Stochastic Timing Error Rate Estimation under Process and Temporal Variations

Shoichi Iizuka Yutaka Masuda Masanori Hashimoto Takao Onoye
Department of Information Systems Engineering, Osaka University
hasimoto@ist.osaka-u.ac.jp

***Abstract*—Reducing design and operational margin is a key factor that makes fabricated chips competitive in terms of speed and power consumption. On the other hand, a smaller margin involves a higher risk that a timing error occurs in field. This paper proposes a stochastic framework that estimates timing error rate under static process variations and dynamic environmental variations for circuits with and without run-time adaptive speed control. The proposed framework extends the state assignment of the continuous-time Markov process used in the previous work so as to take into account within-die random variation, and speeds up the database construction for the transition rate matrix by combining logic simulation and statistical static timing analysis. This paper also demonstrates that the proposed framework can cope with transistor-by-transistor stochastic aging processes. Experimental results show that the within-die random variation deviates the MTTF with σ of 52%. The CPU time for the transition rate matrix computation is reduced to 1/30.**

I. Introduction

Device miniaturization due to technology scaling has made parametric performance variation more and more significant. Lower supply voltage makes circuits sensitive to environmental fluctuation, especially to supply voltage. Furthermore, aging effects, such as NBTI (negative bias temperature instability), HCI (hot carrier injection) and TDDB (time dependent dielectric breakdown), cause unexpected timing failures in field. To overcome manufacturing variability, environmental fluctuation and aging, designers set design margin and production tests give operational margin. When the given margin is large enough, timing failures in filed can be avoided. On the other hand, if the given margin is too large, the chip is often operated at a supply voltage higher than necessary and consequently it consumes larger power. Such excessive design and operational margins involve power, area and cost overhead, and/or performance loss, which deteriorates the competitiveness of the chips.

Figure 1 illustrates the operational margin in the chip lifetime. The operational margin at the beginning of the chip lifetime is large, and the margin decreases as the chip ages. If the delay increase due to aging exceeds the timing margin, timing errors occur in the chip. In addition, it is known that some aging effects vary for every transistor. For example, the threshold voltage variation due to NBTI is randomly distributed because the location and number of traps in the gate oxide are determined by a stochastic process. Furthermore, the speed of aging process depends on the workload (switching activity), supply voltage and temperature. In industry, the worst-case aging effects, i.e. the worst-case transistor, the worst workload, the highest supply voltage, and the highest temperature are assumed for estimating the impact of the aging effects, and the design and operational margins are often determined according to the worst-case aging effects. For some chips, such large margins are really necessary. However, for most of the other chips, such large margins are unnecessary.

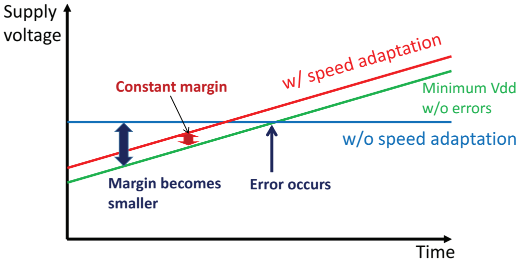

Fig. 1. Margins of circuits with and without adaptive speed control in chip lifetime.

Recently, to self-adjust the operational margin, adaptive performance control is studied [1–6]. Dynamic voltage and frequency scaling and body biasing are popular ways to control performance. With such adaptive speed control, we can ideally set a constant operational margin for the entire chip lifetime, as illustrated in Fig. 1. Especially, we might be able to reduce the margin at the beginning of the chip lifetime. If the large operational margin can be translated into supply voltage reduction, the aging process, i.e. the performance degradation, can be slowed down, and the chip lifetime extends. In addition, even when the aging effects are larger than the prediction at design time, the timing margin can be kept at the cost of higher power consumption.

However, the adaptive speed control reduces the operational margin, and hence the possibility that an unexpected timing error due to, for example, unexpected large supply noise occurs becomes higher. In addition, the margin checking performed in the chip is not perfect due to the limited test costs in area and time. Therefore, there is a fundamental problem that the possibility of timing error occurrence cannot be completely reduced to zero, since, for example, a sudden delay increase larger than expectation can induce a timing error without error detection or before error prediction. Similarly, offline delay testing may miss the error because delay testing is carried out with a certain time interval. It should be noted that the timing errors due to such a sudden delay increase arise even in the chips without adaptive speed control, especially at the end of the chip lifetime because the operational margin is small. Researchers working for any types of adaptive speed control claim that by tuning the operational margin through some design parameter optimization the possibility of timing error occurrence can be reduced to almost zero and the time to failure can be extended to over years. For example, delay testing should be carried out more frequently, earlier error prediction should be enforced, and so on.

However, it is challenging to quantitatively estimate such a long MTTF (mean time to failure) and extremely low probability of error occurrence for guiding design optimization in design time. A naive simulation is totally impractical since

978-1-4673-6578-9/15/$31.00 ©2015 IEEE

one year operation of a processor, for example, includes 3×10^{16} cycles, and to get 10-k samples, 3×10^{20} cycles must be simulated. With a logic simulator processing 3×10^{3} cycles per second, it takes 3×10^{9} years, and hence another approach instead of naive simulation is indispensable.

To reduce such a long MTTF estimation time, [7] presented a stochastic estimation framework aiming at analyzing MTTF of adaptively performance-controlled circuits. This framework models the adaptive speed control under dynamic delay variation as a continuous-time Markov process, and stochastically estimates MTTF. Given a matrix of transition rates between states, the MTTF can be calculated via matrix computations and its calculation time is independent of how long MTTFs are and how rarely the timing error happens, which is an excellent property for evaluating a long-MTTF circuit operation. Thanks to this development, the framework of [7] computes MTTF 10^{12} times faster than a logic simulator in a test case.

The continuous-time Markov process modeling enabled the fast estimation of MTTF of adaptive speed controlled circuit. However, there remain many factors of delay fluctuation that are not taken into consideration, such as inter-die and within-die process variation. Ignorance of the process variation means that the estimated MTTF is only valid for a particular chip and it is not valid for other chips with different inter-die and within-die process variations. In addition, to cover various factors of delay fluctuation, we need to increase the number of states in the Markov model. However, in the previous work [7], the computational time to prepare the transition rate matrix is significant, and hence it is difficult to increase the number of states.

This paper proposes a novel state definition for the stochastic error rate estimation that is able to cope with the inter-die and within-die process variations. The proposed state definition includes the distributions of gate delay for within-die variation at each state, and then the probability distributions of path delay violation can be considered. In addition, we devise a method of fast state transition rate calculation. The proposed method speeds up the computation of state transition rate by obtaining path delay information and path activation probability separately. We also discuss how gate-by-gate aging processes, such as NBTI, can be accommodated in the proposed framework for error rate estimation.

The rest of this paper is organized as follows. Section II exemplifies two speed adaptation systems as representatives to make the explanation of the proposed method easier; one is based on on-line testing and the other is on off-line testing. Section III outlines the continuous-time Markov process modeling of adaptive speed control developed in the previous work [7] and discusses the problems to overcome. Section IV presents the proposed state definition and the fast transition rate extraction method. Experimental results are shown in Section V, and conclusions are given in Section VI.

II. Assumed Adaptive Speed Control

This section explains two adaptive speed control systems that we use for experiments in Section V as representative ones. Note that the proposed stochastic error rate estimation is not tailored for these two adaptive speed control systems, and it is expected to be applicable to any implementations of adaptive speed control. On the other hand, the explanation with concrete examples of adaptive speed control could be helpful for readers to understand the proposed method.

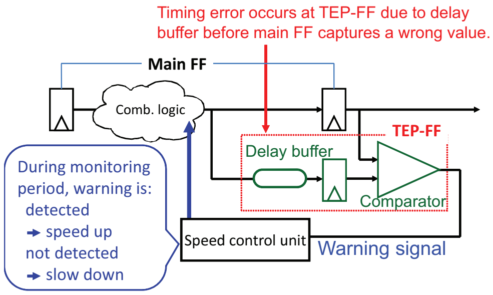
Timing error occurs at TEP-FF due to delay buffer before main FF captures a wrong value.
Main FF
Comb. logic
TEP-FF
Delay buffer
Comparator
During monitoring period, warning is:
detected
➔ speed up
not detected
➔ slow down
Speed control unit
Warning signal

Fig. 2. Run-time Adaptive Speed Control with TEP-FF.

A. On-line Test Based Adaptation using TEP-FF

Figure 2 shows a circuit that adaptively controls the speed and power dissipation using a warning signal generated by a timing-error predictive (TEP) FF [4], and the timing error rate of this run-time adaptive speed control is analyzed in this paper as one of applications of the proposed estimation framework. The TEP-FF consists of a normal flip-flop, a delay buffer and a comparator (XOR gate). When the timing margin is gradually decreasing, a timing error occurs at the TEP-FF before the main FF captures a wrong value due to the delay buffer, which enables us to know that the timing margin of the main FF is not large enough. A warning signal is generated to predict the timing errors, and it is monitored during a specified period. Note that timing errors are predicted, not detected, which is a distinct difference from Razor [3]. Once a warning signal is observed, the circuit is controlled to speed up, in other words, the circuit delay is reduced by voltage scaling and/or body biasing. Note that clock frequency is fixed throughout this paper. If no warning signals are observed during the monitoring period, the circuit is slowed down for power reduction. This proactive speed control overcomes the variation of the timing margin which is different in every chip and varies depending on operating condition and aging.

Even when the TEP-FF is well configured to generate the warning signal, the error occurrence cannot be reduced to zero. This is because when critical paths are not activated for a long time in the circuit operation, the circuit might be slowed down excessively. If a critical path is activated in this condition, a timing error happens. To reduce the error occurrence, we can tune the following design parameters; the number of TEP-FFs, locations where TEP-FFs should be inserted, delay time of the delay buffer in each TEP-FF, monitoring period and fineness of the speed control [4, 8].

B. Off-line Test Based Adaptation

We next explain an adaptive speed control system that repeatedly performs delay test in idle times of the circuit. While the circuit is idle, test-patterns that were prepared beforehand and stored in an internal or external memory are loaded and it is checked if the circuit includes timing-violating paths or not. When a timing-violating path is detected, the minimum speed level that has no timing-violating paths is selected for the operation in the following. Otherwise, the speed level is decremented. For this delay test, scan-based delay test and software-based self-test are possible candidates.

Here, there are two strategies for scan-test execution. One strategy forces the circuit to be idle with a fixed time interval, which can guarantee the time interval between the delay tests. This strategy is helpful to make the timing error rate predictable in addition to the error rate mitigation. A drawback is the performance degradation due to the testing, and in some real-time systems, this strategy could be difficult to adopt. The other strategy is to perform off-line tests only in true idle time. While the performance degradation does not arise, the test interval is less predictable and consequently the error rate tends to be higher. In this paper, we assume the first strategy having the fixed time interval of delay testing.

In this off-line test based adaptation, test coverage of the delay test is a key parameter to determine the rate of timing error occurrence. For appropriate speed adaptation, test coverage should be high not to miss timing errors. However, to make test-patterns that perform high test coverage is difficult and its overhead, test-pattern size and test time, gets large. To reduce the error occurrence while coping with the overhead, we need to carefully prepare the test.

III. Continuous-time Markov process modeling of adaptive speed control

This section explains the conventional MTTF estimation method which models temporal delay fluctuation and adaptive speed control as a continuous-time Markov process [7]. We will extend this method for taking into account the within-die random components of aging and manufacturing variation in Section IV.

A. Overview and State Assignment

The conventional method [7] models adaptive speed control under dynamic delay variation as a continuous-time Markov process. Markov process is a stochastic process having a Markov property that the next state is determined by only the current state and is independent of the previous states. Especially, continuous-time Markov process is a special Markov process whose time parameter is continuous [9][10].

The circuit delay temporally fluctuates due to unintentional temperature change, power supply noise and aging. When a path whose timing constraint is violated due to temporal delay fluctuation is activated, a timing error occurs. In case of adaptive speed control, by sensing such temporal delay fluctuation with online/offline delay testing, the circuit performance is intentionally tuned by supply voltage scaling and/or body biasing. The previous work [7] defines states in Markov process such that each state is associated with a pair of unintentional delay variation and levels of intentional speed control. We often prepare several discrete values of supply voltage scaling and body biasing for intentional speed control. On the other hand, the unintentional delay variation is continuous in nature, but for the model simplicity, we discretize the unintentional delay variation into several representative values. We call these states as normal states. In addition, we add one more failure state meaning that a timing error happened in the past. Note that this state is different from the states of sequential circuit.

Figure 3 illustrates an example of state assignment and a series of state transitions falling into the failure state. In this example, the circuit starts to operate at speed control level of 0 with 0ps delay fluctuation. Then, both the speed control level and delay fluctuation are varying dynamically. At a certain time, a timing error happens at speed control level of 0 with 30ps delay fluctuation, and the state falls into the failure state. For circuits without adaptive speed control, the number of speed control level is one and this state definition becomes one-dimensional.

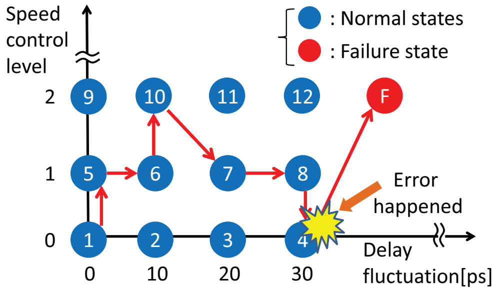

Fig. 3. An Example of State Assignment and Transition.

In a continuous-time Markov process, transition rate of going from state i to state j, $q_{i,j}$, which will be formally defined in the next subsection, is the key parameter that characterizes the process behavior. Given a matrix of the transition rates, we can obtain closed-form expressions of state probability as a function of time t. This means that once the matrix of transition rates is given, the MTTF computation can be carried out with a constant time, and the computation time is independent of the timing error rate and MTTF of the circuit under evaluation. Note that the above computation is applicable to any types of adaptive speed control, since the state assignment explained above is independent of the implementation of adaptive speed control.

B. Deriving Closed-Form State Probability Expressions from Transition Rate Matrix

This section derives closed-form state probability expressions. The state transition probability $p_{i,j}(s,t)$ is defined as a probability that a system being in state i at time s will stay in state j at time t.

$$p_{i,j}(s,t) = P(X(t) = j | X(s) = i). \tag{1}$$

In case of a stationary Markov process, $p_{i,j}(s,t)$ can be simply expressed as $p_{i,j}(t)$.

The transition rate of leaving state i, q_i, is defined as

$$q_i = \lim_{h \to 0+} \frac{1 - p_{i,i}(h)}{h} = -\frac{dp_{i,i}(0)}{dt}, \tag{2}$$

where as $h \to 0+$, $p_{i,i}(h) \to 1$. When the number of state is finite, $q_i < \infty$ holds. The transition rate of going from state i to state j ($i \neq j$) is defined as

$$q_{i,j} = \lim_{h \to 0+} \frac{p_{i,j}(h)}{h} = \frac{dp_{i,j}(0)}{dt}, \tag{3}$$

where $q_{i,j} < \infty$ always holds. Q-matrix, which consists of $-q_i$ and $q_{i,j}$, is expressed by

$$\mathbf{Q} = \begin{bmatrix} -q_1 & q_{1,2} & \cdots & q_{1,N_{state}} \\ q_{2,1} & -q_2 & \cdots & q_{2,N_{state}} \\ \vdots & \vdots & \ddots & \vdots \\ q_{N_{state},1} & q_{N_{state},2} & \cdots & -q_{N_{state}} \end{bmatrix}, \tag{4}$$

where N_{state} is the number of states and $\sum_{j \neq i} q_{i,j}(t) = -q_i$ holds.

Once Q-matrix is given, the state transition probability as a function of time t can be analytically derived by solving a Kolmogorov forward differential equation below [9].

$$\frac{dp_{ij}(t)}{dt} = -q_j p_{ij}(t) + \sum_{\nu \neq j} p_{i\nu} q_{\nu j}. \quad (5)$$

Let us introduce a way to solve this differential equation using matrix computation. Letting λ_i denote the i-th eigenvector of Q-matrix and $\mathbf{u}_i$ denote its corresponding eigenvector, we define the matrices below.

$$\mathbf{U} = [\mathbf{u}_1 \; \mathbf{u}_2 \; \cdots \; \mathbf{u}_{N_{state}}], \quad (6)$$

$$\mathbf{\Lambda}(t) = \begin{pmatrix} e^{\lambda_1 t} & 0 & \cdots & 0 \\ 0 & e^{\lambda_2 t} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & e^{\lambda_{N_{state}} t} \end{pmatrix}. \quad (7)$$

Using $\mathbf{U}$ and $\mathbf{\Lambda}$(t), the matrix of state transition probability is expressed by

$$\mathbf{P}(t) = \begin{bmatrix} p_{1,1}(t) & \cdots & p_{1,N_{state}}(t) \\ p_{2,1}(t) & \cdots & p_{2,N_{state}}(t) \\ \vdots & \ddots & \vdots \\ p_{N_{state},1}(t) & \cdots & p_{N_{state},N_{state}}(t) \end{bmatrix} = \mathbf{U}\mathbf{\Lambda}(t)\mathbf{U}^{-1}. \quad (8)$$

By specifying the initial state ($init$), the state probability being at state i at time t, $p_{init,i}(t)$ can be computed using the analytic expression in Eq. (8).

Once we obtain $\mathbf{P}(t)$, MTTF can be calculated.

$$MTTF = \int_0^\infty t \cdot \frac{dp_{init,fail}(t)}{dt} dt, \quad (9)$$

where $p_{init,fail}(t)$ is the state transition probability from the initial state to failure state. The other indexes of reliability, percentile etc. can be obtained in the same way. In addition, since we know the probabilities of being at each state, other performance metrics, such as power dissipation, can be computed using information of average power dissipation at each state.

C. Extracting Transition Rate with Database

The previous subsection tells us that once the transition rate of going from state i to state j, $q_{i,j}$, becomes available, we can obtain the exact closed-form probability expression of timing failure as a function of time. This section explains how the transition rate is computed in [7].

In the Markov model under consideration, the state transitions correspond to the events of failure occurrence, change of speed control level, or temporal delay fluctuation. Therefore, the rates of these event occurrences are the transition rates that we need to prepare for estimating MTTF. For the temporal delay fluctuation, we need a stochastic model that reproduces the temporal delay fluctuation under consideration, for example NBTI, dynamic power supply noise, etc, and here it is assumed to be given. On the other hand, the rates of failure occurrence and speed level transition depend on the running program and input data, where we call the pair of the program and input data as workload. To extract the transition rate $q_{i,j}$, [7] constructs a database. The database includes the results of logic simulation of various circuit operations at available speed levels in the range of possible delay fluctuation. Using this similarity database, we know the occurrence of events that correspond to the transition from state i to state j, such as speed level transition, during a workload execution. If the execution frequency of each workload and a model of dynamic delay fluctuation are given, we can calculate $q_{i,j}$ by referring the database.

A possible implementation of the database, which is adopted in [7], has the current speed level, the current delay fluctuation and the workload as keys, and the database quickly outputs occurrence of timing error, results of error prediction, results of path delay tests, and so on. For the workload, the exact information is often non-available. In this case, the information of the similar workload in the database substitutes. The information to be stored in the database can be extracted by performing logic simulations. The number of logic simulation executions is the product of the number of states and the number of supposed workloads. This database is constructed for each circuit and technology.

D. Problems

The continuous-time Markov process modeling explained in this section so far enabled us to estimate the MTTF of an adaptively speed controlled circuit in a reasonable time. In a test case, the conventional method computed MTTF 10^{12} times faster than a logic simulator, and the CPU time was reduced to 37 seconds, where the CPU time to construct the database is not included. However, there are some issues to be resolved for putting this stochastic estimation method to practical use.

The first problem is that the MTTF estimation of the conventional method is performed for a particular chip, i.e. all the analysis is based on the timing characteristics of the chip. The database is constructed by performing logic simulation with the SDF (standard delay format) file of the chip. Any manufacturing variability cannot be considered. On the other hand, as everybody knows, even the chips which are fabricated from the same GDS data have different timing characteristics. Circuit designers want to verify that the timing error occurrence is rare enough for all the chips that have passed the production test. However, the conventional method cannot be used for this verification purpose.

Manufacturing variability consists of inter-die and within-die variations, and the gate delay is affected by both of them. For the inter-die variation that is identically applicable to all the devices on a chip, we might be able to extend the state definition shown in Fig. 3 by adding a new dimension for each parameter of the inter-die variation at the cost of the increase in the database construction time, where this database construction time will be discussed in the next paragraph. However, the within-die random variation cannot be represented by extending the state definition, because the device parameter variation is different in every transistor and the state dimension needs to increase by the number of transistors on a chip. It is obvious that such a state extension is not practical since the number of states increases exponentially as the number of transistor increases. The matrix computation in Section III-B is difficult to perform, and more importantly it is prohibitively expensive to prepare the database for such a huge state space. We need to have a new state definition that can accommodate within-die random variation.

Another problem is that there are many physical sources that cause temporal delay fluctuation, such as temperature change, power supply noise, various aging processes, and so on. Even if each source can be represented with a single parameter, we need to increase the dimension of the state definition in Fig. 3 by the number of physical sources. Let us exemplify the impact of this dimension increase on the number of states. Now, we suppose N_{source} parameters of physical sources causing temporal delay fluctuation. If each parameter takes N_{value} values after quantization for discrete Markov modeling, the number of states becomes $N_{value}^{N_{source}}$, and this increases exponentially according to N_{source}. For example, let us suppose that temperature, supply voltage noise and age are selected as the parameters corresponding to physical sources of temporal delay fluctuation. In addition, we have one more dimension representing the speed control level. When each parameter has 10 values, the number of states becomes 10^4+1=10,001, whereas the number of states was 101 in the conventional two-dimensional state definition. To accommodate various physical sources of temporal delay fluctuation, we need to speed up the construction of the database. Even for the two-dimensional state definition, it took 2.6 days to construct the database for 271 states using eight processors [7]. Drastic CPU time reduction is necessary.

IV. Proposed State Definition and Fast Database Construction

This section explains the proposed MTTF estimation method which can take into account the within-die random variation and speed up the database construction.

A. State Definition

The proposed method keeps the basic concept that the state transitions occur when the circuit delay varies by temporal delay fluctuation and the speed level changes. Therefore, the advantage of the previous work, which is the very fast MTTF estimation and the CPU time is independent of the length of MTTF, is preserved in the proposed method. The dimension of the state definition in Fig. 3 is increased by the number of parameters that are necessary to express the physical sources of dynamic delay fluctuation. In addition, the state definition is extended for the inter-die manufacturing variation. The problem of CPU time increase involved with this dimension increase will be solved in Section IV-B.

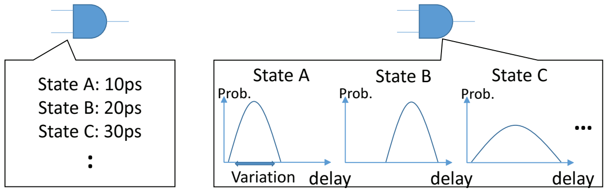

Fig. 4. Gate delay expressions of conventional [7] and proposed methods.

On the other hand, the within-die random variation is taken into account in the database construction. In the previous work [7], the transition rate is computed for a particular chip, while the proposed method calculates the transition rate for a circuit in which each gate delay varies according to probability distributions. Figure 4 illustrates the gate delay expressions in the previous work [7] and the proposed method. In [7], the left figure of Fig. 4, the gate delay value is deterministic as shown in the left. On the other hand, the gate delay value is stochastic in the proposed method and the gate delay value may take any value following the probability distribution as shown in the right figure, which is similar to SSTA (statistical static timing analysis). The important point here is that the probability distribution of the gate delay is different at each state. In other words, each state gives its own stochastic gate delay distributions. With this modification, the within-die random variation can be considered without increasing the number of states. Letting N_{gate} denote the number of gates and N_{value} denote the number of discretized available delay values, the number of necessary states for the naive state expansion would be $N_{value}^{N_{gate}}$ and this number is tremendous even for 10k-gate circuit ($N_{gate} = 10,000$). On the other hand, the proposed method does not need the state expansion. Thanks to the proposed state definition using probability distribution, the number of necessary states for within-die variation is reduced by $1/N_{value}^{N_{gate}}$

In this new state definition, the path delays are also represented by probability distributions. In the previous work [7], the information stored in each entry of the database is the binary information, i.e. whether the path causes timing violation or not. On the other hand, in the proposed method, this information in the database becomes the probability of timing error occurrence which is computed exploiting the probability distribution of the path delay variation. The transition rate is computed from this timing violation information, which will be explained in the following subsections.

At the beginning of this subsection, we extended the dimension of the state to accommodate the physical sources of dynamic delay fluctuation and the inter-die process variation. On the other hand, the inter-die process variation is static, i.e. it is fixed after fabrication. This static property needs to be considered in MTTF estimation. Figure 5 illustrates how this static property is considered. Here, each plane corresponds to the state space in which one parameter of inter-die process variation, in this example inter-die Vth variation, is fixed. $\mathrm{S}_{\mathrm{i,j,k}}$ denotes a state, and i, j, k correspond to speed control level, age and Vth variation. As mentioned above, the parameter of inter-die manufacturing variation is fixed after fabrication, and therefore the state transition should occur only within each plane. This can be easily annotated to the transition rate matrix by setting the transition rates for such inter-plane state transitions zero. In addition, the probability distribution of the inter-die Vth variation can be considered by giving the initial state probabilities according to the probability distribution of the inter-die Vth variation. For example, let us assume the chip operation starts from the state of speed level of two and age of zero, i.e. $\mathrm{S}_{2,0,0}, \mathrm{S}_{2,0,1}, \mathrm{S}_{2,0,2}$. In this case, the initial state probabilities of $\mathrm{S}_{2,0,0}, \mathrm{S}_{2,0,1}, \mathrm{S}_{2,0,2}$ directly correspond to the Vth distribution, where the sum of these three initial state probabilities is one, and the initial state probabilities of the other states are zero. Thus, static inter-die process variation can be considered.

B. Database Construction

In the previous subsection, we extended the state definition for coping with process variations. This section explains how the database is constructed for the new state definition.

We need to store the probability of timing error occurrence in the database. A timing error occurs in case that the following two conditions simultaneously satisfied; (1) the delay of a path

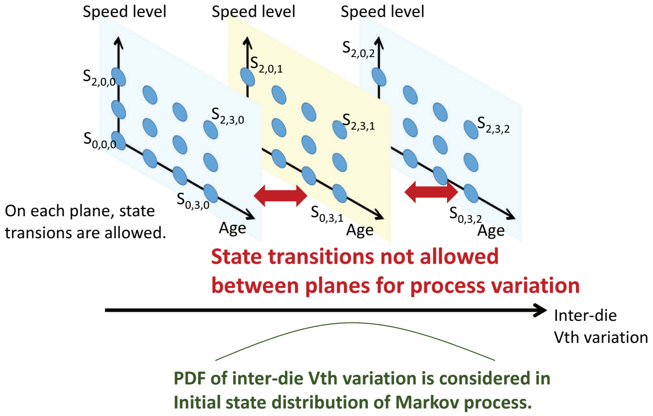
Speed level
Speed level
Speed level
$S_{2,0,0}$
$S_{2,0,1}$
$S_{2,0,2}$
$S_{2,3,0}$
$S_{2,3,1}$
$S_{2,3,2}$
$S_{0,0,0}$
$S_{0,3,0}$
$S_{0,3,1}$
$S_{0,3,2}$
Age
Age
Age
On each plane, state transions are allowed.
State transitions not allowed between planes for process variation
Inter-die Vth variation
PDF of inter-die Vth variation is considered in Initial state distribution of Markov process.

Fig. 5. State definition and transitions for inter-die variations.

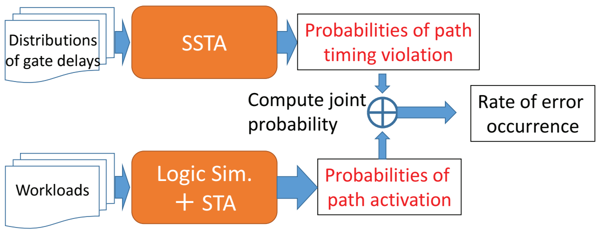
Distributions of gate delays
SSTA
Probabilities of path timing violation
Compute joint probability
Rate of error occurrence
Workloads
Logic Sim. + STA
Probabilities of path activation

Fig. 6. Computing error occurrence rate for database construction.

violates the timing constraint, and (2) the path is activated. In the previous work [7], logic simulation is carried out to know whether a timing error occurs or not, i.e. these two conditions are evaluated simultaneously. In this case, we need to execute logic simulation every time the timing characteristics change. Due to manufacturing variability, every chip has different timing characteristics, and hence logic simulation must be performed in a Monte Carlo manner, which is prohibitively expensive.

On the other hand, in this work, we compute the probability of path delay violation and the probability of path activation for each path separately, and then calculate the probability of timing error occurrence, which is the joint probability of the probabilities of path delay violation and path activation, as illustrated in Fig. 6. The proposed method obtains the probabilities of path delay violation and the path activation as follows.

The probability distribution of path delay variation can be computed by path-based statistical static timing analysis (SSTA). We give the distributions of gate delay at the state under consideration, and perform path-based SSTA. There are a number of proposals of path-based SSTA (a survey is found in [13]), and we need to choose one of them according to the shape of gate delay distribution, accuracy and CPU time. This SSTA is performed for all the states. Once the path delay distribution is available, we can compute the probability of path delay violation, which is the probability that the timing constraint of the path of interest is violated.

The path activation probability is extracted from the logic simulation result since the path activation depends on the workload. However, the logic simulator does not give the information of path activation directly. Logic simulator outputs the locations and timings of signal transitions, and we can know when and where (i.e. at which net) signal transitions occur. However, this signal transition information is not associated with the path on which the signal is propagating. For every transition at the input terminal of each flip-flop, we need to know which path delivers this transition. To associate the transition with the activated path, we use STA result. The proposed method checks the correspondence between the timing of signal transition obtained by the logic simulation and the path delay reported by STA.

This correspondence checking process is performed as follows. We perform the STA of the target circuit and make the list of path information, such as endpoint (input terminal of flip flop), transition direction (rise or fall) at endpoint and path delay. The number of paths in a circuit is too large, and hence we select top N_{path} paths for the list. Next, logic simulation of each workload is performed. Then, we extract the transition timing and direction at FF inputs from the dump file of logic simulation. The information extracted from STA and logic simulation can be expressed as triplets of [endpoint, delay, direction (rise or fall)], where we call this triplet as path triplet. By checking the correspondence between the path triplet from logic simulation and the path triplet from STA using Algorithm 1, the activation probability of path i can be obtained. P_{logic} in Algorithm1 denotes the set of the path triplets extracted from logic simulation and P_{STA} is the set of the path triplets extracted from STA result. P_{STA_SUB} is a subset of P_{STA}, and P_{STA_SUB} corresponds to the top N_{path} paths. For each path triplet from logic simulation $triplet\_logic_j$, Algorithm 1 first finds the identical path triplet from STA $triplet\_STA_i$ and recognizes that the signal transition of $triplet\_logic_j$ propagated through the i-th path. Then, the number of path activation of the i-th path, n_i, is incremented. After all the path triplets from logic simulation are processed, we compute the path activation probability of the i-th path $p_{active,i}$ as $n_i/cycles$, where $cycle$ is the number of clock cycles to complete the workload. If multiple paths have the same delay value, this path identification algorithm cannot find one-to-one correspondence. In this case, by repeatedly performing this algorithm under another delay condition, one-to-one correspondence can be obtained.

In this subsection, we explained how to compute the probability of timing error occurrence. Similarly, the probability of timing error prediction/detection can be computed. The details are omitted due to the space limitation. In the case of the result of path delay tests, the probability of the path delay passage

Algorithm 1 Computation of path activation probability p_{active}

```
for all $triplet\_STA_i \in P_{STA\_SUB}$ do
    $n_i = 0$
end for
for all $triplet\_logic_j \in P_{logic}$ do
    for all $triplet\_STA_i \in P_{STA\_SUB}$ do
        if $triplet\_STA_i == triplet\_logic_j$ then
            $n_i + +$
        end if
    end for
end for
for all $triplet\_STA_i \in P_{STA\_SUB}$ do
    $p_{active,i} \leftarrow n_i/cycles$
end for
```

Algorithm 2 Computation of state transition rate for one workload (on-line test based adaptation using TEP-FF).

```
1: for all $i \in States$ do
2:     $q_{i,delay++} \leftarrow$ dynamic delay fluctuation rate
3:     $q_{i,delay--} \leftarrow$ dynamic delay fluctuation rate
4:     for all $p \in P_{STA\_SUB}$ do
5:         $q_{i,fail} \leftarrow q_{i,fail} + ViolationRate(p) \times ActivationRate(p)$
6:         $\underline{q_{i,SpeedLevel+1}} \leftarrow q_{i,SpeedLevel+1} + (\overline{Violation\_rate(p)} \times ViolationRate(p_{TEPFF})) \times ActivationRate(p)$
7:     end for
8:     $q_{i,SpeedLevel-1} \leftarrow 1/$MonitorTime
9: end for
```

can be calculated only with SSTA. Thus, we can construct the database.

C. Computation of State Transition Rate

This subsection explains how to construct Q-matrix using the database.

State transitions occur in three cases; (1) a timing error arises, (2) the speed level changes and (3) the temporal delay fluctuation occurs. We thus need to compute the probabilities for these three cases. The computation of Q-matrix consists of two steps. The first step computes preliminary state transition rates for each workload executed at a state. On the other hand, the probability of each workload being executed in the actual circuit behavior is different depending on, for example, the processor usage. The second step calculates the overall state transition rates taking into account the processor usage and the dynamic delay fluctuation.

We first explain the computation of Q-matrix for on-line test based adaptation using TEP-FF explained in Section II. The computation of Q-matrix of one workload is shown in Algorithm 2. $States$ represents the set of all states in the state space. The functions $ViolationRate$ and $ActivationRate$ return the timing violation rate and path activation rate of the path of interest referring to the database. p_{TEPFF} represents the path whose endpoint has TEP-FF. In lines 2 and 3, the rates of dynamic delay fluctuation are substituted for $q_{i,delay++}andq_{i,delay--}$. Next, we compute the rate of error occurrence of each path, which is the product of the timing violation rate and the path activation probability, and we sum up the rate of error occurrence for all the paths. Line 6 computes the probability that the main FF does not cause the timing error and the TEP-FF causes the timing error and the path is activated. This corresponds to the probability of the error prediction, $q_{i,SpeedLevel+1}$. The probability of speed control level decrease, $q_{i,SpeedLevel-1}$ is given by the inverse of monitor time of speed adaptation. With Algorithm 1, we can compute necessary transition rates for one workload.

Then, the weighted average of state transition rates is calculated at step 2. In order to consider biased (non-equal) workload execution, the probability of each workload execution is given as the weight. Then, we simply calculate the weighted average of each Q-matrix elements using this weight.

State transition rates for off-line test based adaptation can be computed with Algorithm 3. The transition rates of delay fluctuation and the failure rate, $q_{i,delay++}andq_{i,delay--}, q_{i,fail}$, can be computed in the same way. The difference to the on-line test is the trigger of speed level transition. In off-line test based adaptation, speed level changes by the result of off-line test. The speed level changes to only adjacent states for simplicity in this paper. The function, $TestCoverage$ in this algorithm returns whether the path of interest is testable or not. In line 6, the probability that the off-line test detects the timing violation is computed for each path. In line 7, the probability that the off-line test dose not detect the timing violation is computed for each path. The calculation of the weighted average is the same with that of on-line test.

Algorithm 3 Computation of state transition rate for 1 program (off-line test based adaptation).

```
1: for all $i \in States$ do
2:     $q_{i,delay++} \leftarrow$ dynamic delay fluctuation rate
3:     $q_{i,delay--} \leftarrow$ dynamic delay fluctuation rate
4:     for all $p \in P_{STA\_SUB}$ do
5:         $q_{i,fail} \leftarrow q_{i,fail} + ViolationRate(p) \times ActivationRate(p)$
6:         $q_{i,SpeedLevel+1} \leftarrow q_{i,SpeedLevel+1} + ViolationRate(p) \times TestCoverage(p)/$TestInterval
7:         $q_{i,SpeedLevel-1} \leftarrow q_{i,SpeedLevel-1} + (1 - TestCoverage(p))/$ TestInterval
8:     end for
9: end for
```

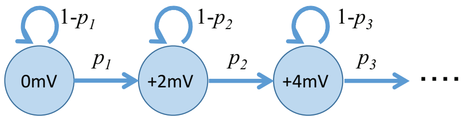

Fig. 7. An example of stochastic aging model for each transistor.

D. Gate-by-Gate Aging Consideration

It is known that aging process is stochastic in nature, and hence each gate has different aging processes. For example, in case of NBTI, the locations and the number of traps in the gate oxide are statistically distributed. In addition, the aging speed depends on the switching activity of the gate. In order to accurately consider the aging process, we need to consider both the long-term tendency and the stochastic property.

We assume that a stochastic aging model for each transistor is given. A simple example is shown in Fig. 7. In this example, the threshold voltage degrades monotonically. Using such stochastic models, we can generate a number of aging processes for each transistor by generating random numbers. Letting $N_{aging\_process}$ denote the number of generated aging processes, we have $N_{aging\_process}$ threshold voltage values at each age. Therefore, we can compute the mean and the standard deviation of the threshold voltage for each transistor at each age. In other words, the stochastic property is available.

To express the long-term tendency, we add one parameter of age to the state definition of Markov process. Figure 8 illustrates the state transition that expresses the age increment. Note that the state definition in Fig. 8 is the same with that in Fig. 5. The states on the left plane correspond to age 0, and the gate delay degradation follows the stochastic property

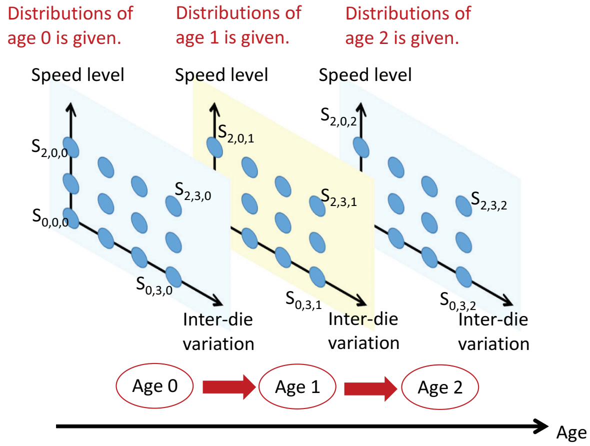

Fig. 8. State transitions for aging.

at age 0. The middle and right planes correspond to age 1 and age 2, respectively. In the Markov process modeling, the transitions to older ages occur stochastically. By appropriately determining the transition rates $p_{i,j}$ to older ages, the mean time to increment an age in the Markov model can match with the time between the ages, $t_j - t_i$, which are assumed in deriving the stochastic property. The state transition rate $p_{i,j}$ is given by

$$p_{i,j} = 1 - \exp(\frac{-\log 2}{t_j - t_i}). \tag{10}$$

V. Experiments

This section shows experimental results. First, we explain the implementation of SSTA necessary for the proposed estimation method. Next, we demonstrate that the proposed estimation method can obtain the state transition rate 30 times faster than the conventional method [7]. In addition, the impact of within-die process variation on MTTF is exemplified. Then, we show some analysis examples for future design optimization.

A. SSTA Implementation

As we explained in Section IV-B, the probability distributions of path delay variation can be obtained by SSTA. In this paper, we adopted Monte Carlo SSTA as the implementation of SSTA for experiments. Monte Carlo SSTA collects statistics by repeating STA with different gate delays. Figure 9 shows the flowchart of the implemented Monte Carlo SSTA. First, we prepared N_{STA} virtual chips, where N_{STA} is the number of trials of Monte Carlo simulation. Each chip includes different within-die random process variations, more concretely different SDF (standard delay format) files. Next, we perform STA for N_{STA} chips. For each execution of STA, we can obtain the path delay value for each path. Then, we can collect the statistics of the path delay.

The above SSTA needs to be executed for each state, since each state has different inter-die variations and ages and then the timing characteristics are different. It should be noted that Monte Carlo SSTA is just an implementation of path-based SSTA, and other SSTA methods can be also used. In addition, Monte Carlo SSTA can be accelerated by smart sampling [12].

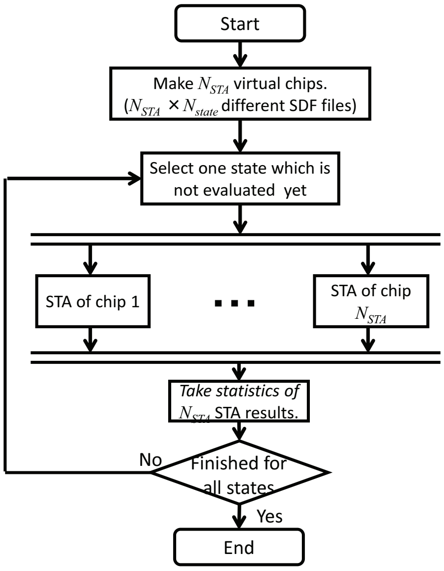
Start
Make N_{STA} virtual chips. (N_{STA} × N_{state} different SDF files)
Select one state which is not evaluated yet
STA of chip 1
...
STA of chip N_{STA}
Take statistics of N_{STA} STA results.
Finished for all states
No
Yes
End

Fig. 9. Flowchart of the implemented Monte Carlo SSTA.

B. Experimental Setup

In this work, the adaptive speed control is applied to OR1200 OpenRISC Processor. OR1200 is a 32-bit RISC microprocessor and implemented with five pipeline stages. The processor was designed by synthesizing RTL hardware description was synthesized by a commercial logic synthesizer with a 45nm Nangate standard cell library. The number of standard cells is 24,000. The maximum clock frequency at 1.1V and 25°C is 217MHz, which corresponds to the critical path delay of 4.6ns.

The number of STA of Monte Carlo SSTA N_{STA} is 100 in this experiment using a commercial STA tool, where the necessary N_{STA} and/or smart sampling needs to be explored in the future work. We selected the top 1,000 paths to be considered, i.e. N_{STA} is 1,000. We selected three benchmark programs (CRC32, SHA1 and Dijkstra) from MIBenchmark [11] and 30 sets of input data for each program. Consequently, 90 (=3 × 30) workloads were prepared.

The three factors, speed control by voltage scaling, dynamic supply voltage fluctuation, and within-die random process variation, are considered in this experiment. The states of Markov process are defined by speed control level and supply voltage fluctuation. The aging is considered in Section V-E. Twelve speed levels, i.e. ten supply voltages (1.2V, 1.15V, 1.1V, 1.05V, 1.0V, 0.95V, 0.9V, 0.85V, 0.8V, 0.75V, 0.70V and 0.65) were prepared. The supply voltage fluctuates between -50mV and 50mV by 10mV with eleven steps. Here, a stochastic fluctuation model expressed as a Markov chain in Fig. 10 is used. The transition rate of supply voltage fluctuation was set to 0.001 ($p = 0.001$). The number of states in this experiment is 12 (speed levels) × 11 (voltage fluctuation) + 1 (failure) = 132. The within-die random variations given to the virtual chip fabrication in the SSTA are $\sigma = 30mV$ for threshold voltages of NMOS and PMOS, and $\sigma = 2$nm for gate length. The delay characteristics of cells at each supply voltage are obtained using SPICE simulations and linear interpolation.

C. CPU Time Evaluation for Database Construction

First, we evaluate the speed-up of the proposed database construction over the conventional one with ordinary logic

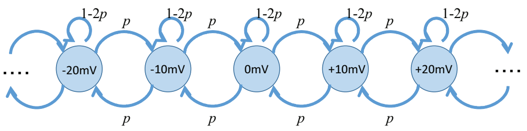

Fig. 10. A stochastic model of dynamic supply noise.

TABLE I. COMPARISON ON CPU TIME FOR DATABASE CONSTRUCTION.

	Conventional [7]	Proposed
Runs of logic sim.	$132 \times 90 = 11,880$	90
CPU time per run [s]	120	90
Runs of SSTA	0	132
CPU time per run [s]	-	300
Total [h]	396	13.25

simulation. The adaptive speed control system based on TEP-FF is analyzed in this subsection.

The proposed method reduced the number of simulations (STA, SSTA, logic simulation) from $N_{state} \times N_{workload}$ to $N_{state} + N_{workload}$. In this experiment, N_{state} is 132, $N_{workload}$ is 90. Table I shows the CPU time of each simulation and total CPU time. These simulations were carried out on a computer with CentOS5, Intel Xeon X5680 processor and 96GB memory. We can see that the proposed method reduced the number of logic simulations from 11,800 to 90. On the other hand the proposed method needs to perform SSTA, and the number of SSTA runs is 132. As a result, the simulation time for database construction is reduced from 396 hours to 13.25 hours, and its reduction ratio is 1/30. This speed-up will be more significant if N_{state} and/or $N_{workload}$ becomes larger. The proposed method reduced the CPU time even while it explicitly considers the within-die random process variation whereas the conventional method cannot consider the random variation. Besides, the impact of within-die process variation will be evaluated in Section V-D. This speed-up facilitates increasing the number of states for taking into account more delay fluctuation sources.

D. Impact of Within-die Variation

Next, this section shows the impact of within-die process variation on MTTF. To evaluate the impact of within-die process variations, we estimated the MTTF of 100 chips which had different within-die variation. The procedure of MTTF estimation for each chip is the same as the proposed MTTF estimation method except that the number of virtually fabricated chip is one. Figure 11 shows the average and the standard deviation of the MTTF. The X-axis of Fig. 11 is the number of buffers inserted in TEP-FFs. As the number of buffer becomes larger, the MTTF becomes longer since the error prediction occurs earlier. More importantly, we can see that all the standard deviations are larger than 9% of the MTTF and the standard deviation of the 6 buffer case is larger than 52%. This indicates that the within-die random variation may change the MTTF several times when thinking of $\pm 3\sigma$. Such MTTF variation cannot be estimated in the conventional method [7]. The proposed method can improve the estimation accuracy.

E. Analysis for Aging

To demonstrate that the proposed method can consider gate-by-gate aging processes, we exemplified an analysis. We used a simple aging model that is expressed by a Markov model. The number of states is 8, and each state corresponds to 0/1/2/3/4/5/6/7 ps increase in the gate delay from the fresh chip. The state transition rates to the next state of delay increase are $[10^{-1}, 10^{-2}, ..., 10^{-7}, 0]$, and other state transitions are not allowed, i.e. the state transition rates are 0.

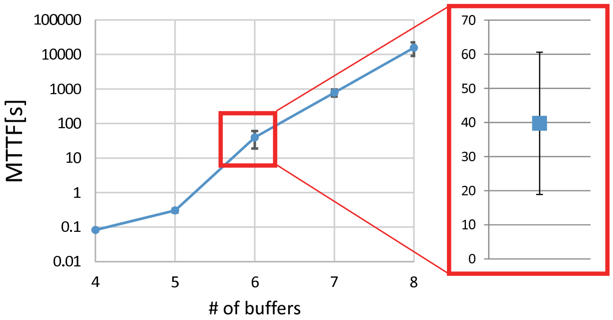

Fig. 11. Effect of within-die random variation on MTTF. X-axis is the number of buffers in the TEP-FFs. Error bars represent $\pm 3\sigma$.

In this experiment, a parameter of age was added, as explained in Section IV-D. This parameter corresponds to the elapsed time, and it is discretized into five states, where the five states correspond to the elapsed times of 0, 2, 101, 1001, 10001 s, respectively.

The upper figure of Fig. 12 shows the MTTFs of the circuit with and without adaptive speed control under the aging. The supply voltage of the circuit without adaptive speed control is fixed to 0.7V. The number of buffers in the TEP-FF is three. We can see that the average supply voltage of the circuit with adaptive speed control increases as the time elapses and the aging effect proceeds. The bottom figure of Fig. 12 shows the accumulated failure probability. At the beginning, the supply voltage of 0.7V gives enough timing margin, and hence the increase in the accumulated failure probability is smaller in the case without adaptive speed control. However, as the time elapses, the timing margin of the circuit without adaptive speed control becomes smaller, and the accumulated failure probability increases faster. On the other hand, with the adaptive speed control, the timing margin is kept almost constant, and hence the increasing rate of the accumulated failure probability, i.e. the failure rate, is almost constant. For the real products, the failure rate needs to be reduced further via design optimization, but the important point here is that such an analysis can be performed with the proposed estimation method.

F. Analysis Examples

Finally, we give two MTTF estimation results using on-line and off-line adaptation. Figure 13 plots the MTTF when the number of inserted TEP-FFs is changed from 10 to 1000. The number of buffers in the TEP-FFs is five. We can see the MTTF improved linearly to the number of TEP-FFs below 300, and above 300 the MTTF saturated. We can estimate how many TEP-FFs are necessary to meet the error rate constraint. Also guided by the proposed estimation method, the inserted locations can be optimized, which is included in our future works.

We next show the experiments of off-line adaptation. To evaluate the impact of test coverage, we varied the test coverage artificially and estimated the MTTF. Here, the test coverage TC is defined so that TC % paths in the N_{STA}

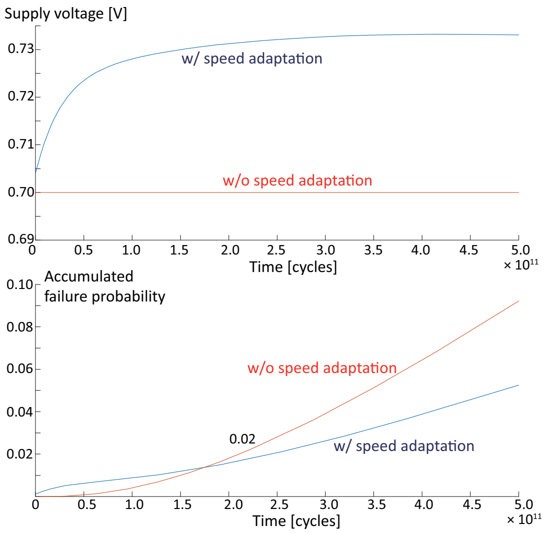

Fig. 12. MTTF and accumulated failure probability under an aging process.

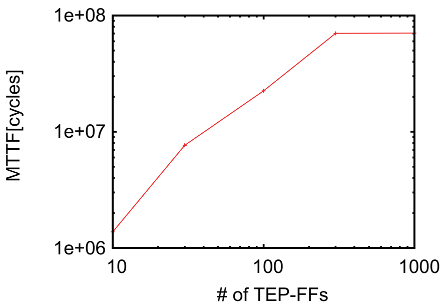

Fig. 13. MTTF versus the # of inserted TEP-FFs.

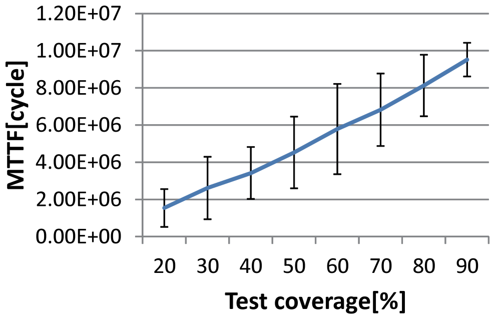

Fig. 14. MTTF versus test coverage TC of off-line adaptation. Error bars represent $\pm 3\sigma$.

paths are testable. In this experiment, TC is varied from 20% to 90% with a step of 10%, and for each TC we generated 30 sets of testable paths randomly. For each set of testable paths, we evaluated the MTTF. Figure 14 shows the mean and 3σ of the MTTF. We can see the mean of MTTF improves as the test coverage TC increases, but there is a considerable MTTF variation depending on the set of testable paths. This observation indicates that the test pattern preparation is influential on the MTTF and we will be able to optimize the test patterns to maximize the MTTF using the proposed estimation method.

VI. Conclusion

In this work, we proposed a stochastic MTTF estimation framework under dynamic and process variations. While keeping the advantages of the modeling of adaptive speed control as a continuous-time Markov process, the proposed method can reduce the CPU time for computing the transition rate matrix. In addition, the proposed statistical state definition enables to consider both inter-die and within-die delay fluctuation. Thanks to the statistical state definition and fast transition rate computation, the CPU time for the database construction decreased to 1/30 while the proposed method newly considered within-die random variation. This enablement of long MTTF evaluation under static process variation and temporal fluctuation contributes to quantitatively deriving appropriate design and operational margins and helps design and validate adaptive speed control system with field delay testing.

Acknowledgements

This work is supported by STARC.

References

[1] M. Agarwal, B. C. Paul, Z. Ming, and S. Mitra, “Circuit Failure Prediction and Its Application to Transistor Aging,” in *Proc. VTS*, pp.277–286, 2007.

[2] Y. Li, S. Makar, and S. Mitra, “CASP: Concurrent Autonomous Chip Self-Test Using Stored Test Patterns,” in *Proc. DATE*, pp.885–890, 2008.

[3] S. Das, et.al., “A self-tuning DVS processor using delay-error detection and correction,” *IEEE JSSC*, vol.41, pp.792–804, Apr. 2006.

[4] H. Fuketa, M. Hashimoto, Y. Mitsuyama, and T. Onoye, “Adaptive Performance Compensation with In-Situ Timing Error Predictive Sensors for Subthreshold Circuits,” *IEEE TVLSI*, vol. 20, no. 2, pp. 333–343, Feb. 2012.

[5] K. A. Bowman, et.al., J. W. Tschanz, S. L. Lu, P. A. Aseron, M. M. Khellah, A. Raychowdhury, B. M. Geuskens, C. Tokunaga, C. B. Wilkerson, T. Karnik, and V. K. De, “A 45 nm Resilient Microprocessor Core for Dynamic Variation Tolerance,” *IEEE JSSC*, Vol. 46 , No. 1, pp.194 –208, Jan. 2011.

[6] D. Blaauw, et.al., “Razor II: In Situ Error Detection and Correction for PVT and SER Tolerance,” in *ISSCC Dig.*, pp.107–114, 2013.

[7] S. Iizuka, M. Mizuno, D. Kuroda, M. Hashimoto and T. Onoye, “Stochastic error rate estimation for adaptive speed control with field delay testing,” in *Proc. ICCAD*, pp. 107–114, 2013.

[8] H. Fuketa, M. Hashimoto, Y. Mitsuyama, and T. Onoye, “Trade-Off Analysis between Timing Error Rate and Power Dissipation for Adaptive Speed Control with Timing Error Prediction,” *IEICE Trans. Fundamentals*, vol. E92-A, no. 12, pp. 3094–3102, Dec. 2009.

[9] A. Papoulis and S. U. Pillai, “Probability, Random Variables and Stochastic Process, Fourth Edition,” McGraw-Hill Higher Education, 2002.

[10] J. R. Norris, “Markov Chains,” Cambridge University Press, 1997.

[11] M. R. Guthaus, et.al., “Mibench: A free, commercially representative embedded benchmark suite,” in *Proc. IEEE Workshop on Workload Characterization*, 2001.

[12] V. Veetil, K. Chopra, D. Blaauw, D. Sylvester, “Fast Statistical Static Timing Analysis Using Smart Monte Carlo Techniques,” *IEEE Trans. CAD*, Vol. 30, No. 6, pp. 852865, June 2011.

[13] D. Blaauw, K. Chopra, A. Srivastava, L. Scheffer, “Statistical Timing Analysis: From Basic Principles to State of the Art,” *IEEE Trans. CAD*, Vol. 27, No. 4, pp. 589607, April 2008.