International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 17 (2015) pp 38377-38383

Survey of Fault-Injection Techniques for Resilient Systems-on-Chip Design

Yeong Seob Jeong and Seong Mo Lee and Seung Eun Lee*

*Department of Electronic Engineering, Seoul National University of Science and Technology, 232 Gongreung, Seoul, Korea *seung.lee@seoultech.ac.kr*

Abstract
The development of process technology has increased system performance, but the system failure probability has also significantly increased. It is important to consider the system reliability in addition to the cost, performance, and power consumption. In this paper, we describe the types of faults that occur in a system and where these faults originate. Then, fault-injection techniques, which are used to characterize the fault rate of a system-on-chip (SoC), are investigated to provide a guideline to SoC designers for the realization of resilient SoCs.

Keywords: System-on-Chips, Resilient Design, Fault Injection, Soft Error, Fault Analysis

Introduction

Recently, the development of process technology has increased system performance, but the system failure probability has also significantly increased. It is important to consider that the system must be robust against failures when designing the circuits in addition to the cost, performance, and power consumption. Thus, resilient design becomes particularly prominent in system design to increase reliability. Many studies have been conducted for over half a century on fault tolerance and fault diagnosis to control the system in anticipation of inevitable defects.

Fault avoidance is a technique that addresses the faults of devices to prevent failure. This technique includes improving the reliability of the product through inspection and testing processes. However, a complete fault-free design and manufacturing process for complex devices such as processors is difficult to achieve. Further, these devices often encounter dangerous situations because of aging and internal and external defects of the hardware. Therefore, researchers have actively studied fault-tolerance techniques to ensure normal operation, although the devices may experience some faults. Fault-tolerance techniques have made much progress for nearly half a century. Among others, the redundancy scheme has been highlighted, which tolerates faults using additional resources. This scheme is simple to implement and provides high reliability. The redundancy scheme can be classified into four general types: hardware, software, information, and time redundancies.

To detect the faults in a system, the hardware redundancy technique uses replicated hardware, the software redundancy technique employs an additional program routine, the information redundancy technique inserts extra bits into the data to be transferred, and the time redundancy technique repetitively performs the same process and compares the results. These techniques involve additional costs and delays because of the additional resources; thus, designers should select one of these techniques after considering the tradeoffs. Structures and methods have been studied in recent years to reduce these additional resources.

System diagnosis is naturally important to handle a system failure caused by faults. Methods that test and evaluate the devices have also been developed following the development of fault-tolerance techniques. Fault-injection (FI) techniques have been widely used as a fault-testing plan. The FI techniques can be classified according to which device injects a fault into a target system. These techniques have their own advantages and disadvantages. Because most of the defects can be eliminated during the testing process, the proper FI technique should be selected in accordance with a particular design.

In this paper, we describe what types of faults occur in a system and where the faults originate. Then, the FI techniques, which are used to characterize the fault rate of a system-on-chip (SoC), are investigated to provide a guideline to SoC designers for the design of resilient SoCs.

The rest of this paper is organized as follows. We first classify the types of faults in Section 2 and explain the causes that lead to faults in Section 3. Section 4 addresses the FI techniques, in which a discussion is included. Finally, we conclude this paper in Section 5.

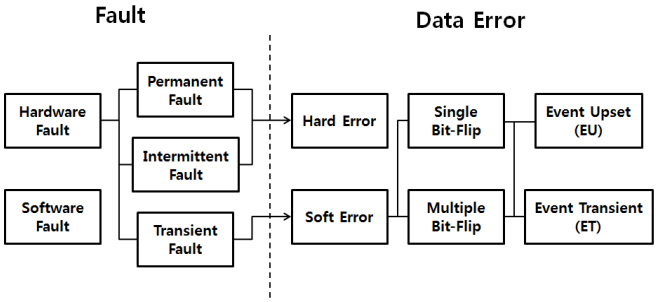

Fig.1. Block diagram of fault and error terminology focused on the hardware fault.

Type of Faults

A fault can be classified into hardware or software fault according to where it occurs. We focus on hardware faults in this study, which greatly affect the device and system. A hardware fault is classified into a permanent, an intermittent, or a transient fault according to how long it exists in a device (see Fig. 1). A permanent fault (stuck-at, stuck-open, and bridging faults) remains permanently in the circuit, a transient fault appears and disappears within a brief time, and an intermittent fault introduces repetitive broken data in a

specific place because of hardware damage. Permanent and intermittent faults occur because of inaccurate specifications, implementation mistakes, or component defects. A transient fault usually occurs because of internal and external noise.

The data errors that result from a hardware fault include hard and soft errors. A hard error causes data corruption because of hardware faults arising from permanent and intermittent faults. A soft error causes data corruption because of a disturbance in the environment, such as alpha particles or neutrons, and originates from transient faults. In contrast to a hard error, a soft error arises under conditions where the device is not damaged. A soft error can be divided into single and multiple bit flips. A single bit flip consists of one data flip, and multiple bit flips consist of several data flips. Further, a single bit-flip can be categorized into a single event upset (SEU) or a single event transient (SET), depending on where it occurs. An SEU occurs at storage element, e.g. in the latch or flip-flop, whereas an SET appears in combinational logic. Erroneous SEU values in storage element can potentially be captured in the following sequential logic. An SET in the combinational logic encounters fewer occurrences of rates of failure than an SEU because the errors are reduced by logical, temporal, or electrical masking. However, higher cost is involved in correcting the error because the result of the operation is directly propagated as soon as the input data are entered.

The soft error rate (SER) is defined as the occurrence rate of a soft error in a device. The number of failures-in-time (FIT) or the mean time between failures (MTBF) are commonly used to express the SER. The main source of SER originates from the flip-flops in most embedded digital applications without a microprocessor [1].

Cause of Hardware Faults

Understanding the types of faults and how they occur is essential for fault modeling and diagnosis. The majority of causes of hardware faults result from the development of process technology. Following the development of process technology, the probability of encountering process variations or external noise sources (alpha particles or neutrons in cosmic rays) increases in the semiconductor manufacturing process, leading to soft errors. Similarly, VLSI circuits that operate under high operating speeds and low supply voltage are susceptible to process variations; thus, these circuits have a higher error probability owing to the switching delay of transistors. In addition, the defects per chip area in a VLSI process increase with the increasing number of on-chip transistors, which increases the probability of failure. Thus, process-technology improvements result in both hard and soft errors.

A. Cosmic-ray Particle

Cosmic rays cause soft errors in the system. Most cosmic rays do not reach the Earth's surface. However, cosmic rays produce energetic secondary particles such as neutrons and protons by collision with a nucleus in the Earth's atmosphere. A neutron by itself cannot interfere with the circuit; however, it is absorbed by the nucleus and causes a "neutron capture" reaction that emits alpha particles. These alpha particles generate an incorrect value when they collide with the circuit. Neutrons also originate from nuclear-fission reactions or from the creation and destruction of radioactive nuclei. Alpha particles originate from various radioisotopes during radioactive decay and are detected in materials such as glasses, fillers, alumina, plastic, and even in the sea [2].

A study that cosmic rays potentially affect devices was presented in 1962 [3]. Communication disruption due to cosmic rays actually occurred, and the cosmic-ray event rate was calculated in 1975 through experiments using a scanning electron microscope [4]. Devices have become more vulnerable to neutrons and alpha particles from cosmic rays because of the development of the process technology [5]. The fact that circuits are more susceptible to atmospheric neutrons was confirmed by a comparison of the SER caused by neutrons depending on the scaling of device size of CMOS transistors [6]. By checking the SER caused by alpha particles and radiation, it observed that the circuits are vulnerable to alpha particles when the operating voltage of the devices was lowered in the sequential logic, static combinational logic, and SRAM [7]. Reference [8] confirmed that the multi-bit error rate for 90-nm SRAM was slightly higher than that for 130-nm SRAM.

B. Noise Sources

Layman and Chamberlain demonstrated that the various noise sources that cause soft errors are thermal, shot, and l/f noise [9]. Thermal noise is caused by heat when the charge carriers (electrons or holes) move erratically in the capacitor. Thermal noise affects the semiconductor threshold voltage and flips the original value in the logic, resulting in a soft error. Thermal noise can be modeled with the voltage or current [10]. Shot noise is generated when the carriers pass over the potential barrier in a semiconductor, and the number of carriers becomes irregular. Because the direction and speed of the electron motion is irregular, each carrier introduces a problem in the semiconductor. The 1/f noise is caused by conductance fluctuation, which is inversely proportional to the frequency. The 1/f noise in the internal components increases significantly in the low-frequency region, and the noise decreases in the high-frequency region. Thus, these additional noise sources attack the noise margins of the semiconductors and increase the SER.

C. Critical Charge

Critical charge is the minimum required amount of charge to change the states of a semiconductor. When enough critical charge is collected, the logic value is changed. By decreasing the semiconductor size, the collected charge required to upset the logic also decreases and becomes susceptible to soft errors. Similarly, critical charge has been confirmed to decrease under lower operating voltages and smaller feature sizes [11]. Reference [12] confirmed that the SER is altered depending on several factors, including the critical charge. A device-level 3D simulation was performed to model the relationship between the bit error rate and the critical charge values in 90-nm SRAM [13].

D. Crosstalk

Crosstalk is electrical interference that occurs when the distance between two conductors is sufficiently small. Narrowing of the distance resulting from deep submicron technology causes electrical distortion and adversely affects reliability. The high-frequency operation of VLSI causes a skin effect propagated along the surface of a conductor [14]. This skin effect causes frequency-dependent interconnection resistance. The reliability problem of a circuit can be easily found in other places because of the increasing signal interference of the crosstalk in a smaller transistor and the interconnect dimensions [15]. Most designs encounter potentially soft errors from the RC delay, noise interference, and crosstalk [16]. To investigate the crosstalk properties, coupled RLC parameter values on four different interconnects were measured for the 0.13-µm and 0.18-µm processes [17].

E. NBTI

Negative bias temperature instability (NBTI) is a type of aging. The time delay of a circuit increases in proportion to the transistor threshold voltage (Vth). NBTI leads to timing error because the initial value of Vth for a PMOS varies with the negative bias and temperature of a circuit that has been used for a long time. This phenomenon was observed in 1967 [18]. Reference [19] demonstrated that a longer exposure time to a negative voltage at the gate results in a larger fluctuation in the threshold voltage. Further, a larger change in Vth results in more occurrences of the critical timing problem.

Schroder and Babcock presented many process conditions such as oxide damage; the temperature; the oxide electric field; the presence of hydrogen, boron, nitrogen, water; and the gate length that affect the NBTI sensitivity [20]. The reliability of the NBTI significantly decreases when the transistor operates at high temperature, has a small gate length, and has a large content of boron, nitrogen, hydrogen, or water. The fact that hydrogen increases the NBTI was proven in [21]. Similarly, the lifetime of a semiconductor is significantly reduced because the change in Vth is different, depending on the boron content of the gate oxide and the thin gate length [22]. The fluctuation in Vth increases according to the NBTI stress in nitrogen oxide compared with that in pure SiO2 [1]. Water can also affect Vth when the gate oxide layer is formed. As the size of CMOS devices is gradually reduced with the development of process technology, nitride oxide is being used in the gate instead of the existing SiO2 to reduce the gate insulator film and improve the performance. However, the thin nitride oxide is very sensitive to NBTI stress; thus, the PMOS transistor easily acquires defects compared with that using the existing SiO2 [1].

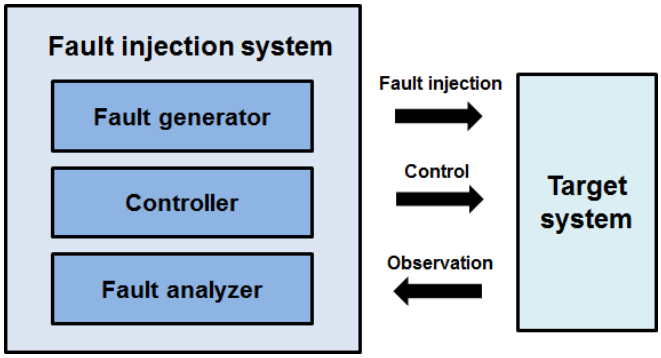

Fig.2. Fundamental environment of Fault Injection

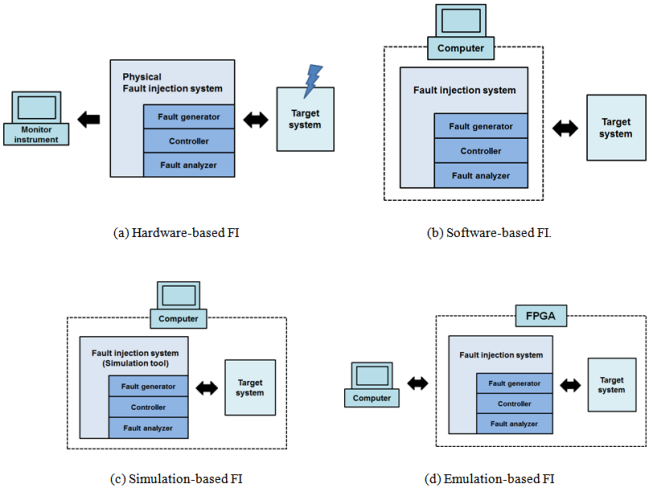

Fig.3. Fault Injection Techniques

Fault Injection (FI) Techniques

FI is adopted to verify the reliability of a system or to perform fault modeling. In this manner, we can ensure the sensitive part of the system against faults and the potential lack of fault tolerance to create a resilient design. The basic environment of the FI method includes FI system and the target system (see Figure. 2). The FI system interacts with the target system for fault generation, control, and fault analysis. The FI methods can be classified into four techniques as follows: Hardware-based FI, Software-based FI, imulation-based FI,and Emulation-based FI.

A. Hardware-based FI

Hardware-based FI is the most realistic method, which makes target system to experience faults in a physical level and measures the occurrence of the failure (see Figure 3(a)). The circuit is tested using the change in the operating power or temperature or the external shocks that cause transient errors. Moreover, this technique directly provides a stimulus at the pins or the sockets. The testing speed is fast owing to the real-time FI structure. By directly changing the environment, a wide range of circuits can be evaluated through these disturbances. However, its processes are difficult to monitor and control because we do not know the exact moment when a fault is injected by the disturbance. In addition, damage can be

done to the target system because the actual circuit cannot be restored after testing [23].

A circuit was validated by using a pin-level FI tool (MESSALINE) by derivation of the experimental measurements such as defective time distribution and size [24]. Wang tested the fault-tolerance capability of a software to changes in the power supply and payload at a satellite's on-board computer [25]. He injected the faults through a cable and monitored the changes in the output port.

Laser injection schemes into a system are available. The reliability of time-resolved ICs exposed to a pulsed laser was evaluated [26]. The SER was confirmed by calculating and normalizing the cumulative error histogram in accordance with the laser pulse delay of the circuits. Pin-level FI was conducted to verify a fault-tolerant multiprocessor system (FASST) [27]. FASST performed a fast fail-silent technique that analyzed the error detection coverage and latencies. A method that used a high-intensity laser in the microcontrollers was proposed [28]. Its drawback was that the disturbance of the circuit could be not completely controlled in the experiment.

B. Software-based FI

Software-based FI causes a software fault by modifying the execution code of the actual running software in the system (see Figure 3(b)). Software-based FI is practical because the required hardware and software are actually used in the device, and additional hardware to inject a fault is not required. However, the method suffers from limitations in terms of the types of faults injected by the software. In addition, detailed information on the hardware and software is necessary to model and control the fault.

Wulf et al. injected faults by using a software-based FI tool for multi-core devices on a cache using MATLAB [29]. This tool can generate a cache error by randomly injecting faults in the data accessed by the load instructions. Roberto et al. suggested a method for selecting appropriate fault locations from the analysis of the circuit complexity by 3.8 million experiments [30], which significantly reduced the load size of the fault effectively and improved the performance. A dynamic software fault injection system that targeted the Apache Web server was proposed using the PIN framework- a dynamic binary instrumentation tool- from Intel [31]. To test the reliability of the server, this tool injected faults dynamically after recording the information of the fault locations. The work in [32] injected faults in microprocessors and the main memory circuits. Some of the FI methods were evaluated for software fault tolerance that detects and masks hardware errors, and the results were then compared. Several fault models were experimented on a communication channel between the serial port driver and the OS kernel to evaluate the effect on the system according to software FI [33]. Through the tests, the results of the average execution time, implementation complexity, coverage, and injection efficiency were manifested in detail.

C. Simulation-based FI

Simulation-based FI injects a fault into the design and observes the failure using computer simulation tools (see Figure 3(c)). Simulation-based FI operates along with the actual workload in the software program and can be used in every process of design for function verification. The reliability can be verified simultaneously by functional verification of the design. Performing fault modeling and control is possible without damaging the real system. Moreover, simulation-based FI can change the data of any location thanks to its superior accessibility.

Additionally, environment construction is cheap because additional hardware is not required. However, simulation-based FI suffers from the drawbacks of long simulation setup process and simulation time.

A fault injector that injects board-level component faults was implemented for board-level built-in test (BIT) software [34], which is suitable for testing the reliability of a BIT system because it is created to handle the lack of validation in the BIT software. A system C hardware simulation model that uses embedded benchmark software was proposed to reduce the hardware resources [35]. This model supports a mixed-level simulation conducted at an electronic system level and RTL. Ruano et al. used a simulation-based FI platform that models soft errors to evaluate the reliability of a system [36]. This platform has low circuit costs and high controllability and can be performed with both synthesizable and non-synthesizable models. Reference [37] revealed that injecting faults into all places in the RTL and gate-level designs is possible, which supports a C function to add new types of faults. Wang et al. tested a method that modifies the data of a processor using a full system simulator-based FI tool (FSFI) on a system level [38]. The FSFI can check the processor components such as the integer register files, ALU, and decoder.

D. Emulation-based FI

Emulation-based FI injects faults into a design implemented in the FPGA (see Figure 3(d)). Emulation-based FI is proposed to overcome the long simulation time of simulation-based FI. Diagnosis can be processed quickly with real-time or partial reconfiguration. However, emulation-based FI is constrained by the precondition that the target design must be optimized in the FPGA before the experiment. Further, flexibly checking the response to the failure of the target design is difficult.

An FI method for any microprocessor implemented on an FPGA with an on-chip debugger (OCD) and a JTAG interface was implemented to complement the time bottleneck of an OCD built in a processor for debugging [39]. This implementation combined hardware and software FI in the FPGA design. The OCD-based method is a balanced technique in terms of tradeoffs with accuracy, cost, and speed. Entrena et al. proposed the FI tool, which can quickly and accurately analyze a failure by combining the gate-level model of FI and the RTL model, and they analyzed the SET and SEU of a logic gate on complex circuits such as microprocessors and memories [40]. Mogollon et al. experimented with a hardware emulation-based platform that optionally uses three modes: an ASIC mode that changes the logical structure, an FPGA mode that injects a fault into a configuration bit of the FPGA, and a beam-testing mode that exposed ion beams [41]. Control flexibility and high speed can be obtained at the same time through these different resources. Rahbaran et al. implemented a target system and

fault injector on an FPGA [42]. The method exhibited high testing-process accuracy and controllability characteristics. The work in [43] sent compressed data after connecting all circuits in the FPGA according to a decision rule to reduce the communication time between an FPGA and a host PC.

Discussions

Hardware-based FI potentially hazardous in that it can damage the hardware during the experiment. In addition, hardware-based FI has low fault coverage and controllability. Therefore, the study of hardware-based FI has been somewhat reduced in recent years. In spite of the lesser fault coverage of software-based FI and the low speed of simulation-based FI, they are widely used because of their other outstanding strengths, and many studies to overcome their weaknesses are currently being conducted. These methods feature considerable controllability, low cost, and hardware safety. Further, simulation-based FI can address many types of faults on a target system. The emulation-based FI method has also been widely adopted owing to its low cost, high speed, and wide fault coverage. Table 1 summarizes the characteristics of the four FI techniques.

TABLE.1. Features of the FI methods

	Cost	Speed	Controllability	Safety	Fault coverage
Hardware-based FI	High	High	Low	Low	Low
Software-based FI	Low	High	High	High	Medium
Simulation-based FI	Very low	Low	High	High	High
Emulation-based FI	Low	High	Medium	High	High

Conclusions

A fault is defined as the loss of normal functionality; thus, a circuit is required for fault modeling. Understanding the cause of a fault is necessary to recognize what type of fault occurs in a circuit. After fault characterization, we must select an appropriate resilient design technique to reinforce the reliability of the system. System reliability must also be improved by checking the requirements and characteristics of the design. Thus, various evaluation techniques such as simulation or emulation techniques are used for accurate assessment.

This paper has described the types of FI techniques available and where to use these techniques. In addition, we have presented the respective strengths and weaknesses of these techniques. FI is quite necessary in SoC design because the system failure probability is increasing. We believe that understanding the faults and FI techniques presented in this paper could provide a guideline to SoC designers for designing resilient SoCs.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIP) [NRF- 2014R1A1A1004150].

References

[1] N. Kimizuka, K. Yamaguchi, K. Imai, T. Iizuka, C.T. Liu, R.C. Keller and T. Horiuchi, "NBTI enhancement by nitrogen incorporation into ultrathin gate oxide for 0.10-μm gate CMOS generation," VLSI Technology, Digest of Technical Papers, 92-93, 2000.

[2] T. Karnik and P. Hazucha, "Characterization of soft errors caused by single event upsets in CMOS processes," IEEE Transactions on Dependable and Secure Computing, vol. 01, no. 02 , pp. 128-143, 2004.

[3] J.T. Wallmark and S.M. Marcus, "Minimum Size and Maximum Packing Density of Nonredundant Semiconductor Devices," Proceedings of the IRE, vol. 50, no. 3, pp. 286-298, 1962.

[4] D. Binder, E. C. Smith and A. B. Holman, "Satellite Anomalies from Galactic Cosmic Rays," IEEE Transactions on Nuclear Science, vol. 22, no. 6, pp. 2675-2680, 1975.

[5] C. Constantinescu, "Trends and Challenges in VLSI Circuit Reliability," IEEE Micro, vol. 23, no. 4, 14-19, 2003.

[6] J.-L. Leray, J. Baggio, V. Ferlet-Cavrois and O. Flament, "Atmospheric Neutron Effects in Advanced Microelectronics," International Conference on Integrated Circuit Design and Technology(ICICDT), pp. 311-321, 2004.

[7] N. Seifert, P. Slankard, M. Kirsch, B. Narasimham, V. Zia, C. Brookreson, A. Vo, S. Mitra, B. Gill and J. Maiz, "Radiation-Induced Soft Error Rates of Advanced CMOS Bulk Devices," IEEE International Reliability Physics Symposium Proceedings, pp. 217-225, March 2006.

[8] J. Maiz, S. Hareland, K. Zhang and P. Armstrong, "Characterization of multi-bit soft error events in advanced SRAMs," IEEE International Electron Devices Meeting(IEDM), pp. 21.4.1-21.4.4, 2003.

[9] P.A. Layman and S.G. Chamberlain, "A compact thermal noise model for the investigation of soft error rates in MOS VLSI digital circuits," IEEE Journal of Solid-State Circuits, vol. 24, no. 1, 79-89, 1989.

[10] Texas Instruments Incorporated, "Noise Analysis in Operational Amplifier Circuits, Digital Signal Processing Solutions," Application Report, SLVA043B, 2007.

[11] Lianlian Zeng and P. Beckett, "Soft Error Rate 18 Sub-micron CMOS," Pacific Rim International Symposium on Dependable Computing(PRDC), pp. 210-216, 2007.

[12] Timothy C. May and Murray H. Woods, "A New Physical Mechanism for Soft Errors in Dynamic Memories," Annual Reliability Physics Symposium, pp. 33-40, 1978.

[13] R. Naseer, Y. Boulghassoul, J. Draper, S. DasGupta and A. Witulski, "Critical Charge Characterization for Soft Error Rate Modeling in 90nm SRAM," IEEE

International Symposium on Circuits and Systems (ISCAS), pp. 1879-1882, 2007.

[14] M.G. Walker, "Modeling the wiring of deep submicron ICs," IEEE Spectrum, vol. 27, no. 3, 65-71, 2000.

[15] C. Constantinescu, "Impact of Deep Submicron Technology on Dependability of VLSI Circuits," International Conference on Dependable Systems and Networks, pp. 205-209, 2002.

[16] D. Sylvester and K. Keutzer, "Rethinking deep-submicron circuit design," IEEE Computer Society, vol. 32, no. 11, pp. 25-33, Nov. 1999.

[17] Sun Lingling and Pew Rong, "Crosstalk estimation in deep sub-micron VLSI," International Conference on Microwave and Millimeter Wave Technology (ICMMT), pp. 891-894, 2004.

[18] B. E. Deal, M. Sklar, A. S. Grove, and E. H. Snow, "Characteristics of the Surface-State Charge (Qss) of Thermally Oxidized Silicon," Electrochemical Society, vol. 114, no.3, pp. 266-274, 1967.

[19] C. Schlundera, R. Brederlowa, P. Wieczorekc, C. Dahld, J. Holzd, M. Rohnerd, S. Kesseld, V. Herolda, K. Goserb, W. Webera and R. Thewesa, "Trapping mechanisms in negative bias temperature stressed p-MOSFETs," European Symposium on Reliability of Electron Devices, Failure Physics and Analysis, vol. 39, no. 6-7, pp. 821-826, 1999.

[20] Dieter K. Schroder and Jeff A. Babcock, "Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufacturing," Journal of Applied Physics, vol. 94, no. 1, 1-18, 2003.

[21] E. Morifuji, T. Kumamori, M. Muta, K. Suzuki, M.S. Krishnan, T. Brozek, X. Li, W. Asano, M. Nishigori, N. Yanagiya, S. Yamada, K. Miyamoto, "T. Noguchi and M. Kakumu, New Guideline for Hydrogen Treatment in Advanced System LSI," VLSI Technology, Digest of Technical Papers, pp. 218-219, 2002.

[22] T. Yamamoto, K. Uwasawa and T. Mogami, "Bias temperature instability in scaled p+ polysilicon gate p-MOSFET's," IEEE Transactions on Electron Devices, vol. 46, no. 5, pp. 921-926, 1999.

[23] Haissam Ziade, Rafic Ayoubi and Raoul Velazco, "A Survey on Fault Injection Techniques," The International Arab Journal of Information Technology, vo. 1, no. 2, pp. 171-186, 2004.

[24] J. Arlat, M. Aguera, L. Amat, Y. Crouzet, J.-C. Fabre, J.-C. Laprie, E. Martins and D. Powell, "Fault injection for dependability validation: A methodology and some applications," IEEE Transactions on Software Engineering, vol. 16, no. 2, pp. 166–182, 1990.

[25] Wang Ping, "The Fault-Tolerant Design and Fault Injection Test for Embedded Software," Second Pacific-Asia Conference on Circuits, Communications and System (PACCS), vol. 1, pp. 307-310, 2010.

[26] 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.

[27] 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.

[28] Jasper G. J. van Woudenberg, Marc F. Witteman and Federico Menarini, "Practical optical fault injection on secure microcontrollers," Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pp. 91-99, 2011.

[29] N. Wulf, G. Cieslewski, A. Gordon-Ross AND A.D. George, "SCIPS: An Emulation Methodology for Fault Injection in Processor Caches," IEEE Aerospace Conference, Big Sky, pp. 1-9, 2011.

[30] Roberto Natella, Domenico Cotroneo, Joao A Duraes and Henrique S Madeira, "On Fault Representativeness of Software Fault Injection," IEEE Transactions on Software Engineering, vol. 39, no. 1, pp. 80-96, 2013.

[31] Ang Jin, Jianhui Jiang, Jiawei Hu and Jungang Lou, "A PIN-Based Dynamic Software Fault Injection System," International Conference for Young Computer Scientists (ICYCS), pp. 2160-2167, 2008.

[32] Ruben Alexandersson and Johan Karlsson, "Fault injection-based assessment of aspect-oriented implementation of fault tolerance," IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN), pp. 303-314, 2011.

[33] S. Winter, M. Tretter, B. Sattler and N. Suri, "SimFI: From Single to Simultaneous Software Fault Injections," Dependable Systems and Networks (DSN), pp. 1-12, 2013.

[34] Jun Xu and Ping Xu, "The Research Of Memory Fault Simulation And Fault Injection Method For BIT Software Test," Instrumentation, Measurement, Computer, Communication and Control (IMCCC), pp. 718-722, 2012.

[35] Dongwoo Lee and Jongwhoa Na, "A Novel Simulation Fault Injection Method for Dependability Analysis," IEEE Design & Test of Computers, vol. 26, no. 6, pp. 50-61, 2009.

[36] O. Ruano, J.A. Maestro and P. Reviriego, "Performance Analysis and Improvements for a Simulation-based Fault Injection Platform," IEEE International Symposium on Industrial Electronics (ISIE), pp. 2299-2304, 2008.

[37] David Kammler, Junqing Guan, Gerd Ascheid, Rainer Leupers, Heinrich Meyr, "A fast and flexible platform for fault injection and evaluation in verilog-based simulations," Third IEEE International Conference on Secure Software Integration and Reliability Improvement (SSIRI), pp. 309-314, 2009.

[38] Wang Chao, Fu Zhongchuan, Chen Hongsong and Cui Gang, "FSFI: A Full System Simulator-Based Fault Injection Tool," Instrumentation, Measurement,

Computer, Communication and Control, pp. 326-329, 2011.

[39] 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.

[40] 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.

[41] J. M. Mogollon, H. Guzman-Miranda, J. Napoles, J. Barrientos and M. A. Aguirre, "FTUNSHADES2: A novel platform for early evaluation of robustness against SEE," European Conference on Radiation and Its Effects on Components and Systems(RADECS), pp. 169–174, 2011.

[42] B. Rahbaran, A. Steininger and T. Handl, "Built-in fault injection in hardware - the FIDYCO example," IEEE International Conference on Field-Programmable Technology, pp. 327-332, 2004.

[43] M.S Shirazi, B. Morris and H. Selvaraj, "Fast FPGA-Based Fault Injection Tool for Embedded Processors," International Symposium on Quality Electronic Design (ISQED), pp. 476-480, 2013.