# Hardware-Simulation Correlation of Timing Error Detection Performance of Software-based Error Detection Mechanisms

Yutaka Masuda Masanori Hashimoto Takao Onoye
Department of Information Systems Engineering, Osaka University
Email: {masuda.yutaka, hasimoto}@ist.osaka-u.ac.jp, TEL: 06-6879-4528, FAX: 06-6879-4529

***Abstract*—Software-based error detection techniques, which includes EDM (error detection mechanisms) transformation, are used for error localization in post-silicon validation. This paper evaluates the performance of EDM for timing error localization with 65-nm test chips assuming the following two EDM usage scenarios; (1) localizing a timing error occurred in the original program, and (2) localizing potential timing errors that vary execution results. Experimental results show that the EDM transformation customized for quick error detection detects 25% of timing errors in the original program in the first scenario and 56% of non-masked errors in the second scenario. However, these hardware measurement results are not consistent with the simulation results of our previous work. To investigate the reason, we focus on the following two differences between hardware and simulation; (1) design of power distribution network, and (2) definition of timing error occurrence frequency. We update the simulation setup for filling the difference and re-execute the simulation. We confirm that the simulation and the chip measurement results are consistent, which validates our simulation methodology.**

***Index Terms*—electrical timing error, software-based error detection, EDM transformation, error detection**

## I. Introduction

Electrical timing error, which causes a system failure in a logically correct design due to electrical property of the chip, is becoming one of the most serious concerns in post-silicon validation. Electrical timing errors originate from supply voltage variation, temperature gradient, crosstalk noise, and so on [1], and these factors fluctuate dynamically depending on the circuit operation, such as a program running on a processor, and operation environment. In design time, accurately predicting the occurrence of electrical timing errors and their conditions is difficult, and hence unexpected electrical timing errors are often observed in post-silicon validation.

Post-silicon validation gives a wide variety of test patterns at various operating conditions. Once an unexpected system behavior is observed, we start on analyzing the circuit operation. In this analysis, we need to (1) notice error occurrence, (2) localize the error in place, e.g. ALU and cache controller, and time, and (3) manifest the occurrence condition [1]. The most efforts for this analysis are made in (1) and (2) [2], and hence reducing these efforts is highly demanded.

Error occurrence is often detected by observing abnormal behaviors, such as system crash, segmentation fault and invalid op code. End-result-check, which compares the execution result with the expected result, can be also used to find error occurrence. The next step is error localization and it is challenging, because the time interval between the error occurrence and the detection of such an abnormal behavior is quite long. It sometimes reaches billions of cycles [3]. Due to such a long error detection latency, it is difficult to know when and where it occurred, since the trace buffer, which is often used to record signals on a chip for post-silicon debug, has limited record depth of, for example, thousands cycles [4]. Therefore, reducing the error detection latency is helpful to facilitate the error localization. Assertion-based error detection with additional hardware is also proposed [5], [6]. In this approach, it is important when, where and how assertions are inserted for efficient detection with smaller area overhead [7].

Another approach that reduces error detection latency is software based approach, and QED (Quick Error Detection) transformation [3] is one of the software based methods. QED decomposes the input program into blocks and duplicates each block within the program at assembly level. Also for every pair of the original and duplicated blocks, QED inserts a register-level consistency check that compares calculation results. With this fine-grained checking, QED succeeded in dramatically reducing error detection latency. Reference [3] reported that for specific logic errors, QED improved error detection latency by six orders of magnitude. This shorter error detection latency helps improve the efficiency of post-silicon validation.

EDM (Error Detection Mechanisms) transformation [8] is another software-based error detection approach that was originally proposed for detecting soft errors. Reference [8] reported that for random bit flips injected to data memory the error detection coverage was over 90%. In [9], the coverage of over 80% was achieved for a single bit flip occurred in registers. EDM adds data and code redundancy to an input program written in a high-level language (e.g. C and C++), and generates a special program. Here, various programs, e.g., random instruction tests, architecture-specific focused tests, and end-user applications such as operating systems and games, can be given as an input program. The main advantage of EDM transformation lies in the fact that it can be applied to a high-level source code independent of the underlying hardware. To detect errors affecting data, EDM duplicates each variable in the program and adds consistency checks after every read operation. Here, the consistency check after the read operation makes the error detection latency longer while it is acceptable for soft error detection. Therefore, [10] devised another EDM implementation (EDM-L) that added consistency checks after every write operation aiming at shorter error detection latency. Reference [10] evaluated the effectiveness of the EDM-L for electrical timing error with noise aware simulation [11] supposing supply noise is the most primary source of electrical timing errors. It is reported that EDM-L detected 86% of the non-masked errors, i.e., errors that affected execution results, within 1000 cycles whereas the original EDM detected 37%

978-1-5090-1507-8/16/$31.00 

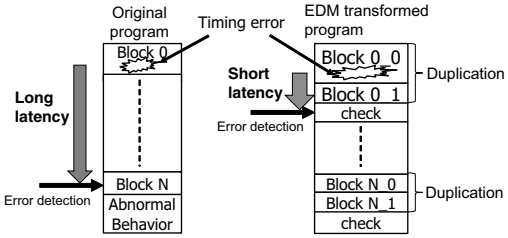


Fig. 1. Error detection by EDM transformation.

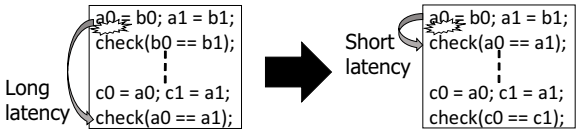


Fig. 2. Difference of error detection latency between EDM-O (left) and EDM-L (right).

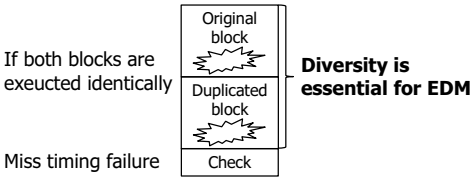


Fig. 3. Diversity is necessary to satisfy detectability.

of the non-masked error within 1000 cycles.

On the other hand, in [10], the EDM performance for electrical timing error is discussed only with simulation, and no silicon results are reported. This paper newly evaluates and reports the EDM performance for electrical timing error localization using 65-nm test chips. In addition, we investigate the reason of the inconsistency between the measurement and simulation results and point out two possible reasons; (1) the design of power distribution network, i.e., the magnitude of dynamic power supply noise, and (2) the definition of timing error occurrence frequency. By updating the simulation setup, we confirm that the measurement and simulation results are well correlated.

The rest of this paper is organized as follows. Section II explains EDM transformations and examines the necessary conditions in which EDM localizes an electrical timing error. Section III presents performance evaluation of EDM with fabricated test chips. Section IV examines the experimental results and discusses the consistency between the measurement and simulation results. Lastly, concluding remarks are given in Section V.

## II. Localizing electrical timing error with EDM

This section explains EDM transformations, discusses two scenarios of EDM usage in post-silicon validation, and describes the necessary conditions for error localization in each scenario.

### A. EDM transformation

To detect an error quickly after its occurrence, EDM converts an input program to a special program using several transformation techniques.

The EDM transformation and error detection described in [8] are exemplified in Fig. 1. First, EDM divides an input program into blocks and duplicates each block. The paired original and duplicated blocks are aligned in sequence. Second, for all the pairs of the original and duplicated blocks, EDM inserts check instructions to compare the execution results. Consequently, the EDM-transformed program executes the original block, the duplicated block and the check instruction in sequence for all the pairs of the original and duplicated blocks. This transformation is performed at C/C++ level.

The original EDM transformation (EDM-O) in [8] is useful for soft error detection, but it can be improved for shortening error detection latency. The left figure of Fig. 2 illustrates such an example. Suppose the memory write of variable $a0$ in the first line failed. In this scenario, EDM-O checks read variable [8], i.e., $b0$ and $b1$. Accordingly, the memory of $a0$ is not accessed for a long time and the inserted check is performed after a long time elapses. To shorten the error latency, the check should be performed immediately after the memory write access (right figure of Fig. 2). Motivated by this, [10] devised EDM-L (EDM for short Latency). EDM-L inserts check instructions for every variable write. Consequently, when an error occurs in the original block, we can expect that the next check instruction detects the error occurrence.

Furthermore, to satisfy detectability in the EDM program, the diversity between the original block and the duplicated block is crucially important. If the original block and the duplicated block are identical, the same error would occur in both the blocks and the check instruction fails to detect the error as illustrated in Fig. 3. In the EDM transformation, the original blocks and the duplicated blocks often split the memory space to gain the diversity, where the different memory addresses are expected to have different access times. EDM can include various transformations to maximize the diversity.

### B. EDM usage scenarios and necessary conditions for error detection

In this section, we list two EDM usage scenarios in post-silicon validation and discuss the necessary conditions that EDM needs to satisfy in each scenario.

We use the following two scenarios considered in [10].

Scenario1:
: When an original program was running, an electrical timing error occurred. We want to localize this error using EDM transformation.

Scenario2:
: We want to localize potential errors that vary execution results.

First, we examine the necessary conditions for the first scenario. In Scenario1, EDM should satisfy the two conditions below simultaneously (Fig. 4).

COND1:
: EDM-transformed program reproduces the error

which occurred in the original input program.

COND2:

EDM gives enough diversity so that the paired original and duplicated blocks output different computational results.

The first COND1 condition is necessary to investigate the root cause of the error observed in the original program. To reproduce the error occurrence, the EDM-transformed program should maintain the similar behavior of the original program. If EDM does not reproduce the same error, the error localization of the original program is impossible.

The second COND2 is the fundamental condition for EDM to work. If the original and duplicated blocks output the same wrong values, the inserted check instruction misses the error. Focusing on the second COND2 condition, dynamically fluctuating factors, such as supply noise, might help increase the diversity. The diversity originates from the timing characteristics of the fabricated chip under test and dynamically fluctuating factors. For example, power supply noise varies depending on the running program, which may improve the diversity.

On the other hand, COND1 is thought to become more difficult to satisfy as dynamically fluctuating factors become more significant. The supply noise, for example, of the chip on which the original program is running can be different from the noise of the EDM-transformed program. In this case, the error observed in the original program may disappear in the EDM-transformed program, and a new error may arise at another program location.

As stated above, Scenario1 requires that both COND1 and COND2 are satisfied. The next section experimentally investigates whether these two conditions are satisfied with 65-nm test chips, and compares the measurement results with simulation results shown in [10]. In Scenario2, the error observed in the original program does not need to be reproduced. Moreover, inducing a new error could be preferable since potential errors could be localized. Therefore, COND1 is not necessary. Only COND2 needs to be satisfied. The necessary condition for Scenario2 is the subset of the condition for Scenario1 and hence the experiments for Scenario1 are valid for Scenario2 as well.

## III. Experimental Evaluation of EDM Transformation

This section experimentally investigates whether EDM transformation works well in Scenario1 and Scenario2 with 65-nm test chips.

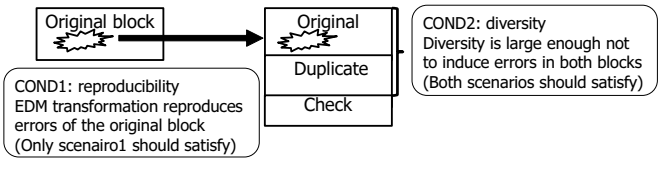


Fig. 4. Two conditions for EDM to localize electrical timing error in Scenario1.

### A. Measurement Setup

First, we explain the experimental setup. We used a 32-bit embedded processor (Toshiba MeP processor) implemented and fabricated in 65-nm CMOS technology. The chip size is 4.2 mm × 2.1mm.

We took three C-language benchmark programs, dijkstra, crc, and sha from MiBench [12] as original input programs. We implemented an EDM translator and used this translator to get full-EDM-L programs.

Fig. 5 shows the measurement setup consisting of a test chip, a DUT board, a DC voltage source and a PC. The packaged test chip is mounted on a DUT (device under test) board. The DUT board, which also includes a Stratix Ⅲ FPGA and SDRAM, is used as a logic analyzer and a pattern generator. For example, the data that should be stored in the instruction memory and the data memory of MeP processor is first transferred from the PC to the DUT board through the USB cable, and then the data is loaded to the on-chip SRAMs. After the program execution of MeP processor, the data in the on-chip data memory is downloaded to the PC through the DUT board. We also use an external DC source (Agilent6611C) to supply the voltage to the test chip.

With this setup, we can obtain the shmoo plot taking the following procedure. In each measurement, we set the clock frequency and supply voltage given to the test chip. Then, the data upload, the program execution and the data download are executed. When the downloaded data is identical to the expected data, the program execution is thought to be correct. When there is inconsistency, it is thought that the program execution failed. This measurement is repeated sweeping the clock frequency and supply voltage. We obtained the shmoo plots of the original and EDM-L programs (dijkstra, crc, and sha) for five test chips. The frequency interval was 5 MHz and the supply voltage was swept between 1.0 and 1.4 V with 0.1 V interval. Figs. 6 and 7 are the shmoo plots of the fastest and slowest test chips among the five chips, where the sha-full-EDM-L program was executed. Even while the five chips were taken from the same wafer, the chip speed is different. Here, we define a term of FMAX. For each program execution, each chip and each supply voltage, we can find the FMAX at which the execution result starts to be incorrect. This frequency is defined as the FMAX. For example, in the shmoo plot of Fig. 6, the FMAX at 1.0 V is 225 MHz, which is 15 MHz lower than that in Fig. 7.

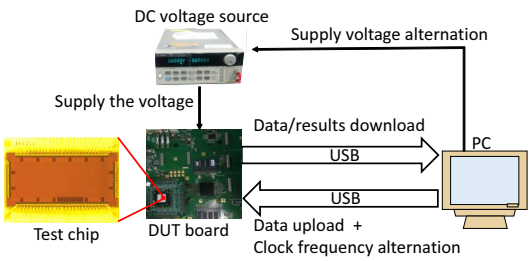


Fig. 5. Die photo of test chip and measurement setup.

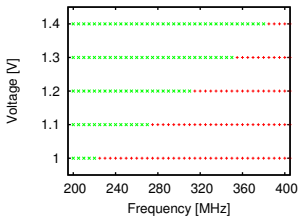


Fig. 6. Shmoo plot of the slowest chip (chip #1).

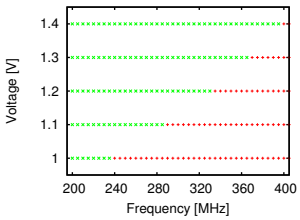


Fig. 7. Shmoo plot of the fastest chip (chip #5).

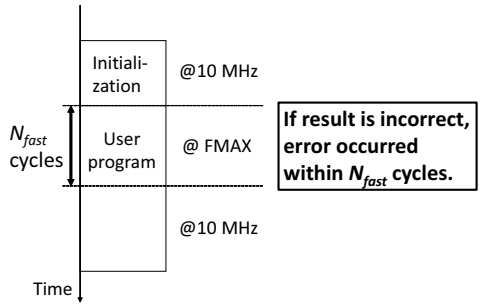


Fig. 8. Procedure of error cycle identification.

### B. Performance Evaluation Procedure

In this work, we focus on the first error that affects the execution result and check whether EDM satisfies COND1 and COND2.

For this purpose, we need to know when the timing error occurred. However, in the hardware it is difficult to know in which clock cycle the timing error occurred unlike the simulation. Therefore, we take the following evaluation procedure. We change the clock frequency during the program execution as shown in Fig. 8. The program execution starts at 10 MHz, and the processor initialization completes at this frequency. Note that 10 MHz is low enough for the correct processor operation. When the user program execution starts, the clock frequency is changed to the FMAX. After $N_{fast}$ clock cycles have passed, the clock frequency is again changed to 10 MHz. Under this configuration, if the execution result is incorrect, we can know the first error occurred within the first $N_{fast}$ cycles. If the execution result is correct, no error occurred. We repeat this measurement by changing $N_{fast}$ in binary search manner and finally identify the clock cycle when the first error occurred.

Besides, when we decomposed the program into blocks, we numbered the blocks from the beginning. We can know in which block the first error occurred from $N_{fast}$. We regard the difference of the error occurrence block numbers as the proximity of error occurrence locations in a similar way to [10]. When the difference is zero, the timing error is reproduced at the same block in the EDM-transformed program and COND1 is satisfied. COND2 was evaluated by checking whether the program was terminated by the check instructions. If the checker works, we subtract the error occurrence clock cycle from terminated clock cycle of the program and obtain the error detection latency.

### C. Evaluation Results

Fig. 9(a) shows the ratio of the samples that satisfied COND1 of reproducibility and COND2 of detectability. For every timing error that affected the execution result in the original program, we checked if COND1 and COND2 were satisfied in the EDM-transformed program. The chip measurement result shows that 25% of the errors in the original program can be reproduced and quickly detected.

On the other hand, in the simulation result of [10], which is shown in Fig. 10(a), EDM could not satisfy COND1 and COND2 simultaneously. In addition, the proportion that only COND1/COND2 is satisfied differs between the chip measurement and simulation. These differences will be discussed in the next section.

*1) COND1:* Fig. 9(b) shows the proximity of the errors occurred in the original and the EDM transformed dijkstra, crc and sha programs in the chip measurement. In our chip measurement, 66% of errors in crc and 20% of errors in dijkstra were reproduced. On the other hand, in sha, no errors were reproduced. As a whole, EDM-L reproduced 29% errors in the chip measurement whereas 4% of errors were reproduced in the simulation as shown in Fig. 10(b). These differences of the reproducibility are supposed to be due to the following two reasons.

The first reason is the difference in the power distribution network (PDN) between the simulation model and the hardware. In the simulations of [10], ten different PDNs are used for the simulation to evaluate the performance against various supply noises, and they are not prepared aiming to model the test chip.

The second reason is the definition difference of the FMAX of the error occurrence between the simulation and hardware measurement, whereas MeP processor was operated at the FMAX in both the simulation and hardware measurement. In the simulation, the FMAX was defined as the frequency at which a timing error started to occur at a flip-flop no matter whether the timing error affected the execution result or not (i.e. no matter whether it is masked or not). Hereafter, this FMAX is called as the FMAX of timing error. On the other hand, in the chip measurement, the FMAX was defined as the frequency at which timing errors started to affect the execution result, because the FMAX of timing error is difficult to obtain in the hardware measurement. This FMAX is called as the FMAX of incorrect execution. Here, the FMAX of incorrect execution is equal to or higher than that the FMAX of timing error. In other words, we executed the original and the EDM programs at higher frequency in the measurement compared to [10]. Fig. 11 exemplifies the cycle time difference between the FMAX of timing error and incorrect execution. This result was obtained by the simulation with EDM programs. We can see that 70% of the samples have $> 0.2$ ns difference.

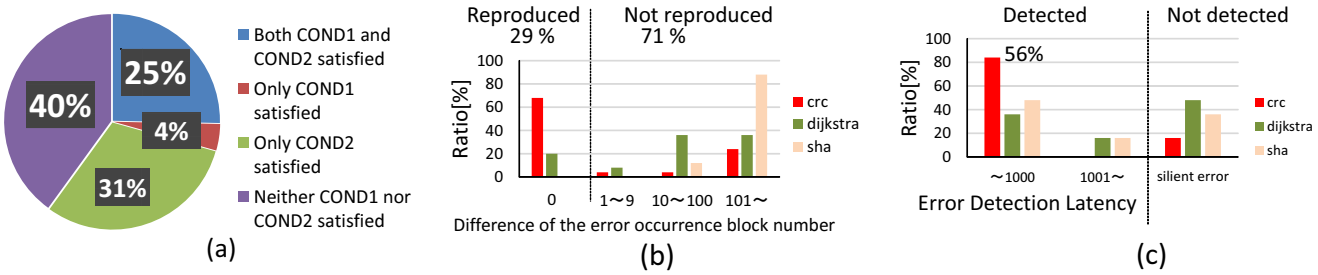


Fig. 9. Results of chip measurement. (a)COND1+COND2(Scenario1), (b)COND1, and (c)COND2(Scenario2).

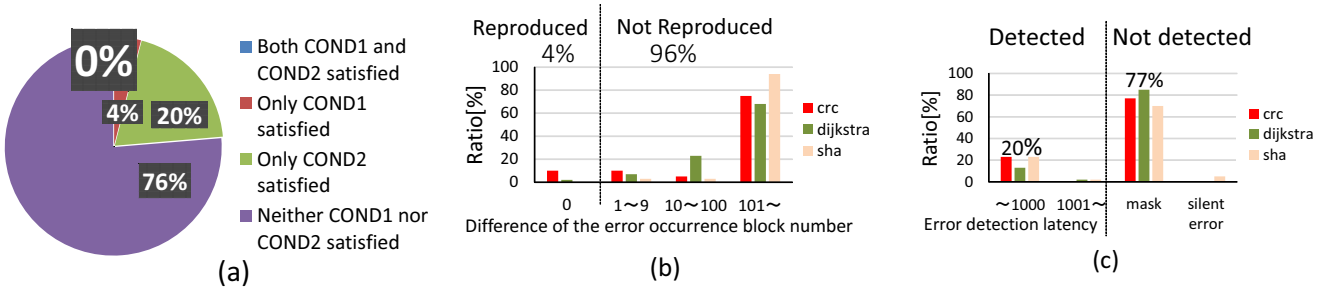


Fig. 10. Results of simulation in [10]. (a)COND1+COND2(Scenario1), (b)COND1, and (c)COND2(Scenario2).

*2) COND2:* Next, COND2 is examined. We first categorized the measured samples into detected samples and not detected samples. In [10], not detected samples were categorized into silent errors and masked errors, where silent errors are the errors that affected the execution results and were not detected by EDM, and masked errors are the errors that did not affect the execution results and were not detected by EDM. On the other hand, in this chip measurement, we focused on the errors affecting the execution result, and hence not detected samples correspond to silent errors.

Fig. 9(c) shows the proportions of detected errors and silent errors in the chip measurement. For detected errors, the histogram of the error detection latency is presented. From Fig. 9(c), we can see that 56% of the errors are quickly detected and 33% are silent errors. Fig. 10(c) is the simulation results of COND2 in [10], and it shows that 86% of the non-masked errors were detected within 1000 cycles. The EDM-L performance of detecting electrical timing errors affecting execution result in [10] is higher than in the chip measurement (86% versus 56%).

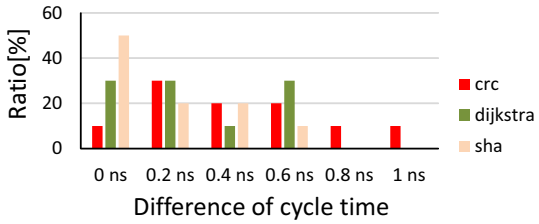


Fig. 11. Histogram of cycle time difference between FMAX of timing error and incorrect execution.

## IV. Simulation Update

The experimental results in the previous section show that in the chip measurement, COND1 is more satisfied and COND2 is less satisfied compared to the simulation in [10]. The possible reasons of these differences are (1) the difference of the power distribution network, and (2) the difference of the FMAX definition, as described in the previous section. In this section, we improve the correlation between the measurement and simulation by updating the simulation setup taking into account these two possible reasons.

### *A. Simulation Update Setup*

Firstly, we update the power distribution network. The chip measurement results in the previous section lead to a hypothesis that the supply noise in the test chip is smaller than that in the simulation and hence the errors are more likely to be reproduced in the chip measurement. To verify the above hypothesis, we suppose the test chip has ideal PDN as an extreme case. In other words, we execute the simulation based evaluation similarly to [10] except that the supply voltage is fixed and the supply noise is zero.

Secondly, we update the definition of the FMAX. In the chip measurement, the FMAX of incorrect execution was used while the FMAX of timing error was used in the simulation in [10]. To clarify the difference originating from this difference of FMAX, we applied the FMAX of incorrect execution to the simulation. In the simulation here, the FMAX of incorrect execution was searched with 200 ps interval, which is also similar to the measurement setup.

In this simulation, we prepared 3 programs and 10 chips, which were virtually fabricated by Monte-Carlo method as similar to [10], and hence totally 30 samples are evaluated.

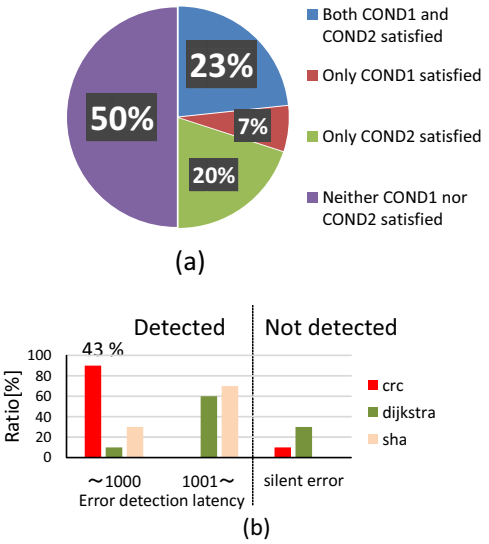


Fig. 12. Simulation results with ideal PDN. FMAX of incorrect execution is used for evaluation. (a) Scenario1, and (b) Scenario2.

### B. Updated Simulation results

Fig. 12(a) shows the result for Scenario1. Figs. 12(a) and 9(a) indicate that the proportions that both reproducibility and detectability are satisfied are almost the same and they are 23% in the simulation and 25% in the chip measurement. We can also see that the portions of only reproducibility/detectability is satisfied and neither satisfied are consistent between the simulation and chip measurement.

Fig. 12(b) shows the Scenario2 results. 43% of the errors are quickly detected and 13% are silent errors. Comparing Fig. 12(b) with Fig. 9(c), we can see that detectability for the non-masked error is similar, that is 43% in the simulation and 56% in the chip measurement.

In addition to the above simulation, other two update situations were evaluated; (1) only PDN was updated, i.e, simulation with ideal PDN and FMAX of timing error, (2) the definition of FMAX was solely updated, i.e., simulation with PDN in [10] and FMAX of incorrect execution. In the simulation of (1), the proportion of COND1 satisfaction was 13% and the detectability for non-masked errors was 70%. In (2), the proportion of COND1 satisfaction was 0% and 38% of non-masked errors were quickly detected. In these simulations, the proportion that reproducibility and/or detectability for non-masked errors were satisfied was still inconsistent with chip measurement.

Based on the discussion above, we can conclude that the simulation with ideal PDN and FMAX of incorrect execution reproduced the chip measurement results. This means that the supply noise in the test chip is smaller than that in the simulation, which is quite natural since the test chip was not designed for the purpose of EDM performance evaluation and hence the PDN was robustly designed.

## V. Conclusion

This work experimentally evaluated the error detection performance of the EDM transformation, which is one of C-level code transformation, for supply noise induced timing errors. To discuss the effectiveness of EDM for electrical timing error localization, we supposed two EDM usage scenarios; localizing an electrical timing error occurred in the original program (Scenario1), and localizing potential timing errors varying execution results (Scenario2). We experimentally evaluated the error detection performance in these two scenarios with a 65-nm test chips. Experimental results showed that the EDM transformation quickly detected 25% of electrical timing errors in the original program in the first scenario. In the second scenario, 56% of non-masked errors was detected. On the other hand, these measurement results were not consistent with the simulation results in the previous work. We found that this inconsistency came from (1) the design of power distribution network, and (2) the definition of FMAX used for evaluation. By updating the simulation setup, we confirmed that the EDM performance evaluated by the simulation was consistent with that by the chip measurement.

## ACKNOWLEDGEMENT

This work is partly supported by STARC.

## References

[1] P. Patra, "On the cusp of a validation wall,"*Design & Test of Computers*, vol. 24, no. 2, pp.193–196, June 2007.
[2] D. Josephson, "The good, the bad, and the ugly of silicon debug," *Proc. DAC*, pp.3–6, 2006.
[3] D. Lin, T. Hong, Y. Li, S. Eswaran, S. Kumar, F. Fallah, N. Hakim, D.-S. Gardner, and S. Mitra, "Effective Post-Silicon Validation of System-on-Chips Using Quick Error Detection," *IEEE Trans. CAD* , vol. 33, no. 10, pp.1573–1590, Oct. 2014
[4] S.-B. Park, T. Hong, and S. Mitra, "Post-silicon error localization in processors using instruction footprint recording and analysis (IFRA)," *IEEE Trans. CAD*, vol. 28, no. 10, pp.1545–1558, Oct. 2009.
[5] A.A. Bayazit, S. Malik, "Complementary use of runtime validation and model checking," *Proc. ICCAD*, pp.1052–1059, 2005.
[6] M. Boule, Z. Zilic, "Incorporating efficient assertion checkers into hardware emulation," *Proc. ICCD*, pp.221–228, 2005.
[7] S. Mitra, S. A. Seshia, and N. Nicolici, "Post-silicon validation opportunities, challenges and recent advances," *Proc. DAC*, pp.12–17, 2010.
[8] M. Rebaudengo, M.S. Reorda, M. Torchiano, and M. Violante, "Soft-error detection through software fault-tolerance techniques," *Proc. DFT*, pp.210–218, 1999.
[9] N. Nicolescu, R. Velazco, "Detecting soft errors by a purely software approach: method, tools and experimental results," *Proc. DATE*, pp.57–62, 2003.
[10] Y. Masuda, M. Hashimoto, and T. Onoye, "Performance Evaluation of Software-based Error Detection Mechanisms for Localizing Electrical Timing Failures under Dynamic Supply Noise," *Proc. ICCAD*, 2015.
[11] M. Ueno, M. Hashimoto, and T. Onoye, "Real-time On-chip Supply Voltage Sensor and Its Application to Trace-based Timing Error Localization," *Proc. IOLTS*, pp.188–193, 2015.
[12] 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," *Proc. Workload Characterization*, pp.3–14, 2001.
[13] L.D. Smith, R.E. Anderson, D.W. Forehand, T.J. Pelc, and T. Roy, "Power distribution system design methodology and capacitor selection for modern CMOS technology," *IEEE Trans. Advanced Packaging*, vol.22, no.3, pp.284–291, Aug 1999.
[14] T. Roy,L. Smith, and J. Prymak, "ESR and ESL of ceramic capacitor applied to decoupling applications," *Proc. EPEP*, pp.213–216, 1998.