI techniques can be classified based on the device injecting the fault into a target system. A hardware-based FI enables the target system to experience faults at a physical level [5, 6]. The test time of a hardware-based FI is fast. However, hardware-based FI approaches cause permanent damage to the target system, because the actual

Received 23 July 2021, **Revised** 23 July 2021, **Accepted** 09 August 2021
***Corresponding Author** Seung Eun Lee (E-mail: seung.lee@seoultech.ac.kr, Tel: +82-2-970-9021)
Department of Electronic Engineering, Seoul National University of Science and Technology, Seoul 01811, Korea.

Open Access **https://doi.org/10.6109/jicce.2021.19.3.175** print ISSN: 2234-8255 online ISSN: 2234-8883

circuit cannot be restored [7]. A software-based FI introduces a software fault by modifying the execution code in the system [8, 9]. A software-based FI is practical, as the required hardware and software are actually used. However, it is limited by the types of faults injected by the software. An emulation-based FI injects faults into a design implemented on a field-programmable gate array (FPGA) [10, 11]. The diagnosis can be processed in real-time, but the target design must be optimized in the FPGA before the experiment. A simulation-based FI observes a failure using a computer simulation tool [12, 13]. The simulation-based FI can change the data in any location and at any time, without damaging the real system and with no additional hardware costs. Both hardware and software faults cause changes of signals. As a result, both faults are considered by a simulation-based FI, which randomly changes the signals of the system through simulations.

In this paper, we propose a simulation-based FI framework named the “SoC Fault Analyzer” (SoCFA), based on the Verilog Procedural Interface (VPI), to measure the failure rates of SoCs caused by soft errors. The SoCFA functions consist of a series of processes for modeling and injecting soft errors into a target system, and then examining the system failure(s). The function of the VPI is subdivided for better efficiency in modification and addition. A case study using an ARM Cortex M0 processor demonstrates the feasibility of using our SoCFA for fault analyses of SoCs.

The rest of this paper is organized as follows. Section 2 provides a detailed description of the SoCFA framework and the overall simulation procedure. Section 3 presents the experimental setup, and the results from the fault analysis using the ARM Cortex M0 processor. Finally, we conclude our paper in Section 4.

II. SYSTEM MODEL

A. Overview

The SoCFA framework includes the following: (a) the SoCFA, which includes an error injection platform, failure-rate extraction unit, and testbench with a duplicated design; (b) a testbench, which instantiates the duplicated design and calls the VPI function; and (c) a gate-level netlist, i.e., the target under diagnosis (see Fig. 1). The fault diagnosis process using the SoCFA is described as follows. (1) First, the *socfa_fault_model (U, U_TEST)* in the testbench calls the SoCFA VPI function, where *U* is the original design under diagnosis, and *U_TEST* is the duplicated design that suffers the soft errors. (2) A list of input and internal node information is extracted from the design for the soft-error generation. (3) Based on this list, the error generator generates random error data and assigns the time information regarding when the error will be injected. The random error could be changed into specific errors such as cosmic-ray particle, thermal, supply-voltage fluctuation, or 1/f noise-induced errors. (4) The generated errors are injected according to the error time information. The VPI process continues observing the data inside the design during the simulation, and then injects the errors at the designated internal node and corresponding simulation time. (5) The failure-rate extraction unit compares the outputs of the two modules and measures the failure rates. Duplicated modules are used to check the failures of the output ports so that changes in the output are observed when the same data enter the identical design modules. The error injection and failure-rate extraction are performed in each clock cycle during the simulation.

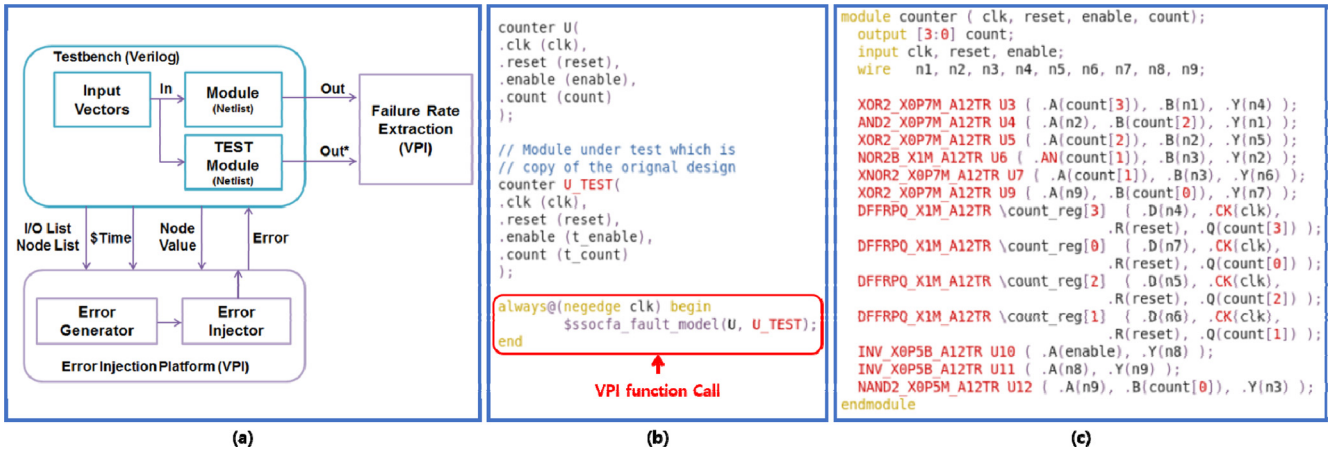


Fig. 1. Overview of the System-on-Chip Fault Analyzer (SoCFA) framework: (a) block diagram of the SoCFA. (b) Verilog Procedural Interface (VPI) function call in the testbench, and (c) gate-level netlist under diagnosis.

B. Verilog Procedural Interface

VPI is a C-programming interface for the Verilog hardware description language (Verilog HDL). It supports interconnection between high-level languages such as C/C++ and Verilog HDL. The VPI was developed to overcome the drawbacks in the Verilog Programming Language Interface (PLI), e.g., too many functions and vulnerabilities in compatibility with other simulators. The VPI has fewer functions than the PLI and is compatible with many simulators such as the Cadence NC Verilog or Synopsys VCS. The VPI can be used in the design verification phase.

The SoCFA functions for modeling errors and measuring the failure rates are accessible only by calling the VPI function in an existing gate-level testbench. Therefore, the SoC designer can easily adopt our SoCFA for fault diagnosis. Moreover, the designer can use an existing gate-level verification environment that includes CAD tools, a testbench, and a netlist, i.e., as already set up for functional and timing validation of the design.

C. System-On-Chip Fault Analyzer(SoCFA) Function

The SoCFA framework provides a convenient way to call the VPI function (*socfa_fault_model*) in the testbench, thereby providing a simulation interface for modeling the errors and measuring the failure rates of a design. Different VPI functions are required to perform the fault diagnosis, as shown in Fig. 1-(a). The SoCFA functions are intended to conveniently modify the existing features and improve new features. The SoCFA functions are divided according to specific roles: (1) extracting design information (e.g., list of inputs, outputs, and internal nodes in the design; simulation time; and size of the design); (2) determining the FI timing according to the error type; (3) injecting the generated errors into the target system; and (4) comparing the system outputs during the simulation. Table 1 summarizes the role of each function in the SoCFA. Fig. 2 shows the system flow of our SoCFA, and the functions used in each sequence. After the *socfa_fault_model* function is called in the testbench, the *socfa_show_ports* function extracts the name and value information from the I/O ports at the first simulation stage.

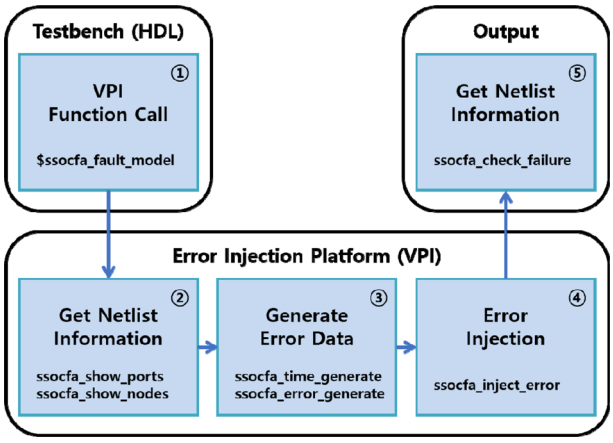


Fig. 2. System flow of the SoCFA for fault diagnosis.

Similarly, the *socfa_show_nodes* function extracts the name and value information from the internal nodes. The *socfa_time_generate* function generates the timing information when the errors are injected into the target system according to the error model. The error data are generated by the *socfa_error_generate* function according to the type of soft errors. Subsequently, the generated errors are injected into the system using the *socfa_inject_error* function. Finally, the output data between two modules are checked by the *socfa_check_failure* function. All SoCFA functions for modeling the errors and measuring the failure rates are accessible only by calling the VPI function call in the testbench.

III. EXPERIMENT

A. Experimental Environment

In our study, the experiments were performed on a server with an Intel Xeon CPU (six cores) running at 2.67 GHz with 24.7 GB of DDR3 memory. The Cadence NC Verilog tool was used for the simulation with the VPI. We adopted the TSMC 65-nm library for the synthesis of the ARM Cortex M0 processor. Six benchmark programs were executed: Dhrystone, Ackermann, Fibonacci, Sieve, Helloworld, and

Table 1. Description of the System-on-Chip Fault Analyzer (SoCFA) functions

Function	Description
$socfa_fault_model (U, U_TEST)	Main Verilog Procedural Interface (VPI) function for error injection and fault analysis
socfa_show_ports	List out the name and value for I/Os
socfa_show_nodes	List out the name and value for nodes
socfa_time_generate	Generate timing to inject errors
socfa_error_generate	Generate errors according to error model
socfa_inject_error	Inject the generated errors into the design
socfa_check_failure	Compare the outputs in U and U_TEST modules and report failure information

Table 2. Information of benchmarks running on an ARM Cortex M0 processor

Benchmark	Simulation time	CPU time	Memory Usage
Ackermann	436.1 ms	70.7 s	158.3 MB
Helloworld	436 ms	73.9 s	158.3 MB
Dhrystone	1,394.8 ms	237.3 s	157.3 MB
Fibonacci	2,187.6 ms	356.8 s	158.3 MB
Bubble	5,124.2 ms	891.9 s	158.4 MB
Sieve	12,174.2 ms	2041.1 s	158.4 MB

Bubble Sort. Table 2 shows the simulation information of the benchmarks, such as the simulation time for each program, CPU time for fault diagnosis, and memory usage.

We injected the soft error into the ARM Cortex M0 processor and checked the failure rate using the SoCFA framework. All experiments were performed with the netlist of the Cortex M0 processor for the gate-level simulation. The entire simulation process was identical to that described in Section 2.1, and the experiment was repeated for 1500 trials for the statistical analysis. The analysis with 1500 trials had a 95% confidence level with a ±2.53% error.

B. Experimental Results

Fig. 3 shows the failure rates of the ARM Cortex M0 processor under the different workloads, assuming a 10^{-8} soft error rate. In Fibonacci, the failure rate is 5.47×10^{-9}, the highest among the benchmarks. In Bubble Sort, Sieve, and Ackermann, the failure rates are approximately 5×10^{-9}. Dhrystone exhibits a 4.21×10^{-9} failure rate. In Helloworld, the failure rate is 2.36×10^{-9}, i.e., roughly 57% lower than that in the Fibonacci. Through the statistical analysis, we can confirm that the failure rates differ depending on the application program, but they are almost uniform, regardless of the number of experimental trials. As a result, our SoCFA framework can reliably measure a target system with consistent results and can provide a feasible means for fault diagnoses of SoCs, along with different application programs.

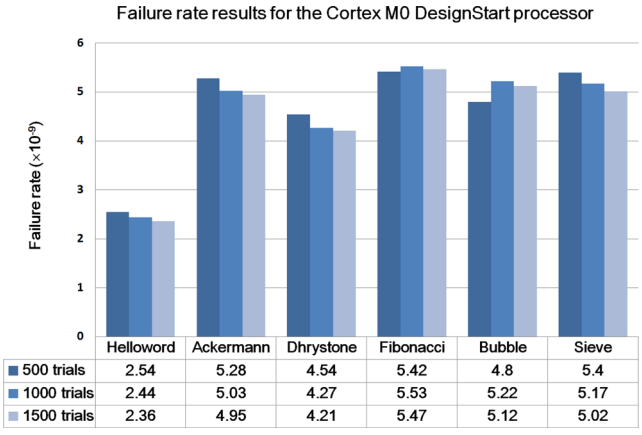


Fig. 3. Failure rates of the Cortex M0 processor with different workloads under a 10^{-8} soft error.

C. Benefits

A study for improving the reliability of a system is required, because semiconductors are vulnerable to external environment and manufacturing process drawbacks following developments in process technologies. Thus, a robust system design and fault-tolerant technique are strongly required. The SoCFA framework is a simulation-based FI technique designed to measure the failure rate of a target system. We verified that our framework can accurately measure the failure rate of a system. In addition, the SoCFA is easily accessible because all routines are performed by only calling the VPI function in the testbench. In addition, the SoCFA can measure the statistical failure rate without concerns regarding damage and additional costs.

IV. CONCLUSIONS

In this paper, we proposed a simulation-based FI framework named SoCFA based on the VPI for fault diagnoses of SoCs. The SoCFA provides an easy way of diagnosing the failure rates of SoCs using an existing gate-level simulation environment. A case study conducted using ARM Cortex M0 demonstrated the feasibility of our SoCFA for fault diagnosis. We believe that the fault diagnosis of SoCs is essential for resilient and low-power circuit designs, and that our SoCFA could provide the means for designing a resilient system.

ACKNOWLEDGEMENTS

This study was supported by the Research Program funded by the SeoulTech(Seoul National University of Science and Technology).

REFERENCES

[1] X. Yang, J. S. Lee, and H. K. Jung, “Fault Diagnosis Management Model using Machine Learning,” *Journal of Information and Communication Convergence Engineering (JICCE)*, vol. 17, no. 2, pp. 128-134, 2019. DOI: 10.6109/jicce.2019.17.2.128

[2] J. Y. Kim, S. H. Park, and Y. S. Suh, “Maximum Current Estimation Method for the Backup of Current Sensor Faults,” *Journal of Information and Communication Convergence Engineering*, vol. 18, no. 3, pp. 201-206, 2020. DOI: 10.6109/jicce.2020.18.3.201.

[3] A. Chatzidimitriou, G. Papadimitriou, C. Gavanas, G. Katsoridas, and D. Gizopoulos, “Multi-Bit Upsets Vulnerability Analysis of Modern

Microprocessors," *2019 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 119-130, 2019. DOI: 10.1109/IISWC47752.2019.9042036.

[4] M. Kaliorakis, D. Gizopoulos, R. Canal, and A. Gonzalez, "MeRLiN: Exploiting dynamic instruction behavior for fast and accurate microarchitecture level reliability assessment," *44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 241-254, 2017. DOI: 10.1145/3079856.3080225.

[5] V. Pouget, D. Lewis, and P. Fouillat, "Time-Resolved Scanning of Integrated Circuits with a Pulsed Laser: Application to Transient Fault Injection in an ADC," *IEEE Transactions on Instrumentation and Measurement*, vol. 53, no. 4, pp. 1227-1231, 2004. DOI: 10.1109/TIM.2004.831488.

[6] R. J. Martínez, P. J. Gil, G. Martín, C. Pérez, and J. J. Serrano, "Experimental Validation of High-Speed Fault-Tolerant Systems Using Physical Fault Injection," *Dependable Computing for Critical Applications 7 (DCCA 7)*, pp. 249-265, 1999. DOI: 10.1109/DCFTS.1999.814299.

[7] H. Ziade, R. Ayoubi, and R. Velazco, "A Survey on Fault Injection Techniques," *The International Arab Journal of Information Technology*, vol. 1, no. 2, pp. 171-186, July 2004.

[8] N. Wulf, G. Cieslewski, A. Gordon-Ross, and A. D. George, "SCIPS: An Emulation Methodology for Fault Injection in Processor Caches," *IEEE Aerospace Conference*, pp. 1-9, 2011. DOI: 10.1109/AERO.2011.5747450.

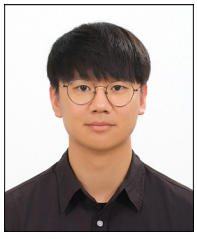
[9] R. Natella, D. Cotroneo, J. A. Duraes, and H. S. Madeira, "On Fault Representativeness of Software Fault Injection," *IEEE Transactions on Software Engineering*, vol. 39, no. 1, pp. 80-96, 2013. DOI: 10.1109/TSE.2011.124

[10] M. Portela-García, C. López-Ongil, M. G. Valderas, and L. Entrena, "Fault Injection in Modern Microprocessors Using On-Chip Debugging Infrastructures," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 2, pp. 308-314, 2011. DOI: 10.1109/TDSC.2010.50

[11] L. Entrena, M. García-Valderas, R. Fernández-Cardenal, A. Lindoso, M. Portela García, and C. López-Ongil, "Soft Error Sensitivity Evaluation of Microprocessors by Multilevel Emulation-Based Fault Injection," *IEEE Transactions on Computers*, vol. 61, no. 3, pp. 313-322, 2012. DOI: 10.1109/TC.2010.262.

[12] J. Xu and P. Xu, "The Research of Memory Fault Simulation and Fault Injection Method for BIT Software Test," 2012 Second *Instrumentation, Measurement, Computer, Communication and Control (IMCCC)*, pp. 718-722, 2012. DOI: 10.1109/IMCCC.2012.174.

[13] D. Lee and J. Na, "A Novel Simulation Fault Injection Method for Dependability Analysis," *IEEE Design & Test of Computers*, vol. 26, no. 6, pp. 50-61, 2009. DOI: 10.1109/MDT.2009.135.



Chang Yeop Han

received a B.S. degree from the Department of Electronic Engineering at the Seoul National University of Science and Technology, Seoul, Korea in 2020, where he is pursuing the M.S. degree. His current research interests include computer architecture and system-on-chip design.

Yeong Seob Jeong

received the B.S. and M.S. degrees in Electronic Engineering from the Seoul National University of Science and Technology, Seoul, Korea, in 2012 and 2014, respectively. His research interests include computer architecture, system-on-chip, and fault-tolerant system designs.

Seung Eun Lee

received a Ph.D. degree in electrical and computer engineering from the University of California, Irvine (UC Irvine) in 2008, and B.S. and M.S. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon in 1998 and 2000, respectively. After graduating, he worked with Intel Labs., Hillsboro, OR, as a Platform Architect. In 2010, he joined the faculty of the Seoul National University of Science and Technology, Seoul. His current research interests include computer architecture, multi-processor system-on-chip, low-power and resilient VLSI, and hardware acceleration for emerging applications.