Abstract— One of the main reliability concerns in the nanoscale logic is the time-dependent variation caused by Negative Bias Temperature Instability (NBTI). It increases the switching threshold voltage of pMOS transistors and as a result slows down signal propagation along the paths between flip-flops, thus causing functional failures in the circuit. In this paper we propose an approach to identify NBTI-critical gates in nanoscale logic. The method is based on static timing analysis that provides delay critical paths under NBTI-induced delay degradation. An analysis on these critical paths is performed in order to select the set of gates that have the highest influence on circuit aging. These gates are to be hardened against NBTI aging effects guaranteeing correct circuit behavior under the given timing and circuit lifetime constraints. The proposed approach is demonstrated on an industrial ALU circuit design.

Keywords—NBTI-critical gate, critical path identification, static timing analysis, aging, logic circuit.

I. INTRODUCTION

In Very Deep Sub-Micron (VDSM) technology, lifetime reliability has become one of the key design factors to guarantee CMOS integrated circuit robustness. One of the most critical downsides of technology scaling beyond the 65nm node is related to the non-determinism of the devices’ electrical parameters due to time-dependent deviations in the operating characteristics of the device. In [11] a brief overview of major sources of time-dependent variations is presented. One of them is Negative Bias Temperature Instability (NBTI) [1]. It is physical/chemical effect that causes a degradation of the oxide and results in a drift of the threshold voltage over time. NBTI has become the most prominent effect due to the fact that it creates interface states along with the entire silicon-oxide interface. NBTI has been shown to be the major reliability limiting factor when the gate oxide is thinner than 4 nm [2].

NBTI is defined as the effect that occurs when a pMOS transistor is negatively biased. The effect manifests itself as an increase of the threshold voltage $V_{th}$ over time. This results in drive current reduction and noise increase, which in turn causes a degradation of the device delay. The threshold variation is estimated to be 5-15% per year [3], [6] depending on the targeted technology and its environment, while the path delay degradation follows the same trend, though with a smaller magnitude. NBTI’s effect on the long-term stability of functional logic expresses itself through the incapability of storing a correct value at memory elements such as flip-flops due to the de-synchronization between clock distribution and signal propagation through the logic paths of a circuit. The de-synchronization effect is observed when the summation of all gate delays along a given path is larger than the time slack allocated by the designer for that path. When this condition happens, the propagated signal reaches the memory elements at the end of the path in a time instant later than the clock front and, as consequence, an incorrect value is stored at that point of the circuit.

The analysis of the NBTI effect is more complicated than other traditional reliability issues, such as hot-carrier injection [2], as it includes stress and recovery phases. The stress phase occurs when a pMOS transistor is under a negatively biased condition, i.e., $V_{GS} = -V_{DD}$. In this situation, the interaction between the inversion layer holes and the hydrogen-passivated $Si$ atoms breaks the $Si-H$ bond generated during the oxidation process. Then, the $H$ atom converts into $H_2$ molecules. When $H_2$ molecules diffuse away, interface traps are left. When interface traps accumulate between silicon and the gate oxide interface, they cause a shift in the threshold voltage $V_{th}$. However, in the recovery phase, when the biased voltage is removed, a reverse reaction is performed. Some hydrogen diffuses back toward the interface and bonds with $Si$, which reduces the number of interface traps and the NBTI effect. Although the recovery phase can reverse the NBTI effect, it does not eliminate all the interface traps generated during the stress phase, and the pMOS threshold voltage will increase in the long term.

Critical path identification is important to stress that the effects of static and dynamic NBTI differ dramatically. Because of the rapid annealing at the beginning stage of the recovery, even a small recovery period (i.e., signal probability $P_e$ very close to 1) reduces the overall degradation by more than 50% of the static
stress [10]. As static NBTI effects only manifest themselves in signal lines that are functionally redundant (i.e. constantly tied to zero), we neglect them in this paper and focus on dynamic NBTI only.

Previous works found in the literature have addressed the NBTI problem. Among them, [3] proposed an approach as a means of alleviating the NBTI-induced aging effects in static random access memories (SRAMs). Another approach [4] uses an experimentally verified NBTI model to study DC noise margins in conventional 6T SRAM cells as a function of NBTI degradation in the presence of process variations. Considering functional logic, authors proposed in [5] a transistor sizing technique that not only mitigates NBTI induced delay of the gate under consideration, but also minimizes its impact on the adjacent gates. This technique seems to be very effective, but it is mandatory to identify the critical gates and paths within the circuit in order to apply such technique. Otherwise, this technique should be applied to all the circuit gates, which would result in an unacceptable area overhead and eventually, excessive power consumption. In [7] the authors present an interesting method characterizing the noise margins in conventional 6T SRAM cells as a function of the signal probability \( P_z \) of each of its inputs. In the sequence, a technology-mapping technique that incorporates NBTI stress and recovery effects in order to ensure optimal performance of the circuit during its entire lifetime is applied. However, even though the use of this technique results in reasonable area and power savings with respect to the worst-case where the \( P_z \) for each gate is not taken into account (i.e., \( P_z \) is fixed to “1”), it must be applied to all nodes of the circuit in order to find the best-delay match for each gate as a function of the \( P_z \). This is the main difference between this technique and the one we propose herein: instead of indiscriminately replacing all the gates by their minimum-NBTI counterpart along with all the logic cones of the circuit, we propose first to identify the delay-critical paths along the circuit. Then, based on \( P_z \) and fanout information for the nodes settled along these critical paths, we select the NBTI-critical paths and only to this latter reduced set of gates and paths, apply NBTI-recovery strategies. Such strategy can be, for instance, the replacement of the gates by their minimum-NBTI counterpart as proposed by [7].

The approach proposed in this paper is hierarchical and consists of NBTI gate delay degradation analysis based on technology dependent information. This analysis is followed by gate-level methods to perform NBTI-aware static timing analysis and NBTI-critical gate identification. The remainder of the paper is organized as follows. Section 2 presents the calculation of NBTI-induced gate delay degradation based on the values of \( P_z \), \( F \), and the expected lifetime of the circuit. Section 3 provides the algorithms for NBTI-aware static timing analysis and NBTI-critical gate identification. It includes a motivational example showing the iterative gate replacement process. Section 4 presents a case study on an industrial ALU design. Section 5 concludes the paper.

II. NBTI-AWARE GATE DELAY DEGRADATION ANALYSIS

In this paper, we propose hierarchical modeling of dynamic NBTI aging, where for each signal \( x_i \), signal probabilities \( P_z(x_i) \) are calculated by gate-level logic simulation and the numbers of gate fanouts \( F(x_i) \) are derived by structural analysis at the gate netlist level, respectively. These parameters together with the expected operation time \( \Phi \) in years are applied as an input to the NBTI-aware gate delay degradation analysis.

Fig. 1 presents the hierarchical NBTI induced gate delay degradation analysis flow implemented in this paper, which takes place as follows. As a preprocessing step [13], complex gates are flattened into NAND, NOR and inverter gates (e.g. an AND gate will be represented by a NAND gate followed by an inverter gate) and nominal delays are calculated. Then, gate-level simulation calculating the signal probabilities \( P_z(x_i) \) for all inputs \( x_i \) of the stages is performed which is followed by structural analysis providing the number of fanouts \( F(x_i) \) for the stages that are driven by signals \( x_i \). These parameters together with the circuit lifetime \( \Phi \) (in years) are included to the NBTI-induced gate delay degradation analysis.

The nominal delays \( d(G_i) \) in the logic gates \( G_i \) were calculated using the relative approach presented in [8]. Calculation of the NBTI-induced delay degradation \( d(G_i) \) is technology dependent. In this paper, we applied the data for 65 nm technology from [10] as follows. Delay \( d \) was calculated for each input signal \( x_i \) separately. The voltage threshold shift \( \Delta V_{th}(x_i) \) was calculated as follows:

\[
\Delta V_{th}(x_i) = (\alpha \cdot P_z(x_i))^\beta
\]

where \( P_z(x_i) \) is the signal probability for input signal \( x_i \) and \( \alpha \) and \( \beta \) are technology dependent constants that were set to 0.018668 and 15·10⁻⁷ for 1 year of aging and to 115·10⁻⁷ for 10 years of aging, respectively.

Further, we calculated the gate delay \( t_{pd} \) taking into account NBTI degradation caused by \( P_z \):

\[
t_{pd} = t_{g} \cdot (1+(\lambda \cdot \Delta V_{th}(x_i))^\mu/(\mu \cdot (\Delta V_{th}(x_i))^\mu))}
\]

where \( t_{g} \) is the nominal gate delay for the gate \( G \) and \( \lambda \) and \( \mu \) are technology dependent constants. (In our experiments \( \lambda \) and \( \mu \) are set to 205 and 20 the NOR gate, to 185 and 18 for the NAND gate and to 195 and 19 for the inverter, respectively).

Finally, in order to calculate the overall gate delay \( t \) related to
input signal $x_i$ we need to take into account also the number of fanouts $F_{G_i}$ for the gate $G$.

$$t_i = t_{pi} \cdot (1 + \phi(F_{G_i} - 1)/100)$$

where $\phi$ is a technology dependent constant and is set to 1.28 in our case. The proposed functions closely match the dynamic NBTI data from [10]. These functions were implemented in the gate-level aging simulator to provide extremely fast evaluation for NBTI-induced delay degradation.

The obtained delay and delay degradation data were applied by the method of identifying NBTI-critical gates presented in the next section.

III. IDENTIFYING NBTI-CRITICAL GATES

In this Section, a method for identifying the most influential gates with respect to NBTI-induced aging to be replaced by their NBTI-robust counterparts is proposed. For that purpose we need to calculate for each gate $G_i$ first, the estimates of NBTI-induced additional delays $\hat{t}(G_i)$ for each $i$-th input of $G_i$, and second, the number of critical paths $N_{OUT}(G_i) \setminus N_{IN}(G_i)$ which cross each input of the gate $G_i$. Here, $N_{IN}(G_i)$ – means the number of paths from the inputs of the circuit which converge at the $i$-th input of the gate $G_i$, and $N_{OUT}(G_i)$ – means the number of paths which start at the output of $G_i$ and spread up to the outputs of the circuit. Then we can calculate the weight of the gate $G_i$ as

$$W(G_i) = N_{OUT}(G_i) \sum_{i=1}^{N_{IN}(G_i)} \hat{t}(G_{i,i})$$ (1)

which characterizes the impact of replacing the gate on the aging reduction for all the critical paths in the circuit which cross the gate $G_i$. Here, $m_i$ is the number of inputs of the gate $G_i$. All the gates are ranked in descending order of the value, and the gates with highest values of $W(G_i)$ can be selected for replacement. The gate replacement procedure can be repeated iteratively until all the delays of the selected critical paths will fall under the given threshold.

Let have the following notations:

- $G_{\emptyset}$ – is the given time-slew threshold, i.e. maximum allowed path delay in the circuit,
- $C_{IN}(G_i)$ – is the input cone of the gate $G_i$ in the given circuit with the bottom of the cone on inputs, and with $G_i$ on the top of the cone,
- $C_{OUT}(G_i)$ – is the output cone of the gate $G_i$ with the bottom on outputs of the circuit, and with $G_i$ again on the top of the cone,
- $d(G_i)$ – is the nominal delay of the fresh gate $G_i$ without considering aging (i.e. its delay at time zero);
- $\hat{t}(G_{i,i})$ – is the increase in the delay of the gate $G_i$ from the $i$-th input to the output of the gate caused by NBTI-induced aging;
- $t(G_i)$ – is the total maximum delay of the gate $G_i$ over all of its inputs, when taking into account NBTI-induced aging;
- $D(G_i)$ – is the delay calculated for the slowest signal path in the cone $C_{IN}(G_i)$ based on the values of $t(G_{i,i})$ for all gates on this path;

$$D(G_i) = \max \{ t(G_{i,i}), t(G_{i,2}), ..., t(G_{i,n}) \}$$

$$D(G_i) = \max \{ (D(G_i) + \hat{t}(G_{i,i})) | G_i \in I_{N}(G_i) \}$$

where $I_{N}(G_i)$ is the set of input gates of $G_i$, and $t(G_{i,i})$ is the total delay of the gate $G_i$ from the output of the gate $G_i$ caused by aging;

- $\Theta(G_i)$ – is the threshold for the value of delay $D(G_i)$ at the output of gate $G_i$, which means that if this delay exceeds the value of $\Delta = D(G_i) - \Theta(G_i)$ then there will be the same overflow of delay $\Delta$ over $\Theta(G_i)$ at least on one primary output;
- $l_{CR}$ – is the set of critical paths consisting of gates $G^*$, where for each $G_i \in G^*$, the following is valid: $D(G_i) \geq \Theta(G_i)$;

- $N_{IN}(G_i)$ – is the number of all critical signal paths $l_{CR}$ in the input cone $C_{IN}(G_i)$ from all inputs of $C_{IN}(G_i)$ up to the gate $G_i$, so that $G_i \in G^*$ for all $G_i \in l_{CR}$;

- $N_{OUT}(G_i)$ – is the number of all critical signal paths $l_{CR}$ in the output cone $C_{OUT}(G_i)$ from the gate $G_i$ up to the all of outputs $C_{OUT}(G_i)$, so that $G_i \in G^*$ for all $G_i \in l_{CR}$;

The method consists of the following steps:

1) Calculation of the values of $D(G_i)$ for all the gates of the circuit based on the estimates of $t(G_i)$ (Algorithm 1: NBTI-aware static timing analysis);
2) Calculation of the values of thresholds $\Theta(G_i)$ for all the gates of the circuit based on the values of $\Theta$ and $D(G_i)$ (Algorithm 2);
3) Finding the number $N_{IN}(G_i)$ of critical paths $l_{CR}$ in the cone $C_{IN}(G_i)$ (Algorithm 3);
4) Finding the number $N_{OUT}(G_i)$ of critical paths $l_{CR}$ in the cone $C_{OUT}(G_i)$ (Algorithm 4);
5) Calculating the set of gates $G^*$ lying on the critical paths $l_{CR}$. The condition of including $G_i$ into $G^*$ is $D(G_i) \geq \Theta(G_i)$ (Algorithm 4);
6) Calculating the weights $W(G_i)$ for all the gates $G_i \in G^*$ according to the formula (1). (Algorithm 4);
7) Ranking the gates $G_i \in G^*$ in descending order of values $W(G_i)$;
8) Replacement of the most NBTI-critical gates of the circuit by their NBTI-robust counterparts.

After the replacement of the first ranked gate $G^* \in G^*$ by its NBTI-robust counterpart the values of $t(G^*_i)$ will reduce. The values of $D(G_i)$ and $\Theta(G_i)$ will be recalculated, and the set $G^*$ will be updated. If $G^* = \emptyset$, then the task of gate replacement is finished. Otherwise, we have to repeat iteratively the described procedure until we get the result $G^* = \emptyset$ or until an allowed cost limit of the gate replacements is reached.
Consider a combinational circuit as a network of gates where all the gates have numbers which show the ranking of gates in a partial order such that:

(1) all the input gates are numbered in an arbitrary order,
(2) all other gates may get their numbers only if all their predecessor gates have already got their numbers.

In the following we present the algorithms for calculating $D(G_k)$, $θ(G_k)$, $G^*$, $N_{IN}(G_k)$, $N_{OUT}(G_k)$ and $W(G_k)$. All of them are based on processing the gate network, gate by gate, either from inputs to outputs (in the case of $D(G_k)$ and $N_{IN}(G_k)$), or vice versa (in the case of $θ(G_k)$, $N_{OUT}(G_k)$, and $G^*$). Let us introduce the following notations:

$NG$ – is the number of gates in the circuit;
$PI$ and $PO$ – are the sets of gates connected to primary inputs and outputs, respectively.

$IN(G_k)$ and $OUT(G_k)$ – are the sets of input and output gates of $G_k$, respectively.

**Algorithm 1.** NBTI-aware static timing analysis

FOR all gates $G_k$, $k = 1, 2, \ldots, NG$:

IF $G_k ∈ PI$ THEN

$D(G_k) = τ(G_k)$

ELSE

$D(G_k) = \max \{D(G_i) + τ(G_{i,k})| G_i \in IN(G_k)\}$

END IF

END FOR

**Algorithm 2.** Calculation of $θ(G_k)$ and $G^*$

Initialize: $G^* = \emptyset$

FOR all gates $G_k$, $k = NG, NG - 1, \ldots, 2, 1$:

IF $G_k ∈ PO$ THEN

$θ(G_k) = θ^*$

ELSE

$θ(G_k) = \min \{θ(G_i) - D(G_i)| G_i \in OUT(G_k)\}$

END IF

IF $D(G_k) ≥ θ(G_k)$ THEN

$G^* = G^* \cup G_k$

END IF

END FOR

**Algorithm 3.** Calculation of $N_{IN}(G_k)$

FOR all gates $G_k ∈ G^*$, $k = 1, 2, \ldots, NG$:

$N_{IN}(G_k) = 0$

FOR all inputs $i$ of the gate $G_k$:

IF $i$ is a primary input THEN

$N_{IN}(G_k) = N_{IN}(G_k) + 1$

ELSE

$N_{IN}(G_k) = N_{IN}(G_k) + N_{IN}(G_i)$

END IF

END FOR

END FOR

**Algorithm 4.** Calculation of $N_{OUT}(G_k)$ and $W(G_k)$

FOR all gates $G_k ∈ G^*$, $k = NG, NG - 1, \ldots, 2, 1$:

IF $G_k ∈ PO$, THEN

$N_{OUT}(G_k) = 1$

ELSE

$N_{OUT}(G_k) = \text{SUM} \{N_{OUT}(G_i)| G_i \in OUT(G_k)\}$

END IF

END FOR

END FOR

$W(G_k) = N_{OUT}(G_k) \sum _{i=1}^{N_{IN}(G_{l,j})} N_{IN}(G_{l,j}) \cdot τ(G_{l,j})$

**Example 1.**

Consider a circuit netlist in Fig.2 where the components are arbitrary logic gates. Table 1 illustrates the Algorithms 1 - 4. We assume that the input data for the values $d(G_k)$ and $τ(G_{i,k})$ for all the gates are precalculated by the approach described in Section 2. In Fig. 2, the values of $d(G_k)$ and $τ(G_{i,k})$ are depicted on the outputs and inputs of gates, respectively.

![Fig.2. Example for NBTI-critical gate analysis of a circuit](image)

In the rows 1 and 2 in Table 1 the values of $D(G_k)$ and $θ(G_k)$ are calculated by Algorithms 1 and 2, respectively. On the basis of values $θ(G_k)$ we define the set of critical gates $G^* = \{G_2, G_5, G_{10}, G_3, G_8, G_{10}, G_{11}\}$. The rows 3 and 4 depict the results of the Algorithm 3 which calculates the values of $N_{IN}(G_k)$ in Formula (1). Finally, the rows 5 and 6 show the results of the Algorithm 4 which calculates the values of $N_{OUT}(G_k)$ in Formula (1). The Algorithms 3 and 4 process only the critical gates, i.e. $G^*$.

After finding the weights $W(G_k)$ we rank the gates $G_k ∈ G^*$ as follows: $G^*_{\text{ranked}} = \{G_3/24, G_2/20, G_5/12, G_{10}/11/8, G_6, G_7/4\}$. For all these gates the threshold is surpassed (i.e. $D(G_k) ≥ θ(G_k)$). Critical paths through the gates $G_k ∈ G^*$ are shown in Fig.1 by bold red lines.

**Table 1.** NBTI ANALYSIS AND REPLACEMENT RESULTS FOR Fig.2

<table>
<thead>
<tr>
<th>$G_k$</th>
<th>$G_{l,j}$</th>
<th>$G_{l,j}$</th>
<th>$G_{l,j}$</th>
<th>$G_{l,j}$</th>
<th>$G_{l,j}$</th>
<th>$G_{l,j}$</th>
<th>$G_{l,j}$</th>
<th>$G_{l,j}$</th>
<th>$G_{l,j}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>D</td>
<td>6</td>
<td>10</td>
<td>19</td>
<td>18</td>
<td>21</td>
<td>28</td>
<td>29</td>
<td>41</td>
</tr>
<tr>
<td>2</td>
<td>$θ^*$</td>
<td>9</td>
<td>8</td>
<td>49</td>
<td>47</td>
<td>20</td>
<td>26</td>
<td>27</td>
<td>39</td>
</tr>
<tr>
<td>3</td>
<td>$N_{IN}$</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>SUM</td>
<td>-</td>
<td>3</td>
<td>-</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>$N_{OUT}$</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>$W$</td>
<td>-</td>
<td>12</td>
<td>-</td>
<td>24</td>
<td>4</td>
<td>4</td>
<td>20</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>$D(G)$</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>20</td>
<td>26</td>
<td>27</td>
<td>39</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>$θ(G)$</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>20</td>
<td>26</td>
<td>27</td>
<td>39</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>$D(G)$</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>20</td>
<td>26</td>
<td>27</td>
<td>37</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>$θ(G)$</td>
<td>-</td>
<td>11</td>
<td>-</td>
<td>23</td>
<td>28</td>
<td>29</td>
<td>39</td>
<td>-</td>
</tr>
</tbody>
</table>

Gates $G_3$ and $G_7$ that receive the highest values of $W(G_k)$ (24 and 20, respectively) are highlighted in Fig.1. Let us assume that after replacing the gate $G_3$ by its NBTI-robust
counterpart, we update the value $\tau_2(\text{G}_3) = 3$ to $\tau_2(\text{G}_3) = 1$, and recalculate the values of $D(\text{G}_i)$ and $\Theta(\text{G}_i)$. The results are depicted in the rows 7 and 8 (i.e. $D^*$ and $\Theta^*$), respectively. As we can see, the threshold is still surpassed for the gates $\text{G}_2$, $\text{G}_5$, $\text{G}_6$, $\text{G}_8$, and $\text{G}_{10}$ but not anymore for $\text{G}_7$ and $\text{G}_{11}$. Assume that after replacing the gate $\text{G}_8$, its NBTI-induced delay degradation drops to $\tau_1(\text{G}_8) = 1$ and $\tau_2(\text{G}_8) = 1$, respectively. Consequently, the delays for all the gates become lower than the threshold (compare the rows 9 and 10, i.e. $D^*$ and $\Theta^*$), i.e. replacing the two gates guarantees that the NBTI-induced delay degradation will not violate the given time-slack $\Theta^*$ under the circuit lifetime constraint.

IV. CASE STUDY ON AN ALU CIRCUIT

The proposed approach is demonstrated on an ALU_4bit circuit shown in Fig. 3, which is a design of the 4-bit ALU circuit 74HC/HCT181 from Philips [9]. It is a gate-level combinational logic design (as outlined by the dashed border). According to the specification of the circuit all input stimuli are possible. Therefore large number of pseudo-random input patterns has been generated and selected as possible functional sequence for input stimuli.

When nominal delays of individual gates as well as delay degradation caused by 10-year NBTI effect have been calculated by the method presented in Section 2, Algorithm 1 is applied to estimate the slowest signal path for each output by static timing analysis taking into account the delay degradation caused by NBTI effect. Note that the delay units $d(\text{G}_i)$ are relative and technology independent whereas for the delay degradation percentage $\tau(\text{G}_{i,k})$, actual data from a 65 nm technology from [10] is applied. Further, Algorithms 2 to 4 are applied to obtain the list of NBTI-critical paths whose delays exceed maximum allowed delay $\Theta^*$ in the circuit. In the study, we set the $\Theta^*$ equal to 228.8, as a meaningful example.

In Fig. 4 the most delayed paths for different outputs that become delay critical due to NBTI effect are depicted. In all, the list contains 93 paths, whereas only paths to outputs $A=B$, $F_3$, $F_2$, and $F_1$ become NBTI-critical. In Fig. 3 the slowest path to each of these outputs through gates is shown. In Fig. 4 timing characteristics for selected paths are depicted. For instance, nominal propagation delay for Path 1 ($A=B$) equals 208. Due to calculated NBTI effect, delay degradation of Path 1 will exceed the maximum allowed delay $\Theta^*$ by 37.97 which makes it the most NBTI-critical path in the circuit.

Next, a list of gates from NBTI-critical paths is extracted and ranked based on weights computed applying Formula 1. Altogether, selected paths contain 49 different gates. In Table 2 10 most NBTI-critical gates are listed where their names, types, number of critical paths the gate appears, nominal delays and weights are shown. Also, in Fig. 3 these gates are marked by boxes and numbered to indicate their location in
the full experiment of ranking NBPTI-critical gates consumed only 0.67 seconds on a dual core 2.2 GHz CPU.

![Fig. 4 NBPTI-critical gates that exceed the maximum allowed delay $\Theta^*$](image)

It is important to stress that the approach considers different cases of NBPTI-critical gates. For example, the gates 156 and 182 are critical because they include a large input cone $C_I$ of critical paths $l_{CR}$. The gates 127 and 134, in turn, include a large output cone $C_{OUT}$ of critical paths. Finally, gates 132 and 135 influence a small number of critical paths (only 8). However, they are critical due to the fact that their NBPTI-degradation is high.

According to Table 2, gate 156 is the first candidate to be replaced with its NBPTI-robust counterpart so as to mitigate its delay degradation and reduce number of NBPTI-critical paths. After one gate replacement delay degradation of each NBPTI-critical path is recalculated. Then the list of these paths is updated by excluding the paths which do not exceed threshold $\Theta^*$ anymore. The process of gate replacement is repeated until all listed NBPTI-critical paths become non-critical or until an allowed cost limit of the gate replacements is reached.

### Table 2. List of NBPTI-Critical Gates Ranked By Weights

<table>
<thead>
<tr>
<th>#</th>
<th>Gate Type</th>
<th># of paths $l_{CR}$ ($N_{IN}N_{OUT}$)</th>
<th>Nominal delay $d(G_k)$</th>
<th>NBPTI-degradation $\tau(G_k)$</th>
<th>Weights $W(G_k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I56 XOR</td>
<td>54 48</td>
<td>11.2 -12.8</td>
<td>607.29</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>I27 NOR</td>
<td>32 46</td>
<td>18.4</td>
<td>588.55</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>I34 NOR</td>
<td>31 46</td>
<td>18.6</td>
<td>575.48</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>I82 AND</td>
<td>63 24</td>
<td>5.0 -5.1</td>
<td>327.30</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>I64 NOR</td>
<td>53 31</td>
<td>6.0 -6.1</td>
<td>321.75</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>I67 XOR</td>
<td>17 48</td>
<td>11.9 -12.6</td>
<td>205.30</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>I60 AND</td>
<td>25 28</td>
<td>5.0 -6.0</td>
<td>129.18</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>I32 NOR</td>
<td>8 44</td>
<td>13.7 -15</td>
<td>119.91</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>I35 NOR</td>
<td>8 36</td>
<td>11.1 -12.9</td>
<td>105.56</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>I72 XOR</td>
<td>8 48</td>
<td>3.3 -4.4</td>
<td>99.56</td>
<td></td>
</tr>
</tbody>
</table>

### ACKNOWLEDGMENTS

The work has been supported in part by CNPq (Science and Technology Foundation, Brazil) under contracts n. 303701/2011-0 (PQ) and n. 483037/2011-7 (Universal), by EU’s FP7 STREP project BASTION, by Estonian ICT program project FUSETEST, by Research Centre CEBE funded by European Union through the European Structural Funds and by Estonian SF grants 8478 and 9429.

### REFERENCES


V. CONCLUSIONS

In this paper, we introduce an approach to identifying NBPTI-critical gates to be hardened against aging effects thereby guaranteeing correct circuit behavior under the given timing and circuit lifetime constraints.

The approach is hierarchical and consists of NBPTI gate delay degradation analysis based on deriving functions matching the dynamic NBPTI data from [10]. These functions were implemented in the gate-level aging simulator to provide extremely fast evaluation for NBPTI-induced delay degradation.

This analysis is followed by gate-level methods to perform NBPTI-aware static timing analysis and a novel method identifying NBPTI-critical gates to be replaced by their minimum-NBPTI counterparts.

The proposed approach was demonstrated on an industrial ALU circuit. The results have shown that the proposed approach requires a very short run-time. Furthermore, it was shown that it is able to detect all the three different types of criticality, i.e. gates that are critical because they influence a large input cone, gates that are critical as they influence a large output cone, and gates that have excessive delay degradation values.

The work has been supported in part by CNPq (Science and Technology Foundation, Brazil) under contracts n. 303701/2011-0 (PQ) and n. 483037/2011-7 (Universal), by EU’s FP7 STREP project BASTION, by Estonian ICT program project FUSETEST, by Research Centre CEBE funded by European Union through the European Structural Funds and by Estonian SF grants 8478 and 9429.