Lifetime Reliability-aware Digital Synthesis

Shengyu Duan, Mark Zwolinski, *Senior Member, IEEE*, and Basel Halak

***Abstract*—CMOS downscaling poses a growing concern for circuit lifetime reliability. Bias Temperature Instability (BTI) is a major source of transistor aging, causing a threshold voltage increase in CMOS devices and affecting circuit timing. This paper presents an aging mitigation approach that can be incorporated in standard synthesis. We propose a technique to restructure the logic expressions for aging-critical gates and to reduce the BTI stress duty cycle. A new technology mapping strategy is demonstrated, including a forward pass to select the proper cells and implement the optimized logic, and a backward pass to validate remapped circuits by restricting the negative slacks. The negative slacks produced in the mapping stage are eliminated by gate-level optimization, which aims to optimize a circuit to improve lifetime reliability under timing and area constraints. It employs a sensitivity metric that can be adjusted according to the design specifications to pick the most favorable transformation in terms of timing, lifetime or both. Our results show a 59.1% lifetime improvement with 0.86% area overhead on average. Compared with conventional over-design, a 28.29% higher lifetime improvement is realized. In addition, our approach can optimize a circuit under each corner case, considering both process variations and input data.**

***Index Terms*—NBTI, Reliability, Aging, Synthesis**

I. Introduction

As CMOS aging effects become pronounced due to technology shrinkage, significant time-dependent variations add to those caused by the fabrication process, posing a growing concern about circuit lifetime reliability [1], [2]. The amount of aging-induced degradation varies due to both intrinsic characteristics (e.g. supply voltage, intrinsic delay, etc.) and extrinsic parameters (e.g. workload, ambient temperature, etc.), which increases the challenge of improving the circuit lifetime at the design stage [3]. Bias Temperature Instability (BTI), one of the dominant aging effects, manifests itself as an increase in the CMOS transistor threshold voltage (V_{th}). It degrades NMOS and PMOS transistors in the form of Positive BTI (PBTI) and Negative BTI (NBTI), respectively. For the combinational parts of a digital circuit, the BTI-induced threshold voltage shift (ΔV_{th}) increases signal delays, eventually leading to timing violations [3], [4].

Aging-aware circuit design is therefore necessary. A number of solutions have been proposed to optimize the circuit lifetime reliability by considering BTI effects, which can be categorized into two groups: Sense & React (S&R) approaches, and precautionary approaches [5].

S&R approaches use aging sensors to monitor performance degradations in timing and/or current [6], [7], and adaptively tune circuit parameters such as supply voltage (V_{dd}) [8]. These techniques capture the performance degradations during actual operational time, and therefore are accurate in terms of aging monitoring. But in respect of lifetime optimization, adaptively scaling circuit parameters could be risky, because the scaling mechanism itself may also affect the circuit performance and even increase degradations [3]. For instance, an increase in V_{dd} results in greater power consumption and higher temperature, which may in turn accelerate aging, as BTI is temperature-/voltage-sensitive; scaling circuit frequency might be another solution to prevent BTI-induced timing violations, but may not be feasible for some time-critical systems.

Precautionary approaches take aging into consideration and optimize a circuit at the design stage. These approaches include two subgroups: aging compensation and aging mitigation [3]. The former method compensates the delay shift due to BTI by allowing an additional timing margin based on prediction. Over-design is one of the compensation methods, which has been widely used in practice to overcome circuit reliability issues. Besides conventional over-design, some compensation methods have been proposed to specifically improve lifetime reliability: Yang and Saluja proposed resizing the gates to assign delay degradation margins to NBTI-sensitive paths, [9]. Ebrahimi *et al* demonstrate an approach that leaves looser constraints on the aging critical paths to compensate for the degradation, and puts tighter constraints on the non-critical paths to reduce the area overhead, [10]. However, these works aim to compensate for a possible delay increase, rather than reduce the amount of delay shift. Thus, degradations could still be large, [11]. In addition, the pre-set extra margin may not be enough to compensate the degradation in practice, because BTI-induced delay shift changes due to different input data and temperature. Other studies aim to mitigate aging by reducing the amount of degradation. In [12], a logic-level mitigation technique is proposed to exhaustively swap the input signals of a sub-circuit and find the optimal structure in terms of NBTI lifetime. A gate mapping strategy to reduce NBTI stress probability has been suggested, [13]. Other works, [11], [14], [15], mitigate BTI at gate-level by resizing the BTI-sensitive gates. Nevertheless, as these techniques are not constraint driven, the lifetime may be improved at the cost of unacceptable overheads, considering other circuit characteristics like delay and area for designs with significant constraints. For this reason, these approaches cannot be used in a synthesis process. Kumar *et al* proposed a NBTI-aware synthesis method, where a circuit is mapped by using a pre-characterized library that includes the post-aging delay for all cells [16]. Nevertheless, the post-aging delay for each cell has to be measured by considering different input signal probabilities, and the new library may therefore contain too much information, largely reducing the efficiency of a synthesis process. Additionally, none of the above techniques has been validated considering

The authors are with Electronics and Computer Science, Faculty of Physical Science and Engineering, University of Southampton, Southampton, UK, SO17 1BJ.
E-mail: {sd5g13,mz,bh9}.ecs.soton.ac.uk

both process variations and input data.

In this work, we propose a digital synthesis approach to mitigate BTI degradation at both logic and gate levels under given constraints. This method is incorporated in the optimization phases of a synthesis flow, and does not require changes to a cell library. Our contributions are as follows:

1) A logic restructuring method to reduce the NBTI/PBTI stress duty cycle by removing NOR/NAND operations, based on signal probability transitions for different Boolean expressions;

2) A cell mapping strategy using a two pass procedure: a forward pass to select the cells to implement the optimized logic, keeping the same circuit depth, and a backward pass to ensure the optimized circuit has small negative timing slacks;

3) A gate-level optimization approach to remove the remaining negative slack, and mitigate BTI based on given area constraint. This approach employs a new sensitivity metric that can be adjusted according to the design specifications, to pick the most favorable transformation in terms of timing, lifetime or both.

It is worth noting that simultaneously mitigating both NBTI and PBTI at the logic level is not possible, as a structure giving a longer NBTI lifetime would generally suffer from more PBTI stress. Thus, one would identify the more critical aging effect based on empirical data, and optimize the circuit by using the corresponding mode of our approach.

The paper is organized as follows. Section II describes the theory and modeling of BTI effects. Section III presents the proposed aging-aware synthesis, including both logic- and gate-level optimizations. In Section IV, we provide experimental results, where it can be seen that our approach is effective, considering process variations and input data, and can realize a 59.1% lifetime improvement with a 0.86% area overhead on average. Finally, the paper is concluded in Section V.

II. BTI Effect and Delay Degradation

A. Transistor-level BTI Modeling

According to the Reaction-Diffusion (R-D) theory, BTI can be physically described as the consequence of charge generation on the transistor oxide interface [17]. The charges are produced while the transistor is stressed (i.e. turned on), and will be partially neutralized in the OFF state, recovery phase. Thus these charges accumulate over time. The ratio of stressed time to the total is known as the stress duty cycle, which also indicates the signal probability (SP) of a logic one/zero on a single NMOS/PMOS transistor.

Based on the R-D mechanism, an analytical model has been proposed, [18], to provide a long-term prediction of BTI-induced ΔV_{th}:

$$\Delta V_{th} = b.\alpha^n.t^n, \tag{1}$$

where b is a constant parameter dependent on the technology node, temperature and V_{dd}; α is the stress duty cycle; t represents the operational time; n is the time exponential constant and is equal to 0.16 [18].

In this work, the above model is used to estimate the lifetime reliability. However, the proposed techniques in the remainder of this paper do not necessarily rely on it. Models proposed in other works, [19], [20], are also compatible, since the trends in degradation will still hold.

B. Gate-level BTI Modeling

Considering multiple stacking transistors connected in series in a logic gate, the one closest to the output is stressed only if all the ones further from the output are in an ON state, and therefore cause a smaller stress duty cycle, [12], [21]. Thus, the stress duty cycle is determined by the SPs of multiple inputs, if there is a stacking effect. Otherwise, it is given by the SP of 0/1 of a single transistor for NBTI/PBTI. Fig. 1 shows the BTI stress duty cycle for each of the transistors in a 2-input NOR gate.

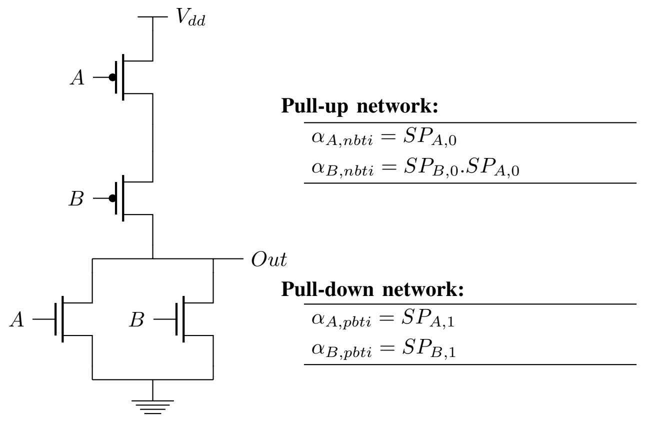

V_{dd}
A
B
Out
A
B
Pull-up network:
$\alpha_{A,nbti} = SP_{A,0}$
$\alpha_{B,nbti} = SP_{B,0}.SP_{A,0}$
Pull-down network:
$\alpha_{A,pbti} = SP_{A,1}$
$\alpha_{B,pbti} = SP_{B,1}$

Fig. 1: Transistor stress duty cycles on 2-input NOR gate

A shift in V_{th}, ΔV_{th}, results in a proportional increase of the signal propagation delay for a CMOS gate, according to the first-order approximation, [3]. Thus, we give the following equation derived from (1) to quantify BTI-induced delay shift, denoted by ΔD_{bti}, and considering the stacking effect:

$$\Delta D_{bti} = K.D_0.\alpha^n.t^n \tag{2}$$

where

$$\alpha = f(SP_i : i \in all\ inputs) \tag{3}$$

and D_0 is the gate intrinsic delay; K is determined by temperature, V_{dd} and technology node. A circuit constructed at a 65-nm technology node suffers from about 20% delay degradation after 10 years due to BTI, [10]. We applied (2) to several circuits from the ISCAS'85 benchmarks assuming a 20% delay shift on average after 10 years, to find a reasonable value of K, namely $0.3year^{-0.16}$.

C. Circuit Degradation and Data Dependence

From (2), BTI is workload dependent (i.e. dependent on the SP). Thus, in order to accurately estimate BTI-induced circuit degradations, we need to capture the correct workload characteristics. Unfortunately, input data may be highly unpredictable during the design stage of some circuits (e.g. for a general purpose processor). Several previous works assume that, for simplicity, the SP at each primary input is 50%, [10],

[12]. This assumption fails to take the unpredictability of data inputs into consideration. Therefore, the effectiveness might be altered by applying a different set of input data.

For a modern digital circuit with a large number of primary inputs, it is impractical to apply all the combinations of input SPs. Nevertheless, Bian *et al*, [22], applied a finite number of SPs to all primary inputs, and showed that the overall delay degradation of a path in a circuit has a Gaussian-like distribution. The Central Limit Theorem under weak dependence [23], suggests that the sum of an infinite number of weakly-dependent variables tends to have a Gaussian distribution. This condition is generally true for a combinational path: the total delay degradation is the sum of ΔD_{bti} for all transistors comprising the path, and the dependence of ΔD_{bti} on two transistors far enough apart from one another is negligible. In fact, path degradations can only approximate a Gaussian distribution, since a real path consists of finite transistors. But path degradations for all combinations of SP should be generally convergent in a certain interval.

Based on the above assumption, a dataset consisting of 2,500 combinations of SPs to all primary inputs was used to capture the range of path degradation in a circuit. The value of each SP is picked from a uniform distribution. The size of the dataset, 2,500, has been found to result in as consistent a distribution as those with more samples, [22]. Fig. 2 shows delay degradations for 10 years on one of the critical paths of the c432 circuit by applying the dataset. The overall shape approximates a Gaussian distribution, and the difference in delay degradation is up to 16.61% with different input data.

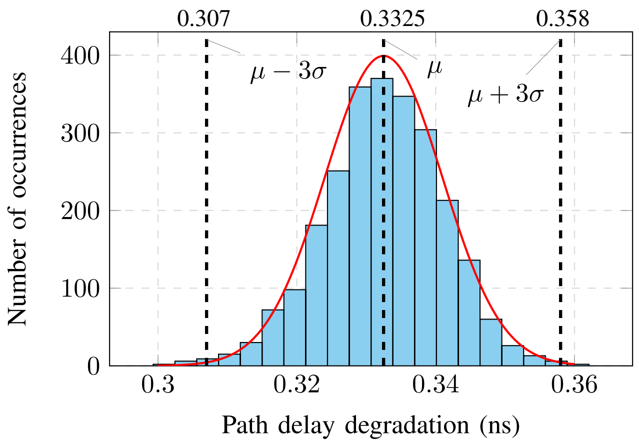


Fig. 2: 10 year degradation in c432 circuit with different input patterns

III. Aging-Aware Synthesis

A. Problem Formulation

Combinational logic optimization in modern synthesis has three phases: logic-level optimization, technology mapping and gate-level optimization [24]. At the logic level, a circuit is represented by a set of Boolean logic equations, which is technology-independent. Afterwards, components from a technology library are selected to implement the logic expressions. During the mapping, the design specifications like timing and area are also taken into consideration. Finally, the gate-level optimization attempts to meet the exact timing/area constraints by applying techniques such as gate resizing, load isolation, etc. Nevertheless, current optimization techniques in commercial synthesis tools are blind to aging effects. Thus, one can only over-design a circuit by putting in pessimistic timing margins to compensate for the impact of aging, introducing a large area overhead. In addition, as conventional over-design is not intended to mitigate aging, the amount of degradation could still be great.

Here, we propose BTI mitigation techniques at both logic and gate levels. The optimization algorithm is designed to be incorporated in a standard modern synthesis flow. The objective is not only to minimize the circuit degradations, but also to meet the constraints for timing and area. Thus we can formulate the problem for a circuit with N paths as follows:

$$\begin{aligned} \textbf{Minimize} \quad & D_{max,j} = \max_{1\le i\le N}(D_{i,0} + \Delta D_{i,j}) \\ \textbf{Subject to} \quad & D_{max,0} = \max_{1\le i\le N}(D_{i,0}) \le D_{cons} \\ & A \le A_{cons} \end{aligned} \tag{4}$$

where $D_{i,0}$ and $\Delta D_{i,j}$ are the delay and delay shift of path i, at time-zero and after j years, respectively. The post-aging circuit delay after j year(s), denoted by $D_{max,j}$, is therefore determined by the most critical path considering both intrinsic path delay and delay degradation. $D_{max,0}$ and A indicate the intrinsic circuit delay and total area; D_{cons} and A_{cons} represent the timing and area constraints respectively, for synthesis.

For digital circuits consisting of sequential and combinational logic, the clock period is determined by the combinational propagation delay and the timing characteristics of the sequential part (i.e. setup time and clock-to-Q delay.) The sequential delay is small or even negligible compared with the combinational delay. Therefore, the lifetime is mostly determined by the delay degradation of the combinational logic. We define the lifetime of a circuit as the point when the degraded delay plus the sequential delay (t_{seq}) can no longer meet the given clock period. The lifetime is computed by the maximum operational time for a circuit to run with a clock period t_{clk}, as given in (5):

$$Lifetime = \max(J) \tag{5}$$

where

$$J = \{j | D_{max,j} + t_{seq} \le t_{clk}\}$$

B. Preliminaries

The actual optimization comes after several preliminary steps, as follows:

1) Initial synthesis: The circuit is initially synthesized using a standard synthesis tool (e.g. Synopsys Design Compiler). The synthesis process is performed under the given constraints. Although lifetime reliability is not taken into account, this tool will give an initial implementation satisfying the timing/area constraints. The lifetime optimization techniques presented in the remainder of this paper are applied to this initial circuit.

2) Potential critical path identification: The potential critical paths are first captured for the initial circuit, based on following definition:

Definition 1: A path is described as a **potential critical path (PCP)** if the post-aging path delay could potentially reach D_{cons} during the required lifetime.

The area cost can be reduced by optimizing the PCPs only and leaving the other paths unchanged, since the lifetime is determined by ΔD on the PCPs. In order to reduce the computation, PCPs are identified by applying the percentage delay shift from average usage patterns, instead of an accurate evaluation based on BTI modeling, as the latter will be performed in the subsequent optimization phases. For example, a circuit at a 65-nm technology node suffers from 20% delay degradation, as stated in Section II-B. Therefore, any paths with delays larger than 80% of t_{clk} will fall into the category of PCPs. Thus, we define potential critical gates/nodes as follows:

Definition 2: A gate is a **potential critical gate (PCG)**, if it is located on one of the PCPs.

Definition 3: A primary input or an internal node is a **potential critical node (PCN)**, if it is on one of the PCPs.

3) Tree partitioning: The initial design is disjointly decomposable into trees, in each of which all cells have a fanout equal to 1, except the root (i.e. output cell), as in previous work, [25].

Definition 4: A tree is defined as a **potential critical tree (PCT)**, if it contains at least one PCG.

Each PCT will be optimized separately to reduce the computational time. Decomposing a circuit into trees may lead to a local optimum, as the logic across the tree boundaries cannot be optimized and mapped. While the problem of mapping a logic structure has been proven to be NP-hard [26], finding an optimal structure is impractical, especially for large-scale circuits. Any heuristic methods to solve the mapping problem in an affordable computational time would result in a local optimum. As one of the heuristic techniques, tree partitioning has a low complexity. Additionally, tree partitioning would exclude high-fanout cells from the optimization, because altering these cells may consequently imply the need to restructure all the driving logic for correct functionality, and thereby introduce more area cost.

We capture the potential critical sub-trees from each PCT, according to the following definition:

Definition 5: A sub-tree in a PCT is called a **potential critical sub-tree (PCST)**, if (i) all gates are PCGs, and (ii) the circuit depth is 2. The circuit depth is given by the maximum number of transistors , instead of the maximum number of gates, per path in a sub-tree. This is because the input signals of some complex gates may be delivered through multiple transistors to the output nodes. For instance, we consider an AND gate has a circuit depth of 2, if it is made by following a NAND gate with an inverter.

We give Fig. 3, as an example, where two PCSTs in a tree overlap with each other. These PCSTs will be optimized in topological order from the inputs to the root, as we will describe in the following subsections.

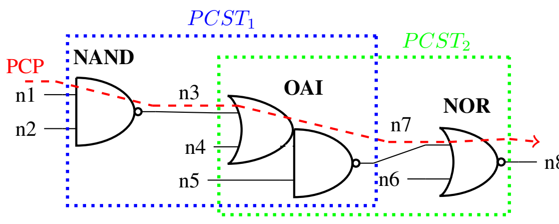


Fig. 3: PCSTs in an example circuit

C. Aging-Aware Logic Restructuring

NAND, NOR and NOT are the basic logic operations that can be used to construct any complex Boolean expression. Thus, we first analyze the SP for these three operations. Assume a signal A is connected to the input of a NOT gate or to one of the inputs of a 2-input NAND/NOR gate. The second input of the NAND/NOR gate is driven by a signal, B. Table I gives the output SPs for all three logic operations. At this step, we can assume input SPs of 0.5, allowing us to quantify the output SPs We sort the output SPs produced by giving 0.5 input SPs for these three logic operations, as shown. NAND logic produces the smallest/largest SP of 0/1 at the output; NOR gives the opposite result, while the output SP of NOT is in the middle. The relationship is independent of the input SPs: consider $SP_{out,0}$ produced by NAND and NOT operations, is equal to $SP_{A,1}.SP_{B,1}$ and $SP_{A,1}$, respectively, as shown. $SP_{A,1}.SP_{B,1}$ is smaller than (or at most equal to) $SP_{A,1}$, as the signal probability cannot be greater than 1. Similarly, the output probabilities of any two of the three operations can be compared in this way, and the relationship shown in Table I will still hold. As explained in Section II-A, the stress duty cycle of NBTI/PBTI is determined by the SP of a logic 0/1 on a single transistor. Therefore, a PMOS/NMOS transistor driven by a NOR/NAND gate is more likely to have a higher probability of NBTI/PBTI stress, assuming the same input SPs for the NOR/NAND gate.

TABLE I: Output SPs for different logic operations

	NAND	**NOT**	**NOR**
$SP_{out,0}$	$SP_{A,1}.SP_{B,1}$	$SP_{A,1}$	$SP_{A,1}+SP_{B,1}$ $-SP_{A,1}.SP_{B,1}$
Sorted (SP=0.5)	Smallest (0.25)	Mid (0.5)	Largest (0.75)
$SP_{out,1}$	$SP_{A,0}+SP_{B,0}$ $-SP_{A,0}.SP_{B,0}$	$SP_{A,0}$	$SP_{A,0}.SP_{B,0}$
Sorted (SP=0.5)	Largest (0.75)	Mid (0.5)	Smallest (0.25)

Motivated by the above observation, we propose an NBTI/PBTI mitigation method by removing NOR/NAND gates, respectively, at the logic level. For NBTI mitigation, the Boolean expression of each PCG is represented in the form of an inverted sum-of-products (SOP), i.e. an And-Or-Invert (AOI) compound gate. The product terms in the ISOP are divided into two groups, if possible: one contains PCNs, and the other does not. Each group is then formed into a new inverted SOP. The output node of the new ISOP that contains PCNs, is then added into the set of PCNs. The two restructured inverted SOPs are connected by an additional AND operation to maintain the logic function. Therefore, in order to keep the circuit depth unchanged, the AND operation is merged with the driving logic. As the original NOR operation between the two groups of product terms is removed, each restructured inverted SOP logic will have a smaller output SP of 0, potentially reducing the NBTI stress duty cycle on the driving transistor. The equations for logic restructuring can be found in the Appendix.

For example, the logic of $PCST_2$ in Fig. 3 can be restructured to reduce the NBTI stress duty cycle on node n7. In other words, we want to minimize $SP_{n7,0}$, as given in (6), in terms of the nodes of Fig. 3.

$$SP_{n7,0} = SP_{n5,1}.(SP_{n3,1} + SP_{n4,1} - SP_{n3,1}.SP_{n4,1}) \quad (6)$$

If we assume all the SPs at the input to $PCST_2$ are 0.5, $SP_{n7,0} = 0.375$.

By definition, the original sub-tree includes three PCNs: n3, n7 and n8. (7) gives the original logic expressions, which can restructured as in (8). As can be seen, the OAI logic is split into two NAND operations, one of which contains the PCN, n3; the other is not PCN-related. The optimized root logic, merging the original NOR expression with an AND operation, is implemented as an And-Or-Invert (AOI) gate, as shown in Fig. 4. The circuit depth of each PCST is the same, compared to the original structure. Assuming 0.5 SP on each of the input of $PCST_2$, $SP_{n7,0}$ is reduced from 0.375 to 0.25 at node n7_1.

$Original\ logic:$

$$\begin{cases} n7 = \overline{n3.n5 + n4.n5} \\ n8 = \overline{n6 + n7} \\ n3, n7, n8 \in PCNs \end{cases} \quad (7)$$

$Restructured\ logic:$

$$\begin{cases} n7_1 = \overline{n3.n5} \\ n7_2 = \overline{n4.n5} \\ n8 = \overline{n6 + n7_1.n7_2} \\ n3, n7_1, n8 \in PCNs \end{cases} \quad (8)$$

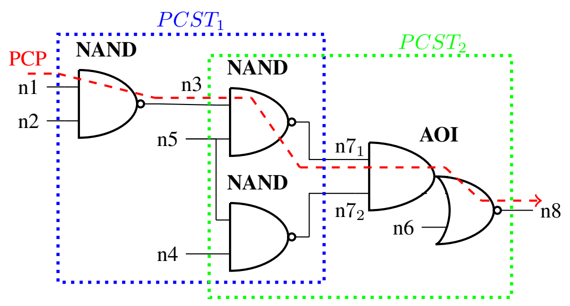


Fig. 4: Optimized PCSTs in the example circuit

The logic restructuring reduces the SP at a certain node. However, according to (3), the stress duty cycle, α, is not only determined by the input SPs, but may also be affected by the transistor-level structure, because of the stacking effect. This might mean that the susceptibility of the restructured PCST to (N)BTI stress might not be reduced. For example, consider two Boolean expressions, OR and AND, with 2 inputs, A and B, and assume these expressions are implemented by NAND, NOR and NOT gates. Table II shows different implementations for each logic expression and compares the NBTI stress duty cycles of the internal node X. In order to sort by α, we assume 0.5 SP for the input A and give two different values (0.4 and 0.6) for $SP_{B,1}$. As can be seen, α is equal to $SP_{X,0}$ if there is no stacking effect. In these cases, a NOT gate produces less NBTI stress than a NOR (in the case of OR logic), while a NAND gate leads to less stress than a NOT (in the case of AND logic). These values are independent of the input SPs, and are consistent with those in Table I. However, the rank is determined by the input SPs if there is a stacking effect on the internal node. Thus, although removing the NOR operation helps to reduce $SP_{X,0}$, reduction of the NBTI stress may also depend on the circuit structure and the SPs for the entire circuit, which are unknown at the logic level. For this reason we refer to the restructured PCST as a *Candidate* PCST. In the next section, the signal probabilities for the entire PCP in which that PCST is located are considered, and the transistor-level structure is taken into account when logic is mapped.

The restructuring process is similar for mitigating PBTI. Each PCG is represented by an inverted product-of-sums (POS), which is then split into two inverted POS expressions based on the PCNs. An OR operation is merged with the root logic, to ensure the correct function. The equations for logic restructuring to mitigate PBTI are also given in the Appendix. As mentioned in Section I, it is not possible to mitigate both BTI effects by the above technique, because a logic operation, reducing the SP of 0/1, would amplify the SP of 1/0, according to Table I.

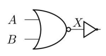
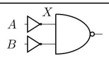
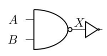
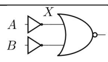
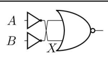
D. Technology Remapping

The optimized logic expressions are implemented by selecting the appropriate cells from the technology library, and applying a two pass procedure: a forward pass to find candidate PCST designs for each PCT in terms of lifetime reliability, and a backward pass to validate the restructured circiit and keep the circuit timing in an acceptable range considering the delay constraints. Fig. 5 presents the algorithm for technology remapping.

1) Forward pass: The forward pass phase attempts to find the minimal degradation design for all PCTs, by restructuring and mapping the logic expression of each candidate or original PCST in topological order from the inputs to the output. A candidate PCST from Section III-C is considered to be lifetime optimal if it satisfies the following conditions:

a) : The candidate PCST has the same circuit depth as the original, which is equal to 2, according to Definition 5. This avoids the risk that extra logic gates are added to the PCP, which would result in an increased path delay and degradation. Although the increased delay and degradation can be eliminated at the following backward pass and gate-level optimization steps, a large area cost will be incurred if many extra gates are put in to the PCPs. The restructuring process may add additional gates to the non-critical paths, as some logic, such as at node n7 in (7), is split into two operations at n7_1 and n7_2 in (8). These gates can be sized into the smallest

TABLE II: Internal node stress duty cycle for different implementations

		OR		**AND**		
Implementation						
$SP_{X,0}$		$SP_{A,1}+SP_{B,1}$ $-SP_{A,1}.SP_{B,1}$	$SP_{A,1}$	$SP_{A,1}.SP_{B,1}$	$SP_{A,1}$	
Stacking effect at node X?		No	No	No	No	Yes
$\alpha_{X,nbti}$		$SP_{A,1}+SP_{B,1}$ $-SP_{A,1}.SP_{B,1}$	$SP_{A,1}$	$SP_{A,1}.SP_{B,1}$	$SP_{A,1}$	$SP_{A,1}.SP_{B,0}$
Sorted $\alpha_{X,nbti}$ (SP_A=0.5)	$SP_{B,1}$=0.4	Larger (0.7)	Smaller (0.5)	Smallest (0.2)	Largest (0.5)	Moderate (0.3)
	$SP_{B,1}$=0.6	Larger (0.8)	Smaller (0.5)	Moderate (0.3)	Largest (0.5)	Smallest (0.2)

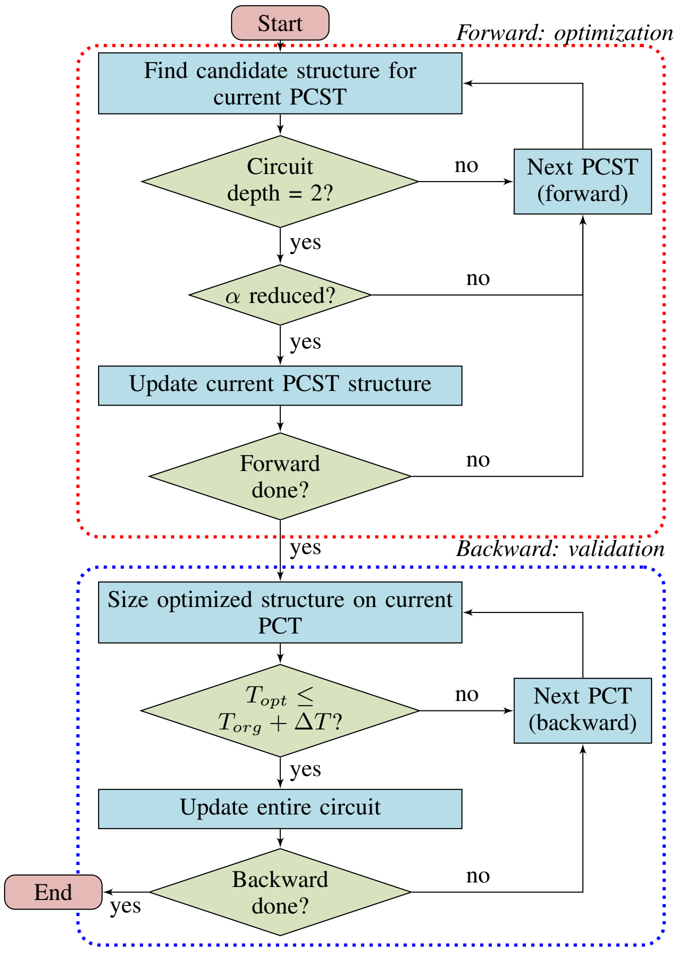


Fig. 5: The overall technology remapping algorithm

area, because the non-critical paths do not affect the overall circuit timing and degradation.

b) : The stress duty cycle of the internal node of the PCST, is reduced. As mentioned in Section III-C, the logic restructuring lowers the SP on a certain node, while the stress duty cycle may or may not be reduced, depending on the transistor structure of the gate. Thus, the stress duty cycle α of the node needs to be computed after remapping to verify the effectiveness. Here, the SP is assumed to be 0.5 at the inputs to the entire path; in Section III-C, the SP was assumed to be 0.5 at the inputs of each PCST. While that made the computation simple, it clearly could be incorrect.

If the above conditions are met, the original PCST is replaced by the remapped candidate PCST for the minimal degradation design. Otherwise, we consider that the mapping phase has failed to find an improved circuit structure for the PCST in terms of lifetime reliability. The above steps are repeated until all candidate PCSTs are mapped, or not. After the forward pass, we have two versions of the circuit for a PCT: one has the original structure, and the other has an optimized structure to minimize NBTI or PBTI degradation.

Considering Fig. 4 again, but this time assume that the SP is 0.5 at each input of the entire circuit ($n1$, $n2$, $n4$ and $n5$), rather than just $PCST_2$ ($n3$, $n4$ and $n5$), we find that α is 0.375 on $n7_1$, compared with α of 0.4375 on $n7$ in the original circuit, assuming there is no stacking effect on both nodes. Thus the NBTI stress is reduced in the restructured circuit.

2) Backward pass: The backward pass validates each optimized PCT in terms of its delay. This ensures the negative slacks of a remapped circuit are in an acceptable range. During the backward pass, the PCTs are analyzed in reverse topological order. Thus, the deepest PCT is examined first.

As the forward pass only determines the circuit structure, rather than the size of each device, we first find a suitable area for each PCT. The original area of each PCT is first recorded. Each optimized PCT is required to have an area as close to the original as possible, since the original PCT is assumed to have a good trade-off between timing and area (Section III-B). This can be done by using a standard synthesis tool and setting the area constraint equal to the area of the original PCT. It can be seen from Figs. 3 and 4 that the total gate count of the remapped circuit is higher than that of the original circuit. Thus, by giving a similar area to the optimized PCT, we would expect an increased delay, because of smaller transistors, compared with the original PCT, which may produce negative slacks.

Any negative slacks will be removed by applying the proposed gate-level optimization, as in the following subsection. Nevertheless, more computation is needed to remove larger negative slacks. Therefore, in order to ensure the entire computational time is acceptable, one can compare the signal arrival time at the output of the optimized PCT, T_{opt}, with that of the original PCT, T_{org}. We set a constant ΔT to restrict the negative slacks to an acceptable range. Here, ΔT is set to 0.5ns. Therefore, any optimized PCT, in which T_{opt} is 0.5ns or less than T_{org}, is selected to construct the circuit. Otherwise, the original PCT is chosen. The above steps are repeated until suitable mappings for all PCTs are found.

E. Gate-level Optimization

The remapping process of Section III-D reduces the stress duty cycles, but it may cause increased path delays or even negative slacks. It is possible to eliminate the negative slacks by using the conventional gate-level optimizations of commercial synthesis tools. However, none of the techniques in these tools is designed for aging. They are less effective, because they fail to optimize the more aging-sensitive devices. Here, we propose a gate-level optimization method to eliminate the negative slacks caused by technology remapping, while also mitigating BTI degradation subject to the given area constraint.

Delay-area sensitivity (S_D), represented by the ratio of delay reduction to area increase, has been used as a standard metric for gate sizing in previous work, [27]. The transformation with the biggest delay-area sensitivity is applied in turn until the timing constraints are met. The optimization aims to add as small as possible an area to satisfy the timing specifications. While this technique does not account for the impact of BTI, degradation-area sensitivity ($S_{\Delta D}$) is proposed to identify the transformation that can give the greatest degradation improvement per unit area increase [11]. Since the degradation is proportional to the intrinsic delay according to (2), up-sizing the gate with the highest $S_{\Delta D}$ would also favor circuit delay reduction. Thus the timing constraint would be met eventually, at the cost of more area.

However, the circuit area is not really controllable by using either of the above sensitivities. We therefore propose a new sensitivity metric combining both sensitivities to realize any customized area/timing constraints, as given by:

$$S_{prop} = C_{cons}.w.S_{\Delta D} + S_D \tag{9}$$

where

$$C_{cons} = \max(0, \frac{\frac{A_{cons}}{A} - 1}{1 - \frac{D_{cons}}{D_{max}}})$$

and C_{cons} is a constraint-related coefficient. w is a weight, indicating the influence of $S_{\Delta D}$ on the proposed sensitivity, S_{prop}. The value of w is adjustable to satisfy any given constraints, as described later. A_{cons}, A, D_{cons} and D_{max} are as defined for (4).

For a better understanding of (9), consider the characteristics of S_{prop} under three scenarios:

Scenario 1: A circuit has an area A very close to or bigger than A_{cons}, while the delay D_{max} is much greater than D_{cons}. In this case, S_{prop} is dependent mainly on S_D, since C_{cons} tends to zero. This situation can be interpreted as follows: if little area is left to accomplish a relatively big delay decrease, the optimization should be dedicated to reducing the delay with as small an area increase as possible;

Scenario 2: At the other extreme, D_{max} is nearly equal to D_{cons} but the difference between A and A_{cons} is big. C_{cons} tends to infinity, and S_{prop} is thereby determined by $S_{\Delta D}$. In this case the remaining area should be used to mitigate the degradation, if the timing constraint has been met;

Scenario 3: If neither of the above holds, C_{cons} is a finite number. Thus S_{prop} is determined by both S_D and $S_{\Delta D}$, representing both delay and degradation in the transformation.

In summary, we can rewrite (9) as follows:

$$S_{prop} = \begin{cases} S_D, & \frac{A_{cons}}{A} - 1 \ll 1 - \frac{D_{cons}}{D_{max}} \\ C_{cons}.w.S_{\Delta D}, & \frac{A_{cons}}{A} - 1 \gg 1 - \frac{D_{cons}}{D_{max}} \\ C_{cons}.w.S_{\Delta D} + S_D, & otherwise \end{cases} \tag{10}$$

Fig. 6 shows the proposed optimization algorithm to minimize BTI degradation under given constraints. Firstly, the weight w from (9) is initialized to a finite number. The initial value of w does not really matter, since it can be adjusted in the remaining steps. However, a greater w indicates $S_{\Delta D}$ has a stronger influence on S_{prop}, and the gate with the highest $S_{\Delta D}$ is more likely to be up-sized. In such a case, the overall optimization would bring about less degradation but more area. Secondly, the critical path, where the path delay is D_{max}, needs to be identified. In addition, because S_{prop} can be determined by both S_D and $S_{\Delta D}$, the path with a degradation equal to ΔD_{max}, or the aging-critical path, is identified as well. Afterwards, S_{prop} for all gates of the critical/aging-critical path is computed, and the gate with the highest S_{prop} is up-sized. The above steps are repeated until D_{cons} is met. Finally, we check whether A_{cons} is satisfied. A pre-defined constant ϵ is used to restrict the acceptable range of area: if the actual area does not lie in $[A_{cons} - \epsilon, A_{cons}]$, w would increase or decrease, depending on the exact value of A, and the design is restored to the initial state, where any gate-level transformations have not yet been applied. A maximum iteration count can be set to give a reasonable runtime.

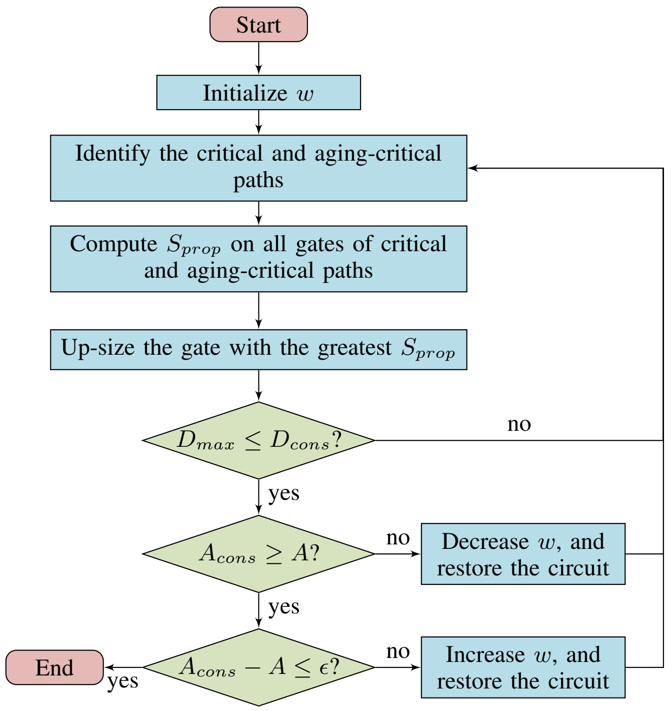

Start
Initialize w
Identify the critical and aging-critical paths
Compute S_{prop} on all gates of critical and aging-critical paths
Up-size the gate with the greatest S_{prop}
$D_{max} \le D_{cons}$?
no
yes
$A_{cons} \ge A$?
no
Decrease w, and restore the circuit
yes
$A_{cons} - A \le \epsilon$?
no
Increase w, and restore the circuit
yes
End

Fig. 6: The gate-level optimization flow to minimize BTI effect under given constraints

IV. Experimental Results

The proposed techniques shown in this paper have been applied to the ISCAS'85 and EPFL benchmark [28] circuits, implemented as standard cells in a 65-nm CMOS technology. As mentioned in Section III-C, NBTI-induced degradation is around ten times bigger than that caused by PBTI for this technology. Thus we use our approach to specifically mitigate NBTI effects, while the circuit lifetime is computed by taking both NBTI and PBTI into account. The experiments are performed on a 24 core 2.6 GHz x86-64 machine.

Each circuit is optimized and evaluated, considering process variations and input data dependence. During the optimization phase, we assume each circuit is implemented using nominal cell values to reduce the computational complexity. The nominal case is defined as all devices having the mean process parameters and the SP at each primary input is 50%. Each circuit is then evaluated comprehensively by applying the nominal, best and worst process corners, where the intrinsic delays are assumed to be μ, $\mu - 3\sigma$ and $\mu + 3\sigma$, respectively, within a Gaussian distribution. In terms of input data, we construct a dataset consisting of 2,500 combinations of SP at all primary inputs in each corner case, assuming the SP at each primary input follows a uniform distribution. As shown in Fig. 2, circuit degradation would also approximate a Gaussian distribution for different input data. Therefore we use a Gaussian distribution to fit the BTI degradations considering both process variations and input data dependence for each circuit.

Fig. 7 shows the degradation change using our approach with the c880 circuit. As can be seen, the optimized distribution is shifted to the left and maintains a similar shape to the original. The mean value of BTI degradations over 10 years is reduced by 6.44% from 0.303 ns to 0.2835 ns, while the standard deviation is very similar. With regard to the lifetime increase compared to the nominal, best (i.e., -3σ) and worst (i.e., $+3\sigma$) cases of the optimized distribution in Fig. 7, the clock period t_{clk} is set to ensure a 10 year lifetime as an example for the original circuit in each case. The operational times when the circuit delay reaches t_{clk} are shown in Fig. 8. The lifetimes are increased by 63.85%, 49.08% and 40.93% in the best, nominal and worst cases, respectively. Our approach shows the greatest lifetime improvement in the case with least circuit delay. This is because our method can reduce the degradation by a similar amount for each corner case, according to Fig. 7. Therefore, the percentage ΔD reduction would be higher for a circuit with a smaller intrinsic delay. This result indicates our approach optimizes the lifetime for all corner cases. It is also noticeable that the lifetime can be significantly extended with a relatively small amount of degradation reduction, due to the exponential behavior of BTI degradation with respect to the operational time, as in (2).

As the proposed approach can mitigate BTI under given constraints, we give two upper limits of area for gate-level optimization: 1) $A_{cons} = 0$, indicating to use the least possible area to satisfy the timing constraint and optimize the lifetime reliability, if possible; 2) $A_{cons} = +5\%$, where the optimized circuit is required to be no more than 5% larger in area than the original. Here, the area is given by the equivalent gate count, with respect to the area of the smallest NAND cell. According to our observations, a 5% area increase is more than sufficient to remove all negative slacks. Thus, the remaining area can be used to further increase the lifetime.

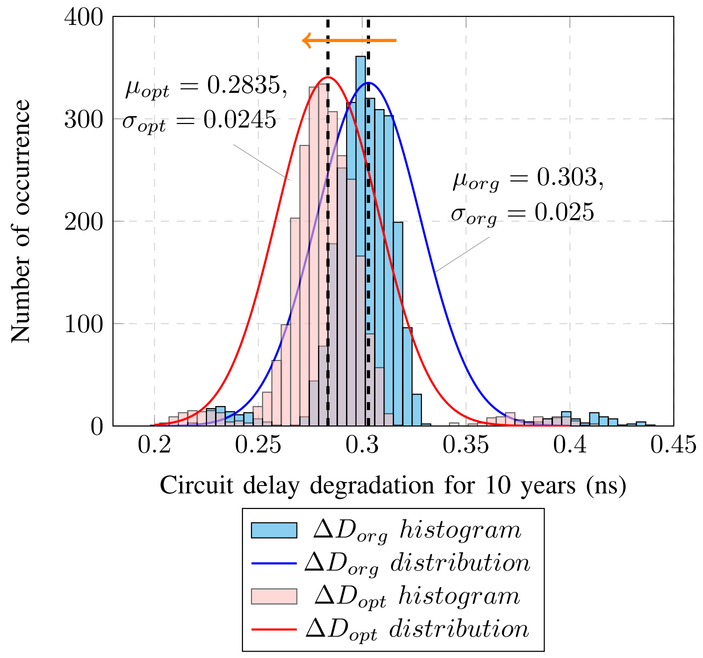
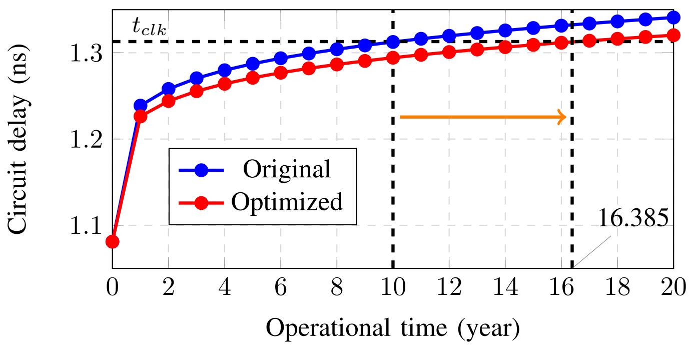
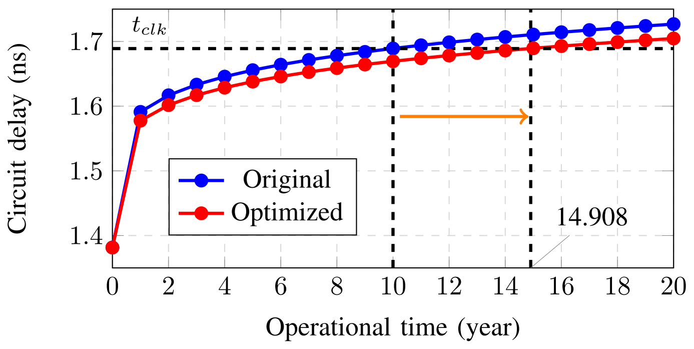


Fig. 7: Degradation decrease by proposed approaches on c880

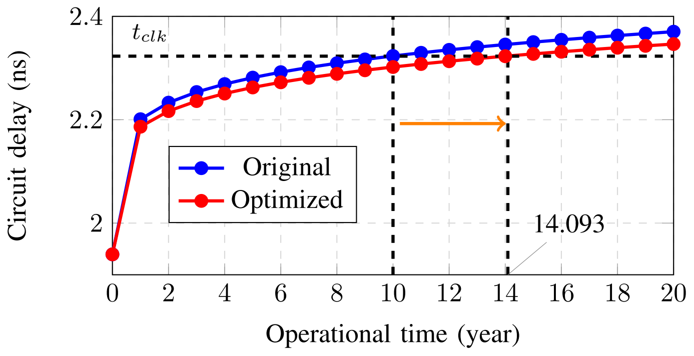
Table III presents the circuit performances for the example circuits. As can be seen from Table III (a), given the least possible area ($A_{cons} = 0$), our approach realizes a 31.72% lifetime improvement on average, while only introducing a 0.3% area increase, which is negligible compared with the original designs. For the circuits c432, c1355, c1908 ,c5315 and Voter, the entire implementations remain unchanged, as our method could not find improved designs for lifetime increase within the given timing constraints and with the least possible area increase. Therefore, we consider these original designs to have a good trade-off between circuit timing, area and lifetime reliability already. For the case of $A_{cons} = +5\%$ shown in Table III (b), our approach generally presents greater lifetime improvements than the case of $A_{cons} = 0$, since more area is provided to mitigate the BTI effect. The lifetime is increased by 59.1% on average, while the area overhead is 0.86%, which is less than the area increase limit (5%). For both cases, our method causes negligible changes for the power consumptions, measured by using Synopsys Design Compiler (DC). The computational time is affordable considering the great lifetime improvement, which can be counted in terms of months or years. Additionally, the computational time is generally independent of the circuit scale. For the logic restructuring and mapping phase, the runtime is dependent on the number of PCGs of a circuit; for the gate-level optimization, it is mainly determined by the amount of negative slacks produced in the technology remapping phase. More computation is required by giving more area to enhance the lifetime reliability. Thus, one can first optimize a design by giving the smallest area constraint, and then decide if it is necessary to increase the budget for area and further mitigate BTI effects. It is also worth noting that we built our approach on top of DC and



(a) Best case



(b) Nominal case



(c) Worst case

Fig. 8: Lifetime increase by proposed approaches on c880

invoked the static timing analysis of DC to measure the circuit delay in each iteration. The optimization for a circuit may take hundreds of iterations and thus, a large amount of time was used to initiate DC. We expect the runtime can be significantly reduced if our approach can be applied as a built-in function of synthesis tools.

We compare the circuits optimized by our approach for the case of $A_{cons} = +5\%$ with the over-design method in Table IV. We use DC to over-design a circuit and leave an extra timing margin. Each circuit is optimized via both techniques, given the same area constraint. As can be seen, our approach has a smaller percentage degradation over 10 years for all circuits, compared with conventional over-design. Thus, our approach mitigates the BTI effect by aggressively reducing the amount of degradation. Our method gives greater lifetime improvements, compared with over-design, on all circuits, with two exceptions on c432 and c5315. As has been mentioned, any heuristic method may fail to find the global optimal solution and thus, a conventional over-design technique may sometimes realize a better trade-off. However, the proposed method results in longer lifetimes on most of the circuits, and a 28.29% lifetime improvement produced by averaging the lifetime increases of Table IV, compared with over-design. These results would be expected, because our approach selectively mitigates the BTI effect on the more sensitive devices. In practice, one can apply both our approach and over-design to optimize a circuit, and select the more robust design. Thus, our method can be used as a complementary technique for conventional over-design during synthesis, to explore the best trade-off between the circuit lifetime and other parameters.

V. Conclusion

In this paper, we have presented a multi-level optimization approach for digital synthesis considering CMOS aging effects. The proposed approach mitigates the amount of BTI-induced circuit degradation, while also taking the design specifications like area and timing into consideration. In particular, based on the signal probabilities in different Boolean expressions, a logic-level optimization method is presented to reduce NBTI/PBTI stress by removing NOR/NAND gates. This method can be used to mitigate the more critical aging effects based on empirical data, as there is no way to mitigate both NBTI and PBTI simultaneously.

We further demonstrate a cell mapping strategy to implement the restructured logic expressions. During the mapping phase, we use a forward pass to select appropriate cells from a given technology library for each logic expression, in topological order from inputs to the outputs. After the forward pass, the minimal degradation structure for each part of the circuit is produced, which is then used to determine the overall circuit structure in a backward pass. The backward pass restricts the negative slacks produced in the mapping stage to an acceptable range.

A gate-level optimization method is then presented to remove the remaining negative slacks, and to further mitigate aging effects according to the given area constraints. In this work, the proposed approach is applied to specifically mitigate the NBTI effect, which is the more severe lifetime-limiting mechanism comparing with PBTI for a 65nm technology node. But it is also applicable for any other technology node dominated by one aging effect. The results show our method manages to find a design within the constraints, and can lead to a 59.1% lifetime improvement with a 0.86% area overhead on average. Our approach gives a 28.29% higher lifetime reliability, compared with a conventional over-design method , and thus can be used as a complementary technique for over-design to find the best trade-off between lifetime and other parameters. In addition, we show our approach is effective for different corner cases, as the lifetime reliability is optimized over different process variations and input data.

Acknowledgment

This work was partly supported by Cisco Research Center Grant # 593688.

TABLE III: Nominal case circuit performances by the proposed approaches

(a) $A_{cons} = 0$

Circuit	**Clock period (ns)**	**Original**	**Optimized**						
		Area A_{org}	Area inc		Power inc (%)	Lifetime inc (%)	Runtime (s)		
			Equivalent gate count	(%)			Logic-level & mapping	Gate-level	Total
c432	1.92	135	0	0	0	0	17.43	853.1	870.53
c499	1.65	370.5	2.5	0.67	1.58	17.56	4.5	390.75	395.25
c880	1.7	270	1	0.37	-0.65	49.08	4.53	112.08	116.61
c1355	1.53	358	0	0	0	0	4.56	361.85	366.41
c1908	2.03	328	0	0	0	0	4.53	543.92	548.45
c2670	1.98	574	4.5	0.78	-0.71	71.62	18.14	139.95	158.09
c3540	2.76	687	1.75	0.25	-1.26	72.94	0.29	199.6	199.89
c5315	2.19	1088.75	0	0	0	0	0.27	243.43	243.7
c6288	11.61	1929.5	7.75	0.4	-2.88	8.09	48.81	1290.94	1339.75
c7552	2.85	1507.75	18	1.19	-0.31	156.58	9.68	641.76	651.44
Sine	22.5	4457.5	9.25	0.21	-0.16	49.07	104.01	41.19	145.2
Voter	7.36	10037.25	0	0	0	0	15.29	40.05	55.34
Square	36.85	16718.75	59	0.35	0.69	16.6	508.81	45.78	554.59
Log2	56	24135.75	10	0.04	0.08	2.5	51.91	43.2	95.11
average				**0.3**	**-0.26**	**31.72**			

(b) $A_{cons} = +5\%$

Circuit	**Clock period (ns)**	**Original**	**Optimized**						
		Area A_{org}	Area inc		Power inc (%)	Lifetime inc (%)	Runtime (s)		
			Equivalent gate count	(%)			Logic-level & mapping	Gate-level	Total
c432	1.92	135	6.5	4.81	-4.35	78.02	17.43	1245.28	1262.71
c499	1.65	370.5	2.5	0.67	1.58	22.25	4.5	402.13	406.63
c880	1.7	270	1	0.37	-0.65	49.08	4.53	114.44	118.97
c1355	1.53	358	3.25	0.91	2.1	13.32	4.56	480.38	484.94
c1908	2.03	328	2.75	0.84	0.75	182.17	4.53	587.01	591.54
c2670	1.98	574	4.5	0.78	-0.71	71.62	18.14	150.02	168.16
c3540	2.76	687	1.75	0.25	-1.26	74.1	0.29	203.53	203.82
c5315	2.19	1088.75	12.5	1.15	-2.02	71	0.27	601.9	602.17
c6288	11.61	1929.5	8.25	0.43	-2.91	20.72	48.81	1314.14	1362.95
c7552	2.85	1507.75	18	1.19	-0.31	156.58	9.68	647.42	657.1
Sine	22.5	4457.5	9.25	0.21	-0.16	75.58	104.01	413.33	507.34
Voter	7.36	10037.25	2	0.02	0.02	58.28	15.29	168.52	183.81
Square	36.85	16718.75	59	0.35	0.7	17.05	508.81	3397.21	3906.02
Log2	56	24135.75	10	0.04	0.08	8.57	51.91	937.29	989.2
average				**0.86**	**-0.51**	**59.1**			

TABLE IV: Nominal case lifetime reliability by conventional over-design and our approach

Circuit	**Over-design**		**Our method**	
	$\Delta D_{max,10}$ (%)	Lifetime (yrs)	$\Delta D_{max,10}$ (%)	Lifetime inc (%)
c432	21.16	19.485	19.2	-8.64
c499	19.1	7.824	18.7	56.25
c880	21.83	12.89	20.22	15.66
c1355	19.2	11.059	18.85	2.47
c1908	20.13	16.724	17.23	68.72
c2670	20.06	14.156	19.27	21.23
c3540	23.2	13.057	21.32	33.34
c5315	22.2	19.183	20.23	-10.86
c6288	22.45	7.214	21.57	67.4
c7552	22	11.477	17.61	123.56
Sine	19.11	16.86	18.26	4.14
Voter	19.86	14.1	19.68	12.26
Square	21.69	11.47	21.17	2.05
Log2	21.9	10	21.61	8.57
average				**28.29**

Appendix

Assume the logic of a node is given the sum of p products, and each product term i contains q^i elements. Equations (11) and (12) are used to restructure the logic of a PCST, for NBTI mitigation. We denote n_{inter} as the internal PCN of a PCST, which is split into two nodes, n_{pcn} and n_{ncpn}; n_{out} is the output node of the PCST.

Original logic representation (NBTI mode):

$$\begin{cases} n_{inter} = \neg \sum_{i=1}^{p} (\prod_{j=1}^{q^i} node(i,j)) \\ n_{out} = f(n_{inter}) \end{cases} \tag{11}$$

Restructured logic representation (NBTI mode):

$$\begin{cases} n_{pcn} = \neg \sum_{i=1}^{p_1} (\prod_{j=1}^{q_1^i} node_1(i,j)), \\ \qquad\qquad where\ \{node_1\} \cap PCNs \neq \varnothing \\ n_{npcn} = \neg \sum_{i=1}^{p_2} (\prod_{j=1}^{q_2^i} node_2(i,j)), \\ \qquad\qquad where\ \{node_2\} \cap PCNs = \varnothing \\ n_{out} = f(n_{pcn} \cdot n_{npcn}) \end{cases} \tag{12}$$

Similarly, we give Equations (13) and (14) to restructure a PCST and reduce the PBTI degradation, where the logic of the internal node is represented by an inverted POS with p sums, each of which contains q^i elements.

Original logic representation (PBTI mode):

$$\begin{cases} n_{inter} = \neg \prod_{i=1}^{p} (\sum_{j=1}^{q^i} node(i,j)) \\ n_{out} = f(n_{inter}) \end{cases} \tag{13}$$

Restructured logic representation (PBTI mode):

$$\begin{cases} n_{pcn} = \neg \prod_{i=1}^{p_1} (\sum_{j=1}^{q_1^i} node_1(i,j)), \\ \qquad where\ \{node_1\} \cap PCNs \neq \varnothing \\ n_{npcn} = \neg \prod_{i=1}^{p_2} (\sum_{j=1}^{q_2^i} node_2(i,j)), \\ \qquad where\ \{node_2\} \cap PCNs = \varnothing \\ n_{out} = f(n_{pcn} + n_{npcn}) \end{cases} \tag{14}$$

REFERENCES

[1] I. Agbo, M. Taouil, D. Kraak, S. Hamdioui, H. Kükner, P. Weckx, P. Raghavan, and F. Catthoor, "Integral impact of BTI, PVT variation, and workload on SRAM sense amplifier," *IEEE Trans. VLSI Syst*, vol. 25, no. 4, pp. 1444–1454, 2017.

[2] Y. Lu, L. Shang, H. Zhou, H. Zhu, F. Yang, and X. Zeng, "Statistical reliability analysis under process variation and aging effects," in *Proceedings of the 46th Annual Design Automation Conference*. ACM, 2009, pp. 514–519.

[3] X. Chen, Y. Wang, H. Yang, Y. Xie, and Y. Cao, "Assessment of circuit optimization techniques under NBTI," *IEEE Design & Test*, vol. 30, no. 6, pp. 40–49, 2013.

[4] K.-C. Wu and D. Marculescu, "Aging-aware timing analysis and optimization considering path sensitization," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*. IEEE, 2011, pp. 1–6.

[5] E. Afacan, M. B. Yelten, and G. Dndar, "Review: Analog design methodologies for reliability in nanoscale CMOS circuits," in *2017 14th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, June 2017, pp. 1–4.

[6] G. Sai, B. Halak, and M. Zwolinski, "Multi-path ageing sensor for cost-efficient delay fault prediction," *IEEE Trans. Circuits Syst. II: Express Briefs*, vol. PP, no. 99, pp. 1–1, 2017.

[7] R. H. Ramlee and M. Zwolinski, "Using Iddt current degradation to monitor ageing in cmos circuits," in *Power and Timing Modeling, Optimization and Simulation (PATMOS), 2016 26th International Workshop on*. IEEE, 2016, pp. 200–204.

[8] T.-B. Chan, W.-T. J. Chan, and A. B. Kahng, "Impact of adaptive voltage scaling on aging-aware signoff," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2013, pp. 1683–1688.

[9] X. Yang and K. Saluja, "Combating NBTI degradation via gate sizing," in *Quality Electronic Design, 2007. ISQED'07. 8th International Symposium on*. IEEE, 2007, pp. 47–52.

[10] M. Ebrahimi, F. Oboril, S. Kiamehr, and M. B. Tahoori, "Aging-aware logic synthesis," in *Computer-Aided Design (ICCAD), 2013 IEEE/ACM International Conference on*. IEEE, 2013, pp. 61–68.

[11] A. Gomez and V. Champac, "A new sizing approach for lifetime improvement of nanoscale digital circuits due to BTI aging," in *Very Large Scale Integration (VLSI-SoC), 2015 IFIP/IEEE International Conference on*. IEEE, 2015, pp. 297–302.

[12] K.-C. Wu and D. Marculescu, "Joint logic restructuring and pin reordering against NBTI-induced performance degradation," in *Proceedings of the Conference on Design, Automation and Test in Europe*. European Design and Automation Association, 2009, pp. 75–80.

[13] M. Ghane and H. R. Zarandi, "Gate merging: An NBTI mitigation method to eliminate critical internal nodes in digital circuits," in *Parallel, Distributed, and Network-Based Processing (PDP), 2016 24th Euromicro International Conference on*. IEEE, 2016, pp. 786–791.

[14] S. Khan and S. Hamdioui, "Modeling and mitigating NBTI in nanoscale circuits," in *On-Line Testing Symposium (IOLTS), 2011 IEEE 17th International*. IEEE, 2011, pp. 1–6.

[15] B. C. Paul, K. Kang, H. Kufluoglu, M. A. Alam, and K. Roy, "Negative bias temperature instability: Estimation and design for improved reliability of nanoscale circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 4, pp. 743–751, 2007.

[16] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "Nbti-aware synthesis of digital circuits," in *Proceedings of the 44th annual Design Automation Conference*. ACM, 2007, pp. 370–375.

[17] A. Campos-Cruz, E. Tlelo-Cuautle, and G. Espinosa-Flores-Verdad, "Advances in BTI modeling for the design of reliable ICs," in *Electrical Engineering, Computing Science and Automatic Control (CCE), 2016 13th International Conference on*. IEEE, 2016, pp. 1–4.

[18] W. Wang, Z. Wei, S. Yang, and Y. Cao, "An efficient method to identify critical gates under circuit aging," in *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on*. IEEE, 2007, pp. 735–740.

[19] S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, and S. Vrudhula, "Predictive modeling of the NBTI effect for reliable design," in *Custom Integrated Circuits Conference, 2006. CICC'06. IEEE*. IEEE, 2006, pp. 189–192.

[20] W. Wang, S. Yang, S. Bhardwaj, S. Vrudhula, F. Liu, and Y. Cao, "The impact of NBTI effect on combinational circuit: Modeling, simulation, and analysis," *IEEE Trans. VLSI Syst.*, vol. 18, no. 2, pp. 173–183, Feb 2010.

[21] D. R. Bild, G. E. Bok, and R. P. Dick, "Minimization of NBTI performance degradation using internal node control," in *Proceedings of the Conference on Design, Automation and Test in Europe*. European Design and Automation Association, 2009, pp. 148–153.

[22] S. Bian, M. Shintani, S. Morita, H. Awano, M. Hiromoto, and T. Sato, "Workload-aware worst path analysis of processor-scale NBTI degradation," in *Proceedings of the 26th edition on Great Lakes Symposium on VLSI*. ACM, 2016, pp. 203–208.

[23] R. C. Bradley, "Central limit theorems under weak dependence," *Journal of Multivariate Analysis*, vol. 11, no. 1, pp. 1–16, 1981.

[24] G. D. Micheli, *Synthesis and optimization of digital circuits*. McGraw-Hill Higher Education, 1994.

[25] Z. Wo and I. Koren, "Technology mapping for reliability enhancement in logic synthesis," in *Quality of Electronic Design, 2005. ISQED 2005. Sixth International Symposium on*. IEEE, 2005, pp. 137–142.

[26] K. Chaudhary and M. Pedram, "A near optimal algorithm for technology mapping minimizing area under delay constraints," in *Proceedings of the 29th ACM/IEEE Design Automation Conference*. IEEE Computer Society Press, 1992, pp. 492–498.

[27] D. G. Chinnery and K. Keutzer, "Linear programming for sizing, Vth and Vdd assignment," in *Proceedings of the 2005 international symposium on Low power electronics and design*. ACM, 2005, pp. 149–154.

[28] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "The epfl combinational benchmark suite," in *Proceedings of the 24th International Workshop on Logic & Synthesis (IWLS)*, no. EPFL-CONF-207551, 2015.

Shengyu Duan received the double B.Eng. degree in telecommunication engineering and electronic & electrical engineering from Huazhong University of Science and Technology, China, and the University of Birmingham, U.K. in 2013. He received the MSc degree in Microelectronics Systems Design from the University of Southampton, U.K., in 2014. He is currently a PhD student in electronic & electrical engineering at University of Southampton. His research interests include design for digital systems reliability and aging-aware analysis.

PLACE
PHOTO
HERE

Basel Halak is the director of the Embedded Systems Master program at Southampton University, he has written over 60 conference and journal papers, and authored two books He received his PhD degree in Microelectronics System Design from Newcastle University. He was then awarded a knowledge transfer fellowship to develop secure and energy efficient design for portable health care monitoring systems. He is a member of the Sustainable Electronics research group, as well as, Cyber Security group at Electronics and Computer Science School (ECS). His background is on the design and implementation of microelectronics systems, with special focus on reliability and security. In particular, Dr Halak is interested in developing secure hardware implementation for cryptographic primitive such as physically unclonable functions. Dr Halak lectures on digital design, Secure Hardware and Cryptography, supervises a number of MSc and PhD students He is also the head of the Master of Science Course(MSc) on Embedded Systems at Southampton University . Dr Basel Halak is a senior fellow of the Higher Education Academy (HEA), a guest editor of the IET CDT, and serves in several technical program committees such as IEEE ICCCA, ICCCS, MTV, IVSW, MicDAT and EWME. He is also member of hardware security working group of the World Wide Web Consortium (W3C).

PLACE
PHOTO
HERE

Mark Zwolinski received the B.Sc. degree in electronic engineering and the Ph.D. degree in electronics from University of Southampton, Southampton, U.K., in 1982 and 1986, respectively. He is currently a Professor in the School of Electronics and Computer Science, University of Southampton. He has authored two textbooks and has co-authored a third. He has written over 190 papers in the areas of EDA and test. His current research interests include high-level synthesis, fault tolerance, and behavioral modeling and simulation. Dr. Zwolinski is a Fellow of IET and BCS and Senior Member of IEEE and ACM.