Model-Based Fault Injection for Large-Scale Failure Effect Analysis with 600-Node Cloud Computers

Yohei Nakata[1*], Yasuhiro Ito[2], Yusuke Takeuchi[1], Yasuo Sugure[2], Shigeru Oho[2], Hiroshi Kawaguchi[1], and Masahiko Yoshimoto[1, 3]

[1]Graduate School of System Informatics, Kobe University, Kobe, 657-8501 Japan

[2]Central Research Laboratory, Hitachi, Ltd., Kokubunji, 185-8601 Japan

[3] Japan Science and Technology Agency (JST) CREST, Tokyo, 102-0076 Japan

E-mail: *nkt@cs28.cs.kobe-u.ac.jp

***Abstract*—We propose a large-scale fault injection system that can execute numerous model-based fault-injection simulations in a reasonable time using a cloud computing environment. We have developed a device model and a fault-injection flow of vulnerable SRAMs in a microprocessor. The fault-injection flow is applicable to very large scale failure effect analysis with the proposed fault-injection scheme. We developed a large-scale simulation environment that has 600 computing nodes in a public cloud computing environment. Leveraging the 600 computing nodes in the cloud, 672,000 model-based simulations with the proposed fault-injection scheme and reliability evaluation were conducted within 12 hr.**

Keywords—failure effect analysis; model-based design; processor-in-the-loop simulation; fault injection; cloud computing; SRAM

I. Introduction

Recently, VLSI is increasingly serving important roles in various industrial products. Therefore, its reliability is important. However, transistors are more vulnerable and sensitive to soft errors and negative bias temperature instability (NBTI) because the process technology is scaled down. In addition, increasing variability in the transistor worsens its reliability and LSI yield. On the LSI, an SRAM is comprised of the smallest-size transistors, which are therefore the dominant factor determining the LSI's reliability. Accordingly, high reliability is necessary for SRAM on the system LSI [1–2].

A fault injection system that can consider physical characteristics of the vulnerable SRAM for the system-level verification has been proposed [3]. The fault injection system can evaluate SRAM reliability in terms of operating stability for a system LSI. Large-scale verification considering the random process variation of each physical LSI can be performed by the fault injection system. Furthermore, the fault injection system can evaluate numerous LSI chips using the concept of a virtual chip that emulates characteristics of an actual silicon chip.

However, to perform exhaustive failure effect analysis of numerous system LSIs integrating numerous SRAMs, very large computational power is needed. In addition, these numerous failure effect analyses are best completed within a realistic time to converge the system design. Herein, we present a model-based fault injection scheme that has large-scale failure effect analysis capability. To complete numerous model-based fault-injection simulations within a realistic time, we developed a large-scale simulation environment in a public cloud computing environment.

II. Fault-Injection System

A processor-in-the-loop simulation (PILS) can provide information related to hardware features and can perform high-accuracy simulations in a prototype system. It tests actual control software running on a dedicated processor with the virtual prototype of the mechanical plant.

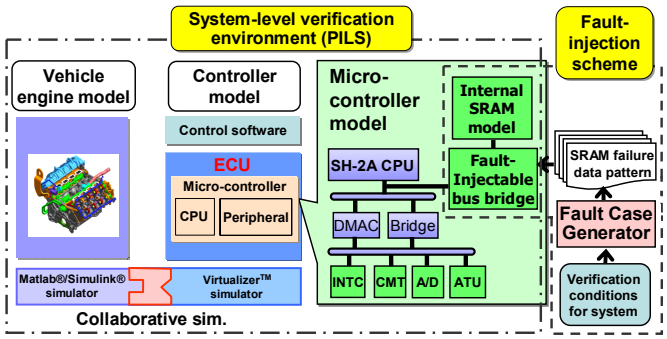

Fig. 1. System-level verification environment has a vehicle engine model and controller (ECU) model in which a micro-controller model is included.

The increase in minimum operation voltage (V_{min}) on an LSI degrades its device reliability because of power supply noise, IR drops, and/or soft errors. V_{min} on the entire micro-controller, including the logic block and SRAM block, is determined by the circuit with the highest value of V_{min} [1]. SRAM has a larger standard deviation for the threshold voltage than the logic block because its transistor size is smaller. To make matters worse, the SRAM capacity on the micro-controller is huge. Consequently, large SRAM blocks such as the cache memory or internal local memory determine V_{min} on the micro-controller.

Fig. 1 presents an overall view of the fault-injection system (FIS). The FIS integrates a system-level verification environment and the fault-injection scheme.

In this study, we handled an electric control unit (ECU) system for vehicle engine control that consists of a vehicle engine with sensors/actuators and the ECU with an SH-2A processor. It can simulate engine revolution control. The mechanical system including the engine, sensors, and actuators is emulated by MATLAB®/Simulink®. The SH-2A processor is emulated by Synopsys Virtualizer™.

Fig. 1 shows that the fault-injection scheme can inject failures based on a precalculated bit error rate (BER) into the internal. Several failure modes are supported as described in the next section. The fault-injectable bus bridge (FIB) is allocated between the SH-2A core and internal SRAM in the micro-controller; it arbitrates a normal access and false access (injected failure). The FIB intervenes in the memory transactions to destroy access data to the internal SRAM and switches to the failure data pattern when a failure occurs.

The fault case generator (FCG) uses various device parameters such as a supply voltage, temperature, and aging. It

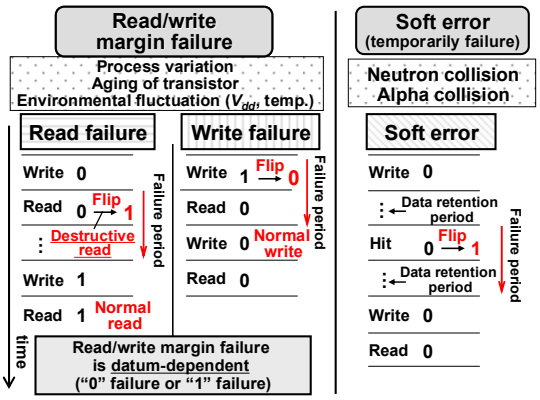

Fig. 2. Failure pattern examples in SRAM memory cell: read margin failure, write margin failure, and soft error.

generates time-series failure data patterns according to the parameters. The time-series failure data patterns are stored once in the FIB, which then injects the failure data into the memory transactions when accessing the failure address.

III. Modeling of Failures in SRAM

In this section, details of the methods for modeling the SRAM failure are described. By injecting SRAM's physical behavior from the device level to the system level, the proposed model can reflect the SRAM well as an actual silicon chip.

A. Behavior of SRAM failures on a device level

To inject the SRAM failure and to estimate the system-level verification, modeling the SRAM failures is necessary. Fig. 2 shows the failure pattern examples of the read/write margin failure and soft error. The models in the figure are derived from physical SRAM behaviors.

The read margin failure emerges as a destructive readout. The stored datum in a memory cell flips when the datum with no read margin is read out. The failure (flipped datum) lasts until it is rewritten. The write margin failure occurs when an attempt exists to write a memory cell with no write margin. In the write operation, the memory cell with no write margin cell does not flip to the write datum. This failure lasts until the flipped memory cell is normally written, similar to the read margin failure.

The read/write margin failure is mainly caused by process variations including random and systematic variations, aging of the transistor device, and fluctuations in the supply voltage and temperature. In addition, the read/write margin failure has datum dependence: either a "0" failure or a "1" failure for each memory cell. It is determined by the random variation of transistors in every SRAM memory cell.

The soft error is modeled as a temporary failure: a datum stored in a memory cell suddenly flips. This failure also lasts until it is rewritten.

Access time violation was not considered in this study because the read/write margin failure and soft error are dominant at low operating frequencies.

B. Fault-Injection Flow for System-Level Verification

Fig. 3 presents an illustration of the proposed fault-injection flow for system-level verification, which starts at the device level and ends at the system level. First, on the device level, SPICE Monte Carlo simulations using a transistor-level SRAM netlist are conducted considering various device parameters. In the following subsection, we explain the device parameter. As

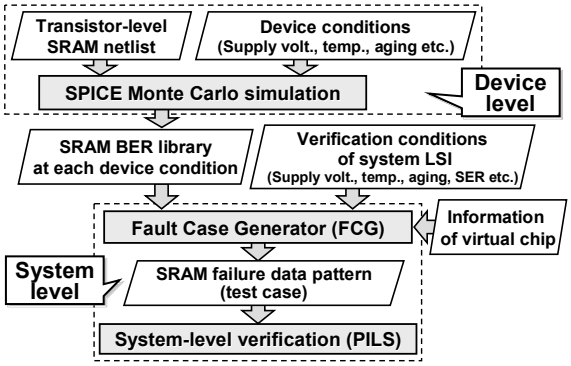

Fig. 3. Proposed fault injection scheme flowing from a device level to a system level.

a result of the Monte Carlo simulations, an SRAM BER library including BERs on various device conditions is obtained. Next, the generated SRAM BER library, the verification condition under which a system LSI designer wants to verify, and information of the virtual chip are used as inputs to the FCG. The virtual chip [3] has information about failure addresses, as described in detail in the next subsection. Eventually, the FCG calculates and outputs SRAM failure data patterns, which are fed to the PILS as system-level verification.

In this way, the device-level behavior of the SRAM is injected into the system-level verification environment. If SRAM of another kind must be evaluated on a system level, it can be achieved by creating a new SRAM BER library. The same fault injection flow is then conducted.

C. Modeling Failures for System-Level Fault Injection

In an actual silicon chip, read/write margin failures and soft errors are distributed randomly across the chip as a result of the random variation derived from transistor physics. The datum-dependence of the read/write margin failure is also determined randomly by the random variation.

The virtual chip can reproduce these features of failure on an actual silicon chip. Therefore, it has repeatability. The failure addresses are determined to be random spatially. The datum-dependences of the read/write margin failure are determined randomly as "0" or "1". The greatest benefit of using the virtual chip is its large-scale verification capability. Each virtual chip has different addresses of failures and therefore has different reliabilities. The failure addresses might make the virtual chip fail or might not. The FIS with the virtual chip concept can readily perform large-scale verification using numerous virtual chips without numerous actual chip samples.

IV. Extension to a Cloud Computing Environment

To perform large-scale verification using numerous virtual chips, great amounts of computing resources are needed. Preparing numerous computers is too costly; it is difficult to utilize so many computers efficiently at all times. Moreover, performing large-scale verifications with small-scale computers is too time-consuming. To obtain numerous computers when the need arises, we introduce a cloud computing environment as a computing resource. Using the cloud computing environment, we can reduce the cost of large-scale verification in accordance with the usage of computing resources in the cloud.

A. Building Large-Scale Fault-Injection Environment in Cloud Computing Environment

Fig. 4 shows an overview of large-scale fault-injection environment in the public cloud computing environment. 600

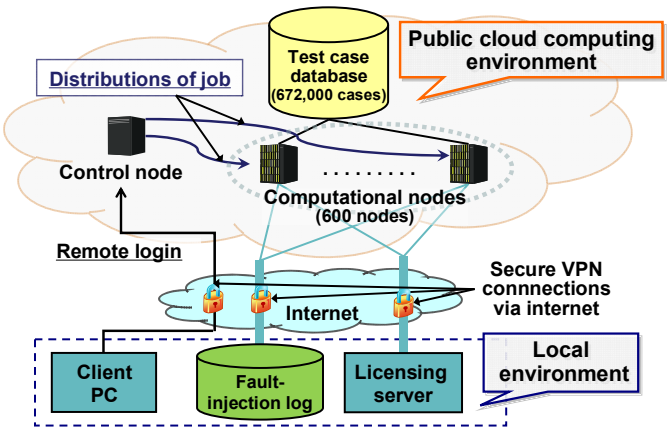

Fig. 4. Overview of large-scale fault-injection environment in a cloud computing environment.

computational nodes are launched for the evaluation of this paper. Each computational node has the equivalent CPU capacity of a 1.0–1.2 GHz 2007 Xeon processor (Intel Corp.). Each computational node can access a test case database that contains 672,000 test cases (SRAM failure data patterns). The 672,000 test cases consist of 56 test cases of 6,000 virtual chips in two-memory mode. The details of test cases are described in Section VI.A. To prevent data access congestion, a copy of the test case database is assigned to each 20 computational nodes. A control node in the cloud distributes jobs to the computational nodes and manages the progress of the entire simulation. Each job includes locations of input test case and output log file, along with parameters of simulation. The control node is operated by a client PC in the local environment via remote access. Software licenses of the Virtualizer™ on each computational node are provided using a licensing server in the local environment via secure VPN connections. By such means, we can use an existing license in the cloud while maintaining its security and the site of the license server. Output fault-injection logs are once output to each computational node; then they are compressed and transferred to a fault-injection log database in the local environment. By compressing the output logs, the log size is reduced to less than 5%. Then we can transfer 600 computational nodes' logs that output in parallel with the secure VPN connections. These fault-injection logs, which can contain any state of a running processor and software, are useful for failure effect analyses.

We obtained 600 computational nodes in this paper by the limit of the amount of the cloud computing environment's resources. This evaluation environment in this paper, however, can augment its scale and throughput as long as computing resources in the cloud are obtainable.

V. 7T/14T Dependable SRAM

A. 7T/14T SRAM

Fig. 5(b) depicts the 7T/14T SRAM memory cell (14T for two memory cells) [9]. Fig. 5(a) shows that two PMOSs are added to internal nodes ("N00 and N10", "N01 and N11") in a pair of conventional 6T SRAM memory cells. The area overhead of the 7T memory cell is 11% greater than that of the conventional 6T memory cell.

Table I shows that the 7T/14T memory cells have two modes.

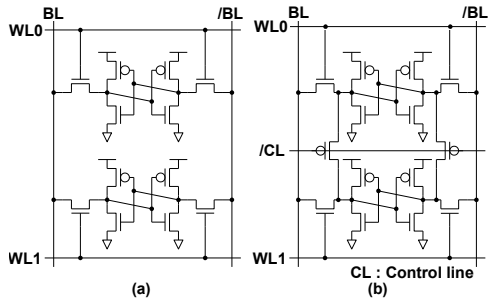

Fig. 5. (a) Conventional 6T memory cell (b) 7T/14T memory cell pair.

TABLE I. Two modes in 7T/14T memory cell

	# of memory cells comprising 1 bit	# of WL drives	CL
Normal	1 (7T/bit)	1	Off ("H")
Dependable(read)	2 (14T/bit)	1	On ("L")
Dependable(write)	2 (14T/bit)	2	On ("L")

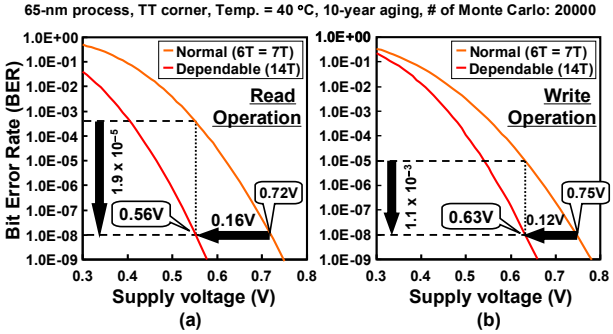

Fig. 6. Bit error rates (BERs): (a) read operation and (b) write operation.

• Normal mode (7T): The additional transistors are turned off (CL = "H"); the 7T cell acts as a conventional 6T cell.

• Dependable mode (14T): The additional transistors are turned on (CL = "L"); the internal nodes are shared by the bitcell pair. In write operation, both WL0 and WL1 are driven, but in read operation, either WL0 or WL1 is asserted, which ensures stable operation.

In the normal mode, a one-bit datum is stored in one memory cell, which means that it is more area-efficient. In the dependable mode, a one-bit datum is stored in two memory cells, although the reliability of the information differs from that of the normal mode. The "more dependable with less failure rate" information is obtainable by combining two memory cells [4]. In addition, the 14T dependable mode has better soft-error tolerance than the 7T normal mode because its internal node has more capacitance.

B. Bit Error Rate (BER)

Fig. 6(a) shows a bit error rate in the read operation. The static noise margin is used as a metric to evaluate read BERs. The dependable mode functions well below 0.56 V with a BER of 10^{-8} kept in the typical-case condition (TT corner, 40°C). The minimum operating voltage and BER are improved by 0.16 V and 1.9×10^{-5} in comparison with the normal mode.

Fig. 6(b) is a BER in the write operation. The write trip point is used as a metric to evaluate write BERs. The dependable mode functions at 0.63 V with a BER of 10^{-8} maintained. The minimum operating voltage and BER are improved by 0.12 V and 1.1×10^{-3} over the normal mode.

TABLE II. PARAMETERS OF EVALUATION

# of computational nodes in cloud	600
# of virtual chips	6,000
# of test cases / one virtual chip	112
Total # of test cases	672,000
Execution time	10 sec.
Range of supply voltage	0.3 V to 1.0 V
Range of temperature	–50 degC to 150 degC
σ_{Vth} of PMOS, NMOS	40 mV, 30 mV
Delta *Vth* of PMOS (aging)	–24 mV
SRAM capacity	128 Kbytes
Soft error rate	300 FIT@1.0 V

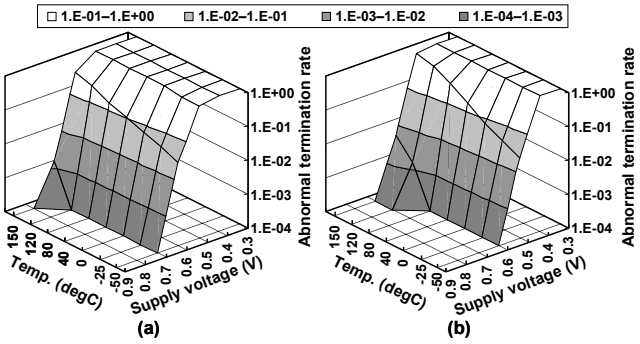

Fig. 7. Abnormal terminating rates of engine control ECU system: (a) 6T SRAM, (b) 14T dependable mode of 7T/14T SRAM.

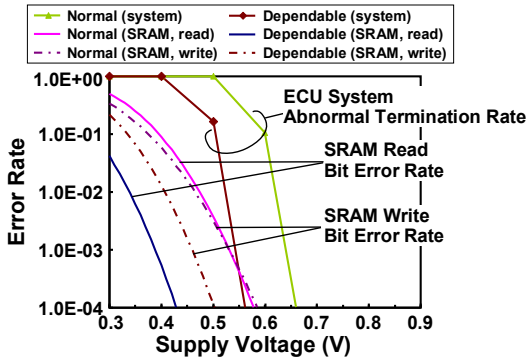

Fig. 8. Comparison of ECU system abnormal termination rates (error rate) and SRAM bit error rates.

VI. SYSTEM-LEVEL EVALUATION

To evaluate the proposed FIS integrated with the fault-injection scheme and system-level verification environment, we used the vehicle engine control ECU system presented in Fig. 1. In this evaluation, we used the 6T SRAM and 7T/14T dependable SRAM as internal SRAM of ECU. Vehicle engine control software first ran on the ECU. Faults were injected to the internal SRAM in the ECU running the vehicle engine control software.

A. Evaluation Methodology

Abnormal termination of the vehicle engine control software is judged in two ways: a watchdog timer interruption triggered by a runaway of the software and an access violation to an illegal address. A normal termination is judged as occurring when no abnormal termination occurs within the predefined execution time. Abnormal behavior of the mechanical system was not considered in this study, only the behavior of the electrical system. The BERs of the SRAMes are shown in Fig. 6. For the degree of aging of the transistor, we assumed degradation of the PMOS threshold voltage as –24 mV, assuming 10-year aging by negative bias temperature instability (NBTI).

Table II presents a summary of the parameters used for the system-level evaluation of the FIS. In the actual silicon chip, mapping of SRAM failure points differed for each chip. As a result, the impact of SRAM failures in each chip to the operating stability of each system was unique. Consequently, to evaluate the functional safety of the system, exhaustive system-level failure analysis for numerous chips is needed. For this evaluation, we generated and evaluated 6,000 virtual chips.

The large-scale fault-injection environment integrated with the 600 computational nodes in the cloud presented in Fig. 4 is used as a computing resource in this evaluation. Throughput of this simulation environment was obtained from this evaluation.

In this evaluation, inputs of supply voltages and operating temperatures did not change in time. We evaluated the static supply voltage (DC) and operating temperature characteristics for the abnormal termination of system. 672,000 test cases were evaluated in all. Results of the evaluation show that knowledge of the operating range for evaluating the functional safety of the system is obtainable.

B. Evaluation Result

As a result of the evaluation, the 672,000 test cases are evaluated by the 600 nodes within 12 hr. 1,146 GB fault-injection logs are compressed to 47.6 GB (reduced to 4.16%) and are transferred to the local fault-injection log database.

Fig. 7 shows the evaluation result of the abnormal termination rates in the vehicle engine control ECU system. The evaluation results obtained using the 6T SRAM and 7T/14T SRAM as the internal SRAM of ECU are, respectively, shown in Figs. 7(a) and 7(b). The evaluation result obtained using the 7T/14T SRAM (in dependable mode) improved V_{min} compared with that using the 6T SRAM. To analyze the reason for this, a statistical analysis of a large amount of virtual chips is necessary to determine what kind of SRAM failure or where it invokes abnormal termination of the system. This kind of analysis is left to another future work.

Fig. 8 depicts a comparison of the ECU system error rates (ECU system abnormal termination rates) and SRAM read/write BERs. Apparently correlation between the abnormal termination rate and the SRAM read/write BERs.

VII. CONCLUSION

As described herein, we proposed a large-scale fault-injection environment in the public cloud computing environment. Large-scale fault-injection integrated with 600 computational nodes has extremely high throughput of execution of model-based simulation. 672,000 simulations with the proposed transistor device-aware fault-injection scheme and reliability evaluation were done within 12 hours.

ACKNOWLEDGMENT

This work was supported by Synopsys Inc. for providing the licenses of Virtualizer™ in this paper.

REFERENCES

[1] K. Itoh, "Low-voltage scaling limitations for nanoscale CMOS LSIs," *International Conference onUltimate Integration of Silicon (ULIS)*, pp. 3-6, Mar. 2008.

[2] L. Chang et al., "A 5.3 GHz 8T-SRAM with Operation Down to 0.41 V in 65 nm CMOS," *Symposium on VLSI Circuits*, pp. 252-253, 2007.

[3] Y. Nakata et al., "Model-Based Fault Injection for Failure Effect Analysis – Evaluation of Dependable SRAM for Vehicle Control Units –," *Fifth Workshop on Dependable and Secure Nanocomputing (WDSN)*, pp. 91-96, Jun. 2011.

[4] H. Fujiwara et al., "Quality of a Bit (QoB): A New Concept in Dependable SRAM," *Ninth Int. Symposium on Quality Electronic Design (ISQED)*, pp. 98-102, 2008.