MAUI: Making Aging Useful, Intentionally

Tien-Hung Tseng, Shou-Chun Li and Kai-Chiang Wu
Department of Computer Science
National Chiao Tung University, Hsinchu, Taiwan
E-mail: {eric830303.cs05g@g2.nctu.edu.tw, scli.cs02g@nctu.edu.tw and kcw@cs.nctu.edu.tw}

***Abstract*—Device aging, which causes significant loss on circuit performance and lifetime, has been a primary factor in reliability degradation of nanoscale designs. In this paper, we propose to take advantage of aging-induced clock skews (i.e., make them useful for aging tolerance) by manipulating these time-varying skews to compensate for the performance degradation of logic networks. The goal is to assign achievable/reasonable aging-induced clock skews in a circuit, such that its overall performance degradation due to aging can be minimized, that is, the lifespan can be maximized. On average, 25% aging tolerance can be achieved with insignificant design overhead. Moreover, we also apply V_{th} assignment to further mitigate the aging-induced degradation of logic networks. Averagely, 39.74% aging tolerance can be achieved when aging manipulation and V_{th} assignment are applied.**

I. INTRODUCTION

The design and manufacturing of semiconductor devices have recently experienced dramatic innovations, at the cost of downgrading reliability of nanoscale integrated circuits (ICs) and system-on-chips (SOCs). The 2013 ITRS [1] projects that the long-term reliability of sub-100nm technology nodes can reach a noteworthy order of 10^3 FITs (failures in 10^9 hours). Soft errors, thermal and aging effects are some of the major challenges driving reliability-aware IC/SOC design techniques. With the continuous scaling of transistor dimensions, device aging, which causes temporal performance degradation and potential wear-out failure, is becoming increasingly dominant for lifetime reliability concerns because of limited timing margins [2]. Therefore, the need of a verification and optimization flow considering aging effects emerges as a key factor in guaranteeing reliable and sustainable operation over a required lifespan. *Bias temperature instability* (BTI) is known for prevailing over other device aging phenomena, in terms of dependence on the scaling of nanometer technologies. BTI [3] is a MOSFET aging phenomenon that occurs when transistors are stressed under bias (positive or negative, i.e., $V_{gs} = \pm V_{dd}$) at elevated temperature. As a result of the dissociation of Si-H bonds along the Si-SiO_2 interface, BTI-induced MOSFET aging manifests itself as an increase in the threshold voltage (V_{th}) and decrease in the drive current (I_{ds}) [4], which in turn lengthen the propagation delays of logic gates/paths. Experiments on MOSFET aging [5] indicate that BTI effects grow exponentially with higher operating temperature and thinner gate oxide. If the thickness of gate oxide shrinks down to 4nm, the circuit performance can be degraded by as much as 15% after 10 years of stress and lifetime will be dominated by BTI [6]. In contrast, when the stress condition is relaxed (V_{gs} = 0), the aging mechanism can be recovered partially [7] and the threshold voltage decreases toward the nominal value by more than 75% if the recovery phase lasts sufficiently long [8]. In addition to the aging effect on the logic gates/paths of a circuit, the impact of aging on the clock network should not be ignored. Considering both "the aging of logic networks" and "the aging of clock networks" is essential since unbalanced aging of clock networks can greatly affect circuit performance by inducing clock skews, as a result of non-uniform increases in clock latency from the clock source to different terminals. Prior work on addressing such clock skews due to aging (i.e., aging-induced clock skews) mainly attempts to balance the aging effects on various clock sub-networks, such that aging-induced clock skews can be minimized [9]–[11]. However, suppressing aging-induced clock skews may be difficult and costly, especially for clock-gated designs where the rate of aging varies from one clock sub-network to another [12].

Then think about the following question: if one has to work hard on (and incur significant overhead for) minimizing aging-induced clock skews but it turns out that there still exist stub-born non-zero skews, why don't we try to take advantage of them? We do; we propose to intentionally make aging-induced clock skews useful for aging tolerance. More specifically, we compensate for the performance degradation (due to aging) of logic networks by exploring "useful" aging-induced clock skews, based on the concept of time borrowing. Note that the rate of aging depends on the stress time, defined as the amount of time during which a PMOS/NMOS transistor is stressed under negative/positive bias. For clock drivers (comprising pairs of inverters), their stress times are proportional to the duty cycle of the clock signal. The key idea of our framework is to manipulate the rates of aging on different clock branches, by changing the duty cycle of a clock waveform delivered to each of the clock branches. The proposed framework succeeds in mitigating effective aging-induced performance degradation with little design penalty.

II. RELATED WORK AND PAPER CONTRIBUTION

A. Previous Work on Aging-Aware Optimization

To deal with aging phenomena, traditional design methods adopt guard-banding by adding extra timing margins, which in practice imply over-design and may be expensive. To avoid overly conservatism, the mitigation of aging-induced performance degradation can be formulated as a timing-constrained area minimization problem with consideration of aging effects. Existing aging-aware techniques basically follow this

formulation. A novel technology mapper considering signal probabilities for NBTI was developed in [13]. On average, 10% area recovery and 12% power saving are accomplished, as compared to the most pessimistic case assuming static NBTI on all PMOS transistors in the design. The authors of [14] proposed a gate sizing algorithm based on Lagrangian relaxation. An average of 8.7% area penalty is required to ensure reliable operation for 10 years. Other methods related to gate or transistor sizing can be found in [15], [16].

Aforementioned methods focus on mitigating the aging of logic networks only. There are some work [9]–[11] addressing the aging problem of clock networks. Methodology of [9] is based on V_{th} assignment for clock buffers. Authors of [10] and [11] explore the use of alternative clock gating cells for clock-gated designs. On the other hand, two new clock gating cells were presented in [12] to balance the delay degradation of clock signal propagation, which reduces aging-induced clock skews between ungated and gated clock branches. The above work [9]–[12] aim to minimize aging-induced clock skew instead of making it useful (i.e., assign specific clock skews in the designs to mitigate the aging-induced performance degradation).

B. Paper Contribution

In this paper, we propose an optimization framework for aging tolerance. Our proposed framework manipulates the rates of aging on different clock branches, which is achieved by changing the duty cycle of a clock waveform delivered to each of the clock branches. In addition, we involve the technique of V_{th} assignment in the framework to further improve aging tolerance of design circuits. The contributions and advantages of this work are threefold:

- ***Exploration of aging-induced clock skews for aging tolerance:*** Existing work on addressing aging-induced clock skews mainly attempts to minimize the skews. This paper presents the first work on *exploring "useful" clock skews (i.e., making them useful)* for aging tolerance*.
- ***Problem formulation based on Boolean satisfiability and optimal solutions:*** The proposed formulation of making clock skew useful is transformed into a Boolean satisfiability (SAT) problem, and its optimal solution can be efficiently found by a SAT solver such as MiniSat.
- ***Low design overhead and little design modification:*** Restrained by the synthesized clock tree whose topology and structure are basically determined, our post-CTS (clock tree synthesis) framework does not involve aggressive modification and thus does not incur significant design overhead.

*We do not minimize "absolute" performance degradation by directly mitigating the aging of logic networks; instead; we minimize "effective" performance degradation by V_{th} assignment and manipulation of aging rates on clock networks. Furthermore, the aging of logic network is tolerated based on timing borrowing, as a result of newly-designated clock skews in designs, which is achieved by V_{th} assignment and aging manipulation on clock networks. Of course, one can mitigate the logic's aging itself by using existing techniques [13]–[16] before applying the proposed framework. This is however beyond the scope of this work and thus not particularly addressed in this paper.

III. MOTIVATING EXAMPLE

This section introduces an illustrative example which motivates our idea of making aging useful. Consider the circuit in Figure 1(a) where FF_X, FF_Y and FF_Z are three edge-triggered flip-flops, and L_{XY} and L_{YZ} are the corresponding in-between logic networks. Other notations to be used later are listed in Figure 1(b).

For each pair of flip-flops (e.g., FF_i and FF_j) between which there exists at least one logic path from FF_i to FF_j, the following setup-time (Equation (1)) and hold-time (Equation (2)) constraints need to be satisfied:

$$C_i + T_{cq} + D_{ij} + T_{su} < C_j + T_c \tag{1}$$

$$C_i + T_{cq} + d_{ij} < C_j + T_h \tag{2}$$

Assume that, at year 10, D_{ij} is degraded by 15%, and both T_{cq} and T_{su} increase by 20%. By using the predictive model presented in [8], [17], we can accurately derive the aging of C_i and C_j due to the regularity and predictability of a typical clock waveform of 50% duty cycle. In the process technology used (TSMC 45nm GP standard cell series), C_i and C_j are degraded by 13% under 10-year BTI, i.e., C_i and C_j become 1.13X larger.

To be aging-aware for the example circuit in Figure 1(a), we have to consider aforementioned aging factors in the setup-time constraints on L_{XY} and L_{YZ}:

$$L_{XY}: \quad \mathbf{1.13}C_X + 1.2T_{cq} + 1.15D_{XY} + 1.2T_{su} < \mathbf{1.13}C_Y + T_c \tag{3}$$

$$L_{YZ}: \quad \mathbf{1.13}C_Y + 1.2T_{cq} + 1.15D_{YZ} + 1.2T_{su} < \mathbf{1.13}C_Z + T_c \tag{4}$$

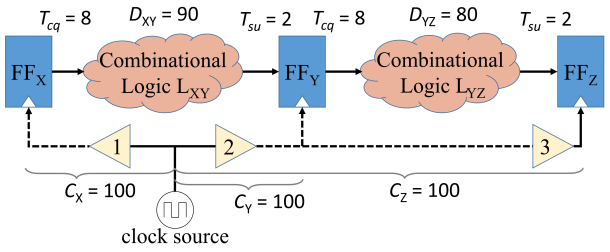
where $C_X = C_Y = C_Z = 100$, $T_{cq} = 8$, $T_{su} = 2$, $D_{XY} = 90$ and $D_{YZ} = 80$, as shown in Figure 1(a).

By re-arranging Equations (3) and (4):

$$L_{XY} : T_c > 115.5$$
$$L_{YZ} : T_c > 104$$

Therefore, the clock period needs to be larger than 115.5 (dominated by L_{XY}) to ensure no setup-time violation over a required lifespan of 10 years. For brevity, we omit the discussion on hold-time constraints, which in our work are actually formulated to ensure no existence of racing due to short paths.

To minimize required T_c under aging, we use *duty-cycle converter* (DCC) to manipulate the aging rates of clock branches. We insert one 20% DCC at the input of buffer 1, and another 80% DCC at the input of buffer 2. The 20% (80%) DCC can decrease (increase) the stress times of downstream clock buffers by converting the clock duty cycle to 20% (80%), from a typical duty cycle of 50%. Therefore, 20% DCC can mitigate/decelerate the aging of C_X and 80% DCC can aggravate/accelerate the aging of C_Y. In the case of no DCC, C_X and C_Y are degraded by 13% under 10-year BTI in TSMC 45nm GP standard cell series, while 20% and 80% DCCs will degrade C_X and C_Y by 9% and 16%, respectively, assuming



(a) Example

Symbol	Description
FF_i	Edge-triggered flip-flop, e.g., $i \in$ {X, Y, Z, ...}
L_{ij}	Corresponding in-between logic network, e.g., $i, j \in$ {X, Y, Z, ...}
T_C	Clock period
C_i	Clock arrival time to flip-flop i (FF_i)
D_{ij}/d_{ij}	Longest/shortest path delay of L_{ij}
T_{cq}	Clock-to-output delay
T_{su}/T_h	Setup/hold time

(b) Notations

Figure 1. Illustrative example and notations for the proposed framework based on DCC deployment/insertion

that the clock paths from the clock source to FF_X and FF_Y are disjoint.

Consider the new aging factors (due to effects of various clock duty cycles) in the setup-time constraints on L_{XY} and L_{YZ}:

$$L_{XY}: \quad \mathbf{1.09}C_X + 1.2T_{cq} + 1.15D_{XY} + 1.2T_{su} < \mathbf{1.16}C_Y + T_c \tag{5}$$

$$L_{YZ}: \quad \mathbf{1.16}C_Y + 1.2T_{cq} + 1.15D_{YZ} + 1.2T_{su} < \mathbf{1.16}C_Z + T_c \tag{6}$$

By re-arranging Equations (10) and (11):

$$L_{XY} : T_c > 108.5$$
$$L_{YZ} : T_c > 104$$

As it can be seen, we can reduce the required T_c from 115.5 to 108.5 (dominated by L_{XY} still), by adding two DCCs in the existing synthesized clock tree to create aging-induced clock skews. The skew for L_{XY} (between FF_X and FF_Y), quantified as $1.16C_Y$ minus $1.09C_X$, is useful/beneficial and accounts for the reduction of required T_c. A certain level of aging tolerance is thus achieved because aging-induced performance degradation of D_{XY} (plus T_{cq} and T_{su} actually) can be tolerated, by exploring such useful aging-induced clock skews.

One may note that *clock skew scheduling* (CSS) [18], which derives unequal delays for all clock branches prior to *clock tree synthesis* (CTS), can also optimize a circuit for aging tolerance. However, the optimization potential of general CSS is limited since it is difficult to precisely implement a wide range of clock delays during [19].

In contrast, post-CTS clock skew scheduling based on buffer insertion is another option. We will demonstrate that, if buffer insertion is employed to match our optimization results based on DCC insertion, the number of inserted buffers is usually much larger than the number of inserted DCCs. Also, as described later in Section IV-D, the overhead of a single DCC can be diminished by integrating a DCC with its downstream buffer, which further reveals the cost effectiveness of our proposed DCC-based framework.

A. Aging Prediction Model

Before discussing the proposed framework, we briefly introduce the aging (BTI degradation) model for logic gates/networks [8], [17] used in our paper. This model enables us to analyze the long-term behavior of BTI-induced MOSFET degradation, with both aging and recovery mechanisms taken into account. First, the degradation of threshold voltage at a given time t can be predicted as:

$$\Delta V_{th} = \left(\frac{\sqrt{K_v^2 \cdot T_{clk} \cdot \alpha}}{1 - \beta_t^{1/2n}} \right)^{2n} \tag{7}$$

where K_v is a function of temperature, electrical field, and carrier concentration, α is the stress probability, and n is the time exponential constant, 0.2 for the used technology. The detailed explanation of each parameter can be found in [8].

Next, the authors of [17] simplify this predictive model to be:

$$\Delta V_{th} = b \cdot \alpha^n \cdot t^n = b \cdot (\alpha \cdot t)^n \tag{8}$$

where $b = 3.9 \times 10^{-3} V \cdot s^{-1/5}$.

Finally, the rising/falling propagation delay of a gate through the degraded P-type/N-type MOSFET can be derived as a first-order approximation:

$$\tau_p' = \tau_p + a \cdot (\alpha \cdot t)^n \tag{9}$$

where τ_p is the intrinsic delay of the gate without BTI degradation and a is a constant.

We apply Equation (9) to calculate the delay of each gate under BTI, and further estimate the performance of a logic circuit. The coefficient a in Equation (9) for each gate type and each input pin is extracted by fitting HSPICE simulation results in 45nm, Predictive Technology Model (PTM). The simplified long-term model successfully predicts the MOSFET degradation, with less than 5% loss of accuracy against cycle-by-cycle simulations [8].

IV. Proposed Framework (MAUI) Based on Boolean Satisfiability (SAT)

The basic idea of our MAUI framework for aging tolerance is to insert *duty-cycle converters* (DCCs) in the existing clock tree, so as to intentionally create aging-induced clock skews which can compensate for the performance degradation of the logic circuit based on time borrowing. We formulate the problem using Boolean satisfiability (SAT) and thanks to the efficiency of existing SAT solvers, the optimal solution can be obtained efficiently. The end result of this formulation

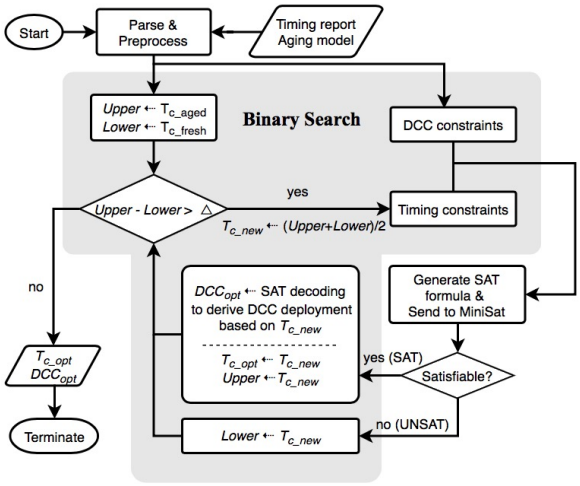


Figure 2. The overall flow of MAUI

is the locations (in the existing clock tree) to insert DCCs such that, when aging-induced clock skews are considered, the required clock period of the given circuit under n-year BTI is minimized. Note that the clock period can be minimized since the performance degradation of the logic circuit is "tolerated" as a result of useful aging-induced clock skews. The minimum required clock period thus implies maximum level of aging tolerance.

The overall flow of our MAUI framework is depicted in Figure 2, where a binary search for the minimum clock period (T_c) is involved. Formulated based on SAT, the key of our framework is to represent the problem in *conjunctive normal form* (CNF). A CNF representation is a conjunction of one or more clauses, where each clause is a disjunction of one or more Boolean variables. In the sequel, Section IV-A explains how the proposed problem of DCC deployment/insertion is encoded by Boolean variables. Section IV-B and Section IV-C describe two major components, DCC constraints and timing constraints, for our SAT-based formulation and how they are translated into legal SAT formula, i.e., CNF representation.

A. Encoding for DCC Deployment

The problem of DCC deployment/insertion needs to be encoded into Boolean representation before being transformed into a SAT-based formulation. Assume that a total of N types of DCCs can be chosen. Including the DCC-free case where no DCC is inserted, there are (N + 1) possibilities of DCC insertion for each clock buffer. Note that, when a DCC/HTV leader is inserted, it is inserted at the input of a buffer. We denote a clock buffer by $p\,(1 \le p \le P)$ where P is the number of buffers. For each buffer, Boolean variables $B_{p,q}\,(1 \le p \le P, 1 \le q \le Q = \lceil \lg(N+1) \rceil)$ are introduced where $\{B_{p,1}, B_{p,2}, \ldots, B_{p,Q}\}$ encode the aforementioned (N + 1) possibilities of DCC insertion at the input of buffer p.

Without loss of generality, we assume N = 3 (types of DCCs) and they are 20%, 40%, and 80% DCCs, as shown in Figure 3. Therefore, two Boolean variables are used for

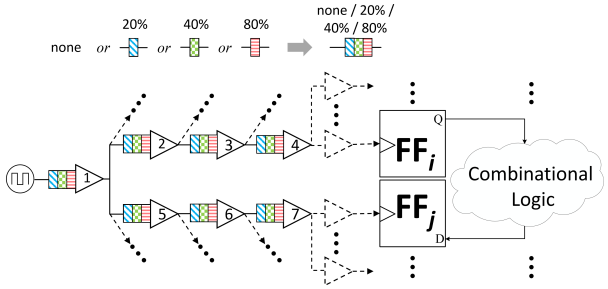


Figure 3. Generalized DCC insertion for a target pair of flip-flops

encoding four possibilities of DCC insertion at the input of any buffer. The four possibilities can be encoded as follows:

	DCC type	$\{B_{p,2}, B_{p,1}\}$
(1)	None	{0,0}
(2)	20%	{0,1}
(3)	40%	{1,0}
(4)	80%	{1,1}

For example, in Figure 4(b), a 20% DCC and an 80% DCC are inserted at buffer 3 and buffer 6, respectively. Therefore, $\{B_{3,2}, B_{3,1}\}$ = {0, 1}, $\{B_{5,2}, B_{5,1}\}$ = {1, 1}, and $\{B_{p,2}, B_{p,1}\}$ = {0, 0} for p = 1, 2, 4, 6 or 7.

The 20% DCC will mitigate the aging of buffer 3 and its downstream buffers, while the 80% DCC will aggravate the aging of buffer 6 and its downstream buffers. It can be found that, if a DCC is deployed deep in the clock tree (i.e., close to the flip-flops), the number of buffers affected is limited and thus the overall impact of aging mitigation/aggravation on C_i/C_j may be insignificant, which diminishes the benefit from deploying DCCs for aging tolerance. To avoid this phenomenon, we set a rule of prohibiting DCC deployment at a clock tree level larger/deeper than a specified boundary. This rule also greatly reduces the complexity of our SAT-based formulation because a significant fraction of buffers are excluded from being considered for DCC deployment. For example, in Figure 3, those dashed buffers and their downstream buffers are excluded.

B. DCC Constraints and Corresponding Clauses

Figure 3 shows a generalized example of DCC insertion for a pair of flip-flops (FF_i and FF_j) where there exist aging-critical paths from FF_i to FF_j. A path is defined as an aging-critical path if, in the presence of aging, it is possible to determine the clock period of the circuit. Every pair of flip-flops between which there exist aging-critical paths needs to be considered and here, we use this generalized example to illustrate our SAT-based formulation.

In Figure 3, buffers $1 - 7$ are candidate locations for DCC insertion and according to the encoding scheme explained in Section IV-A, two Boolean variables are introduced for each of the seven buffers to encode four possibilities of DCC insertion. Considering all seven buffers, there are a total of 16,384 (= 4^7) possibilities, just for this pair of flip-flops. This makes the formulation based on SAT intractable due to clause explosion. Therefore, we set up the following constraint on DCC insertion:

DCC constraint: At most one DCC on a single clock path (from the clock source to one of the flip-flops)

In order to ensure no more than one DCC on any clock path, we can use the Boolean variables introduced for DCC insertion, i.e., $B_{p,q}\,(1 \leq p \leq P, 1 \leq q \leq Q = \lceil \lg(N+1) \rceil)$, to generate some clauses which suppress the occurrence of having two DCCs on a clock path. Consider buffer 2 (encoded by $\{B_{2,2}, B_{2,1}\}$) and buffer 3 (encoded by $\{B_{3,2}, B_{3,1}\}$) in Figure 3. If there is a DCC at buffer 2 (i.e., $\{B_{2,2}, B_{2,1}\} \not\equiv \{0,0\}$), then there must no DCC at buffer 3 (i.e., $\{B_{3,2}, B_{3,1}\} \equiv \{0,0\}$), and vice versa. The constraint can be formally written as:

$$(\{B_{2,2}, B_{2,1}\} \equiv \{0,0\}) \vee (\{B_{3,2}, B_{3,1}\} \equiv \{0,0\})$$

Next, it can be translated into 4 CNF clauses:

$$(\neg B_{2,1} \vee \neg B_{3,1}) \wedge (\neg B_{2,1} \vee \neg B_{3,2}) \wedge (\neg B_{2,2} \vee \neg B_{3,1}) \wedge (\neg B_{2,2} \vee \neg B_{3,2})$$

Any pair of buffers along a single clock path should be constrained in this way. Among buffers $1-7$ in Figure 3, there are 12 pairs: $\langle 1,2\rangle$, $\langle 1,3\rangle$, $\langle 1,4\rangle$, $\langle 1,5\rangle$, $\langle 1,6\rangle$, $\langle 1,7\rangle$, $\langle 2,3\rangle$, $\langle 2,4\rangle$, $\langle 3,4\rangle$, $\langle 5,6\rangle$, $\langle 5,7\rangle$, $\langle 6,7\rangle$. Each pair translates to 4 clauses and a total of 48 clauses will be generated accordingly.

With DCC constraints and corresponding clauses, we can drastically reduce the possibilities of DCC insertion to be formulated. In the above example where 48 clauses associated with DCC constraints are generated, the number of possibilities drops from 16,384 to 103. In the next subsection, we describe what the 103 possibilities are and how they are translated into the final CNF representation.

C. Timing Constraints and Corresponding Clauses

Given a pair of flip-flops, if there exists one logic path between them, the timing (i.e. setup-time and hold-time) constraints must be met based on the inequalities (1) and (2). Different types of DCCs reveal different aging influence on clock latency. Consider the lifespan specification of 10 years in Figure 4(b): the delay of each clock buffer is changed after 10 years. The clock latency of FF_i (i.e., C_i) is the sum of clock buffer delays from clock source to FF_i:
$C_i = \tau_1 + \tau_2 + \tau_3 + \tau_4 + \cdots$ and
$C_j = \tau_1 + \tau_5 + \tau_6 + \tau_7 + \cdots$, where τ_k is the delay of buffer k.
Consider aging effects on C_i and C_j:
$C_{i\_aged} = 1.13 \times (\tau_1 + \tau_2) + 1.09 \times (\tau_3 + \tau_4 + \cdots)$ and
$C_{j\_aged} = 1.13 \times \tau_1 + 1.16 \times (\tau_5 + \tau_6 + \tau_7 + \cdots)$, where C_{i_aged} and C_{j_aged} denote aged C_i and C_j, respectively.

Next, we apply Equations (1) and (2) to check whether timing constraints will be violated under this DCC deployment.

Timing constraint: No existence of timing violation

Given one clock period T_c derived by binary search, one aging-critical path, and its associated clock network, all possible DCC deployments can be classified into 3 classes according to the number of DCCs used. Furthermore, due to the aforementioned DCC constraints, the SAT solver will only output a DCC deployment where there does not exist more than one DCC along any clock path. Thus, in the following discussion, the deployment with more than one DCC along a single clock path can be ignored. In each class, if the DCC deployment causes a timing violation within 10 years (i.e., the lifespan specification), then the deployment will be transformed into CNF clauses, such that the solver will not output the deployment as results. Here, we explain the generation of CNF clauses by using the example in Figure 4.

Class 1: No DCC is inserted on either clock path

Consider the situation that no DCC is inserted at buffers $1-7$. If it causes a timing violation along the aging-critical path within 10 years, then the Boolean representation of the deployment,

$$(\{B_{1,2}, B_{1,1}\} \equiv \{0,0\}) \wedge (\{B_{2,2}, B_{2,1}\} \equiv \{0,0\}) \wedge \cdots \wedge (\{B_{7,2}, B_{7,1}\} \equiv \{0,0\})\,,$$

equivalent to the following CNF clause:

$$(B_{1,2} \vee B_{1,1} \vee B_{2,2} \vee B_{2,1} \vee \cdots \vee B_{7,2} \vee B_{7,1})\,,$$

should be generated such that the solver will not output the corresponding deployment in the result if the CNF is satisfiable. In this case, a total of 1 CNF clause is generated.

Class 2: Inserting one DCC

This class can be further classified into 2 sub-classes based on the location of inserted DCCs.

Class 2-1: Inserting one DCC on the common clock path

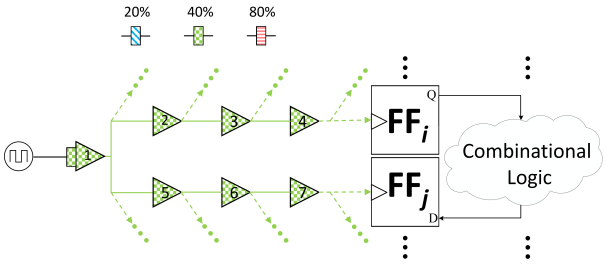
In Figure 4, buffer 1 is on the common clock path. Consider the DCC insertion shown in Figure 4(a): if the insertion of a 40% DCC at buffer 1 causes a timing violation within 10 years, then the Boolean representation of the DCC deployment, $(\{B_{1,2}, B_{1,1}\} \equiv \{1,0\}) \wedge (\{B_{2,2}, B_{2,1}\} \equiv \{0,0\}) \wedge \cdots \wedge (\{B_{7,2}, B_{7,1}\} \equiv \{0,0\})$, equivalent to the following clause: $(\neg B_{1,2} \vee B_{1,1} \vee B_{2,2} \vee B_{2,1} \vee \cdots \vee B_{7,2} \vee B_{7,1})$, should be generated such that the solver will not output the deployment in the result if the CNF is satisfiable. Given that there are 3 choices of DCCs, a total of 3 CNF clauses will be generated in the worst case.

Class 2-2: Inserting one DCC on one of the divergent clock paths

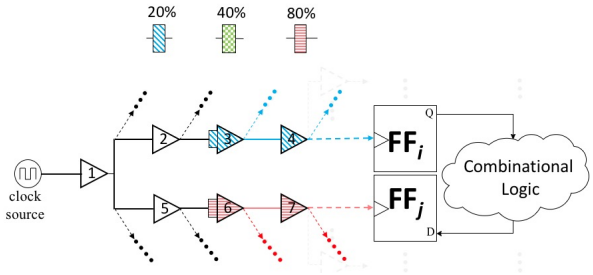
This class targets buffers 2, 3, 4, 5, 6, 7. If the insertion of a 20% DCC at buffer 3 causes a timing violation within 10 years, then the Boolean representation of the DCC deployment, $(\{B_{3,2}, B_{3,1}\} \equiv \{0,1\}) \wedge (\{B_{1,2}, B_{1,1}\} \equiv \{0,0\}) \wedge \cdots \wedge (\{B_{7,2}, B_{7,1}\} \equiv \{0,0\})$, equivalent to the following CNF clause: $(B_{3,2} \vee \neg B_{3,1} \vee B_{1,2} \vee B_{1,1} \vee B_{2,2} \vee B_{2,1} \vee B_{3,2} \cdots \vee B_{7,2} \vee B_{7,1})$, should be generated such that the solver will not output the deployment in the result if the CNF is satisfiable. This class includes 6 candidates: buffers 2, 3, 4, 5, 6, 7, and each has 3 choices of DCCs. Therefore, a total of 18 CNF clauses will be generated in the worst case.

Class 3: Inserting two DCCs on two clock paths respectively

Given the DCC deployment in Figure 4(b) (a 20% DCC inserted at buffer 3 and a 80% DCC inserted at buffer 6), if it causes a timing violation along the aging-critical path within



(a) Case 2-1: one DCC on the common clock path (e.g., at buffer 1)



(b) Case 2-2: one DCC on one of the divergent clock paths, or class 3: two DCCs, one on each of the divergent clock paths

Figure 4. Examples of DCC insertion

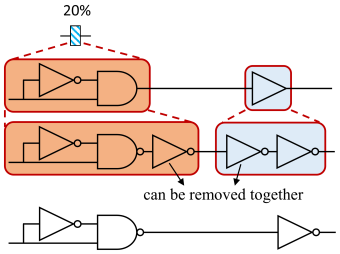


Figure 5. Practical considerations for DCC insertion

10 years, then the Boolean representation of the deployment, $(\{B_{3,2}, B_{3,1}\} \equiv \{0,1\}) \wedge (\{B_{5,2}, B_{5,1}\} \equiv \{1,1\})$, equivalent to the following CNF clause: $(B_{3,2} \vee \neg B_{3,1} \vee \neg B_{5,2} \vee \neg B_{5,1})$, should be generated such that the solver will not output the deployment in the result if the CNF is satisfiable.

Class 3 considers two buffer locations to insert DCCs, one among buffers $\{2, 3, 4\}$ and the other one among buffers $\{5, 6, 7\}$; thus, there are totally 9 combinations of buffer locations. Each combination includes two buffers and thus 9 possibilities of choosing one specific DCC for each of the two buffers. Therefore, a total of 81 CNF clauses will be generated in the worst case.

Considering all of the above cases, a maximum number of 1 + 3 + 18 + 81 = 103 clauses can be derived. This is based on the existence of 48 clauses introduced in Section IV-B.

D. Practical Considerations

The diagram on the top of Figure 5 shows the primitive design of a DCC, consisting of an inverter and an AND gate. In practice, the AND gate is implemented by a NAND gate feeding an inverter. As mentioned earlier, when inserting a DCC, it is inserted at the input of a buffer, which is a pair of inverters in practice. Therefore, we can actually use the diagram on the bottom of Figure 5 to realize the insertion of a DCC. More specifically, we use it as a new cell to "replace" a buffer when a DCC is needed. By doing so, the cost of a single DCC can be significantly reduced and not much more expensive than the cost of inserting a buffer for clock skew scheduling.

V. EXPERIMENTAL RESULTS

The proposed framework for aging tolerance is implemented in C++ and the SAT-based formulation is solved by MiniSat on a 2.83GHz Intel Quad-Core CPU workstation running Linux. The benchmark circuits are chosen from the IWLS'05 and ISCAS'89 suites. The technology used is TSMC 45nm GP standard cell series.

Under 10-year BTI, the aging rates of clock buffers were obtained from HSPICE. The aging rates of clock buffers with duty cycles of 20%, 40%, 50%, and 80% are 8.51%, 12.08%, 13.51%, and 16.41% respectively and the aging rate of logic is obtained by using the predictive model presented in [8], [17] (as in Section III-A).

Table I reports the experimental results and information of each benchmark. Columns 2 to 5 show the total number of gates, total numbers of flip-flops, total numbers of clock buffers, and maximum level of the clock tree in each benchmark respectively. Column 6 demonstrates the fresh clock period that is the circuit delay without aging, denoted by T_{c_fresh}. Column seven demonstrates the clock period of the circuit under 10-year aging, denoted by T_{c_aged}. Column eight demonstrates the optimum clock period of the circuit under 10-year aging after applying our framework, denoted by $T_{c\_aged\_opt_DCC}$. Column 9 demonstrates the used DCC count. A shorter clock period under aging implies better circuit performance and higher level of aging tolerance. Column 10 demonstrates the runtime and the last column demonstrates the improvement, i.e., the level of aging tolerance which is calculated as:

$$1 - (T_{c\_aged\_opt_DCC} - T_{c\_fresh})/(T_{c\_aged} - T_{c\_fresh})$$

For benchmark *des_perf*, T_{c_fresh} is 773.9ps and T_{c_aged} is 897.8ps, which means after 10-year aging the clock period of circuit will increase by 123.9ps. With DCC insertion using the proposed framework, the clock period achieved is 849.9ps, an increment of 76ps against T_{c_fresh} (38.66% improvement). As shown in Table I, the improvement ranges from 7.38% to 49.88% and is 25.05% on average. The number of inserted DCCs is between 2 (for benchmark *s38584*) to 17 (for benchmark *des_perf*), implying very limited degree of circuit modification and insignificant design overhead.

Table I
RESULTS OF AGING TOLERANCE

Benchmark	# Gates	# FFs	# Clock buffers	Max Tree Level	No aging/Fresh	10-year Aging				
					None	None	Aging Manipulation (DCC)			
					T_{c_fresh}	T_{c_aged}	$T_{c\_aged\_opt\_DCC}$	# DCC	Run (s)	Improve (%)
des_perf	7401	8802	3416	11	773.9	897.8	849.9	7	16.45	38.66%
leo3mp	526297	108839	4924	10	3016.7	3530.6	3458.5	1	24.89	14.03%
netcard	561091	97831	7904	12	3367.0	3940.2	3897.9	1	30.69	7.38%
vga_lcd	101496	17079	4345	10	864.2	1013.7	994.1	11	32.77	13.11%
s13207	2627	625	180	8	776.3	891.7	873.3	1	0.65	15.94%
s15850	2967	513	100	5	1011.1	1165.9	1089.9	3	0.2	49.10%
s35932	6466	1728	411	6	822.1	950.8	886.6	4	1.33	49.88%
s38417	8422	1564	376	6	798.1	933.2	909.8	15	1.46	17.32%
s38584	6861	1275	627	10	731.7	842.5	820.3	2	1.43	20.04%
AVG.					(ps)	(ps)	(ps)			25.05%

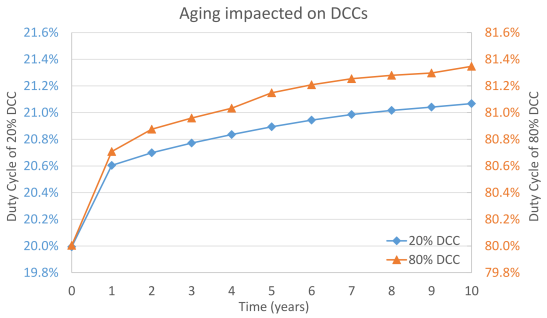


Figure 6. Aging impact on 20%/80% DCC under BTI

Figure 6 shows the change in the duty cycle of a 20%/80% DCC over 10-year aging. The y axis on the left represents the duty cycle of a 20% DCC, and the one on the right represents the duty cycle of an 80% DCC. As it can be seen, the growth in both cases are marginal: $20\% \rightarrow 21.07\%$ for a 20% DCC and $80\% \rightarrow 81.35\%$ for an 80% DCC, which in turn should not affect the benefit of our proposed framework significantly.

VI. FURTHER OPTIMIZATION BY V_{TH} ASSIGNMENT

In the section, the technique of V_{th} assignment is involved in the proposed framework to manipulate clock skew for aging tolerance. The idea comes from the aforementioned prior work [9], which exploits the technique to minimize the aging-induced clock skew, while it does not make the skew useful. Since V_{th} assignment can also manipulate the clock skew, why do not we use the technique to achieve aging tolerance? We do. The two techniques, V_{th} assignment and aging manipulation, are applied together to further optimize the required T_c. In other words, in addition to aging manipulation based on DCC insertion/deployment, we assign the threshold voltages of certain clock buffers, such that the latencies of the buffers are changed, and so are the slacks, which give us the opportunity to further mitigate the aging-induced degradation of logic network based on time borrowing. Note that, we only reassign the threshold voltages of clock buffers rather than the counterparts of logic gates.

The rest of the section is organized as follows: In Section VI-A, we use an example to demonstrate the effectiveness of V_{th} assignment for aging tolerance and in Section VI-B we introduce the rule/mechanism of V_{th} assignment, followed by VI-C, where we explain how we convert the rule/mechanism into CNF clauses, and introduces the new timing constraints while V_{th} assignment is considered. Finally, in Section VI-D, we show the experimental results, which are optimized by inserting DCC and assigning V_{th} of clock buffers.

A. Motivating Example considering V_{th} assignment

We use the illustrative example to explain how we further improve the aging tolerance by comparing the two examples of Section III and this section, based on the design in Figure 1(a). The example in Section III only considers the aging manipulation based on DCC deployment/insertion; however, in this section, the two techniques, DCC deployment and V_{th} assignment are applied together to optimize/reduce required T_c. In this example, we let the DCC deployment same with that in the former example (in Section III), i.e., 20% DCC and 80% DCC are inserted at the inputs of buffer 1 and buffer 2, respectively. Moreover, we begin assigning high V_{th} to the certain clock buffers, which are in the intervals from buffer 2 to FF_x and to FF_y. In other words, buffer 2 and its downstream buffers are assigned to high V_{th}. To include the aging rates of high-V_{th} buffers in the setup-time constraint, we assume that, the intrinsic delay of high-V_{th} buffer is 1.2X longer than that of nominal buffer (whose V_{th} are not reassigned), and aging rates of high-V_{th} buffer, with the duty cycle of 20%, 40%, 50% and 80%, are 0.5%, 4.1%, 5.4% and 8.2%, respectively. Note that, the aging rates of high-V_{th} buffer are lower than those of nominal buffer, because the higher/lower V_{th} leads to lower/higher aging rates [9]. Consider the new aging factors in the setup-time constraints on L_{XY} and L_{YZ}:

$$L_{XY}: \quad \mathbf{1.09}C_X + 1.2T_{cq} + 1.15D_{XY} + 1.2T_{su} < \mathbf{(1.2+0.08)}C_Y + T_c \tag{10}$$

$$\text{L}_{\text{YZ}}: \quad \textbf{(1.2+0.08)}C_Y + 1.2T_{cq} + 1.15D_{YZ} + 1.2T_{su} < \textbf{(1.2+0.08)}C_Z + T_c \tag{11}$$

By re-arranging Equations (10) and (11):

$$\text{L}_{\text{XY}} : T_c > 96.5$$
$$\text{L}_{\text{YZ}} : T_c > 104$$

Apparently, the required T_c is further reduced/optimized from 108.5 (in Section III) to 104 by inserting two DCCs and V_{th} assignment in the existing synthesized clock network. We also observe that, the required T_c is not dominated by L_{XY} anymore (in Section III); instead, it turns out that L_{YZ} dominates T_c. As it can be seen, when the two techniques, V_{th} assignment and aging manipulation (by DCC insertion), are applied together, the skew for L_{XY}, which equals $1.28C_Y$ minus $1.09C_X$, is larger than that in Section III. Therefore, the new skew for L_{XY} is more useful/beneficial and accounts for the better optimization of required T_c. Additionally, when it comes to the timing-borrowing mechanism of the two examples, there exists a difference: The timing-borrowing mechanism in Section III, is achieved by the aging-induced clock skew, caused by manipulating the duty-cycle delivered to flip-flops. However, in the example, the timing-borrowing mechanism is based on aging-induced clock skew and *tech-induced* clock skew, which is caused by manipulating the technology of clock buffers, i.e., assign the threshold voltages of clock buffers to specific value.

B. Technology Leader

When V_{th} assignment is considered, how to select short-listed clock buffers arises as a problem. The so-called short-listed buffers are those whose V_{th} will be assigned (i.e., their technology is changed) to a specific value. Because the possibilities of shortlisted buffers are numerous, we need to reduce the complexity of the problem. Thus, we introduce the rule of V_{th} assignment by explaining *technology leader*.

If the clock buffer is chosen as the *technology leader*, the clock buffer and its downstream buffers are regarded as shortlisted buffers. In other words, the technology of the buffer and the downstream buffers will be replaced with new counterpart. For example, in VI-A, buffer 2 is actually chosen as the technology leader, so that the shortlisted buffers are buffer 2 and its downstream buffers, which also include buffer 3. Note that, there is a difference between DCC and technology leader. DCC is not a clock buffer but a special-purpose gate, which is inserted at the input of the clock buffer, such that the aging rates of downstream buffers are manipulated; however, technology leader is an existing clock buffer. More specifically, it is selected from the buffers in the existing clock tree and indicates where we begin manipulating the technology of downstream buffers toward flip-flops.

C. Proposed Framework considering V_{th} Assignment

The proposed framework, which simultaneously considers aging manipulation (DCC) and V_{th} assignment (technology leader), is depicted in Figure 7. Compared with the former framework in Figure 2, the framework is also based on a

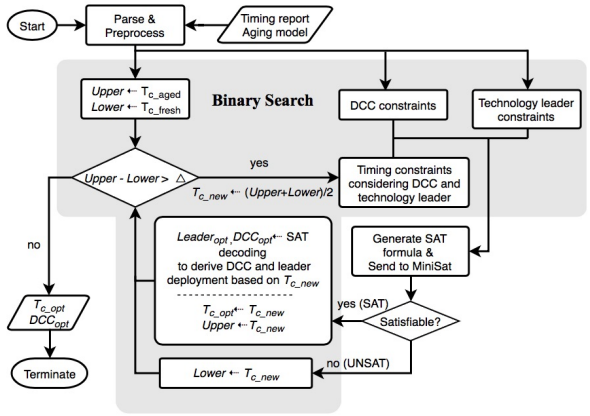


Figure 7. The overall flow of the framework when technology leaders are considered

binary search for the minimum clock period (T_c) and SAT formulation, while timing constraints need to consider the technology leader selection. The problem, technology leader selection from clock buffers, is formulated as SAT problem, and so is the problem of DCC insertion/deployment. In the sequel, Section VI-C1 explains how the problem of DCC insertion and technology leader selection is encoded by Boolean variables. Section VI-C2 reviews DCC constraints and introduces technology leader constriants. Section VI-C3 introduces the timing constraints which consider DCC deployment and technology leader selection.

1) Encoding for DCC and Technology Leader Deployment: The problem of DCC deployment and technology leader selection needs to be encoded into Boolean representation before being transformed into a SAT-based formulation. Assume that a total of N types of DCCs can be chosen. Including the DCC-free case where no DCC is inserted, there are $(N + 1)$ possibilities of DCC insertion for each clock buffer. Furthermore, we also assume that a total of M types of technology leaders can be chosen. Note that, each type of technology leader represents an individual technology. Including the nominal technology, there are $(M + 1)$ possibilities of technology leader selection for each clock buffer. We denote a clock buffer by $p\,(1 \leq p \leq P)$ where P is the total count of clock buffers and p is buffer index. For each clock buffer, there exist two types of Boolean variables, $B_{p,q}$ and $B_{p,r}$ $(1 \leq q \leq Q < r \leq R$, $Q = \lceil \lg(N + 1) \rceil$ and $R = \lceil \lg\{(N + 1)(M + 1\}\rceil)$, where $\{B_{p,1}, B_{p,2}, \ldots, B_{p,Q}\}$ encode the aforementioned $(N + 1)$ possibilities of DCC insertion at the input of buffer p and $\{B_{p,Q+1}, B_{p,Q+2}, \ldots, B_{p,R}\}$ encode the $(M + 1)$ possibilities of leader selection of buffer p.

Without loss of generality, we assume $N = 3$, and $M = 1$. Thus, there are three types of DCCs, which are assumed to be 20%, 40%, and 80% DCCs, as shown in Figure 3. In addition, there is one type of technology leader, which is assumed to be high-V_{th} leader. Note that, if the clock buffer is selected as the high-V_{th} leader, the technology of the buffer and the associated downstream ones is replaced with high-V_{th} counterpart. Since

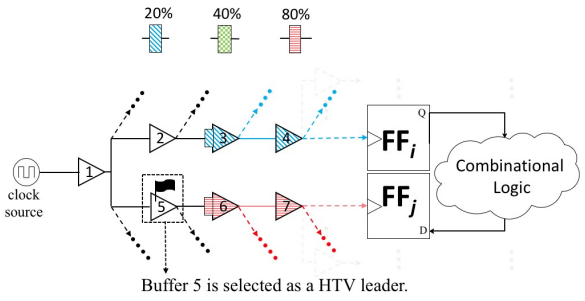


Figure 8. An example with DCC deployment and technology selection

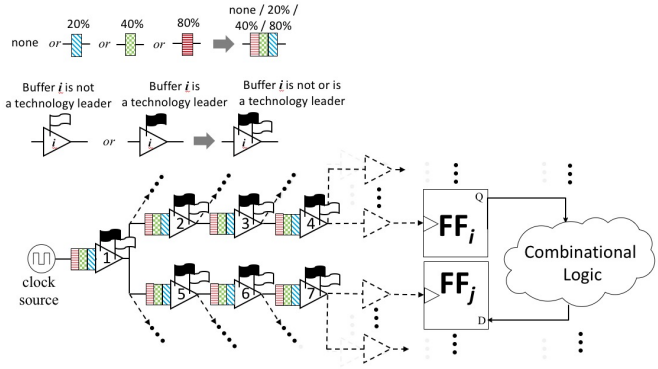


Figure 9. Generalized DCC insertion and technology leader selection for a target pair of flip-flops

we assume three types of DCC and one type of technology leader, three Boolean variables are used for encoding eight possibilities of DCC and technology leader at any buffer. The eight possibilities can be encoded as follows:

	Leader type	DCC type	$\{B_{p,3}, B_{p,2}, B_{p,1}\}$
(1)	Nominal V_{th}	None	{0, 0, 0}
(2)	Nominal V_{th}	20%	{0, 0, 1}
(3)	Nominal V_{th}	40%	{0, 1, 0}
(4)	Nominal V_{th}	80%	{0, 1, 1}
(5)	high-V_{th}	None	{1, 0, 0}
(6)	high-V_{th}	20%	{1, 0, 1}
(7)	high-V_{th}	40%	{1, 1, 0}
(8)	high-V_{th}	80%	{1, 1, 1}

For example, in Figure 8, the DCC deployment is same with that in Figure 4(b) (i.e., 20% DCC at buffer 3 and 80% DCC at buffer 6), but the buffer 5 is selected as the high-V_{th} leader, which is denoted by a black flag. Thus, the technology of buffer 5, 6 and 7 is replaced with the high-V_{th} counterpart. Therefore, $\{B_{3,3}, B_{3,2}, B_{3,1}\}$ = {0, 0, 1}, $\{B_{5,3}, B_{5,2}, B_{5,1}\}$ = {1, 0, 0}, $\{B_{6,3}, B_{6,2}, B_{6,1}\}$ = {0, 1, 1}, and $\{B_{p,3}, B_{p,2}, B_{p,1}\}$ = {0, 0, 0} for p = 1, 2, 4 or 7.

As mentioned in Section IV-A, the 20% DCC mitigates the aging of buffer 3 and its downstream buffers, but the 80% DCC aggravates the aging of buffer 6 and its downstream buffers. Moreover, since the buffer 5 is a high-V_{th} leader, buffer 5, 6 and 7 are changed as high-V_{th} buffers, implying the latency from buffer 5 to buffer 7 is lengthened. This way, the clock skew becomes larger than that in Figure 4(b), so that the required T_c can be further reduced, based on time borrowing. Note that, as we mentioned in Section VI-A, the T_c reduction in Figure 4(b) is only due to the useful aging-induced clock skew; however, the further T_c reduction in Figure 8 is both based on useful aging-induced and tech-induced clock skew, which is caused by the V_{th} assignment of clock buffers.

2) Technology Leader Constraints and DCC Constraints: Figure 9 shows a generalized example of DCC insertion and technology leader selection for a pair of flip-flops (FF_i and FF_j), where there exist aging-critical paths from FF_i to FF_j. As described in Section IV-B, a path is defined as an aging-critical path if it is possible to determine the clock period of the circuit, in the presence of aging. Each pair of flip-flops between which there exist aging-critical paths needs to be considered. Here, we use the generalized example to illustrate our SAT-based formulation. The generalized example in Figure 9 is similar to that in Figure 3. In contrast, we include technology leader selection for each clock buffer in Figure 9. Moreover, we can find that, if the clock buffer, which is selected as a technology leader, is deep in the clock tree (i.e., close to flip-flops), the count of V_{th}-reassigned buffers decreases thus the impact of tech-induced clock skew becomes insignificant. The above phenomenon is similar to that in Section IV-A, which observes that deeper DCC deployment causes less aging-induced clock skew. Thus, we set a rule of avoiding selecting/inserting leaders/DCCs at a clock tree level, which is larger/deeper than a specified boundary. The rule greatly reduce the complexity of SAT-based formulation, because a significant fraction of buffers are not considered being inserted DCC at their inputs and being selected as a technology leader. For instance, in Figure 9, dashed buffers and their downstream buffers are not considered.

In Figure 9, buffers 1 - 7 are candidate locations for DCC insertion and leader selection. According to the encoding mechanism explained in Section VI-C1, one Boolean variable is introduced to encode the two possibilities of leader selection, for each of the seven buffers. Thus, there are totally 128 (=2^7) possibilities of leader selection, only for this pair of flip-flops. In contrast, there is a total of 16,384 (= 4^7) possibilities of DCC deployment. If we combine the two techniques (i.e., DCC insertion and leader selection) together, there is totally 2,097,152 possibilities. The total count of possibilities can be obtained by multiplying the possibilities of the two techniques (i.e., $4^7 * 2^7$ = 2,097,152), or can be explained by the eight possibilities of DCC insertion and leader selection, for each of the seven buffers (i.e., 8^7 = 2,097,152). Apparently, this make SAT-based formulation tricky because of clause explosion. Therefore, we set the following constraints on DCC insertion and technology leader selection.

a) DCC Constraints: The DCC constraints are identical to those in Section IV-B. That is, at most one DCC exists along a single clock path, from clock source to one of flip-flops. The corresponding clauses can be referred to the 48 clauses in Section IV-B. It is worth reminding that, after applying DCC constraints, the total possibilities of DCC insertion can be reduced from 16,384 (= 4^7) to 103.

b) Technology Leader Constraints and Corresponding Clauses: The leader constraints are similar to DCC constraints. They are defined as follow: At most one technology leader exists along a single clock path, from clock source to one of the flip-flops. To ensure that no more than one leader along any clock path, we use the Boolean variables $B_{p,r}$ to encode leader selection of each buffer. It is worth reminding that $B_{p,r}$ ($Q < r \leq R$) or $\{B_{p,Q+1}, B_{p,Q+2}, \ldots, B_{p,R}\}$ encode the (M + 1) possibilities of leader selection of buffer p. This way, we can generate some clauses to suppress the occurrence that more than one leader along a clock path. Consider buffer 2 (encoded by $\{B_{2,3}\}$) and buffer 3 (encoded by $\{B_{3,3}\}$) in Figure 9. If buffer 2 is a leader (i.e., $\{B_{2,3}\} \not\equiv \{0\}$), then buffer 3 must not be a leader (i.e., $\{B_{3,3}\} \equiv \{0,0\}$), and vice versa. The constraint can be formally written as:

$$(\{B_{2,3}\} \equiv \{0\}) \vee (\{B_{3,3}\} \equiv \{0,0\})$$

Next, it can be translated into one CNF clause: $(\neg B_{2,3} \vee \neg B_{3,3})$.

Any pair of buffers along a single clock path should be constrained in this way. Among buffers $1-7$ in Figure 3, there are 12 pairs: $\langle 1,2\rangle$, $\langle 1,3\rangle$, $\langle 1,4\rangle$, $\langle 1,5\rangle$, $\langle 1,6\rangle$, $\langle 1,7\rangle$, $\langle 2,3\rangle$, $\langle 2,4\rangle$, $\langle 3,4\rangle$, $\langle 5,6\rangle$, $\langle 5,7\rangle$, $\langle 6,7\rangle$. Each pair translates to one clause and a total of 12 clauses will be generated accordingly.

With leader constraints and corresponding clauses, we can drastically reduce the possibilities of leader selection to be formulated. In the above example where 12 clauses associated with leader constraints are generated, when we only consider the possibilities of leader selection, the possibility count drops from 128 (= 2^7) to 17. In addition, when we consider the possibilities of DCC deployment and leader selection, the possibility count drops from 2,097,152 to 1751 due to DCC and leader constraints. In the next subsection, we describe what the 17 and 1751 possibilities are and how they are translated into final CNF representation.

3) Timing Constraints considering DCC and Technology Leader: The timing constraints, considering the two techniques, is extended from those in Section IV-C. As seen in Section IV-C, given a pair of flip-flops, between which there exists one logic path, the timing (i.e., setup-time and hold-time) constraints must be met based on the inequalities Equation (1) and (2). The former timing constraints in Section IV-C only consider the impact of DCC deployment on clock latency. Thus, the new timing constraints here must consider the impact of V_{th} assignment on clock latency.

Given one clock period T_c derived by binary search, one aging-critical logic path, and its associated clock network, the former timing constraints are classified into 3 classes, according to the used DCC count (i.e., no, one, and two DCC insertions). The new timing constraints are extended from the 3 classes, each of which are further classified into 3 subclasses, according to the leader count (i.e., the count of buffers which are selected as leaders). Thus, there are totally 9 (= $3 * 3$) subclasses based on the count of DCC and leader. For brevity, 6 of the 9 subclasses are omitted in the following discussion. We only discuss the other 3 subclasses, whose DCC deployments are identical, but leader counts vary from 0, 1 to 2. Furthermore, due to the aforementioned DCC and leader constraints, the SAT solver will only output a DCC (leader) deployment (selection) where there does not exist more than one DCC (leader) along any clock path. Thus, in the following discussion, the deployment (selection) with more than one DCC (leader) along a single clock path can be ignored. In each subclass, if the DCC (leader) deployment (selection) causes a timing violation within 10 years (i.e., the lifespan specification), then the deployment (selection) will be transformed into CNF clauses, such that the solver will not output the deployment (selection) as results. Here, we explain the generation of CNF clauses by using the example in Figure 10.

Class 1: No buffer on either clock path is selected as a technology leader

Consider the situation that, among buffers $1-7$, no buffer is selected as a technology leader. If it causes a timing violation along the aging-critical path within 10 years, then the Boolean representation of the deployment,

$$(\{B_{1,3}, B_{1,2}, B_{1,1}\} \equiv \{0,0,0\}) \wedge (\{B_{2,3}, B_{2,2}, B_{2,1}\} \equiv \{0,0,0\}) \wedge \cdots \wedge (\{B_{7,3}, B_{7,2}, B_{7,1}\} \equiv \{0,0,0\}),$$

equivalent to the following CNF clause: $(B_{1,3} \vee B_{1,2} \vee B_{1,1} \vee B_{2,3} \vee B_{2,2} \vee B_{2,1} \vee \cdots \vee B_{7,2} \vee B_{7,2} \vee B_{7,1})$, should be generated such that the solver will not output the corresponding DCC deployment and leader selection in the result if the CNF is satisfiable. In this case, a total of 1 CNF clause is generated.

Class 2: Only one buffer is selected as a technology leader

This class can be further classified into 2 sub-classes based on the buffer location of technology leader.

Class 2-1: Selected buffer (i.e., technology leader) on the common clock path.

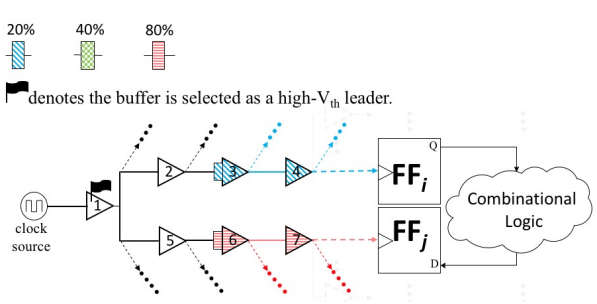
In Figure 10, buffer 1 is on the common clock path. Consider the leader selection shown in Figure 10(a): if buffer 1 is the high-V_{th} leader and the leader selection causes a timing violation within 10 years, then the Boolean representation of the technology leader selection and the DCC deployment,

$$(\{B_{1,3}, B_{1,2}, B_{1,1}\} \equiv \{1,0,0\}) \wedge (\{B_{2,3}, B_{2,2}, B_{2,1}\} \equiv \{0,0,0\}) \wedge (\{B_{3,3}, B_{3,2}, B_{2,1}\} \equiv \{0,1,0\}) \wedge (\{B_{4,3}, B_{4,2}, B_{4,1}\} \equiv \{0,0,0\}) \wedge (\{B_{5,3}, B_{5,2}, B_{5,1}\} \equiv \{0,0,0\}) \wedge (\{B_{6,3}, B_{6,2}, B_{6,1}\} \equiv \{0,1,1\}) \wedge (\{B_{7,3}, B_{7,2}, B_{7,1}\} \equiv \{0,0,0\}),$$

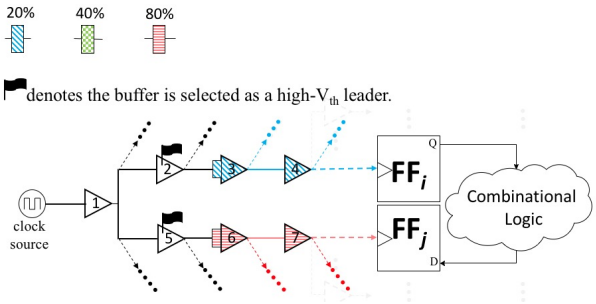
equivalent to the following clause: $(\neg B_{1,3} \vee B_{1,2} \vee B_{1,1} \vee B_{2,3} \vee B_{2,2} \vee B_{2,1} \vee B_{3,3} \vee \neg B_{3,2} \vee B_{3,1} \vee B_{4,3} \vee B_{4,2} \vee B_{4,1} \vee B_{5,3} \vee B_{5,2} \vee B_{5,1} \vee B_{6,3} \vee \neg B_{6,2} \vee B_{6,1} \vee B_{7,3} \vee B_{7,2} \vee B_{7,1})$, should be generated such that the solver will not output the leader selection and DCC deployment in the result if the CNF is satisfiable. Given that there are 1 choices of technology leader, a total of 1 CNF clause will be generated in the worst case.

Class 2-2: Selected buffer (i.e., technology leader) is on one of the divergent clock paths.

This class targets buffers 2, 3, 4, 5, 6, 7. If buffer 2 is the high-V_{th} leader, and it causes a timing violation within 10



(a) Case 2-1: One leader on the common clock path (e.g., at buffer 1), and two DCCs



(b) Case 2-2: one leader on one of the divergent clock paths, or class 3: two leaders, one on each of the divergent clock paths

Figure 10. Examples of DCC insertion

years, then the Boolean representation of the leader selection and the DCC deployment,

$$(\{B_{2,3}, B_{2,2}, B_{2,1}\} \equiv \{1,0,0\}) \wedge (\{B_{1,3}, B_{1,2}, B_{1,1}\} \equiv \{0,0,0\})$$
$$\wedge (\{B_{3,3}, B_{3,2}, B_{3,1}\} \equiv \{0,1,0\}) \wedge (\{B_{4,3}, B_{4,2}, B_{4,1}\} \equiv \{0,0,0\})$$
$$\wedge (\{B_{5,3}, B_{5,2}, B_{5,1}\} \equiv \{0,0,0\}) \wedge (\{B_{6,3}, B_{6,2}, B_{6,1}\} \equiv \{0,1,1\})$$
$$\wedge (\{B_{7,3}, B_{7,2}, B_{7,1}\} \equiv \{0,0,0\}),$$

equivalent to the following CNF clause: $(B_{1,3} \vee B_{1,2} \vee B_{1,1} \vee \neg B_{2,3} \vee B_{2,2} \vee B_{2,1} \vee B_{3,3} \vee \neg B_{3,2} \vee B_{3,1} \vee B_{4,3} \vee B_{4,2} \vee B_{4,1} \vee B_{5,3} \vee B_{5,2} \vee B_{5,1} \vee B_{6,3} \vee \neg B_{6,2} \vee \neg B_{6,1} \vee B_{7,3} \vee B_{7,2} \vee B_{7,1})$, should be generated such that the solver will not output the leader selection and DCC deployment in the result if the CNF is satisfiable. This class includes 6 candidates: buffers 2, 3, 4, 5, 6, 7, and each has 1 choice of technology leader. Therefore, a total of 6 CNF clauses will be generated in the worst case.

Class 3: Two buffers on two divergent clock paths are selected as technology leaders.

Given the DCC deployment and leader selection in Figure 10(b) (a 20% DCC inserted at buffer 3, a 80% DCC inserted at buffer 6, and buffer 2 and 5 are both high-V$_{th}$ leaders), if it causes a timing violation along the aging-critical path within 10 years, then the Boolean representation of the deployment,

$$(\{B_{1,3}, B_{1,2}, B_{1,1}\} \equiv \{0,0,0\}) \wedge (\{B_{2,3}, B_{2,2}, B_{2,1}\} \equiv \{1,0,0\})$$
$$\wedge (\{B_{3,3}, B_{3,2}, B_{3,1}\} \equiv \{0,1,0\}) \wedge (\{B_{4,3}, B_{4,2}, B_{4,1}\} \equiv \{0,0,0\})$$
$$\wedge (\{B_{5,3}, B_{5,2}, B_{5,1}\} \equiv \{1,0,0\}) \wedge (\{B_{6,3}, B_{6,2}, B_{6,1}\} \equiv \{0,1,1\})$$
$$\wedge (\{B_{7,3}, B_{7,2}, B_{7,1}\} \equiv \{0,0,0\}),$$

equivalent to the following CNF clause: $(B_{1,3} \vee B_{1,2} \vee B_{1,1} \vee \neg B_{2,3} \vee B_{2,2} \vee B_{2,1} \vee B_{3,3} \vee \neg B_{3,2} \vee B_{3,1} \vee B_{4,3} \vee B_{4,2} \vee B_{4,1} \vee \neg B_{5,3} \vee B_{5,2} \vee B_{5,1} \vee B_{6,3} \vee \neg B_{6,2} \vee \neg B_{6,1} \vee B_{7,3} \vee B_{7,2} \vee B_{7,1})$, should be generated such that the solver will not output the deployment in the result if the CNF is satisfiable.

Class 3 considers the selection of the two technology leaders, one among buffers $\{2, 3, 4\}$ and the other one among buffers $\{5, 6, 7\}$; thus, there are totally 9 (=3^2) possibilities of high-V$_{th}$ leader selection. Therefore, a total of 9 CNF clauses will be generated in the worst case.

Considering all of the above cases with the given DCC deployment, a maximum number of 1 + 1 + 6 + 9 = 17 clauses can be derived. This is based on the existence of 12 clauses introduced in Section VI-C2a. If we consider the 103 possibilities of DCC deployment in Section IV-C, a total of 1751 (=$103 * 17$) clauses can be derived in the worst case.

D. Experimental Setting and Aging Rate Model for High-V_{th} Buffer

The proposed framework, which simultaneously considers the two techniques, is implemented in C++ and SAT-based formulation is solved by MiniSat. The experimental environment and benchmark circuits are identical to the former setting in Section V. Under 10-year aging influence with BTI, the aging rates of clock buffers are obtained from HSPICE. The aging rates of nominal clock buffers with duty cycles of 20%, 40%, 50%, and 80% are 8.5%, 12.1%, 13.5%, and 16.4% respectively and the aging rate of logic is obtained by using the same predictive model.

The timing information (e.g., intrinsic delay and aging rate) of high-V$_{th}$ buffers is estimated based on the model proposed in [20], which discusses the correlation between BTI and fresh V$_{th}$ offsets among transistors. Note that, the V$_{th}$ offsets are treated as process variations in [20], while here they are regarded as the gaps between high V$_{th}$ and nominal counterpart. The correlation is a long-term phenomenon that bridge the V$_{th}$ differences among the transistors over a period. Further, a positive/negative V$_{th}$ offset leads to a higher/lower fresh V$_{th}$, causing a lower/higher aging speed. Therefore, the gap between high and low V$_{th}$ will be gradually converged, letting threshold voltages of transistors, whose fresh ones are different, reach a convergent value. Thus, we can conclude that the gap between high and nominal V$_{th}$ will be gradually converged over a specific period. The concept will be used in the latter model to derive the aging rates of high-V$_{th}$ buffer. A model in [20] is proposed to estimate the correlation between fresh V$_{th}$ offset and BTI effects:

$$\Delta V_{th\_nbti} = (1 - S_v \cdot \Delta V_{th\_offset}) \cdot A \cdot a^n \cdot t^n \qquad (12)$$

$$V_{th} = \Delta V_{th\_nbti} + \Delta V_{th\_offset} + V_{th\_design} \qquad (13)$$

ΔV_{th_offset} denotes the fresh V_{th} offset, which is the gap between high V_{th} and nominal V_{th} in the framework. ΔV_{th_nbti} denotes the BTI-induced V_{th} shift and V_{th_design} denotes the nominal threshold voltage of the design. S_v depends on ΔV_{th_offset}, and can be derived by following procedures:

a) Assume the value of V_{th} is convergent: We assume threshold voltages of all transistors will be convergent after a long period, even if their fresh values are different. In other words, V_{th} is fixed regardless of various ΔV_{th_offset}, since the aforementioned correlation takes effect.

b) Obtain the convergent value of V_{th}: Since V_{th} is fixed regardless of various ΔV_{th_offset}, we set ΔV_{th_offset} to 0 in Equation (13) to derive the convergent value of V_{th}. This way, Equation (13) can be simplified as Equation (15), where the convergent value of V_{th} equals ΔV_{th_nbti} plus V_{th_design}. Here, V_{th_design} is given by technology and ΔV_{th_nbti} can be simplified as Equation (14) because V_{th_offset} is set to 0. In Equation (15), since V_{th_design} is known and ΔV_{th_nbti} can be derived without unknown S_v, the convergent value of V_{th} can be obtained.

$$\Delta V_{th\_nbti} = A \cdot a^n \cdot t^n \tag{14}$$

$$V_{th} = \Delta V_{th\_nbti} + 0 + V_{th\_design} \tag{15}$$

c) Obtain the value of S_v with specific ΔV_{th_offset}: Given a specific value of ΔV_{th_offset}, our objective is to derive corresponding S_v value. Since the convergent V_{th} value is obtained in the last step and V_{th_design} is known, we can derive the corresponding ΔV_{th_nbti} using Equation (13), such that the corresponding S_v can be obtained in Equation (12).

So far, the conversion from a given specific ΔV_{th_offset} to corresponding ΔV_{th_nbti} has been constructed. Then, ΔV_{th_nbti} must be transformed to aging-induced delay shift. In [17], the delay shift is linearly proportional to ΔV_{th_nbti}:

$$\Delta t_{p\_aged} = C \cdot \Delta V_{th\_nbti} \tag{16}$$

where Δt_{p_aged} is BTI-induced delay shift, and C is a constant and fitted to 0.5 after SPICE simulation. Further, the Equation (16) is modified as following Equation (17) to account for the conversion from ΔV_{th_offset} to intrinsic delay shift.

$$\Delta t_{p\_intrinsic} = C \cdot \Delta V_{th\_offset} \tag{17}$$

where $\Delta t_{p\_intrinsic}$ is the delay shift caused by ΔV_{th_offset}. Up to now, a model is built to convert a given specific ΔV_{th_offset} to corresponding Δt_{p_aged} and $\Delta t_{p\_intrinsic}$. As mentioned earlier, ΔV_{th_offset} is the fresh offset/gap between high and nominal V_{th}. We use the model to derive the intrinsic delay and aging rates of high-V_{th} buffer. In our experiments, ΔV_{th_offset} is set to a specific value, such that the intrinsic delay of high-V_{th} buffers is 1.2X longer than that of nominal buffer, and the aging rates with duty cycle of 20%, 40%, 50%, and 80% are 0.5%, 4.1%, 5.4%, and 8.2%, respectively.

E. Experimental Results

Table II shows the experimental results, where Column 4 - 7 demonstrate the former results after applying DCC deployment and Column 8 - 12 demonstrate the counterparts after applying both DCC deployment and high-V_{th} assignment. Column 8 demonstrates the optimum clock period after the two techniques are applied together, denoted by $T_{c\_aged\_opt\_Leader}$. Column 9 demonstrates the used DCC count, Column 10 demonstrates the count of high-V_{th} buffers, total count of clock buffers and the ratio of high-V_{th} buffers to total clock buffers, and Column 11 demonstrates the run time. The last column demonstrates the improvement, which is the level of aging tolerance and is calculated as:

$$1 - (T_{c\_aged\_opt\_Leader} - T_{c\_fresh})/(T_{c\_aged} - T_{c\_fresh})$$

As it can be seen, the proposed framework, which considers the two techniques, can results in lower clock period, implying better improvement for aging tolerance. We can find that, after high-V_{th} assignment is included in the former framework, resulting framework may lead to different DCC count. Take *des_perf* for example, the former DCC count is 7, while the latter DCC count increases to 33. The DCC counts of the two frameworks differ because the DCC deployments are not identical anymore. Specifically speaking, when V_{th} assignment is considered, some clock buffers become candidate buffers to be inserted DCC at their inputs, because timing constraints (i.e., setup-time and hold-time) are met based on the inequality Equation (1) and (2), such that DCC can be redeployed to obtain lower Tc. Thus, even thought the two frameworks target the identical benchmark, the DCC deployments/counts may differ. Additionally, the run time of the framework dramatically increases due to numerous possibilities of DCC deployment and leader selection. To be specific, given a pair of flip-flops and associated clock paths, the former framework only considers the possibilities of DCC deployment along the clock paths, while the framework further considers the possibilities of leader selection, for each possibility of DCC deployment. Therefore, the total possibilities of DCC deployment and leader selection is equal to their mutual multiplication, i.e., DCC possibilities multiplied by leader counterparts, accounting for the dramatic increase of run time.

VII. CONCLUSION AND FUTURE WORK

In this paper, we propose a novel framework, successfully taking advantage of aging-induced clock skews by inserting limited count of DCCs into the clock tree, to enhance circuit aging tolerance. We transform the problem to a Boolean satisfiability formulation which is then solved by using MiniSat. Experiments demonstrate that our framework gains an average of 25.05% aging tolerance via inserting at most 15 DCCs. Furthermore, we include the technique of V_{th} assignment in the framework, to further improve circuit aging tolerance, by assigning small fraction of clock buffers to high V_{th}. The problem of selecting buffers to be assigned high V_{th} is also transformed to a Boolean satisfiability formulation, which is

Table II
EXPERIMENTAL RESULTS OF DCC DEPLOYMENT AND HIGH-V_{TH} ASSIGNMENT

Benchmark	No aging/Fresh	10-year Aging									
	None	None	Aging Manipulation (DCC)				Aging Manipulation (DCC) & High-V_{th} Assignment				
	T_{c_fresh}	T_{c_aged}	$T_{c\_aged\_opt\_DCC}$	# DCC	Run (s)	Improve (%)	$T_{c\_aged\_opt\_Leader}$	# DCC	# High-V_{th} Buffers / # Total Buffers (ratio%)	Run (s)	Improve (%)
des_perf	773.9	897.8	849.9	7	16.45	38.66%	816.8	33	56/3416 (1.64%)	685	65.38%
leo3mp	3016.7	3530.6	3458.5	1	24.89	14.03%	3406.1	2	19/4970 (0.38%)	923	24.23%
netcard	3367.0	3940.2	3897.9	1	30.69	7.38%	3860.1	1	7/7904 (0.09%)	1370	13.97%
vga_lcd	864.2	1013.7	994.1	11	32.77	13.11%	981.4	17	74/4346 (1.7%)	1285	21.61%
s13207	776.3	891.7	873.3	1	0.65	15.94%	857.7	1	7/180 (3.89%)	11.71	29.46%
s15850	1011.1	1165.6	1089.9	3	0.2	49.00%	1071.7	3	4/100 (4%)	0.59	60.78%
s35932	822.1	950.5	886.6	4	1.33	49.77%	865.8	5	4/411 (5.11%)	8.51	65.97%
s38417	798.1	933.2	909.8	15	1.46	17.32%	901.8	13	29/376 (7.71%)	18.47	23.24%
s38584	731.8	842.2	820.8	2	1.43	19.38%	802	4	9/557 (5.21%)	35.98	36.41%
AVG.	(ps)	(ps)	(ps)			24.95%	(ps)		3.01%		37.89%

solved by MiniSat. Experiments shows that the framework, which considers DCC deployment and V_{th} assignment simultaneously, can result in an average of 37.89% aging tolerance, via inserting at most 33 DCCs and assigning 5.21% of buffers to high-V_{th}.

REFERENCES

[1] *International technology roadmap for semiconductor*, 2013.

[2] J. W. McPherson, "Reliability challenges for 45nm and beyond," in *Proc. of DAC*, Jul. 2006, pp. 176–181.

[3] D. K. Schroder and J. A. Babcock, "Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufacturing," *Journal of applied Physics*, vol. 94, no. 1, pp. 1–18, Jul. 2003.

[4] J. H. Stathis and S. Zafar, "The negative bias temperature instability in MOS devices: A review," *Microelectronics Reliability*, vol. 46, no. 2, pp. 270–286, Feb.–Apr. 2006.

[5] S. Chakravarthi *et al.*, "A comprehensive framework for predictive modeling of negative bias temperature instability," in *Proc. of Int'l Reliability Physics Symp. (IRPS)*, Apr. 2004, pp. 273–282.

[6] N. Kimizuka *et al.*, "The impact of bias temperature instability for direct-tunneling ultra-thin gate oxide on MOSFET scaling," in *Proc. of Symp. on VLSI Technology*, Jun. 1999, pp. 73–74.

[7] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "An analytical model for negative bias temperature instability," in *Proc. of ICCAD*, Nov. 2006, pp. 493–496.

[8] W. Wang *et al.*, "The impact of NBTI effect on combinational circuit: Modeling, simulation, and analysis," *IEEE TVLSI*, vol. 18, no. 2, pp. 173–183, Feb. 2010.

[9] J. Chen and M. Tehranipoor, "A novel flow for reducing clock skew considering NBTI effect and process variations," in *Proc. of ISQED*, Mar. 2013, pp. 327–334.

[10] S.-H. Huang *et al.*, "Low-power anti-aging zero skew clock gating," *ACM TODAES*, vol. 18, no. 2, p. 27, Mar. 2013.

[11] A. Chakraborty and D. Z. Pan, "Skew management of NBTI impacted gated clock trees," *IEEE TCAD*, vol. 32, no. 6, pp. 918–927, Jun. 2013.

[12] L. Lai *et al.*, "BTI-gater: An aging-resilient clock gating methodology," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 4, no. 2, pp. 180–189, Jun. 2014.

[13] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "NBTI-aware synthesis of digital circuits," in *Proc. of DAC*, Jun. 2007, pp. 370–375.

[14] B. C. Paul *et al.*, "Temporal performance degradation under NBTI: Estimation and design for improved reliability of nanoscale circuits," in *Proc. of DATE*, Mar. 2006, pp. 780–785.

[15] K. Kang *et al.*, "Efficient transistor-level sizing technique under temporal performance degradation due to NBTI," in *Proc. of ICCD*, Oct. 2007, pp. 216–221.

[16] X. Yang and K. Saluja, "Combating NBTI degradation via gate sizing," in *Proc. of ISQED*, Mar. 2007, pp. 47–52.

[17] W. Wang *et al.*, "An efficient method to identify critical gates under circuit aging," in *Proc. of ICCAD*, Nov. 2007, pp. 735–740.

[18] J. P. Fishburn, "Clock skew optimization," *IEEE Trans. on Computers*, vol. 39, no. 7, pp. 945–951, Jul. 1990.

[19] L. Li, Y. Lu, and H. Zhou, "Optimal multi-domain clock skew scheduling," in *Proc. of DAC*, Jun. 2011, pp. 152–157.

[20] A. F. G. et al., "Early selection of critical paths for reliable nbti ag-ing-delay monitoring," *IEEE Transactions on Very Larger Scale Integration (VLSI) Systems*, 2016.