IEICE TRANS. FUNDAMENTALS, VOL.E99–A, NO.7 JULY 2016

PAPER Special Section on Design Methodologies for System on a Chip

Fast Estimation of NBTI-Induced Delay Degradation Based on Signal Probability

Song BIAN[†a)], ***Nonmember***, **Michihiro SHINTANI**[†], **Masayuki HIROMOTO**[†], ***and*** **Takashi SATO**[†], ***Members***

SUMMARY As technology further scales semiconductor devices, aging-induced device degradation has become one of the major threats to device reliability. Hence, taking aging-induced degradation into account during the design phase can greatly improve the reliability of the manufactured devices. However, accurately estimating the aging effect for extremely large circuits, like processors, is time-consuming. In this research, we focus on the negative bias temperature instability (NBTI) as the aging-induced degradation mechanism, and propose a fast and efficient way of estimating NBTI-induced delay degradation by utilizing static-timing analysis (STA) and simulation-based lookup table (LUT). We modeled each type of gates at different degradation levels, load capacitances and input slews. Using these gate-delay models, path delays of arbitrary circuits can be efficiently estimated. With a typical five-stage pipelined processor as the design target, by comparing the calculated delay from LUT with the reference delay calculated by a commercial circuit simulator, we achieved 4114 times speedup within 5.6% delay error.

key words: *NBTI, reliability, static timing analysis, timing characterization, aging-aware timing library*

1. Introduction

In a time where semiconductor devices are scaling near the limit of physical laws, they become more and more unpredictable and uncontrollable. Nano-scale transistor devices vary their characteristics greatly even within a single chip, and one of the major variation factors is aging-induced device degradation. It is obvious that the variation due to aging exerts great impact on design decisions. Hence, without knowing how the device ages, designers could potentially waste time for optimizing paths that become non-critical in less than a year.

Among various aging mechanisms, negative bias temperature instability (NBTI) is considered to be one of the most crucial factors that shorten the lifespan of VLSI circuits. NBTI causes a gradual increase in threshold voltage, V_{th}, while a negative bias is applied to a pMOS transistor, i.e., when a pMOS is ON. This state is defined as the stress phase of NBTI. The increase in the threshold voltage (ΔV_{th}) reduces the drain current and consequently increases path delays. Meanwhile, when the pMOS is OFF, the stress is removed then, and its V_{th} gradually decreases to its original value. This state is defined as the recovery phase of NBTI. The V_{th} degradation due to NBTI gradually progresses through circuit operations, while repeating the stress and recovery phases. As a result of the pMOS degradation, the delay of combinational paths becomes longer, eventually violating the design timing constraint.

The state-of-art NBTI mitigation techniques include internal node control (INC) [1], input vector control (IVC) [2], and simple design margin [3]. Obviously, these methods require path delays in the circuit to be predicted correctly, such that the NBTI-induced stresses along those paths can be reduced. In order to be NBTI-aware in design phase, tools like SPICE are adopted. Nonetheless, for large designs like processors, this method becomes extremely or even prohibitively time consuming. Especially in cases of applying heuristic or stochastic methods for design optimization, such as genetic optimization [4], a large amount of path delays have to be known in the matter of seconds while considering ΔV_{th} of each gate on the paths. Hence, a fast and sufficiently accurate method to estimate the aging-induced characteristic variations is preferred.

In this paper, we propose a lookup-table-based (LUT-based) aging-aware delay estimation technique composed of two main elements: signal probability propagation (SPP) and LUT interpolation. The delay of each individual gate is calculated based on various conditions of the particular gate, and the path delays are the sum of these individual delays. Although our method shares similarity with static timing analysis (STA), the essential difference is that STA, in its current manner, cannot handle device degradation with accuracy comparable to SPICE simulation. Thus, the key contribution of this paper is offering the same level of accuracy and speed as STA, with aging-induced delay degradation considered through the signal probability calculations. After conducting actual experiments on real-world processors, the accuracy of our aging-aware delay calculation is obtained through comparison with SPICE simulation, and a detailed error analysis is conducted.

2. Related Works

2.1 NBTI Model

To predict NBTI-induced V_{th} degradation, NBTI measurements and mathematical models are studied in [5], [6]. Our degradation calculation is based on an analytical model in [5]. This model is based on the reaction-diffusion mechanism, and the long-term degradation is defined as the function of the stress probability α. The V_{th} degradation at a

Manuscript received September 18, 2015.
Manuscript revised January 19, 2016.

[†]The authors are with Department of Communications and Computer Engineering, School of Informatics, Kyoto University, Kyoto-shi, 606-8501 Japan.
a) E-mail: paper@easter.kuee.kyoto-u.ac.jp

DOI: 10.1587/transfun.E99.A.1400

given time t is defined as

$$|\Delta V_{\rm th}(t)| = \left(\frac{\sqrt{K_v^2 T_{\rm clk} \alpha}}{1 - \beta(t)^{1/2n}} \right)^n , \quad (1)$$

where, K_v is a function of gate-source voltage $V_{\rm gs}$, threshold voltage $V_{\rm th}$, and temperature. n is the time exponent which holds the value 1/6. $T_{\rm clk}$ is a clock period, and α expresses the stress probability of a pMOS transistor. $\beta(t)$ is a function of time, which is dependent on temperature. The stress probability α captures the effects of stress and recovery phases. In [5], [7], it is reported that $|\Delta V_{\rm th}|$ is hardly dependent on $T_{\rm clk}$ when $t > 1,000$ s. In this case, $|\Delta V_{\rm th}|$ at time t can be written as

$$|\Delta V_{\rm th}(t)| \approx \left(\frac{0.001 n^2 K_v^2 \alpha C t}{0.81 t_{\rm ox}^2 (1 - \alpha)} \right)^n , \quad (2)$$

where, $t_{\rm ox}$ is the oxide thickness, and C is a function of temperature. When $\alpha = 100\%$, $|\Delta V_{\rm th}|$ becomes infinite and the model becomes incorrect. In a similar manner to what has been shown in [7], we define an upper limit by

$$|\Delta V_{\rm th}(t)| = (K_v^2 t)^n . \quad (3)$$

Figure 1 shows the $V_{\rm th}$ degradation as a function of the stress probability α, which is calculated by using Eqs. (2) and (3). In this figure, we use Nangate 45 nm Open Cell Library [8] and assume 400 K and 10 years operation. Significant $V_{\rm th}$ degradation can be observed when α is close to 100%. To estimate the impact of this $V_{\rm th}$ shift, a general circuit simulator like SPICE can be adopted. The amount of voltage displacement, $\Delta V_{\rm th}$, is given to the circuit simulator, and through simulation, the delay can be obtained. Based on this α-$\Delta V_{\rm th}$ relation, along with the $\Delta V_{\rm th}$-Δd delay relation obtained from circuit simulators, the degradation delay Δd can be actually calculated directly from the stress probability α. Utilizing this fact, an aging-aware delay prediction method is realized in this paper.

2.2 NBTI Mitigation Techniques

As mentioned in Sect. 1, many NBTI mitigation techniques have been proposed. Along with all the techniques introduced, a dynamic NBTI mitigation method was proposed in [9]. The method tries to mitigate the NBTI degradation by replacing gates in the design circuit with special gates that reduce the NBTI stress at their fanouts. As suggested in [9], the gates were replaced in the order of fanouts, without realizing the actual impact on the worst-path delay. Hence, the paper [9] reported that a replacement, which intends to reduce the worst-path delay, in fact increases it. It would be viable to solve the problem by recalculating all path delays every time a gate is replaced in the target design. Nevertheless, fast calculation of the path delays considering the changes of $V_{\rm th}$ is non-trivial since it requires a large amount of delay libraries at different aging conditions of the gates. In case it were possible, it still takes excessive time, and stochastic optimization methods (e.g., genetic optimization) were unable to be applied due to this fact. This becomes the motivation for this paper to propose a much faster aging-aware path delay estimator.

2.3 Reliability-Aware Timing Analysis

The traditional STA library is generally a two-dimensional LUT whose inputs are the input slew to the gate, and the output capacitance of the gate. The LUT returns the delay of the gate, and the output slew that will be fed into the next gate. In this subseciton, some existing LUT-based approaches are discussed.

2.3.1 Naïve Degradation Library Approach for NBTI Prediction

Due to the inaccurate nature of STA, a trivial approach can be taken to estimate the NBTI degradation by assuming all gates degrade at a fixed rate, and create a traditional STA library biased by the presumed amount of NBTI degradation. This approach does not work, for that the amount of NBTI degradation is extremely sensitive to average workload. Figure 2 is a preliminary research we conducted by creating an STA library that assumes all signal probabilities to be the value of 0.5. From the figure, it is clear that this STA library is far from usable. Indeed, the correlation found between

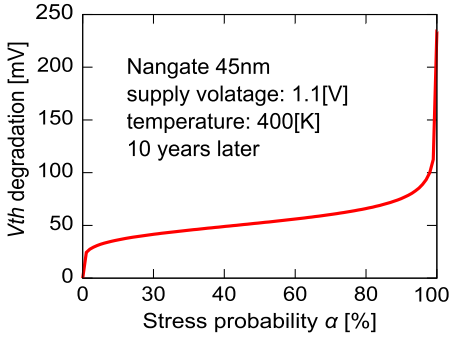

Fig. 1 $V_{\rm th}$ degradation of pMOS with stress probability in Nangate 45 nm Open Cell Library [8], 1.1 V, 400 K, and 10 years operation.

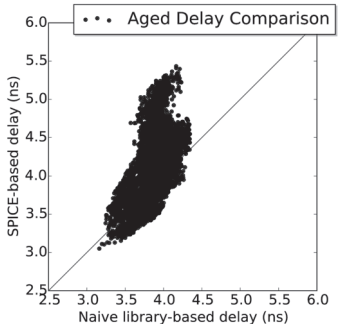

Fig. 2 The aged delay distribution of SPICE-based simulation and naïve-library-based STA.

the two delay vectors is as low as 0.439, which is even not a strong correlation statistically. Hence, it was thus concluded that a new LUT approach is inevitable in accurately predicting the NBTI-induced delay degradation.

2.3.2 LUT Approach for HCI Prediction

Similar LUT-based approach does exist for other reliability issues, as the one used in [10] for calculating the degradation induced by hot-carrier injection (HCI). In the article, the authors describe a way to calculate the HCI-induced degradation by applying the following two formulae.

$$\mathrm{AG} = \int_{\mathrm{tran}} R_{\mathrm{it}}(t)\mathrm{d}t \tag{4}$$

$$\mathrm{age} = \sum_{\text{all transistions}} AG \tag{5}$$

In Eqs. (4) and (5), R_{it} represents the rate of aging over time (i.e., rate of interface trap generation), AG is the HCI-induced degradation from a single transition, and age denotes the cumulative degradation over a certain period of time. The integration over time in Eq. (4) basically means the cumulative amount of degradation during a specific time period, which is a single transition in this case. The authors use a circuit simulator like SPICE to characterize the library and predict the degradation of the circuit over a specific period of time.

This LUT approach designed for HCI degradation prediction looks similar to our proposed method, with several subtle yet fundamental differences. First, being as the most important difference, HCI degrades the circuit in a completely different manner from NBTI. The HCI does not have a recovery phase as NBTI, and the degradation per se only happens during a signal transition. These two important degradation characteristics enabled the authors in [10] to accurately calculate the cumulative HCI-induced delay degradation analytically based on information provided by a two-dimensional LUT that is virtually equivalent to a conventional one.

While this efficient approach works for HCI, it is not the case for NBTI. The amount of degradation induced by NBTI over a certain period of time is not a simple sum over a number of transitions, and the probability-ΔV_{th} relationship is non-linear. This is the reason why we proposed a new LUT approach to the NBTI problem: the amount of degradation induced on a particular pMOS transistor cannot be solved easily without significantly slowing down the calculation process, to an extent that will eventually become unacceptable as the circuit scales. Second, transition probability is different from NBTI stress probability described in 2.1. While the former focuses on the probability of *the transition time being at a particular value at each pin,* the later requires the knowledge of the *average workload at each pin.* Over a clock period, the transition time of a gate tends to occupy only a tiny portion of the period as all gates along the combinational path need to finish their transition in order to meet the timing constraint. The authors of [10] did not systematically examine the impact of their data binning method that approximates the transition probability calculation by reducing the number of transition time values to consider. In contrast, NBTI forces the pMOS to be biased over the entire clock period in the worst-case scenario. Furthermore, as Fig. 1 indicates, the number of ΔV_{th} values (the ΔV_{th} value here is similar to the transition time in the HCI case) to consider needs to be rigorously tested. Otherwise, the error grows unacceptable due to the non-linear relationship between probability and ΔV_{th}. In short, the intrinsic nature of NBTI requires a different LUT approach from the one used in the article [10], and while the new approach could potentially also be used to predict HCI-induced degradation, it is not the case for the other way around.

3. LUT-Based Aging-Aware Delay Calculation

In this section, the basic elements required for performing the proposed aging-aware timing analysis is discussed in detail. As illustrated in Fig. 3, first, the target design is synthesized into a gate-level netlist. Second, a directed acyclic graph (DAG) is formulated from the designed netlist, and SPP is performed on the DAG. After performing the SPP, each node in the DAG will be annotated with its corresponding probability value, and this probability is used by the LUT to lookup for delays. The LUT is constructed in a separate process utilizing the gate design library. Gates in the design library will be characterized using circuit simulators like SPICE, and their propagation delay is recorded into the LUT. As the LUT database and probability-annotated DAG become available, the aging-aware delay calculation is performed. Each gate will have its delay calculated from the LUT, and the path delays are calculated as a simple sum of these gate delays.

In this section, the SPP, specifically two types of SPP, will first be discussed in Sect. 3.1. Second, the aging-aware

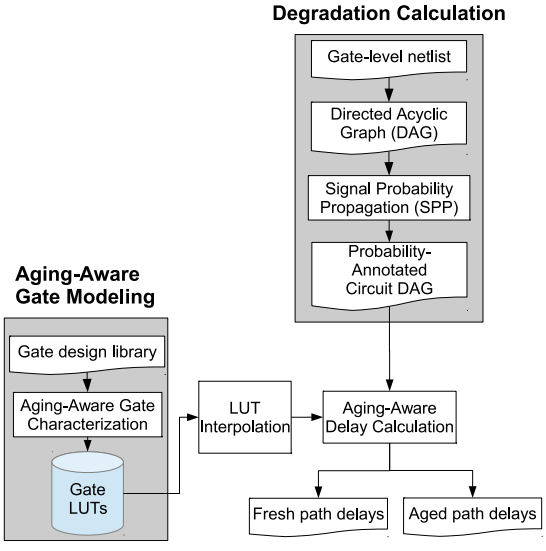

Fig. 3 Overview of the proposed LUT-based delay calculation technique.

Table 1 AND gate under all possible input combinations.

Input Combination	Output Logic	Input 1 Probability	Input 2 Probability	Output Probability
(0, 0)	0	$P(in1)$	$P(in2)$	$P(in1)P(in2)$
(0, 1)	0	$P(in1)$	$1 - P(in2)$	$P(in1) * (1 - P(in2))$
(1, 0)	0	$1 - P(in1)$	$P(in2)$	$(1 - P(in1)) * P(in2)$
(1, 1)	1	$1 - P(in1)$	$1 - P(in2)$	$(1 - P(in1)) * (1 - P(in2))$

gate modeling process is presented in Sect. 3.2. Third, on top of completing previous two steps, it is shown in Sect. 3.3 that by doing simple interpolation and an STA-like delay calculation, path delays in large designs can be calculated in seconds. Finally, in Sect. 3.4, the delay adjustment, a subtle calculation that is specific for aging-aware delay prediction, is discussed, and a fast approximation method for the delay adjustment is proposed.

3.1 Signal Probability Propagation

As mentioned above, a systematical approach to probability calculation is critical in predicting the aging-induced delay degradation. In this paper, intra-cell and inter-cell SPP are considered to be two separate problems, although they could potentially be solved together. Intra-cell signal SPP denotes each pMOS within a logical cell (gate) with the NBTI-stress probability, where the NBTI-stress probability is defined as the probability of the pMOS having its gate and source voltages negatively biased. Whereas, inter-cell SPP focuses on how signal propagates through the combinational logics in a processor.

For intra-cell SPP, we annotate cells for their actual degradation on the pMOS devices. Each input is considered separately in terms of their probability, such that each pMOS has its input annotated relatively to the input signal(s). The NBTI-induced V_{th} degradation is then calculated based on the stress probability on its gate.

Inter-cell SPP is based on the DAG constructed from the gate-level netlist of the target design. To estimate the stress probability for each gate in the design, all the primary inputs applied to the circuit is fixed to follow a certain probability distribution. By constructing a topologically sorted list from the primary inputs, the probability propagates from primary inputs to each and every piece of logic in the circuit, reaching the primary outputs eventually. To make sure that probabilities propagate correctly, not only the probability mapping of a single cell has to be considered, loops in the processor circuit (e.g., forwarding logics) have as well to be handled.

To calculate the SPP along a path, an approach shown in Fig. 4 is taken. Each gate is viewed as a function (f, g) that maps all its input probabilities (α) to its output probability $(f(\alpha))$. Depending on the type of the gate, this function changes. The procedure to generate this probability mapping function is as follows. First, a table for a particular gate, AND gate in this example, is constructed as shown in Table 1. Second, from Table 1, it can be observed that the probability of outputting a logic 0 from an AND gate is the sum of the first three entries in the table. To simplify the formula, this sum can also be thought of 1 subtracting the probability of the AND gate outputting a logic 1, which becomes Eq. (6). Hence, lastly, Eq. (6) becomes the function that maps its input probabilities to the output probability.

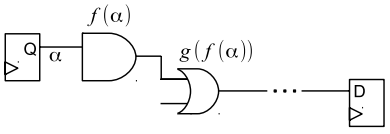

Fig. 4 DFF-to-DFF probability propagation through combinational logics.

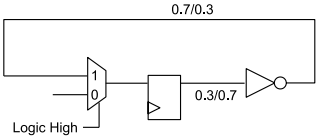

Fig. 5 Inconsistent D-flipflop with oscillating probability propagation.

$$P(out) = 1 - (1 - P(in1)) * (1 - P(in2)) \tag{6}$$

By repeating the above process for each gate, a probability will start to propagate from the Q output of a D-flipflop (DFF), going through each and every gate on the path, reaching the D input of another DFF, as shown in Fig. 4. For combinational logics, this marks the end of the process. However, due to the existence of sequential logics, i.e., DFFs, combinational logics *loop* in the circuit. These logics loop in two ways: non-oscillating and oscillating. An oscillating path example is denoted in Fig. 5. It can be observed that when the DFF's output probability is 0.3, the inverter inverts it to 0.7 and feeds it back to the DFF. When the output probability of the DFF goes to 0.7, the inverter returns 0.3 to the input of the DFF, which never ends. Whereas, a non-oscillating loop converges to a stable value after looping through the path several times.

To resolve this looping of combinational logics, whether oscillating or non-oscillating, an algorithm as in Alg. 1 is used. The algorithm requires only the DAG of the target design, and will first gather a complete list of DFFs in the design. Here, a DFF with its input probability not equal to its output probability is called an *inconsistent* DFF. Furthermore, note that although the circuit DAG contains DFFs, it is not cyclic, for that each node in the graph is a pin of a particular gate (or DFF), and that the D input pin (D) of the DFF is not connected to the Q output of the DFF. Hence,

Algorithm 1 Algorithm for solving looping combinational probability propagations.

```
Require: G = Directed Acyclic Graph of target design
 1: all_DFFs = get all DFFs from G
 2: do
 3:   for each DFF ∈ all_DFFs do
 4:     if P(DFF_out)! = P(DFF_in) &
 5:       DFF is not marked as consistent then
 6:       check DFF oscillation
 7:       if DFF is oscillating then
 8:         Choose the worst NBTI scheme
 9:         Mark DFF as consistent
10:       else
11:         incons_dff_list.append(DFF)
12:       end if
13:     end if
14:   end for
15:   for each DFF ∈ incons_dff_list do
16:     DFF.output probability = DFF.input probability
17:   end for
18:   top_list = make topological list(incons_dff_list)
19:   for all gate in top_list do
20:     gate_output_probability=gate_function(
21:                                 gate_input_probabilities)
22:   end for
23: while (incons_dff_list not empty)
```

there is no cycle in the graph. The algorithm keeps loop until the list of the inconsistent DFF converges, i.e., when the list becomes empty, as indicated by line 2∼16 in Alg. 1. Within the loop, at first, every DFF has its input-output probability consistency checked (line 4∼14). An additional *consistent* flag is added to the DFF node, such that oscillating DFFs can be marked to avoid infinite loops. The oscillation will be checked based on a history of the absolute value of the difference between the input and output probabilities. If oscillating DFFs are found, they are marked as consistent DFFs to ensure loop termination, and the output probability of the DFF is fixed to a particular value that maximizes the amount of NBTI degradation on the gate immediately following the DFF (i.e., the highest probability value). Next, for other DFFs with inconsistent input-output probabilities, the input probabilities will be applied to their outputs, and another round of probability propagation starts (line 15∼22). This procedure is repeated until the probability converges for all non-oscillating loops, as mentioned before.

3.2 Aging-Aware Gate Modeling

Gates are modeled under three conditions that vary from case to case: temperature, years of aging and input vector probability distribution. Different input pins as well as rise/-fall transitions are also considered differently. In addition, since the input/output slew also propagates through combinational paths, two collections of LUTs have to be made: one for delay, another for slew.

Each collection of LUTs consists of many LUTs, where a LUT is specially designed for a particular input pin at rise, or fall, for a particular gate. For each LUT in this case, it will have three input variables: input slew, capacitive load, and NBTI stress probability. The gate delay and output slew are first measured after being simulated under different conditions by varying the three variables defined above. The gate delay will then be gathered into one collection of LUTs, while the output slews are put into another. Finally, two three-dimensional LUTs of the simulated delays and output slews are constructed.

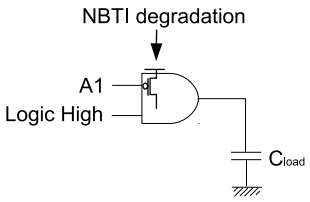

Fig. 6 Example of general simulation circuit setup for LUT construction.

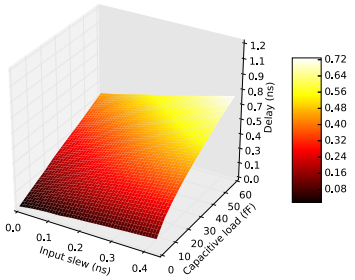

Fig. 7 Example propagation delay LUT simulated with inverter x1 falling input where probability = 0.

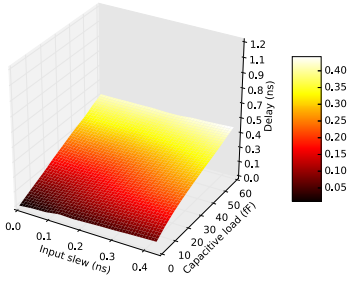

Fig. 8 Example propagation slew LUT simulated with inverter x1 falling input where probability = 0.

The basic circuit schematic for measuring of a gate delay of the A1 input (rising) of an AND gate is shown in Fig. 6. As mentioned, the input slew, capacitive load, and NBTI stress probability are the three variables given to the circuit, and two target values are measured: the propagation delay and the output slew. From the measured values, two tables are constructed: the delay LUT and the slew LUT as shown in Figs. 7 and 8.

One important factor to consider in the process of simulation is the interval into which each variable (slew, capacitive load, and NBTI stress probability) is divided. Con-

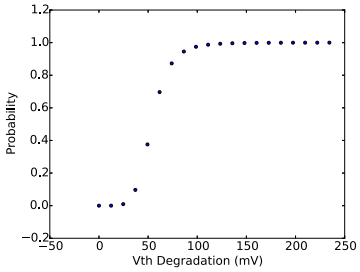

Fig. 9 Inverse of the probability-Vth function divided into 20 intervals.

ventional STA library models slew and load in a simple exponential manner, while such scheme cannot be applied to probability. As previously described in Fig. 1, the V_{th} degradation is a strongly nonlinear function of the stress probability. Thus, in this paper, we propose to use an inverse-function-based step size in determining the interval in the probability dimension. In this approach, we use the approximated inverse function derived from Eqs. (1) and (3) to calculate the probability values, and use these probability values as LUT intervals. To be clear, we did not develop a closed-form analytical equation for this inverse function. Instead, an itervative numerical method is taken towards approximating the inverse function. At each ΔV_{th} step $|\Delta V_{\text{th}}|_i$, a funciton $f(x)$ is established as Eqs. (7) and (8) indicate.

$$g(\alpha) = \left(\frac{0.001n^2K_v^2\alpha Ct}{0.81t_{\text{ox}}^2(1-\alpha)}\right)^n \tag{7}$$

$$f(x) = g(x) - |\Delta V_{\text{th}}|_i \tag{8}$$

The function of using the probability α to obtain the $|\Delta V_{\text{th}}|$ shift is renamed as $g(\alpha)$ which is virtually equivalent to Eq. (1). By setting $f(x) = 0$ and finding the root, temporarily referred to as x_{root}, we can obtain the following equation.

$$g(x_{\text{root}}) = |\Delta V_{\text{th}}|_i \tag{9}$$

In Eq. (9), it is easy to see that x_{root} is the value that could help us obtain the particular $|\Delta V_{\text{th}}|_i$ value we want to know. An example of utilizing this numerical method is illustrated in Fig. 9. We divide ΔV_{th} evenly from 0 mV to the maximum possible degradation into 20 $|\Delta V_{\text{th}}|$ intervals. Then, the corresponding probability at each $|\Delta V_{\text{th}}|_i$ step is calculated using the numerical approximation. It can be observed that the majority of the points (14 of them) locate in the range where the probability is greater than 0.8. The ultimate goal, however, is not to interpolate $|\Delta V_{\text{th}}|$, rather the delay of the gate. Using a linear interpolation method, the number of points will be minimized if the delay increases linearly at each step. If we take a look at the V_{th}-delay relation, we find Eq. (10) as a well-known equation that describes the relationship between delay and V_{th} [11].

$$\text{Delay} \propto \frac{CV_{\text{DD}}}{(V_{\text{DD}} - V_{\text{th}})^{\alpha}}, \tag{10}$$

where C is the load capacitance, and V_{DD} is the supply voltage, which are constants that do not depend on V_{th}. It is also mentioned that α holds a value of 1.3 for short channel device [11]. Although this equation exhibits non-linearity, it can be well approximated by the following equation if the $|\Delta V_{\text{th}}|$ shift from the real V_{th} value, V_{th0}, is small.

$$\text{Delay} = D_0 + \frac{\partial}{\partial V_{\text{th}}}\frac{CV_{\text{DD}}}{(V_{\text{DD}} - V_{\text{th}})^{\alpha}}\bigg|_{V_{\text{th}}=V_{\text{th0}}} \cdot |\Delta V_{\text{th}}| \tag{11}$$

$$= D_0 + \frac{CV_{\text{DD}}\alpha}{(V_{\text{DD}} - V_{\text{th0}})^{\alpha-1}} \cdot |\Delta V_{\text{th}}|, \tag{12}$$

where D_0 is the delay calculated using zero-biased V_{th0} value. Thus, despite the fact that evenly dividing $|\Delta V_{\text{th}}|$ may not result in an optimal delay-division grid, we can say that within certain range, delay is roughly linearly dependent on $|\Delta V_{\text{th}}|$. Thus, by dividing $|\Delta V_{\text{th}}|$ into a linear grid, we ensure that the number of grid points for the third dimension of the LUT is as minimized as we can possibly achieve. Moreover, note that the generation of the aging-aware LUT is only a one-time operation for a specific logic-gate library of a particular process. Hence the aging-aware LUT can be provided as a pre-characterized library just like an existing delay library in general physical design kits.

3.3 LUT-Based Delay Calculation

After calculating the SPP, gate delays along paths are calculated by interpolating the aging-aware LUTs, and path delays are obtained.

Interpolation is an important part of the technique. For a conventional two-dimensional STA library, a simplified linear delay model is developed and generally accepted [12].

$$D = D_0 + D_1S + D_2C, \tag{13}$$

where D denotes the delay of the gate of interest, and D_0, D_1, D_2 are constants. S in the equation means input slew time, and C marks the load capacitance at the output of the gate. As noted in [12], this linear model is already inaccurate for some range of slew time and capacitance for submicron technology processes. Thus, to perfectly model the relationship, it may require a more complex multidimensional interpolation (more than three) with consideration on the physical relationships between voltages and currents. Delay models for slew time and capactiance are well-established [12], [13]. However, for the third dimension of our proposed LUT (the aging dimension), owing to the fact that aging-aware LUT utilizes an inverse-function-based step size, we can simply use a linear interpolation method. Along with gate delays, as mentioned, output slew of a gate also propagates with respect to the input slew of the gate and the capacitive load of the gate. Thus, the output slew table created in the previous step is also interpolated.

After deciding the interpolation scheme for LUTs, path delay calculation is conducted in the target circuit. First of all, capacitances are extracted from the target design. Next, the slew LUT is used to calculate the slews for each gate along the paths depending on their load capacitances. At this

point, the three elements needed for a delay-LUT interpolation are all known, and a LUT interpolation is conducted, returning the propagation delay of the gate of interest. Finally, by summing up the gate delays along each path, the delay can be estimated efficiently along paths.

3.4 Delay Adjustment for Multi-Input Gates

Errors coming from interpolation and slew waveform are inevitable in LUT-based delay calculation systems. However, aging-aware delay calculation has a new and distinct source of error. This type of error comes in the form of significant underestimation observed in logic gates that contains series connection of pMOS transistors. This is caused by the fact that in SPICE simulation, a gate like NOR receives multiple probability annotation on each of its inputs. One of the pin will experience a transition, while the other pins (the propagating pins) are supposed to have no effect on the propagation delay in this scenario. Nevertheless, the NBTI effect induces a V_{th} shift on the propagating pins, which essentially increases the equivalent impedance of the pMOS transistor associated with the pin. However, in creating the LUT, the V_{th} degradation is only annotated on one of the pMOS, leaving the other one "ideal", in the sense that no degradation occurs.

With this observation, it is desirable to use the probability combination of all input pins to index the LUT. However, this would greatly increase the size of the table. Thus, instead, we propose to consider each switching input as a table index at a time, and to compensate the effects of the degradation on the other inputs, i.e., the delay increase due to non-zero degradation of the propagating input, in the process of path delay calculation.

In the above approach, while the effort in making a LUT can be greatly reduced, a simple yet effective compensation technique is required to keep the errors minimal. The method we propose uses a linear approximation towards the V_{th} degradation on the propagating pMOS, and adds a scaled version of the delay of the gate to itself. The analytical equation can be found in Eq. (14), where k is a gate-dependent scaling factor, and ΔV_{th} is the V_{th} degradation of the other pMOS in series with the current pMOS of interest.

$$\text{adjusted delay} = \text{LUT delay} * (1 + k * \Delta V_{\text{th}}) \tag{14}$$

This simple analytical form of the equation allows fast calculations, and by adjusting the k-factor, the technique prevents overly optimistic underestimation at the cost of a small amount of overestimation.

4. Numerical Experiment

4.1 Experiment Setup

Numerical experiments are conducted on a five-stage pipelined processor from a commercial IP library [14]. In the experiment, Nangate 45 nm Open Cell Library [8] is used for circuit implementation. The processor is synthesized using a logic-synthesis tool [15]. The paths with top 8% slack (a total of 25,446 paths) are extracted from the processor by a commercial STA tool [16], whose delays (aged as well as fresh) are calculated based on the proposed LUT and a SPICE simulator [17]. This is the previously mentioned predetermined "critical paths" that are vulnerable to NBTI degradation.

To generate the LUT, input slew and capacitive load are modeled in the range of [0.003ns, 470ns] and [0.01fF, 60fF], respectively. Both input slew and capacitive load are chopped into 11 evenly distributed intervals, and the delay values at the endpoints are simulated. In terms of NBTI stress probability, in this experiment, we divided ΔV_{th} into 100 evenly distributed intervals, and used the previously mentioned inverse function to calculate the actual probability intervals. As for the interpolation, a three-dimensional interpolation library was used [18]. The experiment was conducted on Intel(R) Xeon(R) E5-2630 using a single core.

4.2 Experiment Result

Figure 7 shown previously in Sect. 3.2 demonstrates the evaluation over a dense mesh after the interpolation for a X1 inverter without any NBTI stress. As a general trend, increases in input slew and capacitive load cause the increase in delay. However, for input slew, a log-like curve is derived. This can be explained by the fact that the input slew is becoming too large (> 200 ps) while inverter gates typically have a propagation delay in the range of 30–80 ps. As a result, the transition happens as soon as the input voltage reaches the switching point, giving a saturating delay increase. This is also indicated in Fig. 8 that the input slew to the gate does not have strong effects on the output slew of the gate, but is mainly dominated by the capacitive load driven by the gate.

The runtime and error information are summarized in Table 2. Note that we used SPICE as a reference design, so that the SPICE is considered to be 100% accurate in this case. Compared to SPICE simulation that takes more than 960 minutes to calculate all the path delays, our technique accomplishes the same task within 14 seconds, giving an approximate speedup of 4114x. In addition, our method was implemented in pure Python language without extensive optimization. Hence, further speedup is expected if the programs are written in C or Fortran. The maximum error is found to be +10.5%, −4.5% (actually +9%, −12% without error smoothing, which will be discussed in Sect. 4.3).

Fresh and aged path delay distributions are presented in Fig. 10 and Fig. 12, respectively, with their error histogram plotted in Fig. 11 and Fig. 13. Points are mostly distributed

Table 2 Result of proposed technique compared to SPICE.

	Proposed	SPICE
Calc. Time	14 sec.	960 min.
Abs. Max. Error (ns)	7%	-

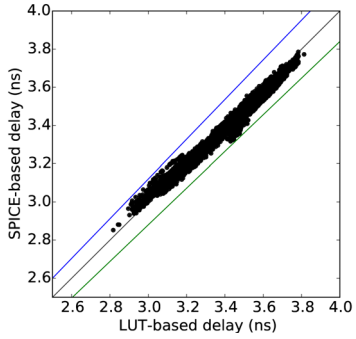

Fig. 10 Fresh delay distribution of SPICE-based simulation and LUT-based interpolation with ±4% error bar.

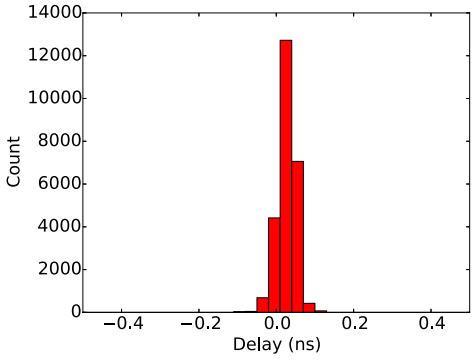

Fig. 11 Distribution error between SPICE-based path delays and LUT-based path delays for fresh processor.

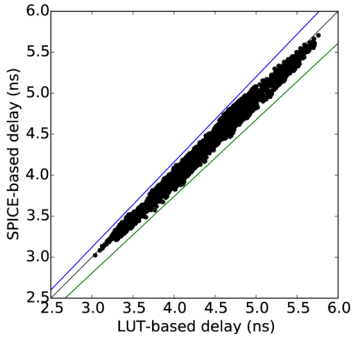

Fig. 12 Aged delay distribution of SPICE-based simulation and LUT-based interpolation with +5.6% and −4% error.

over the diagonal line. However, especially in the aged path delay calculations, a general trend of underestimation is observed. This can be explained by the fact that log-like curve will always suffer from underestimation when the interpolation method is linear. Thus, certain compensation should be added to the interpolation algorithm. The correlations between the LUT-based path delay estimation and SPICE simulation are 0.988 and 0.989 for fresh and aged, respectively. The correlation is retained at the same level, indicating that

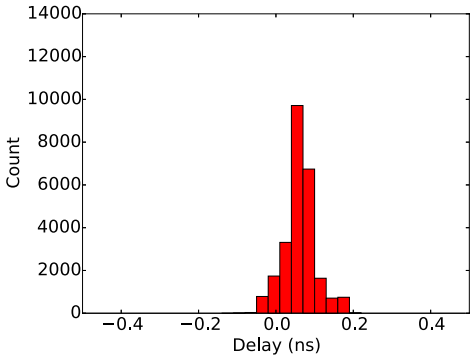

Fig. 13 Distribution error between SPICE-based path delays and LUT-based path delays for aged processor.

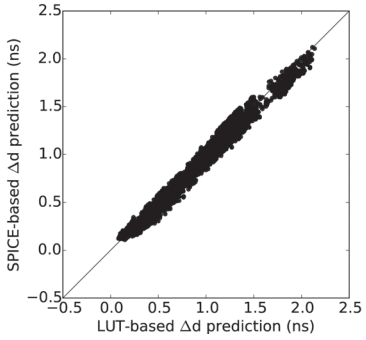

Fig. 14 Distribution of SPICE-based Δd prediction and that calculated from the LUT-based method.

only neglectable errors are introduced in adding the NBTI-stress dimension.

The accuracy of prediction on Δd, the NBTI degradation, is also studied. In Fig. 14, it can be observed that the prediction accuracy has a trend of being underestimated in general. With some paths having extra-large overestimations, as in the path delay prediction. This distribution reveals the fact that quite a huge amount of error was introduced in the process of adding NBTI degradation into consideration.

4.3 Experiment Result of Delay Adjustment

In analyzing the effect of our delay adjustment technique, slew calculation and gate-delay prediction were considered separately. To focus on the gate-delay calculation, the input and output slews for each gate are taken directly from the result of SPICE simulation. We thus obtained the result indicated in Fig. 15, where the proposed delay adjustment was not applied, and Fig. 16, where the proposed technique was applied. As the two figures reveal, the underestimation is greatly reduced, without noticeable increase in overestimation. The maximum error of −3.5% (reduced from −10%), with the small amount of overestimation degradation (from +5% to +5.6%). It is noted that delay adjustment was ap-

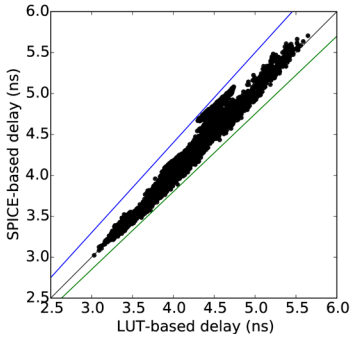

Fig. 15 Distribution of SPICE-based path delays and LUT-based path delays for aged processor without slew error and error smoothing (annotated with −10% and +5% error bar).

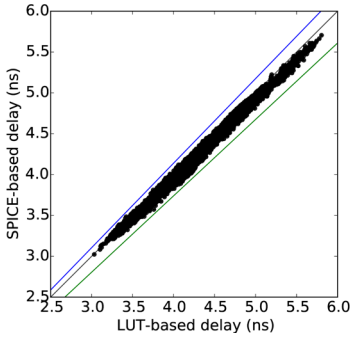

Fig. 16 Distribution of SPICE-based path delays and LUT-based path delays for aged processor without slew error, with error smoothing (annotated with −3.5% and +5.6% error bar).

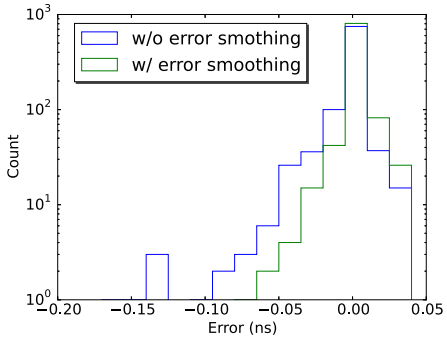

Fig. 17 Error comparison for NOR gates with and without our error-smoothing technique.

plied in obtaining the result in Sect. 4.2.

The errors in each and every NOR gates in the circuit is extracted before and after using the proposed error-smoothing technique, and the comparison is demonstrated in Fig. 17. It is clear that large underestimations are eliminated, with overestimation increased by a tiny amount. It is believed that with a more sophisticated error-smoothing technique, the result can be further improved.

5. Conclusion

In this paper, we proposed a fast and efficient method for predicting the NBTI-induced delay degradation in large designs like processors. The method utilizes signal probability propagation, lookup tables, and static timing analysis to calculate aging-aware path delays. The proposed technique successfully established an systematical approach to accurately calculate the NBTI-induced delay degradation from signal probability. Moreover, through numerical experiment on an example five-stage RISC processor, we demonstrated that this calculation can be done within seconds for relatively large designs. It is shown that using our method, the calculation of delays of 25,446 paths achieved a 4114x speedup when compared to SPICE, with a maximum error of 5.6%.

Acknowledgment

This work was partially supported by MEXT/JSPS KAKENHI Grant No. 26280014, 15K15960. The authors also acknowledge support from VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc.

References

[1] D.R. Bild, R.P. Dick, and G.E. Bok, "Static NBTI reduction using internal node control," ACM Trans. Des. Autom. Electron. Syst., vol.17, no.4, pp.1–30, 2012.
[2] A. Abdollahi, F. Fallah, and M. Pedram, "Leakage current reduction in CMOS VLSI circuits by input vector control," IEEE Trans. Very Lagre Scale Integr. (VLSI) Syst., vol.12, no.2, pp.140–154, 2004.
[3] M. Ebrahimi, F. Oboril, S. Kiamehr, and M.B. Tahoori, "Aging-aware logic synthesis," Proc. 2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp.61–68, 2013.
[4] F.A. Fortin, F.M. De Rainville, M.A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," J. Machine Learning Research, vol.13, pp.2171–2175, 2012.
[5] S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, and S. Vrudhula, "Predictive modeling of the NBTI effect for reliable design," Proc. IEEE Custom Integrated Circuits Conference 2006, pp.189–192, 2006.
[6] H. Awano, M. Hiromoto, and T. Sato, "BTIarray: A time-overlapping transistor array for efficient statistical characterization of bias temperature instability," IEEE Trans. Device Matel. Rel., vol.14, no.3, pp.833–843, 2014.
[7] H. Konoura, Y. Mitsuyama, M. Hashimoto, and T. Onoye, "Comparative study on delay degrading estimation due to NBTI with circuit/instance/transistor-level stress probability consideration," Proc. 2010 11th International Symposium on Quality Electronic Design (ISQED), pp.646–651, 2010.
[8] "Nangate 45 nm Open Cell Library," http://www.si2.org. [Online available].
[9] S. Bian, M. Shintani, Z. Wang, M. Hiromoto, A. Chattopadhyay, and T. Sato, "A processor-level NBTI mitigation technique of applying anti-aging gate control through instruction set architecture," IEICE Tech. Rep., VLD2014-161, March 2015.
[10] J. Fang and S.S. Sapatnekar, "Incorporating hot-carrier injection effects into timing analysis for large circuits," IEEE Trans. Very Lagre Scale Integr. (VLSI) Syst., vol.22, no.12, pp.2738–2751, Dec. 2014.

[11] K. Kanda, K. Nose, H. Kawaguchi, and T. Sakurai, "Design impact of positive temperature dependence on drain current in sub-1-V CMOS VLSIs," IEEE J. Solid-State Circuits, vol.36, no.10, pp.1559–1564, Oct. 2001.
[12] J. Bhasker and R. Chadha, Static timing analysis for nanometer designs: A practical approach, Springer Science & Business Media, 2009.
[13] S. Bhardwaj, P. Ghanta, and S. Vrudhula, "A framework for statistical timing analysis using non-linear delay and slew models," Proc. 2006 IEEE/ACM International Conference on Computer-Aided Design, ICCAD'06, pp.225–230, 2006.
[14] Synopsys, Inc., Processor Designer User Guide Version G-2012.09, 2012.
[15] Synopsys, Inc., Design Compiler User Guide Version D-2010.06, 2010.
[16] Synopsys, Inc., PrimeTime Fundamental User Guide Version D-2010.06, 2010.
[17] Synopsys, Inc., HSPICE User Guide: Basic Simulation and Analysis Version I-2013.12, 2013.
[18] E. Jones, T. Oliphant, and P. Peterson, "SciPy: Open source scientific tools for Python," 2001–. [Online; accessed 2015-07-08].

Song Bian received B.S. from University of Wisconsin-Madison, Wisconsin, USA in 2014. He attended Kyoto University in 2015 in pursuing a master's degree in the field of computer engineering. His main areas of interest include electric deisgn automation (EDA), processor architecture and semiconductor reliability.

Michihiro Shintani received B.E. and M.E. degrees from Hiroshima City University, Hiroshima, Japan, and a Ph.D. degree from Kyoto University, Kyoto, Japan, in 2003, 2005 and 2014, respectively. He was with Panasonic Corporation, Osaka, Japan, from 2005 to 2014, and with Semiconductor Technology Academic Research Center (STARC), Yokohama, Japan, from 2008 to 2010. In 2014, he joined the Graduate School of Informatics, Kyoto University, where he is currently a program-specific researcher. His research interests include reliability-aware LSI design and device-modeling. He is a member of IEEE and IEICE. He received the IEEE Workshop on RTL and High Level Testing 2004 Best Paper Award, IEICE VLD Excellent Student Author Award for ASP-DAC 2014 and IEEE Kansai Section Student Paper Award 2014.

Masayuki Hiromoto received B.E. degree in Electrical and Electronic Engineering and M.Sc. and Ph.D. degrees in Communications and Computer Engineering from Kyoto University in 2006, 2007, and 2009 respectively. He was a JSPS research fellow from 2009 to 2010, and with Panasonic Corp. from 2010 to 2013. In 2013, he joined the Graduate School of Informatics, Kyoto University, where he is currently an assistant professor. His research interests include VLSI design methodology, image processing and pattern recognition. He is a member of IEEE and IPSJ.

Takashi Sato received B.E. and M.E. degrees from Waseda University, Tokyo, Japan, and a Ph.D. degree from Kyoto University, Kyoto, Japan. He was with Hitachi, Ltd., Tokyo, Japan, from 1991 to 2003, with Renesas Technology Corp., Tokyo, Japan, from 2003 to 2006, and with the Tokyo Institute of Technology, Yokohama, Japan. In 2009, he joined the Graduate School of Informatics, Kyoto University, Kyoto, Japan, where he is currently a professor. He was a visiting industrial fellow at the University of California, Berkeley, from 1998 to 1999. His research interests include CAD for nanometer-scale LSI design, fabrication-aware design methodology, and performance optimization for variation tolerance. Dr. Sato is a member of the IEEE and the Institute of Electronics, Information and Communication Engineers (IEICE). He received the Beatrice Winner Award at ISSCC 2000 and the Best Paper Award at ISQED 2003.