IEICE TRANS. FUNDAMENTALS, VOL.E100–A, NO.7 JULY 2017

PAPER *Special Section on Design Methodologies for System on a Chip*

Performance Evaluation of Software-Based Error Detection Mechanisms for Supply Noise Induced Timing Errors

Yutaka MASUDA[†a)], ***Nonmember***, **Takao ONOYE**[†], ***and*** **Masanori HASHIMOTO**[†b)], ***Members***

SUMMARY Software-based error detection techniques, which includes error detection mechanism (EDM) transformation, are used for error localization in post-silicon validation. This paper evaluates the performance of EDM for timing error localization with a noise-aware logic simulator and 65-nm test chips assuming the following two EDM usage scenarios; (1) localizing a timing error occurred in the original program, and (2) localizing as many potential timing errors as possible. Simulation results show that the EDM transformation customized for quick error detection cannot locate electrical timing errors in the original program in the first scenario, but it detects 86% of non-masked errors potential bugs in the second scenario, which mean the EDM performance of detecting electrical timing errors affecting execution results is high. Hardware measurement results show that the EDM detects 25% of original timing errors and 56% of non-masked errors. Here, these hardware measurement results are not consistent with the simulation results. 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.

key words: *electrical timing error, software-based error detection, EDM transformation, error detection*

1. 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 dynamically fluctuate 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 of 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 Quick Error Detection (QED) 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, i.e., from billions of cycles to a few thousand cycles. This shorter error detection latency helps improve the efficiency of post-silicon validation.

Error detection mechanism (EDM) 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 source 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

Manuscript received September 6, 2016.
Manuscript revised January 19, 2017.

[†]The authors are with the Department of Information Systems Engineering, Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565-0871 Japan.
a) E-mail: masuda.yutaka@ist.osaka-u.ac.jp
b) E-mail: hasimoto@ist.osaka-u.ac.jp

DOI: 10.1587/transfun.E100.A.1452

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, we devise another EDM implementation that adds consistency checks after every write operation aiming at shorter error detection latency. We will evaluate both EDM implementations in this paper.

On the other hand, the performance of software-based error detection techniques for electrical timing error has not been studied explicitly, and their effectiveness against electrical timing errors is not clear. This paper focuses on the performance of EDM for electrical timing error localization. Firstly, this paper presents a case study that considers program-dependent supply noise with a supply noise aware simulation framework supposing supply noise is the most primary source of electrical timing errors. Next, we evaluate and report 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. Preliminary results of simulation and measurement were reported in [10] and [11], respectively.

Throughout this paper we consider two scenarios of EDM usage in post-silicon validation; (1) localizing the exact electrical timing error occurred in the original program, and (2) localizing as many potential errors as possible which could lead to abnormal behaviors. For the first scenario, two necessary conditions must be satisfied; error reproducibility, and diversity between the executions of the duplicated blocks. On the other hand, for the second scenario, only the diversity must be satisfied. We discuss the utility of EDM for electrical timing error localization in these two scenarios on the basis of experimental results.

The rest of this paper is organized as follows. Section 2 explains EDM transformations and examines the necessary conditions in which EDM localizes an electrical timing error. Section 3 presents a case study that investigates whether EDM transformation is helpful to localize an electrical timing errors with a supply noise aware simulation framework. Section 4 presents performance evaluation of EDM with fabricated test chips, and Sect. 5 examines the experimental results and discusses the consistency between the measurement and simulation results. Lastly, concluding remarks are given in Sect. 6.

2. 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.

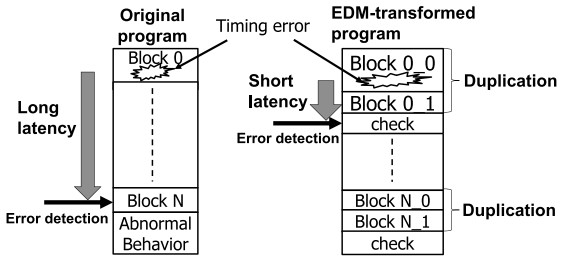

Fig. 1 Error detection by EDM transformation.

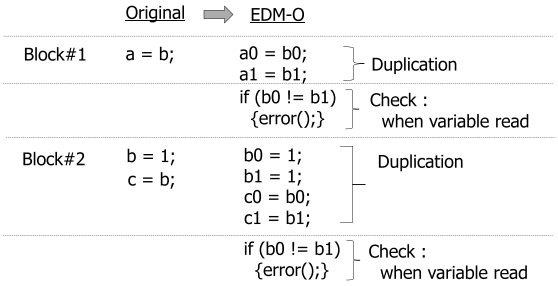

Fig. 2 An example of EDM-O code.

2.1 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, where the original EDM transformation in [8] is hereafter called EDM-O.

The transformation is performed at C/C++ level. Figure 2 gives an example of EDM-O-transformed code. First, EDM-O divides an input program into blocks, where each block consists of a set of operations in series. In the EDM-O block generation, a new block starts when a branch operation or variable read operation is found [8]. Here, variable read operations include those that read values stored in a register or stored in a memory. In the example code in Fig. 2, operations "a = b;" and "c = b;" are variable read operations since the value of variable b stored in a register or a memory is read. On the other hand, operation "b =1;" is not treated as a variable read operation since the constant value of 1 often comes from the immediate field of processor instructions. Second, we duplicate each block. The paired original and duplicated blocks are aligned in sequence. In the example code shown in Fig. 2, operation "a = b;" is duplicated to "a0 = b0;" and "a1 = b1;", and operations "b = 1;" and "c = b;" are duplicated similarly. Third, for all the pairs of the original and duplicated blocks, EDM-O inserts check operations to compare the read values, i.e., in Fig. 2, the values stored in b0 and b1 are compared after "a1 = b1;" and "c1 = b1;". Consequently, the EDM-O-transformed program exe-

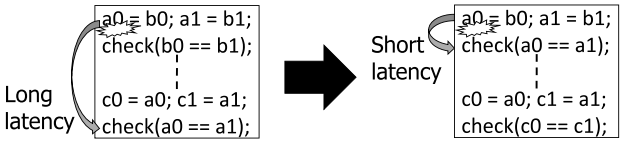

Fig. 3 Difference of error detection latency between EDM-O (left) and EDM-L (right).

cutes the original block, the duplicated block and the check operation in sequence for all the pairs of the original and duplicated blocks, where in some cases a branch operation is sandwiched between the check operation and the next original block.

EDM-O is originally developed for the purpose of improving soft-error detection coverage [8], and hence the check is constantly inserted after each variable read to know whether bit flips occurred in the memory, registers, or FFs. EDM-O is useful for soft error detection, but it can be improved for shortening error detection latency of electrical timing errors, i.e., the elapsed time between occurrence of electrical timing error and its detection. An electrical timing error arises as a write fail to memory, registers or FFs, and once a correct value is stored the value will not be corrupted by electrical timing errors. This means we can know whether an electrical timing error occurred or not immediately after the write operation. This is the main difference from soft error. Therefore, to use EDM for quickly detecting electrical timing errors, EDM should check the values in the memory, register or FFs after they are written. For this purpose, check operations should be performed immediately after variables are written, not read. The left figure of Fig. 3 illustrates such an example. Suppose the memory write of variable $a0$ at the first line failed. In this case, 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 is performed immediately after the memory/register/FF write access (right figure of Fig. 3). Motivated by this, we devised EDM for short Latency (EDM-L). Figure 4 shows an example of EDM-L-transformed code. EDM-L inserts check operations for every variable write. We note that EDM-L performs check operation after "$b1$ = 1;" whereas EDM-O does not check after this operation, which means EDM-L can quickly check whether timing error occurred in $b0$ or $b1$ compared to EDM-O. Consequently, when an error occurs in the original block, we can expect that the next check operation detects the error occurrence.

Furthermore, to satisfy detectability, 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 operation fails to detect the error as illustrated in Fig. 5. 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.

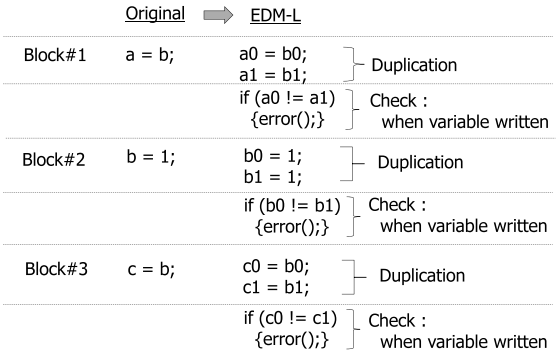

Fig. 4 An example of EDM-L code.

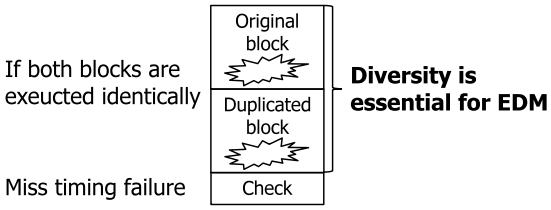

Fig. 5 Diversity is necessary to satisfy detectability.

2.2 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 consider the following two scenarios.

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 as many potential bugs as possible.

We first examine the necessary conditions for the first scenario. In Scenario1, EDM should satisfy the two conditions below simultaneously (Fig. 6).

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

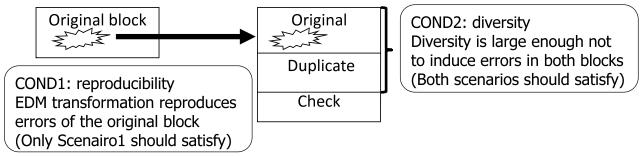

Fig. 6 Two conditions for EDM to localize electrical timing error in Scenario1.

the same wrong values, the inserted check operation 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. However, previous studies did not focus on COND1. It is not clear whether or how often COND1 can be satisfied in the EDM-transformed programs. In addition, it is not clear whether EDM satisfies COND2 for realistic electrical timing error. The next section experimentally investigates whether these two conditions are satisfied under dynamic power supply noise with a noise-aware logic simulation framework.

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 bugs 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.

3. Simulation-Based Evaluation of EDM Transformation

This section experimentally investigates whether EDM transformation works well in Scenario1 and Scenario2. The experiment supposes that dynamic power supply noise is the primary source of electrical timing bugs, and it accurately reproduces the impact of EDM on the dynamic supply noise and the consequent timing variations.

3.1 Experimental Setup

Our experimental evaluation was performed for an industrial embedded processor (Toshiba MeP processor). This processor was synthesized and laid out with an industrial 65nm library. In this experiment, the post-layout design was used for the simulations which will be explained later. 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 EDM-transformed programs.

Table 1 Impact of EDM-L transformation on cycle time and cache miss.

	execution cycles		inst. cache misses		data cache misses	
	orig.	EDM	orig.	EDM	orig.	EDM
dijkstra	24512	69838	45	161	11	20
	(1.00)	(2.85)	(1.00)	(3.58)	(1.00)	(1.82)
sha	30757	97831	44	167	25	42
	(1.00)	(3.18)	(1.00)	(3.80)	(1.00)	(1.68)
crc	19975	57252	9	29	35	71
	(1.00)	(2.87)	(1.00)	(3.22)	(1.00)	(2.03)

A value in parentheses is the ratio of full-EDM-L divided by original.

In EDM transformation, two types of check operations are inserted [8]; (1) data check and (2) code flow check. For the data checking, each variable v is duplicated as $v0$ and $v1$. Then, the consistency check is performed every time $v0$ ($v1$) is read in EDM-O. In EDM-L, the data check is performed every time $v0$ ($v1$) is written. The code flow check aims to detect an illegal change of the code execution flow, such as an illegal jump operation. The code flow check is inserted as follows.

First, EDM identifies all the basic blocks, i.e., branch-free sequences, in the program and checks whether all the statements in every basic block are executed in sequence by numbering the basic blocks. Second, checks for every test statement (e.g. if, else if, while) are inserted. EDM inserts the opposite test to both the true and false clauses to detect an illegal execution flow. The last target is call and return operation. Every procedure, i.e., function in the program, is associated with its unique number, and the number is checked for every call of the procedure. In this section, we duplicated all variables, and inserted all data checks and code flow checks. We call this transformation as full-EDM.

For the original and duplicated blocks, the same input data was stored at two different addresses of data memory, and each block accessed its own data in the data memory. Tables 1 and 2 list increases in the number of execution cycles and the number of cache misses by full-EDM-L and full-EDM-O, respectively. The number of execution cycles increased three to four times, and the increase in the number of instruction cache misses was similar. The increase in the data cache miss was roughly double, which is consistent with a fact that the data size is doubled in the full-EDM-transformed program.

We evaluated and compared the error occurrences in the original and EDM-transformed programs by logic simulation. Our logic simulation framework concurrently simulates two MeP designs; one is at register transfer (RT) level and the other is at gate level. The RT-level logic simulation is performed with zero-delay model, and hence the output is always correct disregarding the clock frequency and the given supply voltage. On the other hand, the gate-level logic

Table 2 Impact of EDM-O transformation on cycle time and cache miss.

	execution cycles		inst. cache misses		data cache misses	
	orig.	EDM	orig.	EDM	orig.	EDM
dijkstra	24512	65693	45	150	11	20
	(1.00)	(2.68)	(1.00)	(3.33)	(1.00)	(1.82)
sha	30757	120487	44	213	25	52
	(1.00)	(3.92)	(1.00)	(4.84)	(1.00)	(2.08)
crc	19975	60000	9	23	35	65
	(1.00)	(3.00)	(1.00)	(2.56)	(1.00)	(1.86)

A value in parentheses is the ratio of full-EDM-O divided by original.

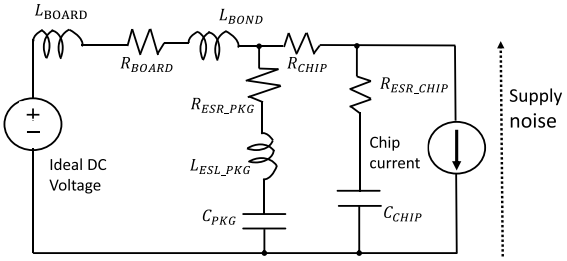

Fig. 7 An equivalent circuit of power distribution network.

simulation includes timing information and then may output wrong values. In this work, a noise-aware logic simulation method [13] is adopted to take explicitly into consideration program-dependent dynamic supply noise. This simulation method will be explained in the next subsection. Once an inconsistency is detected at a FF between RT-level and gate-level simulations, we can immediately notice a timing error occurrence. Thanks to this, we can know the exact location of timing error in time and space.

The comparisons of error occurrence between the original and full-EDM-L programs were performed for the following 300 situations. Due to manufacturing variability, each chip has different delay characteristics. To reproduce this, we virtually fabricated 10 MeP chips by Monte-Carlo method assuming that each instance delay randomly fluctuated with the standard deviation of 25% of the typical instance delay. In addition, depending on the final products, the LSI package may change. We assumed 10 package conditions, i.e. 10 sets of equivalent circuit parameters of power distribution network. The equivalent circuit model will be shown in the next subsection. In summary, $100 = 10 \times 10$ samples were evaluated for each program, i.e. 300 samples in total. Similarly, the full-EDM-O program was evaluated in 300 situations.

We focused on the first error occurred in the program execution, and its location was considered to check whether COND1 was satisfied. The minimum clock cycle that caused timing errors was searched with 2ps interval. When we decomposed the program into blocks, we numbered the blocks from the beginning. We regarded the difference of the block numbers as the proximity of error occurrence locations. 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 operation. Even if a timing error occurred in the EDM-transformed programs, the check operation sometimes miss the error. This case can be categorized into two groups; silent error and masked error. In the silent error case, the execution result is different from the correct result. On the other hand, in the case of masked error, the execution result is correct.

3.2 Noise-Aware Logic Simulation

In this paper, we used a noise-aware logic simulation method that could consider dynamic power supply noise in gate-level logic simulation [13]. The dependence of gate delay on supply voltage was first evaluated with HSPICE and it is expressed using a delay element whose delay is controlled by digital signals representing the supply voltage. Here, this delay element is described at RTL. By attaching this delay element to every gate and dynamically changing the digital signal that represents supply voltage, we can reproduce voltage-dependent gate delay in logic simulation.

When performing the above noise-aware logic simulation, we need to give a waveform of dynamic power supply noise. We prepared noise waveform information with the following two steps. First, we simulated the post-layout MeP design with the original and EDM-transformed programs by a transistor-level circuit simulator, and obtained waveforms of the current consumed by MeP for each program. Here, it should be noted that the transistor-level simulation to obtain the current waveform is very time consuming and it took three days for sha-full-EDM-O program. To make the CPU time needed for the entire evaluation in this work acceptable, the two-step procedure was adopted. Next, we gave this current waveform to the equivalent circuit of Fig. 7 and obtained the waveform of dynamic power supply noise. The nominal supply voltage was 1.0V. To reproduce various package assemblies and obtain corresponding noise waveforms, we used 10 sets of power distribution network (PDN) parameters in Fig. 7. The parameter setting is explained in the following.

We varied three parameters of C_{PKG}, $R_{\mathrm{ESR\_PKG}}$ and $L_{\mathrm{ESL\_PKG}}$ representing the package capacitor, and one parameter of C_{CHIP} representing the on-chip capacitor. The other five parameters were fixed as follows; L_{BOARD}=0.1nH, R_{BOARD}=5mΩ, L_{BOND}=0.3nH, R_{CHIP}=0.1Ω and $R_{\mathrm{ESR\_CHIP}}$=0.3Ω. We prepared five configurations of the package capacitor; (1) no package capacitor, (2) one NPO capacitor, (3) one X7R capacitor, (4) ten NPO capacitors in parallel and (5) ten X7R capacitors in parallel, where NPO and X7R are commercially available popular ceramic capacitors [14]. C_{PKG}, $R_{\mathrm{ESR\_PKG}}$ and $L_{\mathrm{ESL\_PKG}}$ of NPO and X7R are (100pF, 0.3Ω, 0.6nH) and (1nF, 0.6Ω, 0.6nH), respectively [14], [15]. As for the on-chip capacitance C_{CHIP}, two values of 3.5nF and 0.3nF were prepared. Consequently, 10 (=5×2) sets of PDN parameters were prepared and used to obtain the noise waveforms. Examples of the noise waveforms are shown in Figs. 8 and 9. These noise waveforms were given to the noise-aware logic simulation.

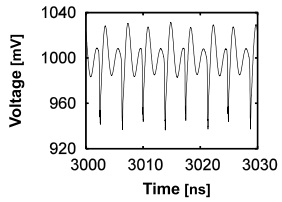

Fig. 8 A waveform example of inductive fluctuation.

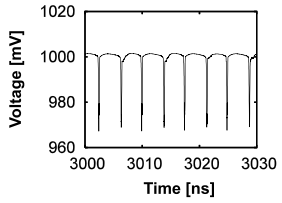

Fig. 9 A waveform example of resistive drop.

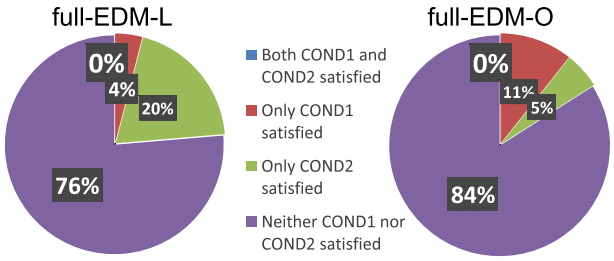

Fig. 10 Evaluation results of full-EDM.

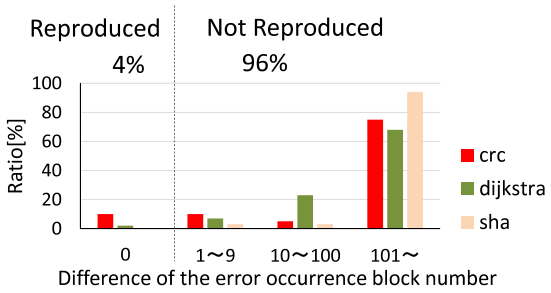

Fig. 11 COND1: difference of block numbers of first error occurrence between original and full-EDM-L programs. For each program, the number of samples is 100.

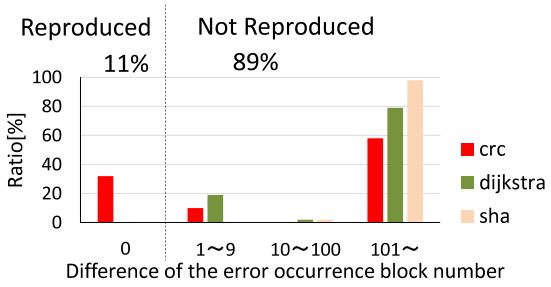

Fig. 12 COND1: difference of block numbers of first error occurrence between original and full-EDM-O programs. For each program, the number of samples is 100.

3.3 Evaluation Results

Figure 10 shows how many samples satisfied COND1 of reproducibility and COND2 of detectability. For every timing error in the original program, we checked if COND1 and COND2 are satisfied in the EDM-transformed program. Among 600 timing error samples, we could not find a sample that satisfied COND1 and COND2 simultaneously, which suggests EDM is less helpful in Scenario1. In addition, over 75% of errors satisfied neither COND1 nor COND2. Comparing full-EDM-L with full-EDM-O, we can see the difference in the proportion that only COND1/COND2 is satisfied. In the following, we examine the results for COND1 and COND2 separately in detail.

3.3.1 COND1

Figures 11 and 12 show the proximity of the errors occurred in the original and full-EDM dijkstra, crc, and sha programs. Remind that the proximity is defined as the difference of the block numbers of the error occurrence. The block number difference of zero means that the same error is observed in the original and EDM-transformed programs. In EDM-L, 10% of errors in crc and 2% of errors in dijkstra were reproduced. On the other hand, in sha, no errors were reproduced. In EDM-O, over 30% of errors were reproduced in crc, but no errors were reproduced in dijkstra and sha. As a whole, EDM-L and EDM-O reproduced only 4% and 11% errors, respectively. Such low reproduction ratios are mainly due to the following two reasons.

The first reason is that EDM changes supply voltage noise since the block duplication and check insertion change the instruction sequence and the usage of circuit blocks, such as memory and general purpose registers. In other words, even when the same instructions are performed, the supply noise could change, because the used registers and memory addresses are different and the inductive noises excited in the previous clock cycles are superposed. Figure 13 shows a comparison of noise waveforms between the original and full-EDM dijkstra programs, where the same instruction was performed in this clock cycle. We can see that the voltage waveforms are not identical. For further investigation, we evaluated the minimum supply voltage within a clock cycle every time mov instruction was performed. Figure 14 shows a histogram of the minimum voltage in the original crc program. We can see that the minimum voltage value ranges from 941 mV to 947 mV even though the same instruction of mov is performed. This waveform difference prevents the error reproduction.

The second reason is that EDM lengthens the program execution as previously shown in Tables 1 and 2. As the program becomes longer, a new timing error, which is different from the error observed in the original program, is more likely to occur. In addition, the duplication and frequent check insertion change the instruction composition of the program. Figure 15 shows the ratios of the instructions

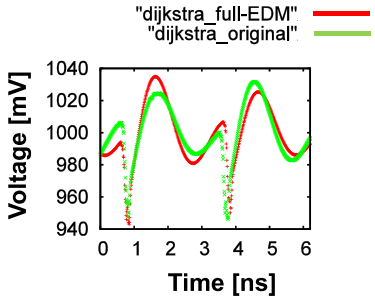

Fig. 13 Voltage inconsistency between original and full-EDM programs.

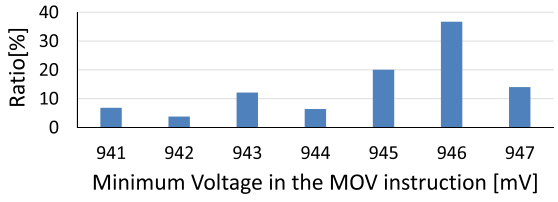

Fig. 14 The histogram of the supply voltage when MOV instructions were executed (crc-original).

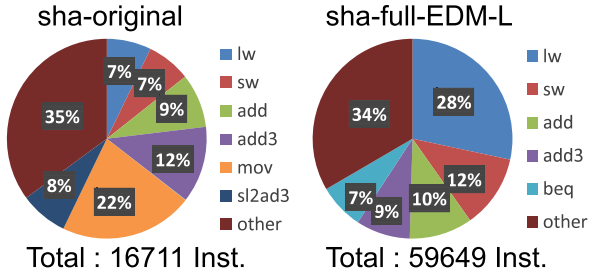

Fig. 15 Proportion of executed instructions in sha-original, sha-full-EDM-L.

executed in sha-original and sha-full-EDM-L programs, respectively. We can see that instruction ratios of the EDM and original programs are considerably different. For example, in EDM, the number of load word (lw) instruction executions increases because the used memory space is doubled, and a number of branch if equal (beq) instructions are introduced due to check insertion. These instruction variations not only affect the processor behavior but also enlarge the noise difference, which makes the error reproduction difficult.

3.3.2 COND2

Next, COND2 is examined. Figures 16 and 17 show the proportions of silent errors, masked errors and detected errors. For detected errors, the histogram of the error detection latency is presented. In EDM-L, we can see that 77% of errors are masked and 2% are silent errors, whereas 87% are masked errors and 7% are silent errors in EDM-O. In other words, most of the electrical timing errors did not propagate to the memory and general purpose registers.

Among the non-masked errors, 86% errors were detected within 1000 cycles in EDM-L, while 38% in EDM-O. This result indicates that the EDM-L performance of detecting electrical timing errors that affect execution results is high. We can say EDM-L is helpful to detect noise induced errors with shorter error detection latency. In other words, we can use EDM-L in Scenario2. For the errors having long error detection latency, we found a tendency that the first error was not detected and the second or later error was detected by the check operation. On the other hand, the EDM-O performance was not good. The proportion of silent errors was larger, and the detection latency was longer. Clearly, for the purpose of quick error detection in post-silicon validation, EDM-L is much better than EDM-O.

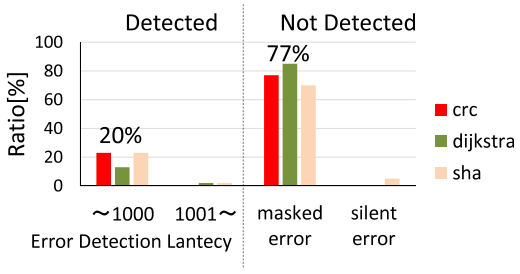

Fig. 16 COND2: Error classification in full-EDM-L. For each program, the number of samples is 100.

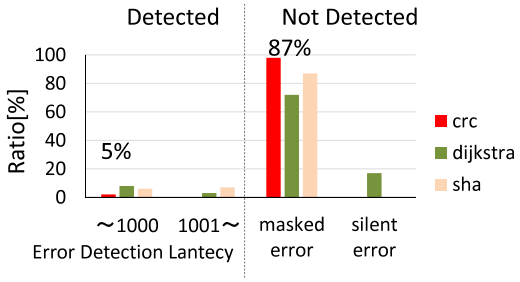

Fig. 17 COND2: Error classification in full-EDM-O. For each program, the number of samples is 100.

4. Hardware Measurement

This section experimentally investigates whether EDM transformation works well in Scenario1 and Scenario2 with 65-nm test chips, and compares theses results with previous simulation.

4.1 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. A chip photo is shown in Fig. 18. The chip size is 4.2 mm × 2.1 mm.

We took three C-language benchmark programs, dijkstra, crc, and sha from MiBench [12] as similar to Sect. 3. Figure 19 shows the measurement setup consisting of a test chip, a device under test (DUT) board, a DC voltage source and a PC. The packaged test chip is mounted on a DUT board. The DUT board, which also includes a Stratix III FPGA and SDRAM, is used as a logic analyzer and a pattern generator.

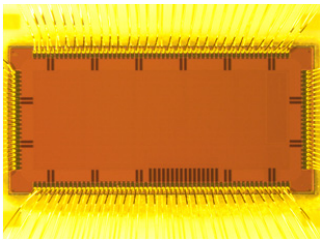

Fig. 18 A photo of 65-nm test chip of MeP processor. Die size is 4.2 mm × 2.1 mm.

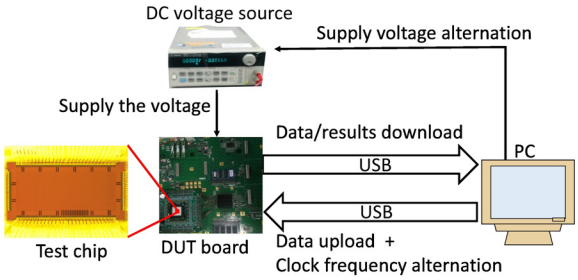

Fig. 19 Measurement setup.

For example, the data that should be stored in the instruction memory and the data memory of MeP processor is first transferred from PC to the DUT board through USB cable, and then the data is loaded to the on-chip SRAMs. Also, after the program execution of MeP processor, the data in the on-chip data memory is downloaded to 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 uploading, the program execution and the data downloading are executed as explained before. 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 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. Figures 20 and 21 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. 20, the FMAX at 1.0 V is 225 MHz, which is 15 MHz lower than that in Fig. 21.

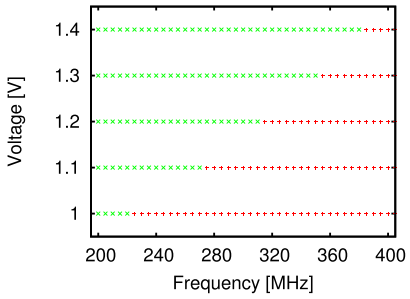

Fig. 20 Shmoo plot of the slowest chip (chip #1).

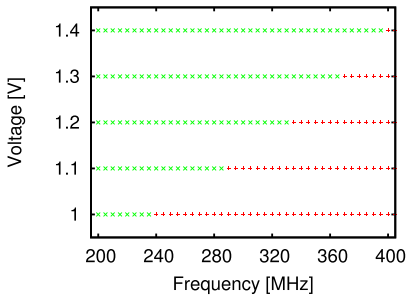

Fig. 21 Shmoo plot of the fastest chip (chip #5).

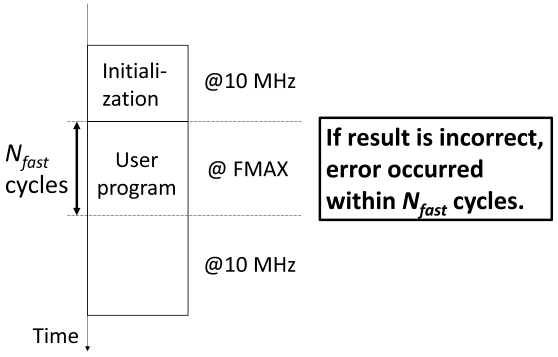

Fig. 22 Procedure of error cycle identification.

4.2 Performance Evaluation Method

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 occurs. However, in the hardware it is difficult to know in which clock cycle the timing error occurs unlike the simulation. Therefore, we take the following evaluation procedure. We change the clock frequency during the program execution as shown in Fig. 22. 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 execu-

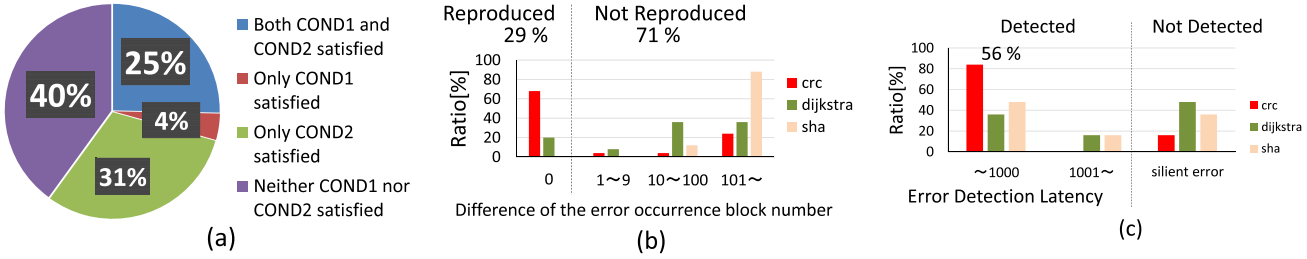

Fig. 23 Results of chip measurement. (a) COND1+COND2(Scenario1), (b) COND1, and (c) COND2(Scenario2).

tion 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.

Remind that, when we decomposed the program into blocks, we numbered the blocks from the beginning. Accordingly, 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 previous simulation. When the difference is zero, the timing error is reproduced at the same block in the EDM-transformed program and COND1 is satisfied. Evaluation for COND2 was also similar to Sect. 3. If checker works, we subtract the error occurrence clock cycle from terminated clock cycle of the program and obtain the error detection latency.

4.3 Evaluation Results

Figure 23(a) shows the ratio of the samples that satisfied COND1 of reproducibility and COND2 of detectability. 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, which is shown in the left figure of Fig. 10, 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.

4.3.1 COND1

Figure 23(b) shows the proximity of the errors occurred in the original and the full-EDM 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. 11. 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 previous simulations, ten different

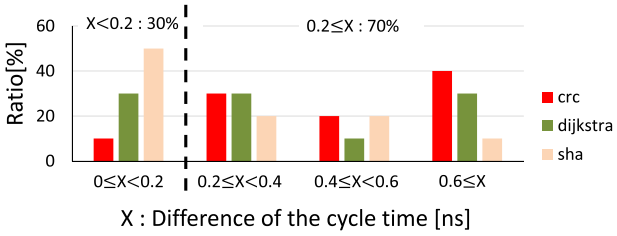

Fig. 24 Histogram of cycle time difference between FMAXs of timing error and incorrect execution.

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 simulation and hardware measurement, whereas MeP processor was operated at the FMAXs 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 cannot be obtained 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 simulation setup. Figure 24 exemplifies the cycle time difference between the FMAXs of timing error and incorrect execution. This result was obtained by the simulation with full-EDM-L programs. We can see that 70% of the samples have $\geq$ 0.2 ns difference.

4.3.2 COND2

Next, COND2 is examined. We first categorized the measured samples into detected samples and not detected samples. In the chip measurement, we focused on the errors affecting the execution result, and hence not detected samples correspond to silent errors, whereas not detected samples include silent errors and masked errors in the simulation. Fig-

ure 23(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. 23(c), we can see that 56% of the errors are quickly detected and 33% are silent errors. Compared with Figs. 16 and 23(c), the EDM-L performance of detecting electrical timing errors affecting execution result in simulation evaluation is higher than in the chip measurement (86% versus 56%).

5. Correlation between the Measurement and Simulation

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. 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.

5.1 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 in a similar way to Sect. 3 except that the supply voltage is fixed and the supply noise is zero. In this simulation, we prepare 3 programs (dijkstra, crc, and sha) and 10 chips, and hence totally 30 samples are evaluated.

Figure 25(a) shows the evaluation results of Scenario1. We can see that EDM could not satisfy COND1 and COND2 simultaneously, which is not consistent with chip measurement results. On the other hand, the proportion of COND1 satisfaction increased from 4% to 13%, approaching to the measurement result of 29% (Fig. 23(a)).

Next, Fig. 25(b) shows the results of Scenario2, and we can see that 76% of the errors are masked error and there are no silent errors. Focusing on the non-masked error, 70% were quickly detected. The ratio of quick detection degraded from 86% but it approaches to 56% of the chip measurement result (Fig. 23(c)).

5.2 FMAX

In the chip measurement, the FMAX of incorrect execution was used while the FMAX of timing error was used in the simulation in Sect. 3. 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

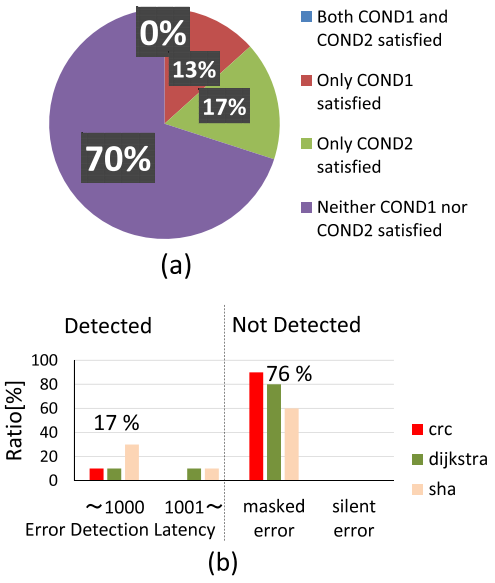

Fig. 25 Simulation results with ideal PDN. FMAX of timing error is used for evaluation. (a) Scenario1, and (b) Scenario2.

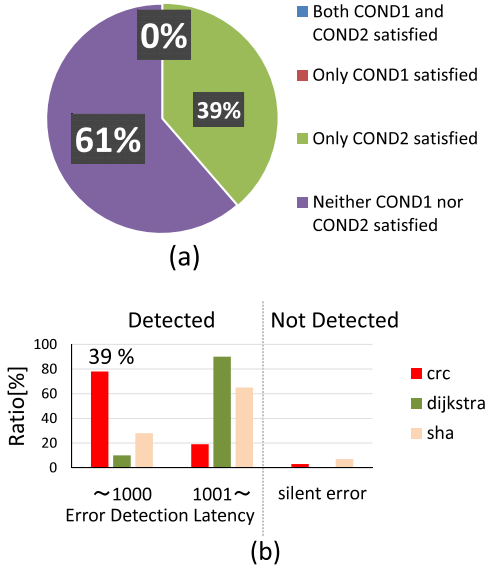

Fig. 26 Simulation results with not ideal PDN. FMAX of incorrect execution is used for evaluation. (a) Scenario1, and (b) Scenario2.

interval, which is also similar to the measurement setup. We used 3 programs and 10 chips in a similar way to the previous evaluation. In addition, we prepared ten PDNs used in Sect. 3. Consequently, we evaluated whether EDM satisfied COND1/COND2 for 300 samples.

Figure 26(a) shows the result for Scenario1. Comparing Figs. 26(a) and 23(a), we can find a large difference in the proportion that both the reproducibility and detectability are satisfied, which is 0% in the simulation and 25% in the chip measurement. Figure 26(b) shows the result for Scenario2. The detectability for the non-masked error has become close to between the simulation and chip measurement, where it is 39% in the simulation and 56% in the chip measurement

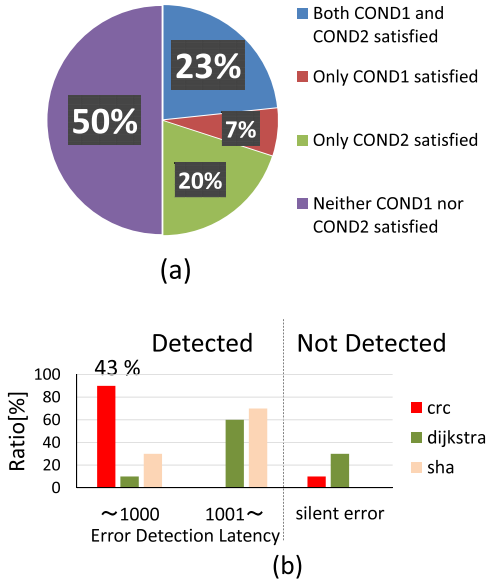

Fig. 27 Simulation results with ideal PDN. FMAX of incorrect execution is used for evaluation. (a) Scenario1, and (b) Scenario2.

(Fig. 23(c)).

5.3 Power Distribution Network and FMAX

Finally, we applied the ideal PDN with no noise and the FMAX of incorrect execution to the simulation.

Figure 27(a) shows the result for Scenario1. Figures 27(a) and 23(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.

Figure 27(b) shows the Scenario2 results. 43% of the errors are quickly detected and 13% are silent errors. Comparing Fig. 27(b) with Fig. 23(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.

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.

6. 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 as many potential errors as possible (Scenario2). We experimentally evaluated the error detection performance in these two scenarios with a noise-aware logic simulator and 65-nm test chips. Simulation results showed that the EDM cannot locate electrical timing errors in the original program in the first scenario, but it detects 86% of non-masked errors potential bugs in the second scenario, which mean the EDM performance of detecting electrical timing errors affecting execution results is high. Hardware measurement results showed that the EDM detects 25% of original timing errors and 56% of non-masked errors. On the other hand, these measurement results were not consistent with the simulation results. 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 and ICOM Foundation, Japan.

References

[1] P. Patra, “On the cusp of a validation wall,” IEEE Design & Test of Computers, vol.24, no.2, pp.193–196, March–April 2007.
[2] D. Josephson, “The good, the bad, and the ugly of silicon debug,” Proc. 43rd ACM/IEEE Design Automation Conf., pp.3–6, San Francisco, CA, 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. Computer-Aided Design Integr. Circuits Syst., vol.33, no.10, pp.1573–1590, Oct. 2014.
[4] S.-B. Park, T. Hong, and S. Mitra, “Post-silicon bug localization in processors using instruction footprint recording and analysis (IFRA),” IEEE Trans. Computer-Aided Design Integr. Circuits Syst., vol.28, no.10, pp.1545–1558, Oct. 2009.
[5] A.A. Bayazit and S. Malik, “Complementary use of runtime validation and model checking,” Proc. IEEE/ACM International Conf. on Computer-Aided Design, pp.1052–1059, 2005.
[6] M. Boule and Z. Zilic, “Incorporating efficient assertion checkers into hardware emulation,” Proc. IEEE International Conf. on Computer Design, pp.221–228, 2005.
[7] S. Mitra, S.A. Seshia, and N. Nicolici, “Post-silicon validation opportunities, challenges and recent advances,” Proc. 47th ACM/IEEE Design Automation Conf., pp.12–17, Anaheim, CA, 2010.
[8] M. Rebaudengo, M.S. Reorda, M. Torchiano, and M. Violante, “Soft-error detection through software fault-tolerance techniques,” Proc. IEEE International Symposium. on Defect and Fault Tolerance in VLSI Systems, pp.210–218, Albuquerque, NM, 1999.
[9] N. Nicolescu and R. Velazco, “Detecting soft errors by a purely software approach: method, tools and experimental results,” Proc. IEEE Design, Automation and Test in Europe Conf. and Exhibition, 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. IEEE/ACM International Conf. on Computer-Aided Design, pp.315–322, Austin, TX, 2015.
[11] Y. Masuda, M. Hashimoto, and T. Onoye, “Hardware-simulation

correlation of timing error detection performance of software-based error detection mechanisms," Proc. IEEE 22nd International On-Line Testing Symposium, pp.84–89, Catalunya, 2016.
[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. IEEE International Symposium on Workload Characterization, pp.3–14, 2001.
[13] M. Ueno, M. Hashimoto, and T. Onoye, "Real-time on-chip supply voltage sensor and its application to trace-based timing error localization," Proc. IEEE 21st International On-Line Testing Symposium, Halkidiki, pp.188–193, 2015.
[14] 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. Adv. Packag., vol.22, no.3, pp.284–291, Aug. 1999.
[15] T. Roy, L. Smith, and J. Prymak, "ESR and ESL of ceramic capacitor applied to decoupling applications," Proc. IEEE 7th Topical Meeting on Electrical Performance of Electronic Packaging, pp.213–216, West Point, NY, 1998.

Yutaka Masuda received the B.E., M.E degree in Information Systems Engineering from the Osaka University, Osaka, Japan, in 2014 and 2016, respectively. He is now a Ph.D. course student of Information Science and Technology, Osaka University. His research interests include error localization in post-silicon validation. He is a student member of IEEE, IPSJ.

Takao Onoye received the B.E. and M.E. degrees in Electronic Engineering, and Dr.Eng. degree in Information Systems Engineering all from Osaka University, Japan, in 1991, 1993, and 1997, respectively. He is currently a professor in the Department of Information Systems Engineering, Osaka University. His research interests include media-centric low-power architecture and its SoC implementation. He is a member of IEEE, IEICE, IPSJ, and ITE-J.

Masanori Hashimoto received the B.E., M.E. and Ph.D. degrees in Communications and Computer Engineering from Kyoto University, Kyoto, Japan, in 1997, 1999, and 2001, respectively. Since 2016, he has been a Professor in Department of Information Systems Engineering, Graduate School of Information Science and Technology, Osaka University. His research interest includes computer-aided design for digital integrated circuits, and high speed circuit design. Dr. Hashimoto served on the technical program committees for international conferences including DAC, ICCAD, ITC, Symposium on VLSI Circuits, ASP-DAC, DATE, ISPD and ICCD. He is a member of IEEE, ACM, IEICE and IPSJ.