A Failure Prediction Strategy for Transistor Aging

Hyunbean Yi, *Member, IEEE*, Tomokazu Yoneda, *Senior Member, IEEE*, Michiko Inoue, *Member, IEEE*, Yasuo Sato, *Member, IEEE*, Seiji Kajihara, *Member, IEEE*, and Hideo Fujiwara, *Fellow, IEEE*

***Abstract*—This paper presents a novel failure prediction technique that is applicable for system-on-chips (SoCs). Highly reliable systems such as automobiles, aircrafts, or medical equipments would not allow any interruptive erroneous responses during system operations, which might result in catastrophes. Therefore, we propose a failure prediction technique that can be applied during an idle time when a system is not working, such as power-on/-off time. To achieve high reliability in the field, the proposed technique should take into consideration various types of aging mechanisms and the testing environment of voltage and temperature which is uncontrollable in the field. Therefore, we propose: 1) an accurate delay measurement technique considering the variation due to voltage and temperature and 2) an adaptive test scheduling that gives more test chances to more probable degrading parts. Experimental results show the required memory space and area cost for implementing the proposed technique.**

***Index Terms*—Design for testability, reliability, testing VLSI.**

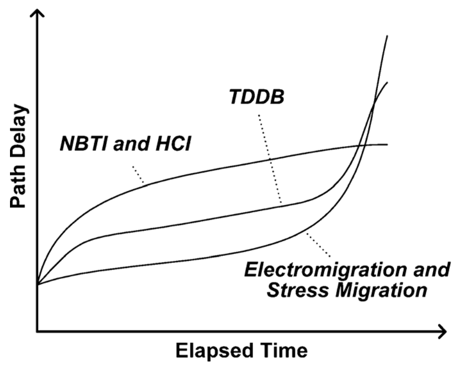


Fig. 1. Different types of delay increase.

I. Introduction

TRANSISTOR aging have been known as troublesome phenomena in the deep sub-micrometer process. It is well known that an aging results in performance degradation and a failure [1]–[4], especially in applications requiring high field reliability, such as automobiles, aircrafts, medical equipments, or power plants, in which performance degradation and a failure are life-threatening. Accordingly, a failure prediction technique with a thorough analysis of aging mechanisms is strongly required.

Negative bias temperature instability (NBTI), hot carrier injection (HCI), electro migration (EM), stress migration (SM), and time dependent dielectric breakdown (TDDB) are the main aging mechanisms. NBTI, which is the dominant one in the latest process technology, causes threshold voltage degradation in a pMOS transistor stressed with negative-biased gate voltages over a couple of decades [5]. HCI, which increases the threshold voltage of an nMOS transistor under a source-drain voltage stress, causes gradual delay degradation like NBTI. EM and SM, which respectively occur due to an excessive current density stress or an excessive structural stress, lead to open or short faults. These phenomena cause a sudden delay increase or a failure. TDDB, in which continuous stresses to a gate oxide-film causes the insulating film breakdown, results in slow delay degradation up to a certain point and causes a sudden delay increase or a failure as shown in Fig. 1 [6]. However, delay variations are caused not only by aging mechanisms but also by variations of environmental conditions such as voltage and temperature. Accordingly, profiling voltage and temperature is needed for accurate delay measurement [6].

In this paper, we focus on an aging test for system-on-chips (SoCs) in systems that require high-reliability. Many systems require hard real-time responses to meet the real-time requirements. Then, during normal operation, an interruptive operation may not be allowed. Therefore, we utilize power-on/-off time to apply our aging test scheme. Like the existing failure prediction techniques in the next section, the proposed aging test scheme performs a test scheduling, controls the test clock, and reports aging test results to the system. However, our proposed test strategy can be differentiated from the existing techniques in the following respects:

1) various types of delay degradation are detected or predicted;
2) an accurate delay measurement is conducted by referring to the measured voltage and temperature;
3) based on the amount of aging delay shift of each part-under-test (PUT; a core or a group of gates or paths), the proposed method can dynamically adapt a test schedule so

Manuscript received October 28, 2010; revised March 16, 2011 and June 15, 2011; accepted August 12, 2011. Date of publication October 06, 2011; date of current version July 27, 2012. This paper is a version of an earlier paper titled "Aging Test Strategy and Adaptive Test Scheduling for SoC Failure Prediction," presented at IEEE 16th International On-Line Testing Symposium (IOLTS), 2010.

H. Yi is with the Department of Computer Engineering and Graduate School of Information and Communications, Hanbat National University, Daejeon 305-719, South Korea, and also with Japan Science and Technology Agency, CREST, Tokyo 102-0075, Japan (e-mail: bean@hanbat.ac.kr).

T. Yoneda and M. Inoue are with the Graduate School of Information Science, Nara Institute of Science and Technology (NAIST), Nara 630-0192, Japan, and also with Japan Science and Technology Agency, CREST, Tokyo 102-0075, Japan (e-mail: yoneda@is.naist.jp; kounoe@is.naist.jp).

Y. Sato and S. Kajihara are with the School of Computer Science and Systems Engineering, Kyusyu Institute of Technology (KIT), Iizuka 820-8502, Japan, and also with Japan Science and Technology Agency, CREST, Tokyo 102-0075, Japan (e-mail: sato@aries30.cse.kyutech.ac.jp; kajihara@cse.kyutech.ac.jp).

H. Fujiwara is with Faculty of Informatics, Osaka Gakuin University, Osaka 564-8511, Japan, and also with Japan Science and Technology Agency, CREST, Tokyo 102-0075, Japan. (e-mail: fujiwara@ogu.ac.jp).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2011.2165304

that a PUT with larger amount of delay shift can be tested more often to reduce the possibility to miss a failure.

The proposed strategy can be applied for cores in idle during normal operation of the SoC. However, for cores which are too busy to go in a test mode or for systems which need to continuously work for a long time once they are turned on, a backup circuit or system is required.

The rest of this paper is organized as follows. Section II reviews related works. We present the aging test architecture with the test flow for aging prediction in Section III and our adaptive aging test scheduling scheme in Section IV. Experiment results are given in Sections V and VI concludes this paper.

II. Related Work

Most of the transistor aging mechanisms can be observed by measuring delay shift. F. Ahmed and L. Milor [7] proposed a scheme to measure delay of selected paths where paths in a chip are converted into ring oscillators in test mode with additional hardware. M. Agarwal *et al.* [8], [9] and T. Nakura *et al.* [10] designed aging sensors checking whether the signal transition of the combinational logic output occurs out of the guard-band time interval. These techniques enable aging to be observed concurrently during normal operation by directly checking aging of actual data paths during normal operation. However, the sizes of the sensor [8], [9] and the flip-flop [10] are large. As an online self-test architecture, Y. Li *et al.* [11] introduced the concurrent autonomous chip self-test using stored test patterns (CASP) which performs online testing using the test patterns pre-stored in a non-volatile memory in the system. Y. Li *et al.* [12] also used CASP to test uncore components such as cache controllers, DRAM controllers, and I/O controllers. When an uncore component goes in test mode, they try to minimize performance impact by enabling other components with the similar functionality or a backup function block to accomplish the function instead of the uncore component. They used an ordinal scheduling of CASP, where each core or uncore component is selected and tested in a round-robin manner. However, a simple hardware-level static round-robin scheduling method cannot give cores many chances of test because the hardware scheduler simply waits until the next core becomes idle. Therefore, in [13], they applied the higher-level online scheduling support techniques such as virtualization-assisted concurrent autonomous self-test (VAST) [14] and CASP-aware OS scheduling [15]. They took the unavailability of cores into consideration, thereby attempting to test each core as often as they can as well as minimizing the impact on application performance. O. Khan *et al.* [16] proposed a method that is more tightly connected to the OS. A functional test that measures the maximum frequency $F_{\max}$ and minimum voltage $\mathrm{VDD}_{\min}$ is applied at checkpoints that OS controls. As the device ages, system performance is tuned with a dynamic frequency or voltage control. However, they do not measure the amount of delay degradation.

We presented a circuit failure prediction mechanism named DART which stands for degrade factor, accuracy, report, and test coverage [6], [17]. In this paper, we propose detailed techniques to realize DART such as a delay measurement methodology and a test scheduling and show some experimental data regarding area cost.

III. Aging Test Architecture

A. Architecture Overview

Fig. 2 shows the proposed aging test architecture. In this SoC model, there are multiple cores including one or more processor cores and memories. In our test scheme that utilize system's vacant time such as power-on/-off time, test patterns have to be preloaded on a memory [e.g., read only memory (ROM) or non-volatile memory (NVM)]. However, in a memory, available data volume to hold the entire test patterns is limited. In order to reduce the data volume of test patterns, an aging path selection [18], [19] or a test compression technique is needed [20].

Available vacant time varies from milliseconds to seconds according to the systems. Therefore, we divide the entire test patterns so that one or more small test pattern sets (TPSs) can be applied and their results can be observed within a limited test time. When the system enters a test mode, the SoC test controller selects the next TPSs and transfers them to cores-under-test (CUTs). Then, each core test controller enables its core test components, such as the decompressor, the compactor, and the test clock generator. The test clock generator has a delay logic so that capture timing can be adjusted which is shown in the next subsection. During normal operation or production test, the core functional clock or SoC test clock will be bypassed to the core logic. When a test with a TPS is completed, the test and measurement results are transferred to the SoC test controller. The results will include the voltage and the temperature as well as the measured delay. The SoC test controller collects the information and analyzes them to figure out whether or not the PUT is aged. If the internal memory is too small to record the test results, an external memory may be used. Basically, the test infrastructure such as test access mechanisms (TAMs) and test wrappers are reused for aging test. However, since the normal operation is not conducted during a power-on/-off time, the functional interconnects can be used with modification of the protocol interface logic. For the case where the functional state of the CUT needs to be recovered after testing, an additional memory space or a shadow scan chain to hold the core state is required.

B. Test Strategy for Aging Prediction

In order to measure the minimum test timing at which the test for a PUT is passed, we use a launch timing shift technique (e.g., on-die clock shrink (ODCS) [21]), as shown in Fig. 3. In a launch timing window, there are several steps of the launch-to-capture period (LCP) from the shortest one, $\mathrm{LCP}_{\min}$, to the longest one. We define an LCP which is selected for a delay test as $\mathrm{LCP}_{\mathrm{test}}$. During the test mode, the minimum $\mathrm{LCP}_{\mathrm{test}}$, at which a delay test for the PUT is passed, is found and reported. As aging goes on, the minimum $\mathrm{LCP}_{\mathrm{test}}$ increases gradually. The aging delay increase of each PUT is analyzed using the recorded test results and the minimum $\mathrm{LCP}_{\mathrm{test}}$ by SoC test controller in Fig. 2. $\mathrm{LCP}_{\min}$, the interval between two adjacent LCPs, and the number of LCP levels (from $\mathrm{LCP}_{\min}$ to

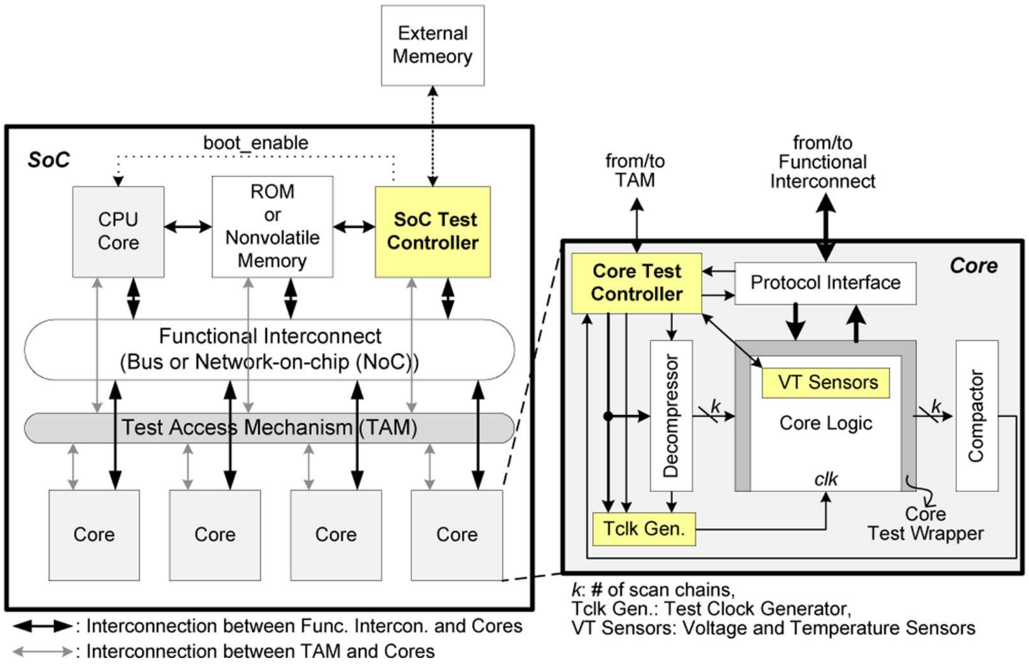

External Memeory
SoC
boot_enable
CPU Core
ROM or Nonvolatile Memory
SoC Test Controller
Functional Interconnect (Bus or Network-on-chip (NoC))
Test Access Mechanism (TAM)
Core
Core
Core
Core
: Interconnection between Func. Intercon. and Cores
: Interconnection between TAM and Cores
from/to TAM
from/to Functional Interconnect
Core
Core Test Controller
Protocol Interface
Decompressor
VT Sensors
Core Logic
clk
Compactor
k
k
Tclk Gen.
Core Test Wrapper
k: # of scan chains,
Tclk Gen.: Test Clock Generator,
VT Sensors: Voltage and Temperature Sensors

Fig. 2. SoC aging test architecture.

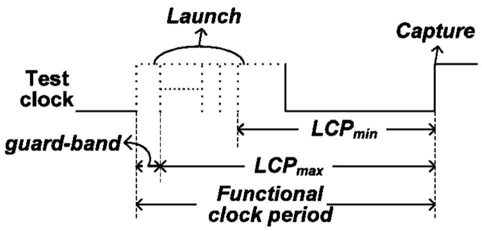


Fig. 3. Launch timing window.

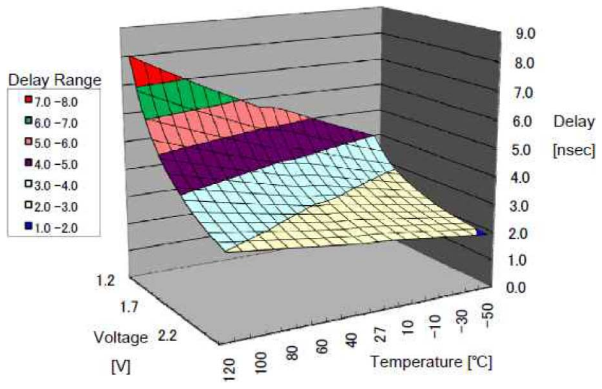


Fig. 4. Delay variation according to voltage and temperature.

$\mathrm{LCP}_{\max}$) are decided by the possible clock resolution in the SoC.

Delay variation is caused not only by aging mechanisms but also by environmental conditions such as voltage and temperature variations. Fig. 4 shows an example of simulation for a delay variation of 27-state ring oscillator according to voltage and temperature variations using TSMC 0.18-μm parameters. Variety of voltage/temperature sensors have been proposed [22]–[24]. Selecting proper sensors, their sizes, accuracy, and design cost will be important issues. We assume that ring oscillator-based sensors and the thermal-aware test patterns [25], which minimize the spatial temperature variation when test patterns shift in scan-chains, are used. In order to remove a hazard or glitches which may be created during delay measurement, hazard-free robust test patterns [26] can be used.

Fig. 5 is the proposed test flow to detect and predict the aging of a PUT. In order to estimate the degree of aging due to delay degradation, we measure minimum delays of PUTs over time. However, since delay also varies with voltage and temperature variations at each test mode, translation equations or lookup tables which map a measured delay to a delay ($\mathrm{LCP}_{\mathrm{typ}}$) at the typical voltage (V_{typ}) and temperature (T_{typ}) are used. The translated delay is compared to the past delay values, which are stored in a log memory, and the aging delay increase is analyzed. Logged test results are also used at the beginning of the test flow to decide the initial condition of the current test. Thus, we can avoid binary-search-like time-consuming tests [21] to find the minimum launch-to-capture period. If equations or lookup tables are prepared for paths, we can improve translation accuracy, but too many parameters or lookup tables are required. Therefore, by adaptively preparing a translation equation or a look-up table for each core using a careful test element group (TEG) analysis and applying an adaptive test, we are able to improve the translation accuracy with a small memory overhead.

The test flow is able to be divided into two parts: 1) detecting a sudden delay increase from Process 1 to 2 and 2) measuring delay, voltage, and temperature to analyze a gradual delay increase from Process 3 to 10. In this paper, a test session is defined as a round of the process steps from "Start" to "End" of the test flow. In the first process, Process 1, a selected TPS is applied to detect a sudden delay increase using the $\mathrm{LCP}_{\max}$. When the test is passed, the next processes try to find out the minimum $\mathrm{LCP}_{\mathrm{test}}$. In Process 3, the initial $\mathrm{LCP}_{\mathrm{test}}$ is decided

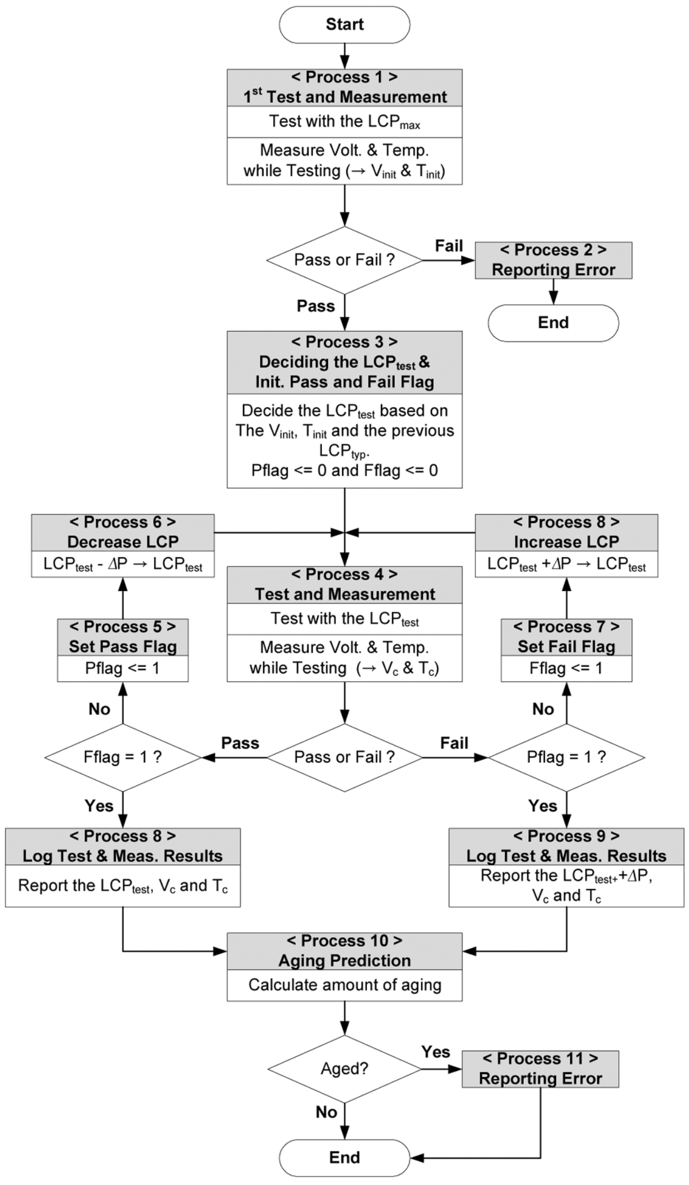

Start
< Process 1 >
1st Test and Measurement
Test with the LCPmax
Measure Volt. & Temp. while Testing (→ Vinit & Tinit)
Pass or Fail ?
Fail
< Process 2 > Reporting Error
End
Pass
< Process 3 > Deciding the LCPtest & Init. Pass and Fail Flag
Decide the LCPtest based on The Vinit, Tinit and the previous LCPtyp. Pflag <= 0 and Fflag <= 0
< Process 6 > Decrease LCP
LCPtest - ΔP → LCPtest
< Process 8 > Increase LCP
LCPtest +ΔP → LCPtest
< Process 4 > Test and Measurement
Test with the LCPtest
Measure Volt. & Temp. while Testing (→ Vc & Tc)
< Process 5 > Set Pass Flag
Pflag <= 1
< Process 7 > Set Fail Flag
Fflag <= 1
No
Fflag = 1 ?
Pass
Pass or Fail ?
Fail
Pflag = 1 ?
No
Yes
Yes
< Process 8 > Log Test & Meas. Results
Report the LCPtest, Vc and Tc
< Process 9 > Log Test & Meas. Results
Report the LCPtest+ΔP, Vc and Tc
< Process 10 > Aging Prediction
Calculate amount of aging
Aged?
Yes
< Process 11 > Reporting Error
No
End

Fig. 5. Test flow for aging detection and prediction.

based on the voltage $(\mathrm{V}_{\mathrm{init}})$ and temperature $(\mathrm{T}_{\mathrm{init}})$ measured in Process 1 and the previously measured delay. To obtain a proper $\mathrm{LCP}_{\mathrm{test}}$, some equations or mapping tables which compute the correlation among voltage, temperature, and delay are used. Two flags, Pass flag (Pflag) and Fail flag (Fflag), are also initialized in Process 3. The two loops, Process 4, 5, and 6 and Process 4, 7, and 8, try to find out the minimum $\mathrm{LCP}_{\mathrm{test}}$, decreasing or increasing $\mathrm{LCP}_{\mathrm{test}}$ according to the test result in Process 4. If the previous test was failed $(\mathrm{Fflag} = \text{“1”})$ and the current test is passed, then the current $\mathrm{LCP}_{\mathrm{test}}$ becomes the minimum $\mathrm{LCP}_{\mathrm{test}}$. On the other hand, if the previous test was passed $(\mathrm{Pflag} = \text{“1”})$ and the current test is failed, then the previous $\mathrm{LCP}_{\mathrm{test}}$ is the minimum $\mathrm{LCP}_{\mathrm{test}}$. Therefore, in Process 8, the current $\mathrm{LCP}_{\mathrm{test}}$ itself is logged and in Process 9, the previous $\mathrm{LCP}_{\mathrm{test}}$ (the current LCP $_{\mathrm{test}} + \Delta\mathrm{P}$) is logged.

Since gradual delay degradation goes on over time, we expect a TPS is applied three times in one test session. The minimum $\mathrm{LCP}_{\mathrm{test}}$ and the measured voltage (V_c) and temperature (T_c)

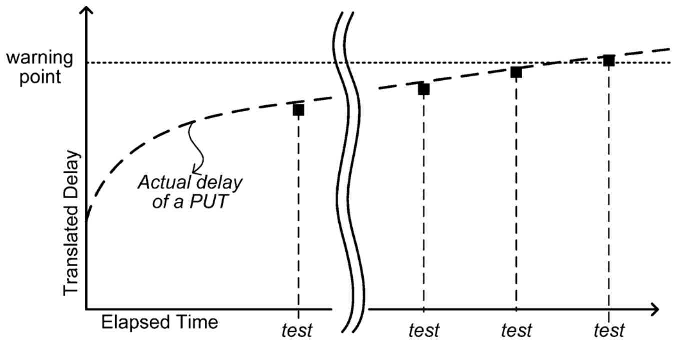


Fig. 6. Detection of a gradual delay increase.

are reported. Then, the SoC test controller translates the reported $\mathrm{LCP}_{\mathrm{test}}$ into a delay at the typical condition and logs it in a memory with the measured voltage and temperature. Finally, based on the logged data, the amount of aging delay increase is analyzed.

Fig. 6 shows detection of a gradual delay increase of a PUT. The Y-axis is the translated delay that is delay under typical temperature and voltage. During life cycle of the product, we apply test to the same PUT periodically. At every test, we find the minimum $\mathrm{LCP}_{\mathrm{test}}$ that is the minimum LCP where the test is passed, and then it is translated into an $\mathrm{LCP}_{\mathrm{typ}}$ that is a delay under a typical temperature and voltage. The figure plots the minimum $\mathrm{LCP}_{\mathrm{typ}}$ for several tests. After capturing the minimum $\mathrm{LCP}_{\mathrm{typ}}$, we analyze an amount of aging, and if the $\mathrm{LCP}_{\mathrm{typ}}$ is greater than or equal to some warning point, an error is reported.

IV. Adaptive Aging Test Scheduling Scheme

We use a degree of aging-based weighted test scheduling where more probably aged PUTs are tested more often. In terms of workload, once a part is aged to some degree, the part is more likely to be aged than the other parts because a user of a system tends to repeat a similar use (e.g., automobiles). In this section, we introduce a degree of aging comparison method and describe our adaptive test scheduling scheme in detail.

A. Comparison of Degree of Aging

If α is the probability in one clock cycle that a pMOS connected to the corresponding gate input has $\mathrm{Vgs} = -\mathrm{Vdd}, t$ is the total circuit operation time, and $n(= 0.16)$ is a characteristic of the NBTI effect, then the increase in the gate delay due to NBTI aging Δd_g is

$$\Delta d_g = A \cdot \alpha^n \cdot t^n \cdot d_0 \tag{1}$$

where A is a constant parameter and d_0 is the time 0 delay of the gate [27]. A, α, and d_0 are given from the design process technology used. Therefore, (1) can be simplified to $\Delta d_g = B \cdot t^n$ for some constant value $B = A \cdot \alpha^n \cdot d_0$. Consequently, since a logical path is a serial connection of gates, the increase in the path delay due to NBTI aging Δd_p can be obtained as follows:

$$\Delta d_p = S \cdot t^n \tag{2}$$

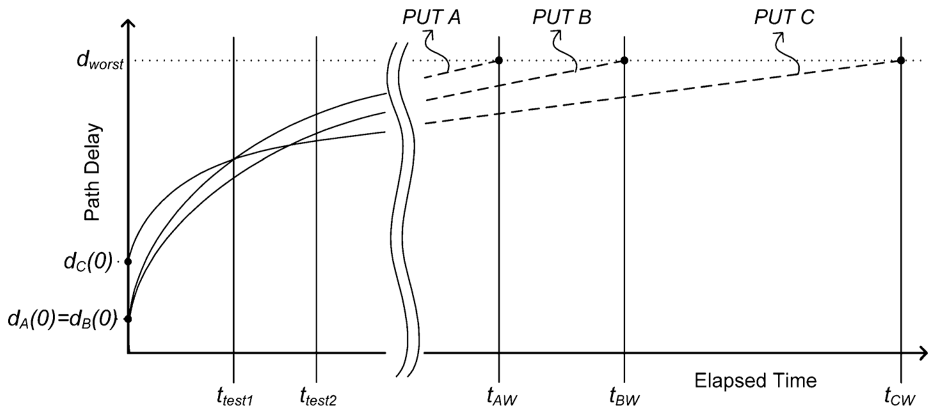


Fig. 7. Examples of delay increases of PUTs.

where S is the sum of the constant values of the gates which are on the path. Continuously, if we let $d_A(t)$ be the measured delay of a PUT, PUT A, at t, then we have

$$d_A(t) = d_A(0) + S_A \cdot t^n \tag{3}$$

where $d_A(0)$ is the initial delay (time 0 delay) of PUT A.

We use (3) to compare degrees of aging of PUTs. Fig. 7 shows three gradual delay increases. d_{worst} is the delay with which the PUTs are identified to be aged. We consider the following two cases.

Case 1) The initial delays of two PUTs are the same, but their delays increase with different speeds of aging from the beginning and their amount of aging are not switched all the way like PUT A and B.

Case 2) Two PUTs start with different initial delays, but the delay of one with the smaller initial delay increases faster than that of the other and its amount of aging overpasses the other one at a certain point in time like PUT A and C.

1) Case 1—Comparison of PUT A and PUT B: Let us assume that we found $d_A(t_{\text{test2}}) > d_B(t_{\text{test2}})$ where t_{test2} is the point of time when PUT A and PUT B were tested. Then, since, from (3) and Fig. 6

$$\begin{aligned} d_A(t_{\text{test2}}) &= d_A(0) + S_A \cdot t_{\text{test2}}^n \\ d_B(t_{\text{test2}}) &= d_B(0) + S_B \cdot t_{\text{test2}}^n \\ d_A(0) &= d_B(0) \end{aligned}$$

we can have

$$d_A(t_{\text{test2}}) - d_B(t_{\text{test2}}) = (S_A - S_B) \cdot t_{\text{test1}}^n > 0. \tag{4}$$

We can let $d_A(t_{\text{test2}}) = q \cdot d_B(t_{\text{test2}})$ and $S_A = q \cdot S_B$ for a constant $q (q > 1)$. If we assume that PUT A and PUT B will reach d_{worst} at t_{AW} and t_{BW}, respectively, then since

$$\begin{aligned} d_A(t_{\text{AW}}) &= d_A(0) + S_A \cdot t_{\text{AW}}^n \\ d_B(t_{\text{BW}}) &= d_B(0) + S_B \cdot t_{\text{BW}}^n \\ d_A(0) &= d_B(0) \\ d_A(t_{\text{AW}}) &= d_B(t_{\text{BW}}) = d_{\text{worst}} \end{aligned}$$

we can finally obtain the relationship between t_{AW} and t_{BW} as follows:

$$t_{\text{AW}} = \left(\frac{1}{q}\right)^{\frac{1}{n}} \cdot t_{\text{BW}}. \tag{5}$$

From (5), we can say that PUT A will reach the worst case delay $(1/q)^{1/n}$ times faster than PUT B, which means PUT A is much more dangerous than PUT B. For example, if we assume $q = 1.5 (q > 1)$, then $t_{\text{AW}} \fallingdotseq 0.079 t_{\text{BW}}$ for $n = 0.16$.

2) Case 2—Comparison of PUT A and PUT C: Let us assume that we found $d_A(t_{\text{test1}}) = d_C(t_{\text{test1}})$ where t_{test1} is the point of time when PUT A and PUT C were tested. Then, since, from (3) and Fig. 6

$$\begin{aligned} d_A(t_{\text{test1}}) &= d_A(0) + S_A \cdot t_{\text{test1}}^n \\ d_C(t_{\text{test1}}) &= d_C(0) + S_C \cdot t_{\text{test1}}^n \\ d_A(0) &< d_C(0) \end{aligned}$$

we can have

$$d_C(0) - d_A(0) = (S_A - S_C) \cdot t_{\text{test1}}^n > 0. \tag{6}$$

If we assume that PUT A and PUT C will reach the worst case delay d_{worst} at t_{AW} and t_{CW}, respectively, then since

$$\begin{aligned} d_A(t_{\text{AW}}) &= d_A(0) + S_A \cdot t_{\text{AW}}^n \\ d_C(t_{\text{CW}}) &= d_C(0) + S_C \cdot t_{\text{CW}}^n, d_A(0) < d_C(0) \\ d_A(t_{\text{AW}}) &= d_C(t_{\text{CW}}) = d_{\text{worst}} \end{aligned}$$

we can obtain

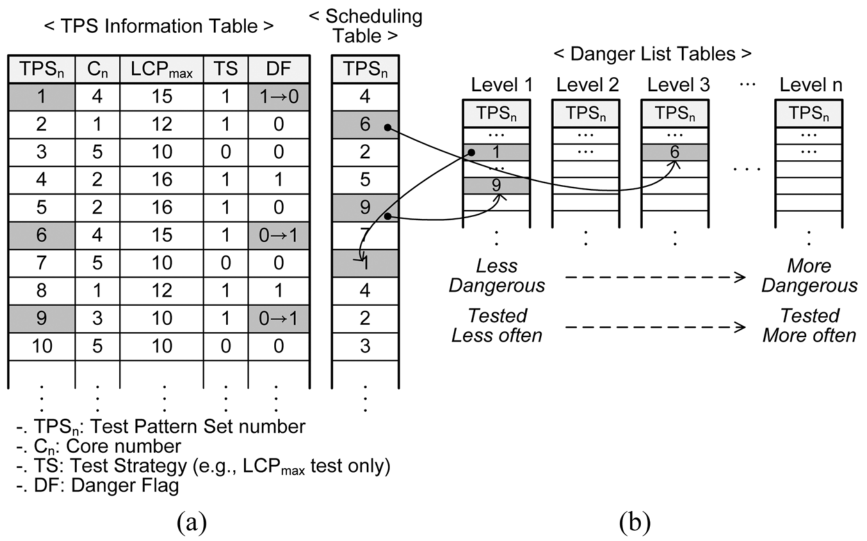
$$d_C(0) - d_A(0) = S_A \cdot t_{\text{AW}}^n - S_C \cdot t_{\text{CW}}^n. \tag{7}$$

From (6) and (7), the increased delay of PUT A is easily driven as follows:

$$S_A \cdot t_{\text{AW}}^n = (S_A - S_C) \cdot t_{\text{test1}}^n + S_C \cdot t_{\text{CW}}^n. \tag{8}$$

For a parameter $p (0 < p < 1)$ and a constant $q (q > 1)$, $t_{\text{test1}} = p \cdot t_{\text{CW}}$ and $S_A = q \cdot S_C$. Then, from (7) and (8) we can finally obtain

$$t_{\text{AW}} = r^{\frac{1}{n}} \cdot t_{\text{CW}} \tag{11}$$


< TPS Information Table >

TPS_n	C_n	LCP_{max}	TS	DF
1	4	15	1	1→0
2	1	12	1	0
3	5	10	0	0
4	2	16	1	1
5	2	16	1	0
6	4	15	1	0→1
7	5	10	0	0
8	1	12	1	1
9	3	10	1	0→1
10	5	10	0	0
⋮	⋮	⋮	⋮	⋮

< Scheduling Table >

TPS_n
4
6
2
5
9
7
1
4
2
3
⋮

-. TPS_n: Test Pattern Set number
-. C_n: Core number
-. TS: Test Strategy (e.g., LCP_{max} test only)
-. DF: Danger Flag

Fig. 8. Tables for adaptive aging test scheduling. (a) TPS information table and scheduling table. (b) Danger list tables.

for $r = ((q-1)\cdot p^n + 1)/q$. The parameter r is always positive and less than one $(0 < r < 1)$ because $q > 1$ and $0 < p < 1$. Therefore, from (11) we can say PUT A will reach the worst case delay $r^{1/n}$ times faster than PUT C, which means also PUT A is much more dangerous than PUT C. For example, let us assume that it is estimated $p = 10^{-3}$ from $t_{\text{test1}} = p \cdot t_{\text{CW}}$. If we let $q = 1.8$, then $t_{\text{AW}} \fallingdotseq 0.1 t_{\text{CW}}$ for $n = 0.16$. This shows that PUT A becomes aged almost 10 times faster than PUT C.

B. *Adaptive Aging Test Scheduling*

We use a set of tables as shown in Fig. 8(a). The scheduling table refers to the TPS information table. A TPS is mapped to a core number (C_n) because it covers a core or a part in a core (i.e., PUT). For each TPS, the $\text{LCP}_{\max}$, the test strategy (TS), and the danger flag (DF) are defined. The value of an $\text{LCP}_{\max}$ is decided by the worst case delay and the resolution of the test clock generator. An LCP_{test} will be mapped to a less value than that of the $\text{LCP}_{\max}$. In this example, if $\text{TS} =$ "0", then the TPS is only tested with the $\text{LCP}_{\max}$. However, if $\text{TS} =$ "1", then the TPS is tested with the test and measurement flow in Fig. 4. In a scheduling table, a TPS is shown at least once and the TPSs are served in a round robin manner from the top entry to the bottom. A TPS for a part which is expected to be vulnerable to aging or has a heavy workload can be listed several times. Since each part has a different speed of aging, the scheduling table needs to be updated at times to reduce the possibility to miss a failure. However, if there are many TPSs, then updating the scheduling table every time delay degradation is detected is time consuming. Therefore we use the danger flag (DF) and danger list tables as shown in Fig. 8(b).

The DF field is used to indicate whether or not the TPS is moved into a danger list table. Each danger list table has its own danger level from the lowest one, Level 1, to the highest one, Level n. The number of danger levels can be decided according to the number of LCP levels. To give the TPSs in the higher level danger tables more chances of testing, the period indicators are used. The shorter the period is, the higher the danger level is. The value of a period indicator simply increases by one every power-on and -off time like a counter. As time goes by, some TPSs will move to the danger list tables while the DF is set to "1". Then, in the scheduling table, only the TPSs of which the DF is "0" have chances of testing. TPSs in a danger list table are served when the period indicator is full. Once the period indicator is full, TPSs in the corresponding danger list table are served in a first-in first-out (FIFO) manner. If a danger list table has too many TPSs to serve in a power-on or -off time, the remaining TPSs are first served in the next power-on or -off time. When a test session starts, the next TPSs are selected based on the following priority order:

1) the remaining TPSs in the danger list table which was served in the previous test session;
2) the TPSs in the danger list table with a highest danger level of the danger list tables whose period indicators are full;
3) in the scheduling table, the next TPSs of which DF is "0".

Some examples of the TPS movements are also shown in Fig. 8. According to the result of an aging analysis, a TPS can move from the scheduling table to the first level danger table like the TPS 9, but can jump to a two or more higher level table like the TPS 6 if it is more aged. The TPS 4 has already been aged and it would be shown somewhere in the danger list tables. However, TPSs are not always moved from the left to the right. We cannot guarantee that the previous analyses are 100% accurate because the test and measurement circuits can have an error. Therefore, by the current analysis, a TPS can move back from a danger list table to the scheduling table like the TPS 1 or a lower level danger list table. When a TPS moves out of or in the scheduling table, the corresponding DF is changed from "0" to "1" or the other way around, respectively. In the example, the TPS 2 and the TPS 5 are estimated that the parts covered by them are still not aged and the TPS 7 and the TPS 3 are not moved because they use the different test strategy with only the $\text{LCP}_{\max}$.

With our test strategy and scheduling technique, it is obvious that the proposed method will be able to make a more accurate

TABLE I
ENVIRONMENT SETTINGS AND ASSUMPTIONS

Power-on time (= Power-off time)	*10 ms*
Scan Shift Clock Frequency	*75 MHz*
Maximum Number of Scan Chains in a Core	*32*
The Size of Selected Aging Test Patterns for an SoC	*1/4 of the total test patterns of an SoC given*
Compression Ratio for Test Patterns	*50x*
Number of LCP_{test} Levels	*16*
Number of Logged Data for a TPS	*10*

failure prediction than a simple worst case delay test. Therefore we estimate memory space and logic area overhead required to implement the proposed method in Section V.

V. EXPERIMENTAL RESULTS

To estimate memory space overhead for various cases, we used the ITC'02 SoC test benchmarks [28]. A TPS is expected to be applied three times in a test session as described in the Section III-B. The sizes of TPSs will vary according to the size of the core that the TPS is applied to, the test strategy that the TPS uses, and the number of core internal scan chains. The basic environment settings and assumptions are shown in Table I. The size of a TPS is limited by the power-on/-off time, the scan shift clock frequency, and the number of scan chains in the core SC_n. However, if the size of a TPS is too big, we cannot test many parts in a given test time. We assume that at least ten test sessions can be performed in a power-on/off time. Then, the maximum size of a TPS of a core becomes $\mathrm{SC}_n \times 25\,000$ bits $(= 750\,000/10/3)$ since the total number of scan shifts is $750\,000$ ($\fallingdotseq$ 10 ms/75 MHz) in a power-on/-off time, ignoring test control time, and a TPS is applied three times in a test session. In order to estimate the amount of log data, we assume that ten logged delay values of a PUT are used to calculate the degree of its aging.

Table II shows the sizes of the entire test pattern sets, tables, and log data for each SoC test benchmark. If the size of the selected test patterns of a core is equal to or less than 25 000 bits, then only one TPS is assigned to the core. For the cores of which the size of the selected test patterns is greater than 25 000 bits, we simply divided the patterns into groups so that the size of each TPS is less than $\mathrm{SC}_n \times 25\,000$ bits and balanced. $\mathrm{TP}_{\mathrm{size}}$ simply represents the total amount of test patterns. If there are many large cores in a SoC, the number of TPSs may not be big even though $\mathrm{TP}_{\mathrm{size}}$ is big. The benchmark only gives the numbers of test patterns and some interface information of cores. If some cores have the same numbers of inputs, outputs, flip-flops, and test patterns, we regard them as the same cores so that they share the same TPSs.

The sizes of the scheduling table and the TPS information table mainly depend on the number of TPSs because the number of rows is equal to or greater than the number of TPSs. The scheduling table has only one field for TPS_n. The TPS Information table has four fields except the field for TPS_n because it is built in numerical order of TPS_n as shown in Fig. 8(a). With regard to the danger list tables, since at least ten test sessions are conducted in a power-on/off time, if the number of

TABLE II
SIZES OF TEST PATTERNS, TABLES, AND LOG DATA

SoC Bench.	# of cores	TP_{size}[a] (bytes)	# of TPSs	$S\&I_{size}$[b] + DLT_{size}[c] (bytes)	Log_{size}[d] (bytes)
u226	*9*	*23.5 K*	*29*	*57.4*	*145*
d281	*8*	*2.3 K*	*12*	*21.5*	*60*
d695	*10*	*0.4 K*	*10*	*17.5*	*50*
h953	*8*	*0.7 K*	*8*	*12*	*40*
g1023	*14*	*0.3 K*	*14*	*26.5*	*70*
f2126	*4*	*3.1 K*	*3*	*3.8*	*15*
q12710	*4*	*11.3 K*	*7*	*9.6*	*35*
p22810	*29*	*4.1 K*	*29*	*63*	*145*
p34392	*19*	*8.5 K*	*21*	*47*	*105*
p93791	*32*	*15.7 K*	*29*	*63*	*145*
t512505	*31*	*99.6 K*	*79*	*202.3*	*395*
a586710	*7*	*2.3 M*	*1184*	*3 K*	*5.8 K*

[a]TP_{size}: Size of Test Patterns which are Selected and Compressed.
[b]$S\&I_{size}$: Sum of Sizes of Scheduling Table and TPS Information Table
[c]DLT_{size}: Size of Danger List Tables
[d]Log_{size}: Size of Log Data

TABLE III
MEMORY AND LOGIC AREA OVERHEAD

TP_{size} (bytes)	$S\&I_{size}$ + DLT_{size} (bytes)	Log_{size} (bytes)	Logic Area Inc. (%)
3.1 K	*257*	*350*	*8.8*

PUTs is less than ten, danger list tables are not needed. Therefore, the SoC benchmarks, d695, h953, f2126, and q12710 do not need danger list tables. For the other benchmarks, the danger list tables and the values of the period indicators are decided so that the TPSs in danger list tables can be applied more often than the TPSs in the scheduling table. Last, if we assume that a translated delay value is logged in a byte, the log size of each TPS is 10 bytes. However, if we log only differences from the typical delay expected, then we can reduce log size by half. As a result, the average sizes of test patterns and tables are about 206.2 kbytes and 299.6 bytes, respectively and the average log amount is about 593.7 bytes.

ITC'02 SoC test benchmarks do not provide logic area information. Therefore, to estimate logic area overhead as well as memory space overhead, we constructed another SoC benchmark with AMBA-based IP cores [29]. There are 32 cores such as four Leon3 Processors, Ethernet MACs, VGAs, GPIOs, PS/2s, Timers, UARTs, three SDRAM Controllers, and one AHB-to-PCI Bridge in the SoC benchmark size of which is about 457 K in 2-input NANDs. We obtained 100 TPSs after dividing all test patterns into TPSs according to the above environment settings as shown in Table I, assuming all the cores are different and any TPSs are not shared among cores. We designed the SoC Test Controller and the Core Test Controllers.

Table III shows the memory and the logic area overhead. As a result, the required memory space is about 3.7 kbytes and the gate counts (# of 2-input NANDs) of the SoC Test Controller, and the Core Test Controller are 3891 and 1141. The area of the cores by Core Test Controller increased by 16.6% on average, ranging from 2.5% for the largest core, Leon3 Processor, to 29.3% for the smallest one, VGA. This is rather high for some cores. However, the area for the entire SoC has increased only by 8.8% and if we design the SoC so that homogeneous cores can share a Core Test Controller, the overall SoC area can be reduced. The sizes of tables and log data and the area overhead due to the control logic and sensors are small compared with the size of test patterns. Thus, in order to reduce design costs, aging path selection, and compression techniques to create more compact test patterns for target aging mechanisms are required.

VI. Conclusion

In this paper, we proposed a built-in test strategy and a test scheduling scheme to realize an accurate SoC aging prediction. Existing failure prediction techniques perform their aging test and scheduling methods, trying to minimize the system performance degradation. We took both the gradual delay increase and the sudden delay increase into consideration and used the launch clock shifting technique to estimate the amount of aging. To make an accurate delay measurement, we assumed that voltage and temperature sensors are used and the delay test timing is adjusted according to voltage and temperature values measured during test. We also presented a degree of aging-based weighted test scheduling scheme. By testing the more aged parts more often, we can reduce the possibility to miss a system failure. Although the proposed test strategy and test scheduling scheme were designed to work in a power-on/-off time, they can be easily applied on line during the test mode assigned by the operating system.

Acknowledgment

The authors would like to thank Prof. Y. Miura of Tokyo Metropolitan University for providing them with the simulation results on delay variation according to voltage and temperature. They would also like to thank Prof. S. Ohtake of Nara Institute of Science and Technology for his invaluable comments.

References

[1] ITRS, "International Technology Roadmap for Semiconductors," 2009.

[2] W. Wang, V. Reddy, A. T. Krishnan, R. Vattikonda, S. Krishnan, and Y. Cao, "Compact modeling and simulation of circuit reliability for 65-nm CMOS technology," *IEEE Trans. Device Mater. Reliab.*, vol. 7, no. 4, pp. 509–517, Dec. 2007.

[3] T. W. Chen, K. Kim, Y. M. Kim, and S. Mitra, "Gate-oxide early failure prediction," in *Proc. IEEE VLSI Test Symp.*, 2008, pp. 111–118.

[4] M. Noda, S. Kajihara, Y. Sato, and Y. Miura, "On estimation of NBTI-induced delay degradation," in *Proc. IEEE Euro. Test Symp.*, 2010, pp. 107–111.

[5] V. Reddy, J. Carulli, A. Krishinan, W. Bosch, and B. Burgess, "Impact of negative bias temperature instability on product parametric drift," in *Proc. Int. Test Conf.*, 2004, pp. 148–155.

[6] Y. Sato, S. Kajihara, Y. Miura, T. Yoneda, S. Ohtake, M. Inoue, and H. Fujiwara, "A circuit failure prediction mechanism (DART) for high field reliability," in *Proc. Int. Conf. ASIC*, 2009, pp. 581–584.

[7] F. Ahmed and L. Milor, "Built-in self test circuit for delay degradation detection," in *Proc. Conf. Design Circuits Integr. Syst.*, 2009, pp. 65–70.

[8] M. Agarwal, B. C. Paul, M. Zhang, and S. Mitra, "Circuit failure prediction and its application to transistor aging," in *Proc. IEEE VLSI Test Symp.*, 2007, pp. 277–284.

[9] M. Agarwal, V. Balakrishnan, A. Bhuyan, K. Kim, B. C. Paul, Y. Cao, and S. Mitra, "Optimized circuit failure prediction for aging: Practicality and promise," in *Proc. Int. Test Conf.*, 2008, no. 26.1, pp. 1–10.

[10] T. Nakura, K. Nose, and M. Mizuno, "Fine grain redundant logic using defect-prediction flip-flops," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 2007, pp. 402–403.

[11] Y. Li, S. Makar, and S. Mitra, "CASP: Concurrent autonomous chip self-test using stored test patterns," in *Proc. Design Autom. Test Euro.*, 2008, pp. 885–890.

[12] Y. Li, O. Mutlu, D. S. Gardner, and S. Mitra, "Concurrent autonomous self-test for uncore components in system-on-chips," in *Proc. IEEE VLSI Test Symp.*, 2010, pp. 232–237.

[13] Y. Li, Y. M. Kim, E. Mintarno, D. S. Gardner, and S. Mitra, "Overcoming early-life failure and aging for robust systems," *IEEE Design Test Comput.*, vol. 26, no. 06, pp. 28–39, Nov./Dec. 2009.

[14] H. Inoue, Y. Li, and S. Mitra, "VAST: Virtualization-assisted concurrent autonomous self-test," in *Proc. Int. Test Conf.*, 2008, pp. 1–10.

[15] Y. Li, O. Mutlu, and S. Mitra, "Operating system scheduling for efficient online self-test in robust systems," in *Proc. Int. Conf. Comput.-Aided Design*, 2009, pp. 201–208.

[16] O. Khan and S. Kundu, "A self-adaptive system architecture to address transistor aging," in *Proc. Design Autom. Test Euro.*, 2009, pp. 81–86.

[17] Y. Sato, S. Kajihara, M. Inoue, T. Yoneda, S. Ohtake, H. Fujiwara, and Y. Miura, "Circuit failure prediction by field test (DART) with delay-shift measurement mechanism," *Integr. Circuits Devices Vietnam*, pp. 5–10, Aug. 2010.

[18] A. B. Baba and S. Mitra, "Testing for transistor aging," in *Proc. IEEE VLSI Test Symp.*, 2009, pp. 215–220.

[19] M. Noda, S. Kajihara, Y. Sato, and Y. Miura, "A path selection method for delay test targeting transistor aging," in *Proc. IEEE Int. Workshop Reliab. Aware Syst. Design Test*, 2010, pp. 57–61.

[20] T. Yoneda, M. Nakao, M. Inoue, Y. Sato, and H. Fujiwara, "Temperature-variation aware test pattern optimization," in *Proc. IEEE Euro. Test Symp. (ETS)*, 2011, p. 214.

[21] D. D. Josephson, S. Poehlman, and V. Govan, "Debug methodology for the McKinley processor," in *Proc. Int. Test Conf.*, 2001, pp. 451–460.

[22] S. Kaxiras and P. Xekalakis, "4 T-decay sensors: A new class of small, fast, robust, and low-power, temperature/leakage sensors," in *Proc. Int. Symp. Low Power Electron. Design*, 2004, pp. 108–113.

[23] A. Mason, A. V. Chavan, and K. D. Wise, "A mixed-voltage sensor readout circuit with on-chip calibration and built-in self-test," *IEEE Sensors J.*, vol. 7, no. 9, pp. 1225–1232, Sep. 2007.

[24] S. Remarsu and S. Kundu, "On process variation tolerant low cost thermal sensor design in 32 nm CMOS technology," in *Proc. IEEE Great Lakes VLSI*, 2009, pp. 487–492.

[25] T. Yoneda, M. Inoue, Y. Sato, and H. Fujiwara, "Thermal-uniformity -aware X-filling to reduce temperature-induced delay variation for accurate at-speed testing," in *Proc. IEEE VLSI Test Symp.*, 2010, pp. 188–193.

[26] B. Kruseman, A. K. Majhi, G. Gronthoud, and S. Eichenberger, "On hazard-free patterns for fine-delay fault testing," in *Proc. Int. Test Conf.*, 2004, pp. 213–222.

[27] S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, and S. Vrudhula, "Predictive modeling of the NBTI effect for reliable design," in *Proc. IEEE Custom Integr. Circuits Conf.*, 2006, pp. 189–192.

[28] E. J. Marinissen, V. Iyengar, and K. Chakrabarty, "ITC'02 SOC Test Benchmarks," Oct. 2002. [Online]. Available: http://itc02socbenchm.pratt.duke.edu

[29] J. Gaisler, S. Habinc, and E. Catovic, "GRLIB IP library user's manual," Ver. 1.0.16 2007.

Hyunbean Yi (M'08) received the B.S., M.S., and Ph.D. degrees in computer science and engineering from Hanyang University, Ansan, Korea, in 2001, 2003, and 2007, respectively.

Currently, he is a Professor with the Department of Computer Engineering/Graduate School of Information and Communications, Hanbat National University, Daejeon, South Korea. He was with Korea Electronics Technology Institute (KETI) from 2002 to 2007. He was a Research Scholar with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, from 2007 to 2009. He was a Postdoctoral Researcher with the Graduate School of Information Science, Nara Institute of Science and Technology (NAIST), Japan from 2009 to 2011. His research interests include high-speed communication system design, design-for-testability (DfT), SoC/NoC testing/debugging, NoC architecture optimization, and system reliability.

Prof. Yi is a member of the Institute of Electronics Engineers of Korea and the Institute of Semiconductor Test of Korea.

Tomokazu Yoneda (M'04–SM'08) received the B.E. degree in information systems engineering from Osaka University, Osaka, Japan, in 1998, and the M.E. and Ph.D. degrees in information science from Nara Institute of Science and Technology, Nara, Japan, in 2001 and 2002, respectively.

Presently, he is an Assistant Professor with the Graduate School of Information Science, Nara Institute of Science and Technology. His research interests include VLSI CAD, design for testability, low power test and SoC test.

Prof. Yoneda is a member of the IEICE.

Michiko Inoue (M'95) received the B.E., M.E., and Ph.D. degrees in computer science from Osaka University, Osaka, Japan, in 1987, 1989, and 1995, respectively.

Currently, she is a Professor of Graduate School of Information Science, Nara institute of Science and Technology (NAIST), Nara, Japan. She worked with Fujitsu Laboratories Ltd. from 1989 to 1991. Her research interests include distributed algorithms, parallel algorithms, graph theory and design and test of digital systems.

Prof. Inoue is a member of the Institute of Electronics, Information, and Communication Engineers, the Information Processing Society of Japan, and the Japanese Society for Artificial Intelligence.

Yasuo Sato (M'09) received the B.S. and M.S. degrees in mathematics from Tokyo University, Tokyo, Japan, in 1976 and 1978, respectively, and the Ph.D. degree in engineering from Tokyo Metropolitan University, Tokyo, Japan, in 2005.

He joined Hitachi, Ltd. in 1978, and began working in computer-aided design. He had been a Senior Manager of Test Methodology Group, Semiconductor Technology Academic Research Center (STARC) from 2003 to 2005. He had been a Chief Engineer with Hitachi Micro Device Division from 2006 to 2008. He joined Kyusyu Institute of Technology, Iizuka, Japan, in 2009, where he is currently a Professor. His research interests include reliability-aware testing, design for testability, delay testing, defect-based testing, and fault diagnosis.

Prof. Sato is a member of the IEICE.

Seiji Kajihara (S'87–M'92) received the B.S. and M.S. degrees from Hiroshima University, Hiroshima, Japan, and the Ph.D. degree from Osaka University, Osaka, Japan, in 1987, 1989, and 1992, respectively.

From 1992 to 1995, he worked with the Department of Applied Physics, Osaka University, as an Assistant Professor. In 1996, he joined the Department of Computer Science and Electronics of Kyushu Institute of Technology, Iizuka, Japan, where he is currently a Professor. His research interests include test generation, delay testing, and design for testability.

Prof. Kajihara was a recipient of the Young Engineer Award from IEICE in 1997, the Yamashita SIG Research Award from IPSJ in 2002, and the Best Paper Award from IEICE in 2005. He is a member of the IEICE and the IPSJ. He serves on the steering committee chair of IEEE Asian Test Symposium and the editorial board of the Journal of Electronic Testing: Theory and Applications.

Hideo Fujiwara (S'70–M'74–SM'83–F'89) received the B.E., M.E., and Ph.D. degrees in electronic engineering from Osaka University, Osaka, Japan, in 1969, 1971, and 1974, respectively.

He was with Osaka University from 1974 to 1985 and Meiji University from 1985 to 1993, Nara Institute of Science and Technology (NAIST), Nara, Japan, from 1993 to 2011, and retired in 2011. Presently, he is Professor Emeritus of NAIST and Professor at Faculty of Informatics, Osaka Gakuin University, Osaka, Japan. His research interests include logic design, digital systems design and test, VLSI CAD and fault tolerant computing, including high-level/logic synthesis for testability, test synthesis, design for testability, built-in self-test, test pattern generation, parallel processing, and computational complexity. He has published over 350 papers in refereed journals and conferences, and nine books including the book.

Prof. Fujiwara was a recipient of the IECE Young Engineer Award in 1977, the IEEE Computer Society Certificate of Appreciation Awards in 1991, 2000, and 2001, the IEEE Computer Society Meritorious Service Awards in 1996 and 2005, the IEEE Computer Society Continuing Service Award in 2005, and the IEEE Computer Society Outstanding Contribution Award in 2001 and 2009. He is a Golden Core member of the IEEE Computer Society, a fellow of the Institute of Electronics, Information and Communication Engineers of Japan (IEICE) and a fellow of the IPSJ (the Information Processing Society of Japan.