Identification and Rejuvenation of NBTI-Critical Logic Paths in Nanoscale Circuits

Identification and Rejuvenation of NBTI-Critical Logic Paths in Nanoscale Circuits

Maksim Jenihhin[1], Giovanni Squillero[2], Thiago Santos Copetti[3], Valentin Tihhomirov[1], Sergei Kostin[1], Marco Gaudesi[2], Fabian Vargas[3], Jaan Raik[1], Matteo Sonza Reorda[2], Leticia Bolzani Poehls[3], Raimund Ubar[1], Guilherme Cardoso Medeiros[3].

[1]*Department of Computer Engineering, Tallinn University of Technology,*
Akadeemia 15A, Tallinn 12618, Estonia
E-mails: {maksim | valentin | skostin | jaan | raiub}@ati.ttu.ee

[2]*Politecnico di Torino, Department of Control and Computer Engineering,*
Corso Duca degli Abruzzi, 24, Torino 10129, Italy
E-mails: {giovanni.squillero | marco.gaudesi | matteo.sonzareorda} @polito.it

[3]*Catholic University – PUCRS,*
Av. Ipiranga, 6681, Porto Alegre 90619-900, Brazil
E-mails: thiago.copetti@acad.pucrs.br, {vargas | leticia.poehls}@pucrs.br

Abstract The Negative Bias Temperature Instability (NBTI) phenomenon is agreed to be one of the main reliability concerns in nanoscale circuits. It increases the threshold voltage of pMOS transistors, thus, slows down signal propagation along logic paths between flip-flops. NBTI may cause intermittent faults and, ultimately, the circuit's permanent functional failures. In this paper, we propose an innovative NBTI mitigation approach by rejuvenating the nanoscale logic along NBTI-critical paths. The method is based on hierarchical identification of NBTI-critical paths and the generation of rejuvenation stimuli using an Evolutionary Algorithm. A new, fast, yet accurate model for computation of NBTI-induced delays at gate-level is developed. This model is based on intensive SPICE simulations of individual gates. The generated rejuvenation stimuli are used to drive those pMOS transistors to the recovery phase, which are the most critical for the NBTI-induced path delay. It is intended to apply the rejuvenation procedure to the circuit, as an execution overhead, periodically. Experimental results performed on a set of designs demonstrate reduction of NBTI-induced delays by up to two times with an execution overhead of 0.1% or less. The proposed approach is aimed at extending the reliable lifetime of nanoelectronics.

Keywords *hardware rejuvenation, aging, NBTI, critical path identification, logic circuit, evolutionary computation, MicroGP, zamiaCAD.*

1. Introduction

Guaranteeing lifetime reliability is a key challenge in current nanoscale semiconductor manufacturing processes. One of the most critical downsides of technology scaling beyond the 65nm node is the non-determinism of the devices' electrical parameters caused by time-dependent

variations [1] in the operating characteristics of the device. Two essential sources of time-dependent variations have been identified: Bias Temperature Instability (BTI), and Hot Carrier Injection (HCI) [2]. These physical/chemical effects result in the degradation of the oxide thus causing a drift of the Threshold Voltage (V_{TH}) over time. In terms of magnitude, BTI has become the most prominent effect. In fact, BTI creates the interface traps along the entire silicon-oxide interface and is thus sensitive to temperature and the vertical electric field. On the contrary, hot carrier generation only damages the interface near the drain side, which makes it basically depend on the lateral electric field. As devices shrink, the influence of the temperature and the vertical electric field has largely surpassed the influence of the lateral field [5].

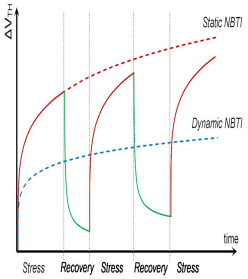
BTI manifests in two distinct forms, depending on the type of transistor involved: Negative BTI (NBTI), which affects pMOS transistors, and the its counterpart Positive BTI (PBTI), which affects nMOS devices. In current technologies, the impact of PBTI is much lower than NBTI. Therefore, this paper specifically addresses the *Negative Bias Temperature Instability (NBTI)* phenomenon [4]. It is worth mentioning that the importance of PBTI is expected to increase, particularly with the adoption of high-k, Hafnium-based dielectrics in the gate-oxide for leakage reduction [3].

1.1 Preliminaries

NBTI is defined as the effect that occurs when a pMOS transistor is negatively biased. The effect manifests itself as an increase of the pMOS transistor threshold voltage $|V_{THp}|$ over time. This leads to drive current reduction and noise increase, which in turn causes a degradation of the device delay. NBTI's impact on the long-term stability of functional logic is expressed through the incapability of storing a correct value in memory elements such as flip-flops. This effect is due to the de-synchronization between clock distribution and signal propagation through logic paths of a circuit. Therefore, after several years of circuit operation time, the NBTI-induced delays may cause, first, intermittent faults and, ultimately, permanent functional failures in the circuit [6].

The analysis of the NBTI effect is more complicated than other traditional reliability issues, e.g., HCI [11]. Despite of several theories proposed over the last decade in order to explain NBTI degradation, common understanding of the process in the scientific community is still subject for research. Nowadays, two widely accepted theories coexist, namely the *reaction-diffusion model* (R-D) [4],[5] and the *trapping/detrapping model* (T-D) [7],[8]. In this paper, we will rely on the R-D model. Generally, NBTI includes *stress* and *recovery* phases (see Fig. 1a). The stress phase occurs when a pMOS transistor is in a negatively biased condition, i.e., $V_{GS} = -V_{DD}$ (see Fig. 1b for an example of NBTI in a CMOS invertor gate). However, when the biased voltage is removed, i.e., $V_{GS} = 0$, the pMOS transistor is in the recovery phase and the NBTI effect is partially reversed. The V_{THp} will still increase over time, however in case of sufficient in the recovery phase, the aging process may be slowed down considerably.

a)



b)

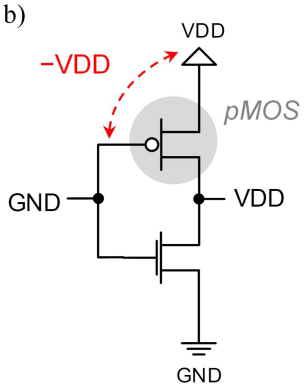


Fig. 1 (a) Illustration of NBTI stress and recovery phases; (b) CMOS invertor gate under NBTI stress.

The variation of V_{THp} of pMOS transistors due to *dynamic NBTI* (i.e. when the stress and recovery phases are iterating) is estimated to be 5-15% per year [12],[13]. The exact value depends on the targeted technology and the environment (e.g. ambient temperature and user workload). The V_{THp} shift due to *static NBTI* (i.e. when a transistor is under constant stress) can be significantly higher. The path delay degradation follows the same trend, though with a smaller magnitude. It has been shown that NBTI depends on many factors [16], but its strongest correlation is with the *signal probability* P_z (input duty cycle). The signal probability $P_z(x_i)$ for a logic gate's input x_i is defined as the ratio of time during which the input signal x_i is set to logic 0.

1.2 Contributions

In this paper, we propose a novel approach to mitigate NBTI using *rejuvenation* of nanoscale logic at NBTI-critical paths with dedicated stimuli sequences. The method is based on fast and hierarchical identification of NBTI-critical paths at gate level and the rejuvenation stimuli generation using an Evolutionary Algorithm. Note that this work does not aim at contributing to the development of a new transistor-level model for the underlying NBTI physical and chemical processes and assumes the existing models as an input.

First, based on SPICE electrical simulations, an *accurate gate-level model for fast computation of NBTI-induced path delay degradation* Δt_{path} and *identification of NBTI-critical paths* is developed. In more detail, this method is based on the estimation of NBTI-induced delays for individual gates Δt_{gate} along selected timing-critical paths followed by a static timing analysis. Experiments with an industrial *ALU* circuit expose the good match of the gate-level approach to the electrical simulation results with the simulation speed-up of several orders of magnitude.

Second, an approach to create *rejuvenation stimuli* for the identified NBTI-critical paths using an *evolutionary stimuli generation algorithm* is proposed. In this paper, we exploit a general-purpose evolutionary toolkit called *µGP* [36],[37] and find a suitable fitness function using an open source hardware analysis framework *zamiaCAD* [39]. The advantage of such flow lies in its flexibility for solving the dependencies of impacts by individual gates to the most critical NBTI-induced path delay by using evolutionary optimization processes.

The generated rejuvenation stimuli are applied at predefined periods in order to drive pMOS transistors to the recovery phase. Thus, the proposed approach aims at extending the reliable lifetime of nanoelectronics.

The remainder of the paper is organized as follows. Section 2 provides an overview of the related work. Section 3 introduces SPICE-inspired models for NBTI-induced gate delay calculation. Section 4 proposes a method for fast gate-level identification of NBTI-critical logic paths. Section 5 introduces a flow for evolutionary generation of rejuvenation stimuli and presents corresponding experimental results. Section 6 discusses applicability and limitations the rejuvenation procedure. Section 7 concludes for the paper.

2. Overview of Related Work

Previous works appearing in the literature address the NBTI problem both for memories [12],[18] and for functional logic. Usually, to mitigate the impact of NBTI on the circuit's lifetime these approaches adopt *redesign* strategies, *voltage and frequency scaling* and *internal node control* guided by monitoring attributes or design structure analysis.

The work in [19] proposes a *redesign approach* for functional logic *based on 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 appears to be very effective, but it is mandatory to provide the critical gates and paths to which it should be applied. Otherwise, this technique will result in an unacceptable area overhead and eventually, excessive power consumption. In [20] the authors present a method, which characterizes the delay of every gate in a standard cell library as a function of the signal probability (P_z) of each of its inputs and suggests an *NBTI-aware synthesis* accordingly. It demonstrates an average of 10% area recovery for 65nm technology under the pessimistic assumption that *all pMOS transistors in the design are under constant static NBTI stress*. Although the calculation process in [17] and [20], allowing the derivation of aging curves for logic components, is handy, it is also prohibitively time consuming. This is due to an extremely large number of stress recovery cycles that have to be computed. The work in [21] proposes an approach for temporarily hiding NBTI-induced aging by applying *changes to voltage and frequency* of the circuit.

Approaches to analyze the efficiency of *controlling input signal probability* for mitigating NBTI at circuit level were proposed in [22],[23]. Works in [24],[25] propose to exploit the idle time of processors and unused bits in source operands [26]. A very relevant approach for processor circuits' rejuvenation is presented in [27], where authors propose to replace the default NOP with a special "maximum aging reduction NOP instruction" that, while having no effect on the program state, minimizes the NBTI effect. The results show that this method can extend circuit lifetime by an average of 37%, with performance, power, and area overhead within 1%.

Different from state-of-the-art, in this paper, a novel approach of approximation by *mathematically convenient functions* is used to calculate the gate and path delay degradations caused by NBTI aging. In addition, to the best of our knowledge, it is the first time that evolutionary algorithms are applied to the task of rejuvenation stimuli generation.

The following points resume the advantages of the approach proposed in this paper:

a) it is based on fast, yet accurate hierarchical gate-level identification of NBTI-critical paths where rejuvenation has to be applied;

b) it proposes efficient rejuvenation stimuli generation using an evolutionary algorithm;

c) it does not require redesign and can be applied to an existing circuit, exploiting , if necessary, the existing design-for-testability instruments.

3. NBTI-Induced Gate Delay Models

In this section, a hierarchical modeling of NBTI aging process [29] is introduced. Here, the gate delay degradation is calculated by polynomials matching previously measured or simulated NBTI-degradation data on devices, e.g. [17] or [8], and SPICE simulations at transistor level. The NBTI-induced gate delay degradation analysis flow implemented in this paper can be represented by the following three steps:

- *Step 1*: Obtaining a curve for ΔV_{THp} as a function of P_z for selected environmental variables, technology and a given time of operation in years. (Section 3.1 presents the details for deriving the equation).
- *Step 2*: Performing extensive electrical simulations of individual gates in SPICE to obtain curves of degradation of delay Δt for each input of each gate type under the stressed conditions, i.e. when the corresponding input switches from 1 to 0. Δt will be represented as the percentage of change of the nominal delay t for the given gate. (Section 3.2 explains the process of obtaining the curves for Δt).
- *Step 3*: Developing approximated polynomial equations for the aging curves obtained in Steps 1 and 2. This step enables mathematically convenient calculation of NBTI-critical paths at the gate-level, as to be presented in Section 4.

Note, that as a preprocessing step, complex gates are flattened into NAND, NOR and inverter stages (e.g. an AND gate is represented by a NAND gate followed by an inverter gate).

3.1. Modeling dependency of pMOS transistor V_{THp} shift on signal probability P_z

In the NBTI effect analysis, we rely on a reaction-diffusion (R-D) mechanism based predictive model for dynamic NBTI presented in [9],[10] and verified with an industrial 65-nm technology as stated in [10]. The proposed model predicts the long term threshold voltage V_{THp} degradation due to NBTI at a time t > 1000s and is proven to be independent of the frequency at high frequencies [9]. It captures the dependence of NBTI on the gate oxide thickness t_{ox} of pMOS transistor, a variety of the dominant diffusion species (H or H_2) expressed by the time exponent parameter n, a duty cycle P_z (probability that pMOS transistor is under stress) in addition to its dependence on other key process and design parameters as presented in [9]:

$$|\Delta V_{THp}| \approx \left(\frac{n^2 K_v^2 P_z C t}{\xi_1^2 t_{ox}^2 (1-P_z)}\right)^n \quad (1)$$

where $C = T_o^{-1} exp(-E_a / kT)$, E_a is the activation energy of hydrogen species, k is the Boltzmann's constant, T – temperature, ξ_1 and T_o are technology dependent constants, and K_v is expressed by following formula [9]:

$$K_v = \left(\frac{q\, t_{ox}}{\varepsilon_{ox}}\right)^3 K_1^2\, C_{ox}\left(V_{gs} - V_{th}\right)\sqrt{C}\, \exp\left(\frac{2E_{ox}}{E_{o1}}\right) \quad (2)$$

where q is electron charge, $C_{ox} = \varepsilon_{ox} / t_{ox}$ is the oxide capacitance per unit area, in the strong inversion region the vertical electrical field $E_{ox} = (V_{gs} - V_{th}) / t_{ox}$; ε_{ox} K_1 and E_{01} are technology dependent constants.

Equation 1 is valid only for dynamic NBTI, since if P_z reaches the value 1, the value of ΔV_{THp} becomes infinite and the formula incorrect. Therefore, the upper limit of ΔV_{THp} is defined by following equation, which models only static NBTI [9]:

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

When values of all key design parameters are entered into Equation 1, then arithmetical operations with known values are performed and the result is stored in parameter γ, the equation gains the following form:

$$|\,\Delta V_{THp}\,| = \gamma\left(\frac{P_z}{1-P_z}\right)^n \quad (4)$$

Equation 4 represents a mathematically convenient function of threshold voltage V_{THp} degradation dependence on the signal probability for input signal $P_z(x_i)$ of pMOS transistor, where n = 1/6 and γ = 0.0904. In Fig. 2 the corresponding dependence is illustrated for PTM 65 nm technology after 10 years of NBTI-induced degradation.

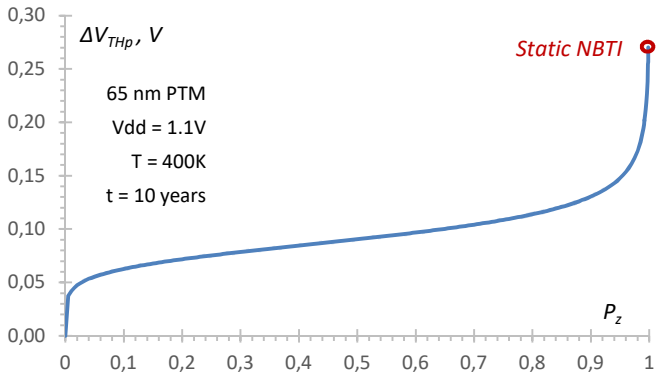


Fig. 2 Threshold voltage shift ΔV_{THp} as a mathematically convenient function of signal probability P_z

Alternative technologies or alternative silicon measurements data may result in variations of the function shape and, thus, changes of the coefficients' values. Moreover, threshold voltage V_{THp} degradation depends also on temperature T and supply voltage Vdd as shown in Fig. 3. However, the procedures of NBTI-critical paths identification and rejuvenation proposed in this paper maintain their validity. Equation (1) allows fast computation of NBTI-induced V_{THp} shift, as it depends on a signal probability at the inputs of the considered logic gate. These voltage shift values serve as input values for modeling the NBTI-induced gate delays.

3.2. Modeling NBTI-induced gate delay degradation based on electrical simulations in SPICE

Modeling of NBTI-induced gate delay degradation relies on intensive SPICE electrical simulations, in case of this work, performed using Synopsys HSPICE simulator. The selected technology was the 65nm PTM [30] with V_{DD} = 1.1V and V_{THp} = -0.365V. For an alternative technology, it would be necessary to repeat the same gate aging characterization procedure in SPICE.

The SPICE simulation process consisted of simulating the basic cells of the technology library for different V_{THp} shift values ΔV_{THp} in order to capture the dependence of gate output delay on V_{THp}. The output load capacity C_L for gate output was chosen to be 1.0fF in accordance with the selected 65nm technology. Fig. 3 displays the typical *n*- and *p*-networks device interconnections for *Inverter*, *NAND* and *NOR* gates considered for simulation.

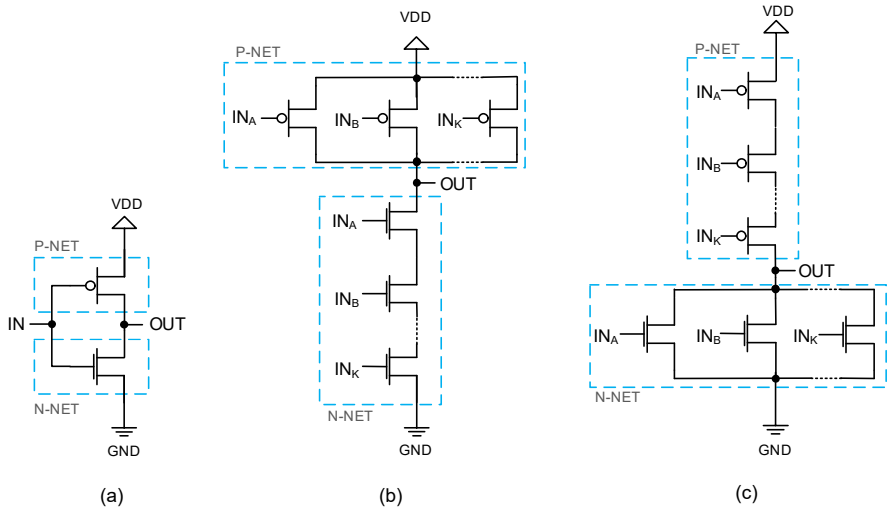


Fig. 3 Typical n- and p-networks displaying device interconnections for (a) Inverter, (b) NAND and (c) NOR gates considered in this work.

According to the Equation (3), the maximum V_{THp} shift under static NBTI can be up to 0.27V (for 10-year induced NBTI aging). Therefore, we have performed SPICE simulations increasing the $\Delta V_{THp}(x_i)$ value step-by-step from 0V to 0.27V applying a 2.5% sampling step to obtain gate output delay degradation for basic gates.

Fig. 4 shows the gate-aging characterization curve in SPICE for an Inverter. It captures the dependence of the gate output delay on ΔV_{THp} for the rising input transition 0→1.

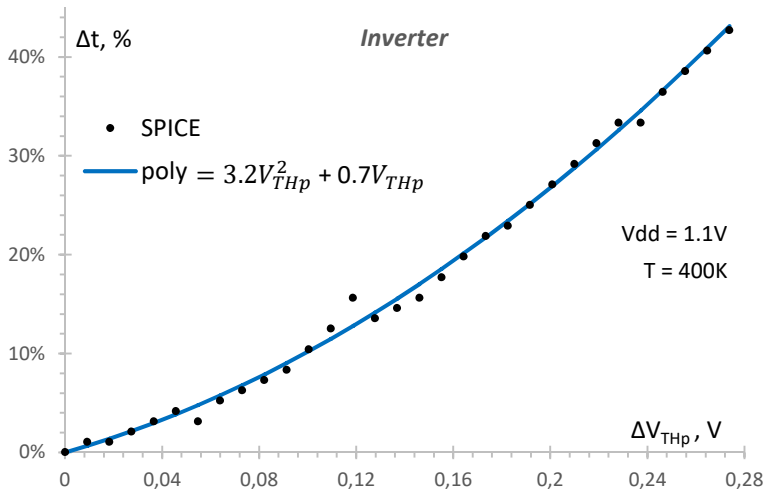


Fig. 4 Dependence of the gate output delay Δt on the V_{THp} shift ΔV_{THp} in an Inverter gate. Results of SPICE simulations (blue curve) and of the proposed mathematically convenient function (black dashed curve).

The following mathematically convenient function (black dashed curve) was matched to the curve produced by the characterization using SPICE (blue curve) in order to extract gate output delay dependency on NBTI-induced aging due to ΔV_{THp}:

$$\Delta t_{gate} = \lambda \cdot \Delta V_{THp}(x_i) + \mu \cdot \Delta V_{THp}(x_i)^2 \qquad (5)$$

where Δt_{gate} is the gate output delay increase (in percent) compared to nominal gate delay, $\Delta V_{THp}(x_i)$ is the change of V_{THp} for pMOS transistor at the gate input x_i, while λ and μ are technology dependent constants. In our experiments λ and μ are set to 1.63 and 5.3 for the Inverter gate. The maximum and average deviation values of the fitting function from the SPICE results were 4.22% and 1.19%, correspondingly.

In case of many-input gates embracing pMOS transistors networks (see Fig. 4), both the physical location of each pMOS transistor relative to the output node and the combination of gate inputs 1→0 transitions have impact on the level of gate delay degradation. Assume IN_A and IN_B are logic input values for a 2-input NAND gate switching as follows. In this case, 3 different curves exist to describe the NBTI effect in the gate depending on the input values case the gate output switches 0→1. These are depicted in Fig. 5a. Note, that our approach is pessimistic in case c), because, in fact, the two pMOS transistors will not degrade with the same speed.

a) IN_A= 0→1, IN_B = 1 (poly1);

b) IN_A= 1, IN_B = 0→1 (poly2);

c) IN_A= 0→1; IN_B = 0→1 (poly3).

The largest difference appears between the values of curves a) and b). It increases up to 15% when ΔV_{THp} surpases 0.24V that corresponds to a signal probability $P_z(x_i)$ very close to 1. Moreover, for $P_z(x_i)$ values below 0.9, the mean difference is still significant – up to 4%. This difference is caused by the stressed pMOS transistor location in the net (see Fig. 4) and identifies the necessity to consider which pMOS transistor of the gate is being under NBTI stress since it impacts on the gate output delay degradation.

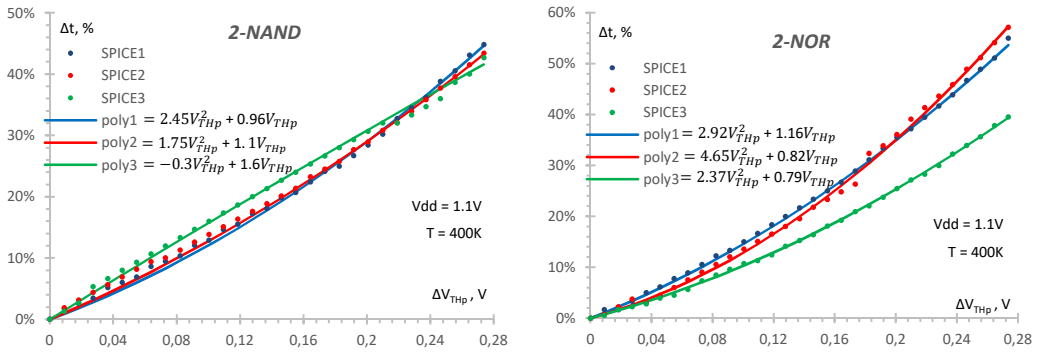


Fig. 5 NBTI impact on gate delay degradation for combinations of input values at a) 2-input NAND and b) 2-input NOR.

This dependence becomes more significant for NOR gates where pMOS transistors are connected in series (compared to the parallel connection in NAND gates). Fig. 5b presents corresponding curves for a 2-input NOR gate, where:

a) IN_A= 0→1, IN_B = 0 (poly1);

b) IN_A= 0, IN_B = 0→1 (poly2);

c) IN_A= 0→1; IN_B = 0→1 (poly3).

Here, the difference between curves b) and c) is over 20% if the V_{THp} is shifted by 0.2V and more.

The degradation variance is even more dramatic for logic gates with a larger number of inputs. Each combination of input values corresponds to an individual function of delay degradation Δt over V_{THp} shift. Considering a 4-input NOR gate output delay degradation described by 15 different curves presented in Fig. 6, where up to 4 simultaneous transitions of the 4 inputs are analyzed. In our approach, each curve is modelled by different values of constants λ and μ in Equation (2).

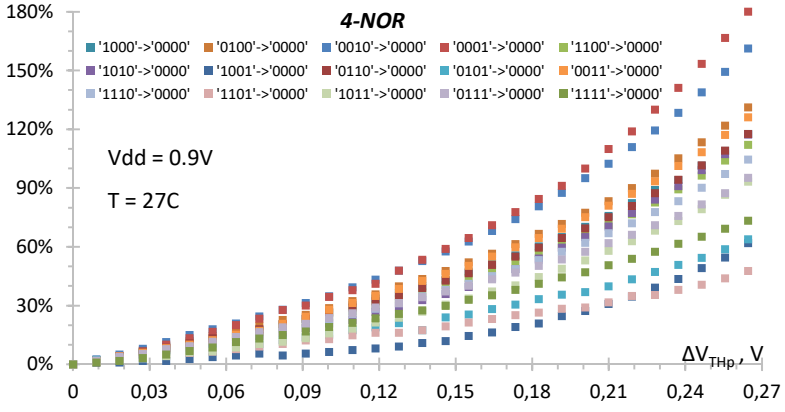


Fig. 6 SPICE simulation for 15 different scenarios of a 4-input NOR gate delay degradation.

The SPICE experiments show that only the 0→1 transition at gate inputs reveals the NBTI-induced gate delay, while the 1→0 transition is computed by the gate with no or negligible NBTI-induced gate delay Δt_{gate}. This can be explained by the fact that during the time the pMOS device in the p-network ages, it facilitates the task of discharging the gate output capacitance by

the nMOS device, placed in the n-network. The exceptions are NOR gates, especially the NOR gates with multiple inputs, where gate delay degradation Δt_{gate} for input 1→0 transition becomes even negative, i.e. the transition delay is decreased compared to the nominal one.

Environmental variables such as ambient temperature T have significant impact on the NBTI-induced delay. SPICE simulations for 27°C, 105°C and 400K (126.85°C) performed for the three basic gates are presented in Fig. 7.

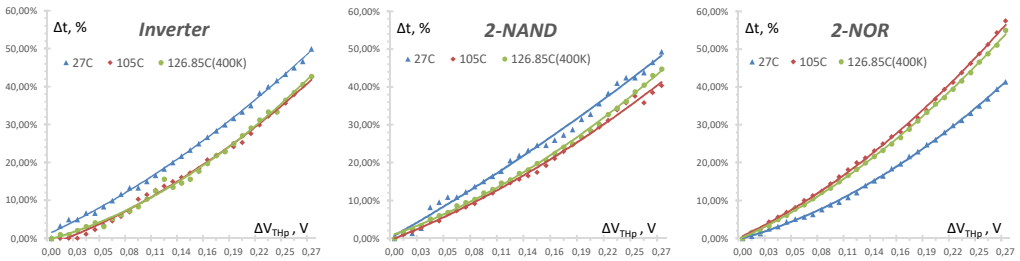


Fig. 7 Influence of the ambient temperature on NBTI-induced delays at basic gates.

Both the gate nominal delay and the NBTI-induced delay are impacted by the gate output load capacitance, which itself is defined by the circuit structure, e.g. the fan-out size at the gate output (available at the gate-level modelling stage), the length of wires (available at the layout floor planning phase). While its impact on the nominal delay is very significant, it was able to conclude from our SPICE simulations that output load capacitance does not significantly impact the NBTI-induced delay even in the case of realistic long wires, i.e. C_L is up to 10 fF (see Fig 8). As a simplification, our NBTI-induced path delay modelling approach does not explicitly model gate-output load capacitance for aging and applies it to nominal delay calculation only. Our experiments (see Section 4.3) demonstrate that this simplification yields acceptable NBTI-induced path delay estimation with inaccuracy lower than 2% when compared to SPICE simulations.

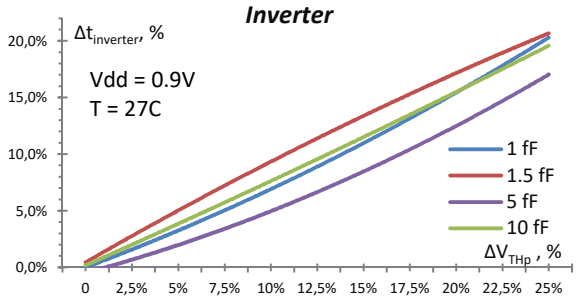


Fig. 8 Impact of output load capacitance on NBTI-induced delay

4. NBTI-Critical Logic Paths

The approximation of the curves to mathematically convenient polynomial equations proposed in Section 3 enables fast hierarchical identification of NBTI-critical logic paths at the gate-level. It is based on simulation of signal probabilities, static timing analysis with nominal delays and calculation of the longest NBTI-degraded path using NBTI-induced path delays. In order to explain the main concepts, let us introduce some basic definitions.

***Definition 1*:** An *NBTI-critical path* is a path in the circuit whose delay d is greater or equal than a ratio of B/D, where B is the *time budget* for the computations along the path to complete, i.e. the time during which a signal can arrive without making the clock cycle longer than desired and D is the coefficient by which a path can be slowed down by NBTI in the current technology and for the given workload.

For example, if it is known that in the particular case NBTI may cause up to 20% of delay degradation for the given time period of interest, then D is equal to 1.2.

***Definition 2*:** The *longest NBTI-degraded path* is the path that has the longest total delay when considering NBTI-induced additional delays Δt for the gates along that path. Here, NBTI-critical paths have to be analyzed both for 0→1 and 1→0 transitions at their primary inputs.

This Section is organized as follows. First, we introduce an algorithm to identify NBTI-critical logic paths and the longest NBTI-degraded path. This is followed by an example explaining the identification process. Finally, we provide experimental results for accuracy assessment of the hierarchical NBTI-critical paths calculation.

4.1. Identification of NBTI-critical logic paths

Consider a combinational gate-level logic circuit represented as a netlist $N = (G, X)$ where $G=\{ g_j|j=0,...,m\}$ is a set of gates and $X=\{x_i|i=0,...,n\}$ is a set of signal lines. The set X is partitioned into three subsets: a set of lines *IN* to represent the primary inputs of the network, a set of internal lines *FN* to represent the signal lines connecting the gates, and a set of lines *OUT* to represent the primary outputs of the netlist. An example of such a netlist is presented in Fig. 9.

1) The first task is to calculate the values of signal probabilities $P_z(x_i)$ for signal lines x_i by applying logic simulation of the expected workload to netlist N.

2) The next task is to characterize each signal line x_i of the circuit by a restricted number of L paths represented by pairs $M_k(x_i)=(d_k(x_i), s_k(x_i) = \{x_{PI}, ..., x_i\})$, where $k = 1, ..., L$ and x_{PI} is the primary input at the start of the corresponding path.

$d_k(x_i)$ – is the delay of the k-th selected path from the primary input x_{PI} to the line x_i;

$s_k(x_i)$ – is the set of signal lines on the path from a primary input x_{PI} to the signal x_i.

The calculation of the pairs $M_k(x_i)$ starts from the primary inputs of the netlist N. The delay value $d_k(x_i)$ for an output signal x_i of a gate g_j is calculated as $d_k(x_i) =\{\max d_k(x_{in,l})\}+t(g_j)$, where $x_{in,l}$ is the l-th input signal of the gate g_j and $t(g_j)$ is the nominal delay of the gate g_j. In order to avoid an explosion of the number of paths to be analyzed, we introduce a threshold L for the number of longest paths traced up to the current signal line x_i under analysis for continuing the calculations of the pairs. As a result we obtain a list of L_i pairs $M_k(x_i)$ for each primary output signal $x_i \in OUT$, where $L_i \le L$.

Fig 9 shows the values of $M_k(x_i)$ calculated for each signal line x_i in the netlist N.

3) As the next task, all the obtained pairs $\{M_k(x_i) \mid x_i \in OUT\}$ for which $d_k(x_i) \ge B/D$, where B is the time budget for the path to complete and D is the maximum expected delay degradation ratio, will be added to the set of NBTI-critical paths C.

4) Finally, all the paths $s_k(x_i)$ in $C=\{M_k(x_i)=(d_k(x_i), s_k(x_i) = \{x_{PI}, ..., , x_i, ..., x_{PO}\} \mid k=0,...,K\}$ will be analyzed for both 0→1 and 1→0 transitions at their primary inputs in order to calculate their delays after NBTI-degradation. We will calculate the NBTI-degraded delay for the paths $s_k(x_i)$ by summing up the delays of gates along these paths obtaining the NBTI-degraded path delay $d'(s_k(x_i))$ for the given transition. Since all the gate stages g_j along the paths invert the values at their input, for 0→1 (1→0) transition at the primary input of the path $s_k(x_i)$, in the case of even order (odd order) gates on path $s_k(x_i)$ their *nominal delays* $t(g_j)$ are summed, while *NBTI-degraded delays* $\tau(g_j)$ are summed for the odd order (even order) gates, respectively. This is due to the fact that the NBTI-induced delay $\Delta t(g_j)$ manifests itself only under the stressed condition, i.e. when the output of g_j is switching from 1 to 0. The NBTI-degraded gate delay $\tau(g_j)$ is calculated as $t(g_j)\cdot(100\% + \Delta t(g_j))$, where $\Delta t(g_j)$ is provided as percentage of delay degradation (see Section 3). Degraded delays for the different gate inputs are calculated separately.

As a result, we obtain NBTI-degraded delay values $d'(s_k(x_i))$ for both input transitions for all the NBTI-critical paths in the set C and we identify the overall delay of the longest NBTI-degraded path. The latter will be applied as the fitness value to the evolutionary optimization presented in Section 5.

Table 1 shows an example of calculating the values of NBTI-degraded delays $\tau(g_j)$ for each input of gates g_j. In addition, it shows the resulting values for the NBTI-degraded path delays $d'_{0\to1}$ and $d'_{0\to1}$ for the identified NBTI-critical paths.

The four above-described tasks are summarized in Algorithm 1 presented in the following:

Algorithm 1. Identification of NBTI-critical logic paths

Calculate $P_z(x_i)$ for signal lines x_i by logic simulation of the expected workload
For all lines $x_i \in$ IN **Do** Assign $d_k(x_i)$: =0; **End for**
For all lines $x_j \in$ FN $\cup$ OUT **Do**
proceed along the numeration of the lines calculating the following:
$d_k(x_i)$:={max $d_k(x_{in,l})$}+$t(g_j)$, where $x_{in,l}$ is the l-th input signal of the gate g_j;
Save highest $d_k(x_i)$ values only up to L for further calculation;
End for
C:={$M_k(x_i) \mid x_i \in$OUT} for which $d_k(x_i) \geq$ B/D;
For all pairs $M_k(x_i)$ =($d_k(x_i)$, $s_k(x_i)$)$\in$ C **Do**
For 0→1 and 1→0 transitions at the primary input of $s_k(x_i)$ **Do**
$d'(s_k(x_i))$:=$\sum t'(g_j)$, where g_j are the gates forming the path $s_k(x_i)$ and
$t'(g_j)= \tau(g_j)$ when 0→1 at the input of g_j, otherwise $t'(g_j)= t(g_j)$;
End for
End for
Return {max $d'(s_k(x_i))$}

4.2. NBTI-critical path identification example

Consider the example circuit shown in Fig. 9 consisting of 7 gates. The nominal delays for each gate $t(g_j)$ are marked next to them. As the starting point, we set a limit L for the number of paths to be considered at each signal line x_i. In other words, only up to L paths with the most significant delay $d(x_i)$ will be propagated to the next gate from x_i. In our simple example, L is set to 4. An additional parameter is the time budget B. Let the time budget for the circuit in Fig. 9 be 45 time units. Assuming that a maximum NBTI degradation for a path is estimated to be e.g. 20%, we can calculate a threshold for a path to be considered NBTI-critical. According to the values in this example, any path longer or equal to B/D=45/1.2=37.5 time units will be NBTI-critical.

The first task is to characterize a restricted number of L paths represented by pairs $M_k(x_i)$ for each primary output signal line $x_i \in OUT$ of the circuit. As we can see from Fig. 9, four paths (represented by pairs $M_1(x_{10})$, $M_2(x_{10})$, $M_1(x_{11})$ and $M_2(x_{11})$) exceed the threshold value of 37.5 time units and are therefore included to the set of K=4 NBTI-critical paths (highlighted by bold lines in the figure).

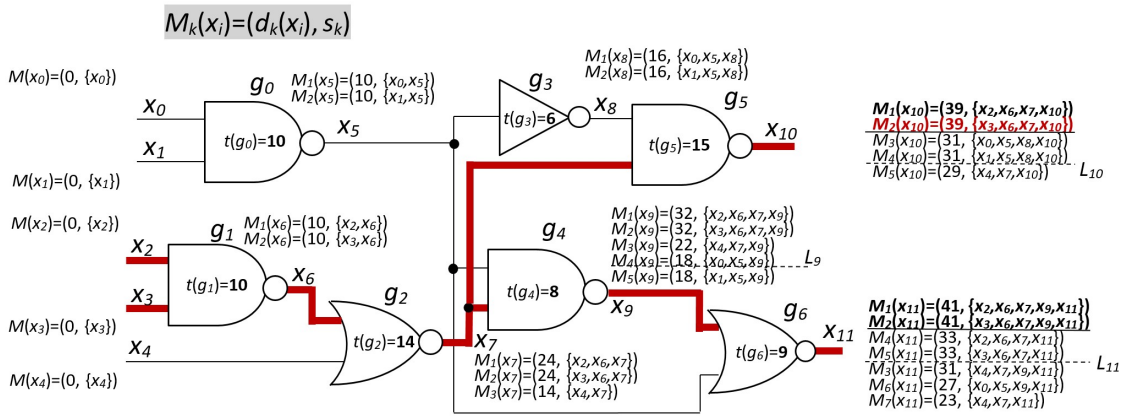


Fig. 9. An example of NBTI-critical paths identification.

Table 1. An example identification of NBTI-critical paths and of the longest NBTI degraded path.

	Delays (in time units)						
Gates	g_0	g_1	g_2	g_3	g_4	g_5	g_6
$t(g_j)$	10	10	14	6	8	15	9
$d(x_i)$	10	10	24	16	32	39	41
$\tau(g_j)$ for IN_A	12	11	16	8	10	18	10
$\tau(g_j)$ for IN_B	12	12	17		9	20	11
$d'_{0\to1}(\{x_2,x_6,x_7,x_9,x_{11}\})$		11	25		34		43
$d'_{1\to0}(\{x_{2/3},x_6,x_7,x_9,x_{11}\})$		10	26		34		44
$d'_{0\to1}(\{x_3,x_6,x_7,x_9,x_{11}\})$		12	26		35		44
$d'_{0\to1}(\{x_2,x_6,x_7,x_{10}\})$		11	25			45	
$d'_{1\to0}(\{x_{2/3},x_6,x_7,x_{10}\})$		10	26			41	
$d'_{0\to1}(\{x_3,x_6,x_7,x_{10}\})$		12	26			**46**	

Table 1 presents the identification of NBTI-critical paths and calculation of the longest NBTI degraded path. The third row "$t(g_j)$" provides the nominal delays for each gate g_j. The fourth

row "$d(x_i)$" shows the longest delay from primary inputs to the signal corresponding to the gate output.

In order to calculate the delays of the NBTI-critical paths after the NBTI-degradation we apply $\Delta t(g_j)$ derived for each gate input based on the corresponding P_z values at these inputs obtained by gate-level simulation of the user workload. Rows 5 and6 in Table 1 show the NBTI-degraded delay $\tau(g_j)$ for gates in the stressed state, i.e. when the input is switching from 0 to 1. Degraded delays for both gate inputs IN_A and IN_B are calculated separately (g_3 being an inverter has only one input).

Subsequently, the degraded delay d' of each NBTI-critical path is calculated separately with $\tau(g_j)$ used for 0→1 transitions (stressed condition) and $t(g_j)$ used for 1→0 transitions (relaxed condition) at the respective gate inputs. Two degraded path delay calculations will be performed for each NBTI-critical path, one for primary input transition 0→1 and one for the transition 1→0. Thus, for K=4 NBTI-critical paths $2 \cdot K=8$ calculations are made. The Table lists the NBTI-degraded path delay calculations for the eight paths on six rows, due to the fact that the calculation results for the paths starting with x_2 and x_3 (denoted by $x_{2/3}$ in the Table) are equivalent with 1→0 primary input transition and are consequently combined into single rows. As it can be seen from the Table, the path $M_2(x_{10})=(46, \{x_3,x_6,x_7,x_{10}\})$ for a 0→1 primary input transition is the longest NBTI degraded path with the given user workload.

In order to explain obtaining the delay of the longest NBTI degraded path $M_2(x_{10})=(46, \{x_3,x_6,x_7,x_{10}\})$, consider the last row "$d'_{0\to1}(\{x_3,x_6,x_7,x_{10}\})$" of Table 1. For the delay from input x_3 to the output of gate g_1, we have to select the "$\tau(g_j)$ for IN_B" since 0→1 transition represents the stressed condition for the gate g_1. Thus, the delay at g_1 is 12. For the next gate, gate g_2, we have to apply the nominal delay "$t(g_j)$" because the transition at the gate input is 1→0 and the gate is in a non-stressed state. Thus, the delay at g_2 is 12+14=26. Finally, for g_5 we have to select again the "$\tau(g_j)$ for IN_B" since x_7 has the 0→1 transition, which represents the stressed condition for the gate g_5. Thus, the final NBTI degraded path delay for path $d'_{0\to1}(\{x_3,x_6,x_7,x_{10}\})$ at the output of g_5 is 12+14+20=46.

4.3. Experimental results for accuracy assessment of model-based NBTI-critical paths calculation

NBTI-critical paths identification is demonstrated on a 4-bit ALU 74HC/HCT181 design from Philips [44] with minor modifications, i.e. its XOR gates have been replaced by NAND and Inverter gates. In Fig. 10, the design's combinational logic is outlined by a dashed border. We assume that the interfaces can be connected to flip-flops (FF) and consider logic paths between them. In our experiment, we compare the proposed fast gate-level approach and SPICE simulation results for path delay calculation and assess accuracy using 5 selected NBTI-critical logic paths shown in as colored lines in Fig.10.

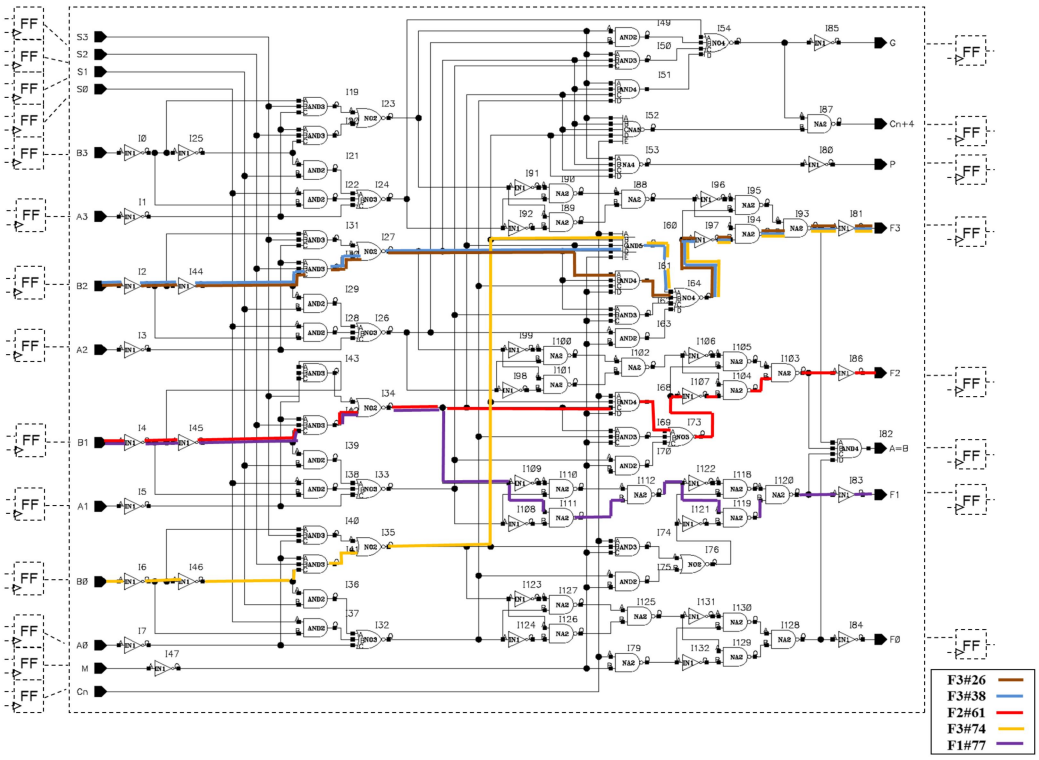


Fig. 10 A set of NBTI-critical paths identified in the 74HC/HCT181 design.

According to the circuit specification [44] all input stimuli combinations are allowed. In our experiment, an exhaustive set of input patterns (16,384 vectors) has been generated. This set has been repetitively applied in order to represent a potential user workload. Firstly, input signal probabilities P_z for all related pMOS transistors are calculated based on logic simulation results. Secondly, these values are used as input in order to compute the ΔV_{THp} values for pMOS transistors using Equation (1). The resulting ΔV_{THp} values serve as input for: a) Equation (2) followed by application of Algorithm 1 and b) path delay simulations in SPICE.

Table 2 demonstrates the path delays induced by NBTI during 10 years at the selected NBTI-critical paths in the 74HC/HCT181 design calculated by the proposed approach and accurate transistor-level simulation in SPICE. Columns 2 and 3 show path delay degradation Δt_{path} in percentage for the rising-edge output transition calculated by the proposed approach and SPICE simulation, respectively. Columns 4 and 5 represent the same information for the falling-edge output transition. The proposed fast gate-level NBTI-induced delay estimation closely correlates with the results obtained in SPICE. The small deviation is mainly caused by the impact of output load capacitance values on gate delay degradation considered in SPICE simulation and not taken into account by the abstraction level of the proposed model. The proposed approach provides a reduction of several orders of magnitude regarding the time required to compute NBTI-induced delays for NBTI-critical paths compared to respective SPICE simulations (0.56 seconds versus 5 minutes). The resulting accuracy for NBTI-induced path delays computation by the proposed approach is acceptable (always lower than 2%) and within the accuracy margins required by the rejuvenation approach described below.

Table 2. NBTI-induced path delays at selected NBTI-critical paths in the 74HC/HCT181 design. A comparison of values calculated by the proposed fast gate-level approach and simulations in SPICE.

Path	Delay for path output transition			
	0→1 (rise-edge delay)		1→0 (fall-edge delay)	
	Proposed	SPICE	Proposed	SPICE
	Δt, %	Δt, %	Δt, %	Δt, %
F3#26	13.76	13.95	11.12	11.71
F3#38	13.16	13.73	12.23	10.92
F2#61	13.06	12.09	12.27	13.91
F3#74	13.26	12.93	11.68	11.76
F1#77	7.78	9.76	17.20	16.38

5. Evolutionary Generation of Rejuvenation Stimuli

While several test generation techniques can be applied to create rejuvenation stimuli for the identified NBTI-critical paths, an evolutionary algorithm is an efficient option due to its inherent properties. Primarily, it is by construction "blind" regarding the structure of the circuit under analysis, and, therefore, it is able to solve dependencies caused by impacts of individual gates and capable to obtain a cost-effective global solution with respect to all critical paths.

5.1. Evolutionary Algorithms

Evolutionary Algorithms (EAs) [31] are algorithms loosely inspired by mechanisms of the biological theory of evolution. When EAs are used to tackle a specific problem, an *individual* is a single candidate solution, and its *fitness* is a measure of its capacity to solve the problem to be tackled; the set of all candidate solutions that exists at a particular time represents the *population*. Evolution proceeds through discrete steps called *generations*. In each of them, the population is first expanded (using a set of *operators* applied to the existing population), and then collapsed, mimicking the processes of breeding and struggling for survival. The population in the first generation may be either completely random or seeded with existing solutions (see Fig. 11).

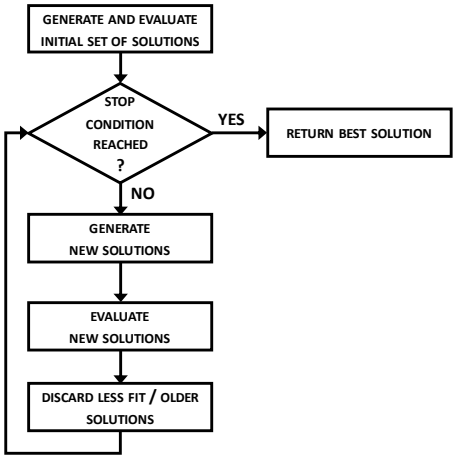


Fig. 11 An evolutionary algorithm.

Over the years, EAs have proven capable of solving difficult problems even within highly complex fitness landscapes, such as open problems related to networking or protocols.

Evolutionary optimizers have been successfully exploited both in stationary and dynamic situations. They were demonstrated to be able to identify either single optima or Pareto sets in multi-objective problems. Since 1990s, the complexity of the electronic circuits dramatically increased, and evolutionary heuristics started to be seen as alternatives to classic approaches in the EDA area. Researchers proposed EA-based methodologies for tackling several well-known NP-hard problems, such as placement, floor-planning, routing [32], and automatic test-pattern generation [34], [35]. EAs have also demonstrated to be efficient for evolving assembly test programs for microprocessors, for validation, post-silicon verification, and test [33].

The task of generating rejuvenation stimuli calls for the use of an EA. The evolutionary approach is intentionally "blind" towards the structure of the circuit. As a result, the EA might be able to sort out dependencies of impacts by individual gates, and obtain a global solution with regard to all critical paths in a cost effective manner. Note, that, generally, the internal information about the circuit is still required by external tools that provide the EA with feedback on the generated solutions' quality (i.e. fitness).

5.2. The Evolutionary Toolkit µGP

The approach presented here makes use of an evolutionary toolkit named µGP. µGP was developed at the Politecnico di Torino in the early 2000s [36], [37], [38], and it is now available under the GNU Public License from Sourceforge.

µGP allows a high degree of customization, but most of its parameters can also be self-adapted, that is, it can autonomously set them to a reasonable value during the optimization process. This self-adaptation mechanism is also used to shift the algorithm's focus between *exploration*, i.e., seeking new solutions changing significantly the current ones, and *exploitation*, i.e., tweaking the current good solutions changing them slightly, increasing both the speed and the quality of result.

µGP implements a large variety of genetic operators that can handle the specific characteristics of the individuals. Moreover, two operators mimic *differential evolution* [38] to more efficiently handle real-valued parameters, while one operator performs a pseudo exhaustive search on a single element of the solution. All operators may be activated with a specific probability, and, further, self-adaptation regulates these probabilities [39].

5.3. Flow for evolutionary generation of rejuvenation stimuli

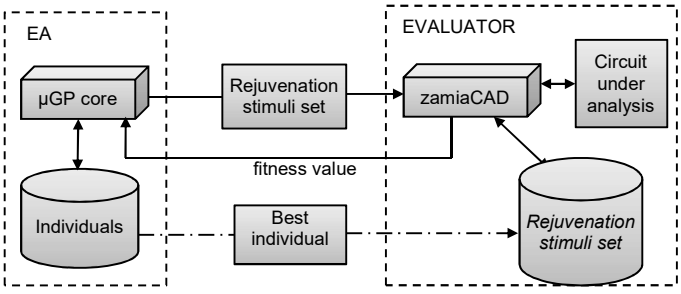


Fig. 12 A general flow of the evolutionary system developed in this work.

The approach for rejuvenation stimuli generation proposed in this paper was implemented on top of the open source scalable hardware design and analysis framework zamiaCAD [40],[41]. The front-end of zamiaCAD includes a parser and an elaboration engine that supports full VHDL-2002 standard specification and a set of VHDL-2008 extensions. On the back-end side, the framework allows design simulation, static analysis and other applications for debug [42]. zamiaCAD has an Eclipse IDE plug-in based graphical user interface for advanced design entry and navigation.

The general flow of the evolutionary system is presented in Fig. 12. It is composed of two main parts: an EA represented by µGP and an *EVALUATOR* represented by the zamiaCAD framework. The evolutionary optimizer devises a set of new stimuli vectors by evolving a population of candidate solutions. The usefulness of each candidate stimuli vector is evaluated with respect to the existing *Rejuvenation Stimuli Set* by the zamiaCAD framework by simulating the NBTI-degraded path delays for the *K* most timing delay critical paths that were identified as described in Section 4. The longest NBTI-degraded path delay value over these *K* paths is reported back to the µGP core in form of fitness values. The best individual from the population, i.e. the one with the highest fitness, is added to the *Rejuvenation Stimuli Set (RSS)*. Then, the process iterates. When the process starts, the RSS is empty. In our experiments we selected *K* longest paths from the circuit obtained by static timing-analysis to be included into the optimization process.

The evolutionary optimizer evolves the population until a *steady state* condition is detected (i.e., non-improvement is recorded for a given number of generations), or a maximum number of steps has been performed. The outer loop (see Fig. 12), on the other hand, is repeated until the RSS reaches a satisfying rejuvenation capability.

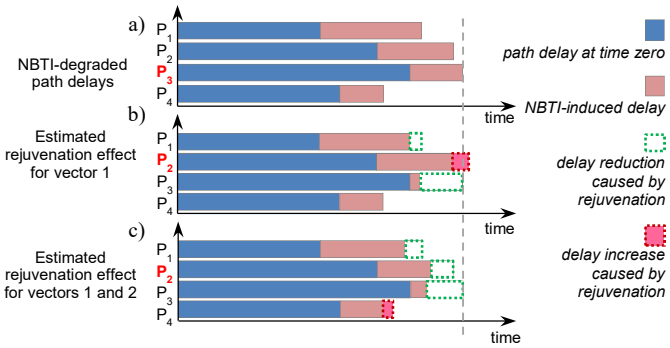


Fig. 13 An example of evolutionary generation of rejuvenation stimuli.

Fig. 13 demonstrates the dependencies being solved during the evolutionary generation of rejuvenation stimuli. Here, P_1 through P_4 represent a set of NBTI-critical paths. In the given example, the final rejuvenation stimuli sequence consists of two vectors. Here, only the combination of two vectors achieves the targeted effect of rejuvenation, i.e. reduction of the longest path delay in the circuit (Fig. 13c). Reduction of one path delay can lead to increase of delay for another one and individual vectors can in fact increase the longest path delay (Fig. 13b). In the performed experiments, the accurate impact of very small execution overheads is calculated via estimation of the NBTI-induced path delay for the accordingly weighted sum of input signal

probabilities P_z. The allowed execution overhead is distributed equally between all vectors in the RSS.

5.4 Experimental results for rejuvenation stimuli generation

The proposed approach is applied to several combinational circuits representing sets of logic paths between flip-flops. The benchmarks include 4-bit and 32-bit implementations of the ALU (Arithmetic Logic Unit) core extracted from a MIPS processor design Plasma [41]. The designs were initially described in VHDL at the RT level and the gate level was synthesized with Synopsys Design Compiler.

First, consider a detailed analysis of the 4-bit ALU design. For the purpose of the rejuvenation experiments, a set of workloads for the circuit were generated.

- *Functional User Workloads* are realistic exploitation scenarios of the ALU circuit, when only one, two or three functions out of the 16 implemented by the ALU logic were used.
  - The workload *1F-OR* repeatedly uses only the function OR with all possible combinations for 4-bit operands. *1F-NOR, 1F-AND* and *1F-ADD* are similar workloads for corresponding single functions.
  - The workloads *2F-ADD_NOR, 2F-ADD_OR* and *2F-OR_AND* activate only the two corresponding functions during their execution.
  - The workloads *3F-OR_ADD_NOR* and *3F-OR_AND_NOR* exercise 3 functions each.
- The *Random* workload is a set of stimuli repeating 150 random vectors.
- The *Artificial* workloads were generated to represent near-maximum and near-minimum aging scenarios (i.e. the P_z values for a large number of gate inputs are either very high or very low).

The 4-bit ALU circuit was first simulated with the given workloads to obtain values for P_z at each node (gate input). Further, NBTI-induced path delays were estimated based on these P_z values (corresponding structural details and initial path delays were calculated by static analysis). Rejuvenation stimuli sequences were individually generated for each workload and their contribution to reduction of path delays was calculated. Table 3 presents an overview of the experimental results.

Table 3. Rejuvenation Stimuli Generation for 4-bit ALU design

Workloads		***Functional User Workloads***									***Random***	***Artificial***	
		1F-OR	*1F-NOR*	*1F-AND*	*1F-ADD*	*2F-ADD_NOR*	*2F-ADD_OR*	*2F-OR_AND*	*3F-OR_ADD_NOR*	*3F-OR_AND_NOR*		*near-max*	*near-min*
Nodes at static NBTI (%) total (along the longest path))		22.5 (14.3)	20.1 (11.5)	23.7 (14.3)	16.0 (14.3)	8.3 (7.7)	12.4 (10.7)	13.6 (10.7)	6.5 (7.14)	5.3 (7.14)	0.0 (0.0)	39.6 (50.0)	16.6 (0.0)
Δt by NBTI (%)		14.91	18.49	28.15	30.05	18.83	28.49	17.87	18.66	14.97	12.37	51.28	8.82
Δt^R after rejuve-nation for the given overhead, (%)	*1.0E-14*	14.91	18.49	28.15	30.05	18.83	28.49	17.87	18.66	14.97	12.37	51.27	8.82
	1.0E-12	14.89	18.37	28.01	29.9	18.71	28.35	17.85	18.61	14.94	12.37	50.94	8.82
	1.0E-10	14.49	17.2	26.71	28.43	17.93	27.01	17.46	18.13	14.71	12.37	47.6	8.82
	1.0E-08	13.79	15.36	24.55	26.02	17.18	24.89	16.75	17.37	14.35	12.37	42.32	8.82
	1.0E-06	13.05	13.23	21.99	23.16	16.25	22.33	15.89	16.45	13.9	12.37	35.89	8.82
	0.001%	12.65	12.63	20.55	21.53	15.72	20.9	15.4	15.93	13.65	12.37	32.64	8.82
	0.01%	12.2	12.15	19.02	19.8	15.17	19.36	14.87	15.37	13.38	12.37	28.35	8.82
	0.1%	11.81	11.73	17.36	17.95	14.58	17.7	14.3	14.77	13.11	12.37	24.19	8.82
	1.0%	11.48	11.37	15.63	16.02	13.93	15.97	13.68	14.12	12.9	12.31	19.8	8.82

	10.0%	11.19	11.06	13.68	13.98	13.02	14.02	13.07	13.21	12.47	11.88	15.52	8.82

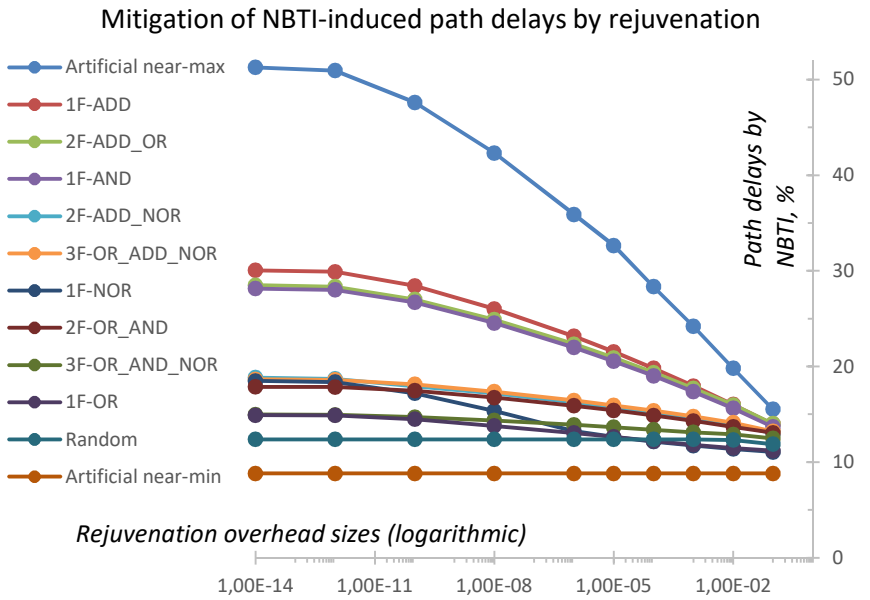


Fig. 14 Smooth decrease of NBTI-induced Δt_{path} after application of rejuvenation stimuli at increasing execution overheads.

In case of different workloads, the NBTI-induced path delay over 10 years was estimated to reach 9% to 51%. The highest path delay increase due to NBTI (the "artificial near-max" workload) is less realistic (just 5 deterministic stimuli vectors are repeatedly applied keeping many gates on the NBTI-critical paths in the static NBTI state). It is provided here mainly in order to present one possible worst-case scenario. However, the increased path delays resulting from the Functional User Workloads can be considered realistic and still provide very high delay increments Δt_{path}, in the range of 15% to 30%. The workload with random stimuli produces a very smooth distribution of P_z probabilities close to 0.5 in the whole design structure and therefore in case of longer random sequences can cause only small and well-distributed NBTI-induced path delays (around 12%, which is also a common estimation found in literature). The last column shows an NBTI-induced path delay of 9% that is possible only in case of deterministic workloads targeted at minimal NBTI.

Two of the most notable parameters characterizing the workloads are depicted in the third row. The first one is the relative number of those nodes (i.e. gate inputs) whose NBTI was observed to be static in relation to the total number of nodes in the circuit. The second is the subset of these nodes, which fall on the longest NBTI-degraded path, i.e. the one whose NBTI-delay is presented in the red row (the value is also given in percentage relative to the total number of nodes along the path and it is put in brackets). It can be observed that the functional workloads, indeed, include a significant number of nodes at static NBTI, many of them are on the longest NBTI-degraded path. Larger and smaller NBTI-induced path delays caused by the Artificial near-maximum and Random workloads, correspondingly, also correlate with these parameters. The Artificial near-minimal workload illustrates that even with the presence of nodes at static NBTI in

the circuit, but being located beyond the NBTI-critical paths, combined with moderate dynamic NBTI elsewhere, may generate an small overall impact on circuit delay degradation.

The last 10 rows in the table demonstrate the efficacy of the generated rejuvenation stimuli mitigating the paths delays execution overheads of 10^{-1} (i.e. 10%) to 10^{-14}. These dependencies of NBTI-induced Δt_{path} reduction on growing overheads are illustrated graphically Fig. 14 for all 12 workloads.

A use case of rejuvenation can be depicted in the following example. Consider a scenario where the circuit's time slack is set to 15%. In this case, during ten years of operation, 7 out of 9 user workloads (columns 4-10 in Table 3) will result in larger delays induced by NBTI and may functionally fail because of desynchronization (the red row). Here, application of the generated rejuvenation stimuli limited, for example, to 0.1% execution overhead (the green row) will mitigate NBTI and reduce the induced delays to fit into the time slack margin for 4 functional user workloads (columns 4, 7, 9, 10). In case of workloads with a large number of gate inputs at close-to-1 signal probability (i.e. column 13), rejuvenation may result in NBTI-induced path delay reduction by factor two. The path delays induced by NBTI caused with *random* and *artificial near-min* workloads cannot be reduced efficiently by application of rejuvenation stimuli.

Table 4 demonstrates similar efficacy of the generated rejuvenation stimuli for a 32-bit implementation of the Plasma ALU core.

Table 4. Rejuvenation stimuli generation for 32-bit Plasma ALU core.

Workloads		***Functional User Workloads***			***Random***	***Artificial***	
		1F-NOR	*1F-OR*	*1F-AND*		*near-max*	*near-min*
Nodes at static NBTI (%) total (along the longest path)		20.7 (26.7)	24.0 (26.8)	20.8 (26.7)	0.0 (0.0)	36.0 (50.0)	13.4 (0.7)
Δt by NBTI (%)		23.88	33.33	24.56	12.35	54.80	9.17
Δt^R after rejuve-nation for the given overhead, (%)	*1.0E-12*	23.78	33.17	24.45	12.35	54.44	9.17
	1.0E-08	20.71	28.73	21.27	12.35	45.34	9.17
	1.0E-04	18.39	25.42	18.86	12.35	38.56	9.17
	0.01%	15.76	21.51	16	12.35	30.53	9.17
	0.1%	14.56	19.35	14.42	12.34	26.09	9.17
	1.0%	13.45	17.05	13.56	12.31	21.8	9.17
	10.0%	12.27	14.54	12.14	12.02	16.58	9.17

The time the proposed approach required to generate the individual rejuvenation stimuli sequences for each user profile was about 20 minutes on a moderate workstation (i.e. 3GHz iCore7 Windows 64 bit, 1 GB of memory used by JVM). This time includes iterative execution of the evolutionary algorithm with the circuit simulation by dedicated workload stimuli and NBTI-critical path identification calls.

6. Discussion

Despite the fact that static NBTI may occur significantly more rarely compared to dynamic NBTI in powered nanoscale logic, it is still very probable in practice. Even a correctly designed circuit with no redundancy may be exploited in various applications that keep parts of the logic unused for a very long time. Consider an ALU circuit or its logic as a part of a complete processor design. Because of system-level architecture properties or due to the end user's habits

and needs [14], the logic of the ALU implementation, e.g., 16 functions can be exploited throughout the use of computations with one or two functions only, thus leaving a part of the ALU logic unused and under constant static NBTI stress for years. The same may happen with particular parts of the control logic that activates some service regimes of a product. A particular concern for vulnerable logic can be long-life reliability-critical applications [15]. In both cases, a user profile can change after long periods of product exploitation (e.g. a car changing the owner or an airplane/satellite changing the flying route) after several years, consequently the logic part degraded over years under static NBTI may be activated and, therefore, potentially cause a functional failure.

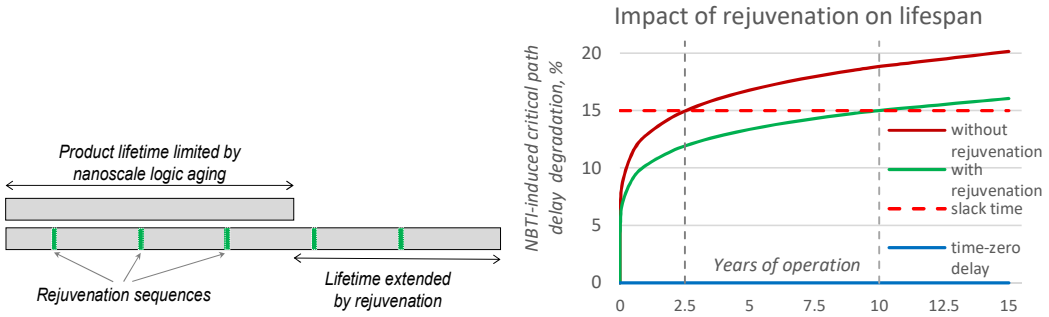


Fig. 15 An illustrative schedule for rejuvenation stimuli execution and the resulting impact

Mechanisms for the application of the rejuvenation stimuli sequences depend on the particular design architecture and if necessary may exploit existing design-for-testability structures and, hence, are out of scope for the proposed approach. Application of the rejuvenation stimuli implies execution overhead and, consequently, the related power budget overhead while utilizing the existing logic. A schedule for rejuvenation stimuli execution may be predetermined in advance or kept dynamic based on the actual workload and idle times of the targeted logic, e.g. a real-time system. For illustration, see Fig. 15. The chart on the right-hand side illustrates a scenario with the slack time set to 15% of the most critical path's delay at time zero (i.e. the fresh circuit). The NBTI-induced delay at the longest path is estimated to reach 15% after 2.5 years and 19% after 10 years of operation for a given workload without rejuvenation. At the same time, the delay is still 15% after 10 years if the rejuvenation procedure is applied. A similar rejuvenation impact is demonstrated in Table 3 (e.g., for the 2F-ADD-NOR workload). Therefore, reduction of the NBTI-induced delay by just several percent can significantly extend reliable lifetime of nanoscale logic.

Different from other known NBTI mitigation approaches, the proposed rejuvenation method does not require redesign of the physical layer before fabrication and can be introduced to a product in the field even after years of operation (e.g. through a firmware update). It can be also an attractive alternative for products, which aim at avoiding frequency or voltage scaling techniques that in return may reduce performance or accelerate transistor's aging.

7. Conclusions

The proposed approach is aimed at extending the reliable lifetime of nanoelectronics. It addresses the time-dependent variation caused by Negative Bias Temperature Instability (NBTI) as one of the main reliability concerns in the nanoscale logic circuits. Different from existing works in the state-of-the-art, the proposed approach:

- is based on accurate and fast hierarchical gate-level identification of NBTI-critical paths and particular gates where rejuvenation has to be applied;
- proposes efficient rejuvenation stimuli generation with evolutionary algorithm;
- does not require redesign and can be applied to the existing circuit, i.e. exploiting if necessary the existing design-for-testability instruments.

The proposed rejuvenation approach may have a very significant impact on the circuit's lifetime extension. The experimental results clearly demonstrate the feasibility and the efficiency of the proposed approach to generate rejuvenation stimuli, as well as efficacy of the rejuvenation stimuli sequences to mitigate NBTI, especially in cases with static NBTI or dynamic NBTI in which a high number of extreme close-to-1 signal probabilities are involved in the pMOS V_{TH} degradation. It was demonstrated that NBTI-induced path delays can be reduced by up to two times with an execution overhead of 0.1% or less.

Acknowledgements The work has been supported in part by EU FP7 STREP project BASTION and H2020 RIA IMMORTAL, by CNPq (Science and Technology Foundation, Brazil) under contract n. 303701/2011-0 (PQ) and FAPERGS/CAPES under contract n. 014/2012, by European Union through the European Structural and Regional Development Fund, and by Estonian SF grant 9429.

We would like to acknowledge Dr. Christoph Werner, from TU Munich, Germany for valuable comments regarding the proposed approach.

REFERENCES

[1] S. Hamdioui, D. Gizopoulos, G. Guido, M. Nicolaidis, A. Grasset, P. Bonnot, "Reliability Challenges of Real-Time Systems in Forthcoming Technology Nodes", Proc. ACM/IEEE Conference on Design, Automation and Test in Europe, pp. 129–134. Mar. 2013

[2] S. Mahapatra, D. Saha, D. Varghese, and P. B. Kumar, "On the generation and recovery of interface traps in MOSFETs subjected to NBTI, FN, and HCI stress", IEEE Trans. Electron. Dev. 53(7): 1583-1592, 2006

[3] S. Kumar, S. Kim, S. Sapatnekar, "Adaptive techniques for overcoming performance degradation due to aging in digital circuits", Proc.Asia and South Pacific Design Automation Conference, pp. 284–289, 2009

[4] M. A. Alam and S. Mahapatra, "A comprehensive model of PMOS NBTI degradation," Microelectronics Reliability, 45(1): 71–81, Jan. 2005

[5] M. A. Alam, "Reliability- and process-variation aware design of integrated circuits", Microelectron. Reliabil. 48(8): 1114–1122, 2005

[6] T. Grasser and B. Kaczer, "Negative bias temperature instability: Recoverable versus permanent degradation," Proc. 37th European Solid State Device Research Conference, Munich, 2007, pp. 127–130.

[7] Yu Cao; Velamala, J.; Sutaria, K.; Chen, M.S.-W.; Ahlbin, J.; Sanchez Esqueda, I.; Bajura, M.; Fritze, M., "Cross-Layer Modeling and Simulation of Circuit Reliability," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 33(1): 8–23, Jan. 2014

[8] G. I. Wirth, R. da Silva and B. Kaczer "Statistical model for MOSFET bias temperature instability component due to charge trapping", IEEE Trans. Electron. Devices 58: 2743–275, 2011

[9] S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, and S. Vrudhula, "Predictive modeling of the NBTI effect for reliable design," Proc. IEEE Custom Integr. Circuits Conf., pp. 189–192, Sep. 2006

[10] W. Wang, V. Reddy, A. Krishnan, R. Vattikonda, S. Krishnan, Y. Cao, et, "Compact Modeling and Simulation of Circuit Reliability for 65nm CMOS Technology" IEEE Trans. on Device and Materials Reliability 7(4): 509-517, Dec. 2007

[11] Ing-Chao Lin, Chin-Hong Lin, Kuan-Hui Li, "Leakage and Aging Optimization Using Transmission Gate-Based Technique", IEEE Trans. on Computer-Aided Design of Integratedd Circuits and Systems 32(1): 87–99, Jan. 2013

[12] C. Ferri, D. Papagiannopoulou, R. Iris Bahar, A. Calimera, "NBTI-Aware Data Allocation Strategies for Scratchpad Memory Based Embedded Systems", Proc. IEEE 12th Latin American Test Workshop, pp. 1–6, Mar. 27-30, 2011.

[13] A. Ceratti, T. Copetti, L. Bolzani, F. Vargas, "Investigating the use of an on-chip sensor to monitor NBTI effect in SRAM," Proc. IEEE 13th Latin American Test Workshop, pp.1-6, Apr. 10-13, 2012.

[14] Q. Li, Q. Han and L. Sun, "Context-Aware Handoff on Smartphones," Proc. IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems, pp. 470-478, Oct. 2013

[15] A.T Tai, L. Alkalai, S.N. Chau, "On-board preventive maintenance for long-life deep-space missions: a model-based analysis," Proc. Computer Performance and Dependability Symposium, pp.196-205, Sep 7-9, 1998

[16] H. Kukner, S. Khan, P. Weckx, P. Raghavan, S. Hamdioui, B. Kaczer, F. Catthoor, L. Van der Perre, R. Lauwereins, and G. Groeseneken, "Comparison of reaction-diffusion and atomistic trap-based BTI models for logic gates," IEEE Transactions on Device and Materials Reliability 14(1): 182-193, 2014

[17] W. Wang, S. Yang, S. Bhardwaj, S. Vrudhula, F. Liu, Y. Cao, "The Impact of NBTI Effect on Combinational Circuit: Modeling, Simulation, and Analysis", IEEE Trans. On VLSI 18(2): 173-183, 2010

[18] F. Ahmed, L. Milor, "Reliable Cache Design with On-Chip Monitoring of NBTI Degradation in SRAM Cells using BIST", Proc. 28th IEEE VLSI Test Symposium, pp. 63-68. 2010

[19] S. Khan, S. Hamdioui, "Modeling and Mitigating NBTI in Nanoscale Circuits", Proc. 17th International On-Line Testing Symposium, pp. 1-6, 2011

[20] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "NBTI-aware synthesis of digital circuits," Proc. Design Automation Conference, pp. 370–375, 2007

[21] A. Tiwari and J. Torrellas, "Facelift: Hiding and slowing down aging in multicores", Proc. International Symposium on Microarchitecture, pp. 129–140, 2008

[22] F. Firouzi, S. Kiamehr, and M.B. Tahoori, "A linear programming approach for minimum NBTI vector selection", Proc. Great Lakes Symposium on VLSI, pp. 253–258, 2011

[23] Y. Wang, X. Chen, W. Wang, V. Balakrishnan, Y. Cao, Y. Xie, and H. Yang, "On the efficacy of input Vector Control to mitigate NBTI effects and leakage power", Proc. Quality Electronic Design Int'l Symp., pp. 19–26, 2009

[24] J. Abella, X. Vera, et al. "Penelope: The NBTI-aware processor", Proc. 40th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 85–96, 2007

[25] L. Li, Y. Zhang, J. Yang, and J. Zhao, "Proactive nbti mitigation for busy functional units in out-of-order microprocessors", Proc. ACM/IEEE Conference on Design, Automation and Test in Europe, pp. 411–416, 2010

[26] X. Fu, T. Li, and J. Fortes. "NBTI tolerant microarchitecture design in the presence of process variation", Proc. Int. Symposium on Microarchitecture, pp. 399–410, 2008.

[27] F. Firouzi, S. Kiamehr, and M.B. Tahoori, "NBTI mitigation by optimized NOP assignment and insertion". Proc. ACM/IEEE Conference on Design, Automation and Test in Europe, pp. 218–223, 2012

[28] R. Ubar, F. Vargas, M. Jenihhin, J. Raik, S. Kostin, L. Bolzani Poehls, "Identifying NBTI-Critical Paths in Nanoscale Logic", Proc. Euromicro Conference on Digital System Design, pp. 136 – 141, Sep. 2013

[29] S. Kostin, J. Raik, R. Ubar, M. Jenihhin, F. Vargas, L. M. Bolzani Poehls, T. Copetti, "Hierarchical identification of NBTI-critical gates in nanoscale logic", Proc. IEEE 15th Latin American Test Workshop, pp.1-6, 2014

[30] W. Zhao and Yu. Cao, "Predictive Technology Model for Nano-CMOS Design Exploration", Journal on Emerging Technologies in Computing Systems 3(1), Article 1, April 2007, (http://ptm.asu.edu/modelcard/2006/65nm_bulk.pm)

[31] A. E. Eiben, and J. Smith, "Introduction to Evolutionary Computing", Springer, 2015

[32] R. Drechsler, "Evolutionary Algorithms for VLSI CAD", Springer, 1998

[33] G. Squillero, "Artificial evolution in computer aided design: from the optimization of parameters to the creation of assembly programs", Computing, 93(2): 102-120, 2011

[34] F. Corno; M. Sonza Reorda, G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results", IEEE Design & Test of Computers, 17(3): 44-53, Jul/Sep 2000

[35] F. Corno; E. Sanchez, M. Sonza Reorda, G. Squillero, "Automatic test generation for verifying microprocessors", IEEE Potentials, 24(1): 34-37, Feb/Mar 2005

[36] G. Squillero, "MicroGP - An Evolutionary Assembly Program Generator", Genetic Programming and Evolvable Machines, 6(3): 247-263, Sep 2005

[37] E. Sanchez, M. Schillaci, G. Squillero, "Evolutionary Optimization: the µGP toolkit", Springer, 2011

[38] Rainer Storn and Kenneth Price, "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces", J. of Global Optimization 11(4): 341-359, 1997

[39] Belluz, Jany, Marco Gaudesi, Giovanni Squillero, and Alberto Tonda. "Operator Selection using Improved Dynamic Multi-Armed Bandit", Proc. ACM Genetic and Evolutionary Computation Conference, pp. 1311-1317, 2015

[40] zamiaCAD framework web page, [http://zamiaCAD.sf.net] (accessed 2015-09-01)

[41] A. Tšepurov, G. Bartsch, R. Dorsch, M. Jenihhin, J. Raik, V. Tihhomirov, "A Scalable Model Based RTL Framework zamiaCAD for Static Analysis", Proc. IFIP/IEEE International Conference on Very Large Scale Integration, pp. 171-176, 2012

[42] M. Jenihhin, A. Tsepurov, V. Tihhomirov, J. Raik, H. Hantson, R. Ubar, G. Bartsch, J.M. Escobar, H.-D. Wuttke, "Automated Design Error Localization in RTL Designs", IEEE Design & Test, 31(1): 83-92, Feb. 2014

[43] Open Cores Plasma CPU project, [http://opencores.org/project,plasma] (accessed 2015-09-01)

[44] Data sheet “74HC/HCT181 4-bit arithmetic logic unit”, Philips, 1998