ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

# Adaptation of the Berkeley Short-channel IGFET Model Code Base for Atomistic BTI/RTN Simulation

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δημήτριος Α. Σταμούλης

**Επιβλέπων :** Δημήτριος Σούντρης
Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2013

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

# Adaptation of the Berkeley Short-channel IGFET Model Code Base for Atomistic BTI/RTN Simulation

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δημήτριος Α. Σταμούλης

**Επιβλέπων :** Δημήτριος Σούντρης
Επίκουρος Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή τη 13η Ιουνίου 2013.

| | | |
|---|---|---|
| ..... | ..... | ..... |
| Κιαμάλ Πεκμεστζή | Δημήτριος Σούντρης | Ιωάννης Ξανθάκης |
| Καθηγητής Ε.Μ.Π. | Επίκουρος Καθηγητής Ε.Μ.Π. | Καθηγητής Ε.Μ.Π. |

Αθήνα, Ιούνιος 2013

.....

Δημήτριος Α. Σταμούλης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

*Στη μνήμη της Γρηγορίας Αλιφέρη*

# Table of Contents

# Abstract

The aggresive downscaling of CMOS technology intensifies both time-zero and time-dependent device variability. During the last years, extensive research on degradation effects has demonstrated their importance for the reliability of downscaled devices. Two mechanisms with an increasingly adverse impact on threshold voltage $V_{th}$ variations are Bias Temperature Instability (BTI) and Random Telegraph Noise (RTN). Accurate and user-friendly simulation frameworks are crucial for the development of appropriate mitigation techniques.

During the last years, an atomistic model that incorporates the transient nature of BTI/RTN mechanisms has been presented. However, this approach is only implemented on top of commercial tools that feature reduced flexibility and increased CPU times. The current thesis proposes the adaptation of the BSIM source code, in order to include the atomistic BTI/RTN model. Using an open source SPICE distribution as a concept vehicle, we developed a SPICE simulator that seamlessly integrates BTI and RTN in transient simulations. The resulting implementation is far more elegant than previous attempts, both from the point of the developer (a single code base) and that of the end-user (no wrapper scripts; only four parameters added to the modelcard). The correctness of the implementation presented herein will be verified against already published results.

**Keywords:** Berkeley Short-channel IGFET Model (BSIM), Bias Temperature Instability (BTI), Gate Stack Defects, Random Telegraph Noise (RTN), Reliability, Simulation Program with Integrated Circuit Emphasis (SPICE), Transient Circuit Simulations

# Acknowledgments

First and foremost, I would like to thank my thesis supervisor Prof. Dimitrios Soudris. I am thankful for his valuable guidance that he gave me in working upon my thesis. His advices were really helpful during my search for graduate studies. I am grateful for his support and trust expressed through his recommendation letters. Being part of his research team was my best academic experience at NTUA.

During my thesis research, I was fortunate to cooperate and interact with Dimitrios Rodopoulos. I would like to sincerely thank him for his mentorship and support. His insightful feedback during our meetings helped me tremendously to improve this current work. His thoughtful and rigorous remarks had always motivated me to push myself to work harder. Moreover, his mentality and his organizational skills will be always useful and inspiring experiences for my future.

I would like to acknowledge Prof. Kiamal Pekmestzi and Prof. John P Xanthakis for serving on my thesis committee. I would like to thank Prof. Nectarios Koziris and Dr. Konstantinos Nikas for their collaboration during the MILE project. From Microlab, NTUA I would also like to thank Harry Sidiropoulos and Dr. Kostas Siozios for generously sharing their experience with FPGAs and Embedded Systems.

I give my special thanks to Katerina Mousteraki for her enjoyable support and cheering me up whenever I needed it. I also thank all my friends whom I met during my time at NTUA. Yannis Chatzimichos, Evangelos Nikoloudakis, George Pavlakos, Stratis Skoulakis, Costas Tokas and Nikolaos Tsiamitros have been great friends and colleagues.

I find myself fortunate to have the guidance of Anastasios and Yannis Stamoulis. I would like to thank them for ours discussions covering a wide range of subjects and for sharing their experiences with me. Their advices taught me to be an honest, critical and respectful man.

Finally, my very special gratitude goes to my father Anastasios Stamoulis, my mother Anastasia Aleiferi and my brother Thomas Stamoulis for their unconditional love, support, and wisdom. I am really blessed to have parents who dedicate their lives for the happiness of their children.

# List of Figures

# Chapter 1
# Introduction

As the device dimensions decrease, adverse degradation effects emerge for many device parameters. Quantities that were considered constant in devices of older technologies, now display flunctuations. Accurately simulating these fluctuations, is of vital importance for researchers and designers, as it will trigger all the appropriate mitigation techniques that will guarantee the reliability of modern electronics. A device parameter that encounters important variability threats is the threshold voltage $V_{th}$ of a MOSFET transistor.

Two mechanisms responsible for $V_{th}$ variations are Bias Temperature Instability (BTI) and Random Telegraph Noise (RTN). Several models can be found in related literature. A novel approach is the atomistic BTI model. This model incorporates the stohastic properties of individual gate oxide defects and realistically captures their transient behaviour. Thus, transient simulation approaches has been enabled. In the context of the current thesis, a SPICE-based BTI/RTN simulation framework will be created. The implementation flow, as well as the simulation results of the inspected examples, will be presented within the following Chapters.

The next Chapter depicts the state of the art on the simulation of Bias Temperature Instability (BTI). Different BTI models will be presented. Moreover, the landscape of BTI modeling will be categorized into several levels of classification. Various simulation approaches will be then reviewed and properly classified. Thereby, any insufficiencies observed in the state of the art will be revealed.

In Chapter 3, the simulation tool will be described. The selected open source SPICE distribution and its MOSFET transistor models will be extensively explored. More specifically, all the $V_{th}$ evaluation steps during setup and at run time will be pointed out. That way, it will be demonstrated where all the BTI/RTN-related commands and data structures should

be added. In addition, a brief user manual of the implemented simulation framework along with a descriptive usage example will be provided.

Next, in Chapter 4 the correctness of the BTI simulation tool will be verified against already published results. For that purpose, phenomenological and detailed validation will be demonstrated by inspecting representative examples. The results of the selected test cases will be properly presented and will be compared to results of related state-of-the-art simulation frameworks. Moreover, the verification of BTI/RTN impact will be investigated by simulating an array multiplier with and without the BTI/RTN effect.

Finally, the last Chapter lists a series of conclusions from the work presented herein. The potential for future work will be also pointed out.

# Chapter 2

# Research Landscape

## 2.1 Introduction

The current Chapter illustrates the state of the art on the simulation of Bias Temperature Instability (BTI) and Random Telegraph Noise (RTN). During the last years, research on the BTI/RTN physical principles has demonstrated its importance for the reliability of aggressively downscaled devices. A model that realistically captures the impact of BTI/RTN is required to achieve reliable designs. Designers are asked to meet certain lifetime reliability specifications. Accurate simulation frameworks are necessary for the development of proper mitigation techniques. Consequently, a particular BTI/RTN model would not be sufficient on its own. It should be followed by an accurate and user-friendly simulation tool.

Several BTI/RTN models have been presented, and therefore various simulation approaches exist and should be reviewed. In the following Section, we will employ a systematic classification to produce the state-of-the-art landscape. The resulted framework will be used as a guideline in order to map prior art related to BTI/RTN simulations. Having addressed the above subjects, we will provide a summary of what is missing from the state of the art. The intention of this thesis is to provide meaningful and novel contribution, in order to complete the insufficiencies observed in the state of the art.

## 2.2 Landscape of BTI modeling

In this Section we will present the landscape of BTI modeling. The absence of a single, unified BTI model yields a heterogeneous landscape to be explored. Consequently, we have

to wonder through different BTI simulation approaches. We will classify all these state-of-the-art approaches into several levels of categorization. In this way, our state-of-the-art categorization will cover all relevant aspects, without suppressing hybrid works.

To identify the initial step of our methodology, it is important to reflect on the degradation of a device throughout its lifetime. We notice that devices of older technologies exhibit uniform degradation [1]. Therefore, older simulation approaches could assume a similar behavior of all devices under the same operating and stress conditions. However, the downscaling trend in device dimensions resulted in extensive variability during the device lifetime. A single device is not representative of the way all devices of the same technology behave. As a consequence, a fundamental criterion for the first level of our classification is the device individuality of the implemented BTI model.

If we employ the first splitting criterion during state-of-the-art readings, we come across a notable observation : The majority of the reviewed works that do not take device individuality into consideration are based on reaction and diffusion mechanisms. More specifically, they are founded upon the Reaction-Diffusion (RD) model [12]. Since the RD model is one of the predominant approaches towards BTI, a brief presentation is required here.

The Reaction-Diffusion model is mainly used for the explanation of NBTI. Under the negative bias condition (i.e. $V_{GS} = -V_{DD}$), Si-H bonds gradually break. Interface traps are generated as hydrogen diffuses toward the gate. Carriers are trapped in the broken bonds and the gate drive current is reduced. Thus, the NBTI phenomenon causes a gradual increase in the $V_{th}$ value. The NBTI relaxation phase is accordingly attributed to diffusion phenomena. When no stress condition is present, hydrogen is expected to diffuse back and anneal some of the broken bonds.

According to the RD model, the change in the threshold voltage due to the increase in interface charge can be given by [13]:

$$\Delta V_{th}(E_{ox}, t) = (1 + m)\frac{qN_{IT}(E_{ox}, t)}{C_{ox}}$$

where $m$ is a constant representing the equivalent $V_{th}$ shift due to mobility degradation for

a given technology. The number of interface traps $N_{IT}$ can be described by several different equations, depending on the form of the detached hydrogen (i.e atoms, molecules, ions). For the dominant case of hydrogen atoms, $N_{IT}$ is given by [14]:

$$N_{IT} = \sqrt{\frac{k_F N_0}{2k_R}}(D_H t)^{1/4}$$

If we combine the two equations above, we can easily have the 1/4 exponent dependence in threshold voltage shifts throughout the device lifetime. But such an expected threshold voltage degradation can mainly be verified in devices of older technologies, which exhibit uniform reliability behaviour (Figure 2.1a). One the contrary, RD based approaches are deemed unrealistic for modern downscaled devices (Figure 2.1b). From the latter observation it becomes apparent that the splitting criterion regarding device individuality is quite relevant for our classification of the state-of-the-art.

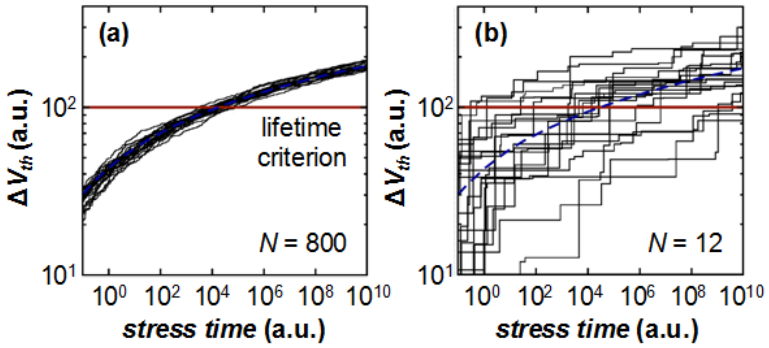


Fig. 2.1: Expected threshold voltage degradation in devices of (a) older and (b) modern technologies [1]

A novel BTI model that captures the time-dependent variability not only at a circuit level, but more extensively at the device level, is the atomistic BTI model presented in [1]. This approach is based on the stohastic properties of individual gate oxide defects. Each particular FET device has a number of defects. Each one of these traps, that is characterized by a set of time constants, can be either occupied or not. By monitoring these occupancy transitions, the impact on FET operation under actual circuit workloads can be properly modeled. Thus, this approach incorporates the time-dependent variability of degradation mechanisms such as BTI and RTN. Note that our implementation is founded upon the

atomistic BTI model. More details and related works will be reviewed within the following Sections.

Having covered possible instantiations of BTI models in terms of device individuality, we assess prior work in terms of the actual reliability analysis tool. During prior work readings, we come across different simulation approaches. In some cases, the expected threshold voltage degradation is precomputed and added as a known $V_{th}$ shift before the actual circuit simulation [13]. This means that any consequent simulation approach will be unable to follow the degradation mechanisms in a transient way. Thus, another interesting classification criterion is whether the BTI analysis that is performed is transient or not.

The outcome of our classification is demonstrated in Figure 2.2. The subdomain that is the residence of the current thesis is highlighted in red.

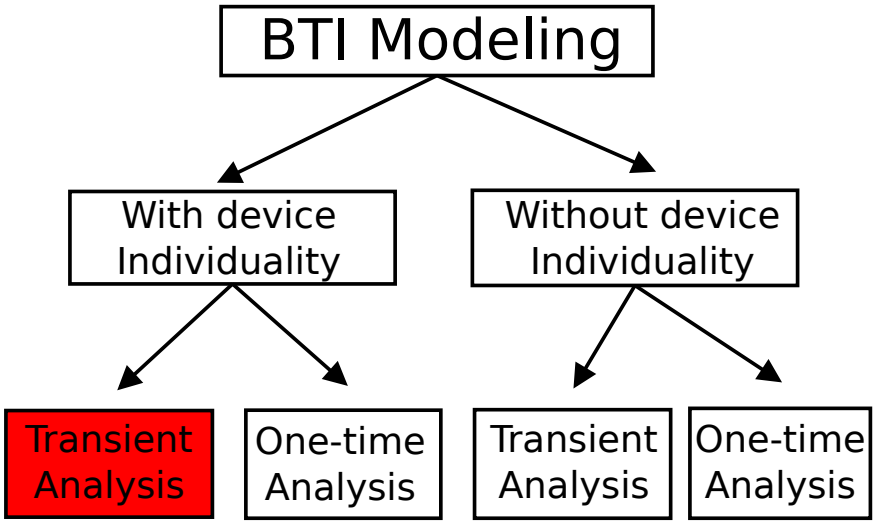


Fig. 2.2: A systematic classification describing the landscape of BTI modeling

## 2.3 Mapping State-of-the-Art on the Landscape

In the current Section, we will be mapping specific papers/works from the literature on the classification domains that have been presented in the previous Section. Therefore all the reviewed approaches will be accordingly classified into the four categories appearing as the leaves of the Figure 2.2.

### 2.3.1 One-time Analysis without Device Individuality

A representative work that is based on one-time BTI simulations is [13]. The utilized circuit simulation software is HSPICE [15]. Several combinatorial circuits were tested and the impact of NBTI on the performance degradation was evaluated. An important simplification is that the threshold voltage degradation ($\Delta V_{th}$ ) due to NBTI was precomputed before the SPICE simulation and no time-dependent view of BTI is provided. The $V_{th}$ variation was represented as an additional voltage source to the gate of each PMOS transistor. Reaction-Diffusion (RD) model is used for these $\Delta V_{th}$ values to be computed. For a given signal switching probability, the ON-time ($V_{GS} = -V_{DD}$) is computed. Using this total stress time the $\Delta V_{th}$ of each PMOS transistor is calculated.

By studying this work [13], we can make one major observation: Only the total stress time is taken into consideration. Hence, the computed threshold voltage degradation does not properly incorporate the effect of recovery, even though it is proved to have a significant effect on the overall NBTI impact. Such a result will be valuable if used and interpreted only as the worst case of the NBTI effects. Otherwise, if used as a design criterion, it can lead to overly pessimistic NBTI estimation and over constrained circuits [3].

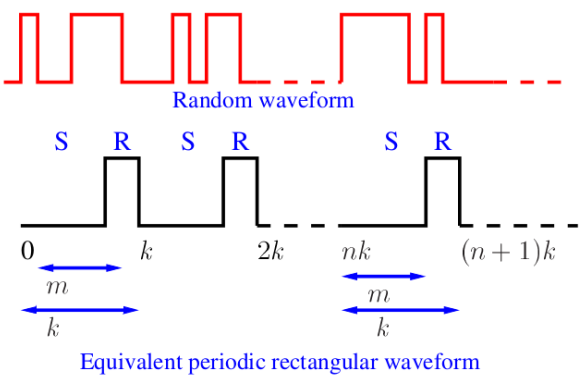


Fig. 2.3: SPAF method, as presented in [2]

A work that targets to properly address the aforementioned flaw is [2]. This work aims to extend the classical RD model to an analytical model that captures the physical effects of multiple stress and relaxation phases. Unlike previous works where the RD models are restricted to a single stress cycle followed by a single relaxation cycle [16] [17], this work provides an analytical solution for the first two cycles, as well as for all the subsequent phases. The Signal Probability and Activity Factor (SPAF) method is presented. The concept of signal propability (SP) is therefore introduced. All different input signals are treated as random waveforms. These waveforms are converted into equivalent

deterministic periodic rectangular signals. More specifically, a random waveform with signal probability equal to SP is converted into a periodic rectangular waveform with the same SP value (Figure 2.3).

The researchers of the above work used their SP method in [3], in order to compute the delay degradation of several ISCAS85 and LGSYNTH benchmarks. The SPAF method is used to convert the probabilistic waveforms of the various nodes into deterministic periodic signals with the same SP values. The threshold voltage degradation of all PMOS transistors is computed and the gate delays are chatacterized as a function of SP. The computed delays are used in the timing library, instead of the nominal values. The NBTI simulation results have been extended to NTBI-aware technology mapping testcases. Both worst-case NBTI and SP-based synthesis have been performed. For a particular benchmark and a selected target delay, the SP-based requires 15% less area compared to the worst-case NBTI, as shown in Figure 2.4. This result corroborates our aforementioned observation that a worst case NBTI criterion can lead to over-constrained circuits.

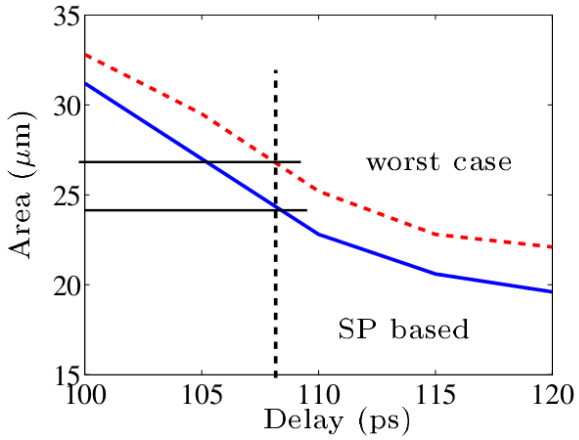


Fig. 2.4: Area-delay curve for the simulated benchmark used in [3]

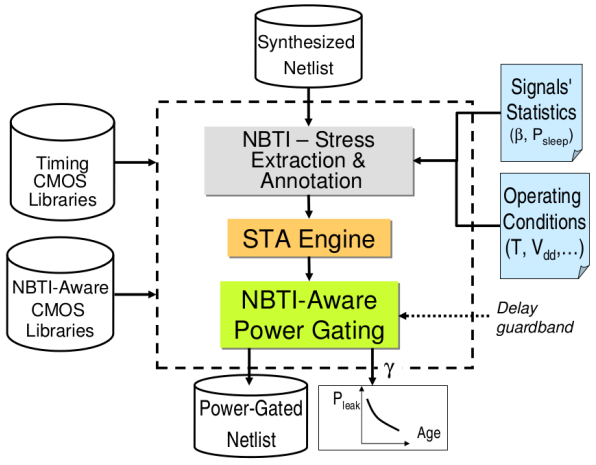


Fig. 2.5: NBTI-aware exploration framework used in [4]

We should point out that the SP method [2] is used as a valid assumption in other simulation approaches. Such an example is a pair of works of A. Calimera et al. [4][5]. Both works examine the usage of power-gating as a technique to reduce the effects of NBTI. On the one hand, the implemented exploration framework in [4] is a SPICE-based flow (Figure 2.5). In the context of our currect classification, we will concentrate on

those blocks that are related to NBTI simulation. More specifically, a pre-stress simulation is performed for the threshold voltage degradation to be computed. During the post-stress simulation, each $\Delta V_{th}$ value is represented as an negative voltage source to the gate of each PMOS transistor.

On the other hand, the usage of power-gating in [5] is focused specifically on SRAM cells. Instead of delay degradation that was inspected before, this work uses the signal-to-noise margin (SNM) as the simulation metric. The total degradation of the SNM is extracted from a SPICE-based framework. Once again the SP method is used for the signal statistics of the stored bits to be converted into equivalent deterministic periodic rectangular signals. Depending on these resulted signals and the operations conditions, the $\Delta V_{th}$ is extracted. This threshold voltage degradation is emulated with additional voltage-controlled voltage source on the gate terminal of each pMOS transistor. SPICE DC simulations are performed to the new NBTI-aware netlist. SNM is numerically extracted by using the graphical method illustrated in Figure 2.6. By comparing the simulation results without and with power-gating, the latter technique seems to reduce the degradation of read/write stability of a SRAM cell.

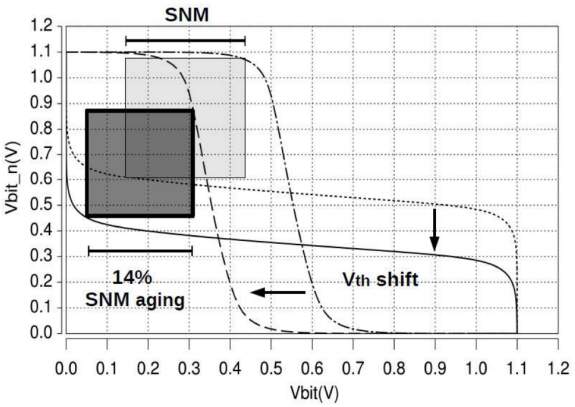

Fig. 2.6: Graphical representation of the SNM as presented in [5]

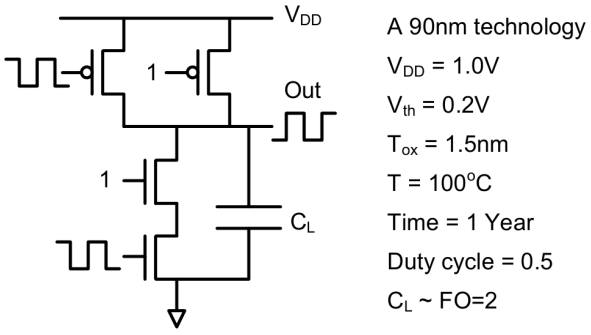

Fig. 2.7: Setup of BTI simulations as presented in [6]

A BTI simulation approach that shares a similar goal with our work is [6]. The intension of the researchers is quite self- explanatory: "Research works on NBTI have been active only within the communities of device and reliability physics [..] The compact model of NBTI bridges the gap between the technological community and CAD tool developers" [6]. The predictive NBTI model, a SPICE compatible model is presented. In

terms of BTI simulation, the selected simulation setup is a 2-input NAND gate, as shown in Figure 2.7. Note that the presented results are extended to design techniques exploration. Different cases of $V_{dd}$, $V_{th}$, duty cycle and gate size tuning have been simulated. Although several claims about the models scalability have been made, no other simulation testcases have been presented throughout this work.

The aforementioned predictive NBTI model is used in [18]. This work exhibits similarities with the work of A. Calimera et al.[5] that is presented in the previous page. More specifically, stability parameters (SNM and WNM) are selected as the evaluation metrics and the same graphical method (Figure 2.6) is therefore used as before. Once again, PMOS transistor models with shifted threshold voltage due to NBTI were added in the HSPICE netlist. However, the basic difference is that the performed simulations are not limited to a specific SRAM cell configuration, but the abstraction level is expanded to whole cache memory units. Different power saving strategies and SRAM cells based cache configurations were simulated and the impact of NBTI is evaluated.

Another pair of noteworthy approaches are the works of Seyab and Said Hamdioui [19][7]. Note that the RD model assumes that NBTI degradation is temperature dependent[20]. However, the researchers make an interesting observation that NTBI temperature dependent sub-processes and the temperature increment in scaling techologies were not fully investigated [7]. Therefore, they use the formulas of each RD model sub-process as presented in [21] (i.e. bond breaking and recovery rates, atomic to molecular H conversion rate and H species diffusion in oxide layer) and they refine them in order to highlight their temperature dependence. As far as the NBTI simulation is concerned, the NBTI impact is emulated with a temperature dependent voltage source on the gate terminal of each pMOS transistor (Figure 2.8). The output value $V_m$ of the module is defined by the 4 temperature dependent formulas and the modified netlist to be simulated in HSPICE.

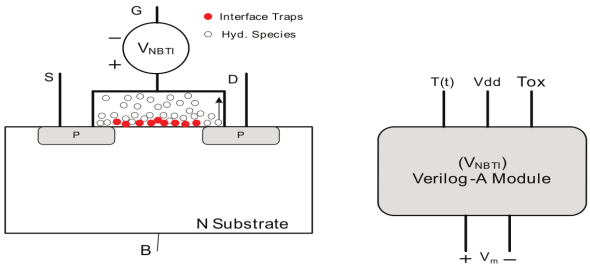


Fig. 2.8: The Verilog-A source representing NBTI degradation under temperature variation [7]

### 2.3.2 Transient Analysis without Device Individuality

Up to this point we have seen that the majority of the approaches on BTI modeling, consist of two basic parts: (i) an evaluation of an analytical transistor degradation model as a pre-simulation step and (ii) a one-time (usually SPICE-based) simulation of the modified, NBTI-aware netlist. Although such a type of processes could be very fast, it has a major drawback that should be pointed out: "The transistor aging calculation is only based on DC stress voltages; time-varying stress voltages are not included" [22]. Simulators and commercial tools such as ELDO and RelXpert, use short transient simulations to solve this problem. Related transient-simulation approaches will be presented within the current Subsection.

A representative example of commercial tool is RelXpert [8]. From the available documentation [8], we quote: "RelXpert will drive the commercial SPICE-like simulator for the lifetime and degradation computations. RelXpert will generate a new netlist allowing an aged circuit simulation". Its simulation flowchart is illustrated in Figure 2.9. More specifically, to simulate the NBTI effects, RelXpert is linked to a SPICE simulator, which reads in the SPICE circuit netlist and generates the voltage and current waveforms of all MOSFETs. The reliability tool then uses these waveforms to evaluate the corresponding age for each device. Based on this computed age effect, the degraded model is provided into the SPICE simulator for second-pass circuit simulation.

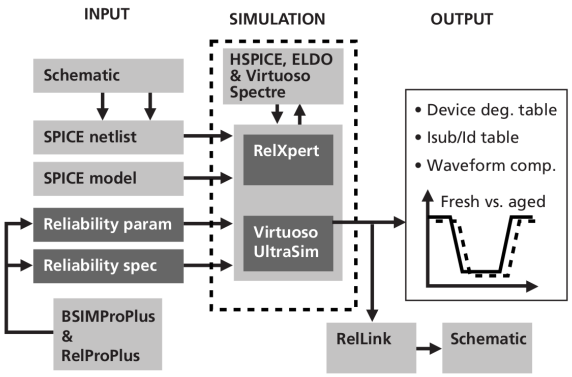


Fig. 2.9: Reliability simulation framework with Virtuoso UltraSim or RelXpert, as presented in [8]

Another commercial option from Cadence® Design Systems, Inc. is Virtuoso® UltraSim [8]. UltraSim is a Fast SPICE isomorphic simulator. It has a similar simulation flowchart with RelXpert (Figure 2.9). Its fundamental difference compared to RelXpert is that UltraSim does not use a simulated SPICE circuit netlist as an input to generate reliability effects. All

the reliability parameters are instead built inside its own simulation framework so it can directly account for NBTI effects.

A noteworthy NBTI simulation approach that aims to analyze both temporal and spatial reliability variations is [22]. In order to reduce transient simulation times without sacrificing simulation accuracy, the framework uses an automatic step-size algorithm. The reliability behaviour of the simulated circuit is evaluated through a vector of performance parameters $P^i$, where exponent $i$ refers to each $t_i$ time step. The total degradation due to NBTI is expressed as the integral equation of $A_{NBTI}$, which is a function of design, environmental and process-related parameters. Moreover, these technology parameters may not be properly defined by their nominal values, as they exhibit a process-dependent variability. A method is therefore presented for this variability to be mapped on the aforementioned $P^i$ vector.

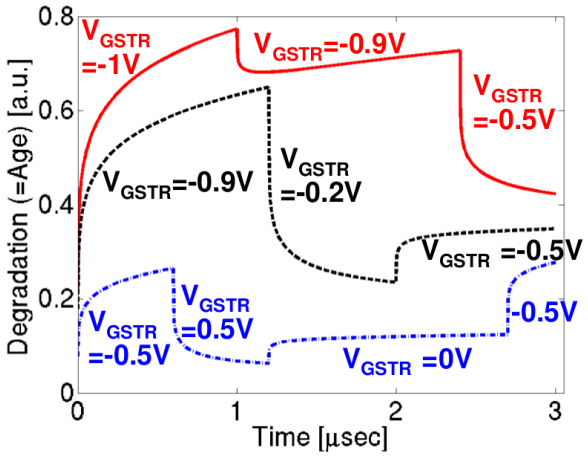


Fig. 2.10: Several stress patterns, where the final degradation depends on the stress history[9]

An approach that aims to properly adress the aforementioned flaw is [9]. In this work, it is assumed that the impact of degradation could be quantifiable in an encapsulated form. Therefore, the degradation $D(t)$ is evaluated by the convolution of the input function $g(t)$ and the response function $h(t)$ of this degradation mechanism. The pre-

Based on the RelXpert simulation flowchart presented above, we can make an important observation: For each simulation step, no memory of the stress history is taken into consideration between the independent subsequent transient steps. However, it is crucial to incorporate the past stress of a device when considering the evolution of BTI degradation. Such an argument is adequately demonstrated in Figure 2.10.

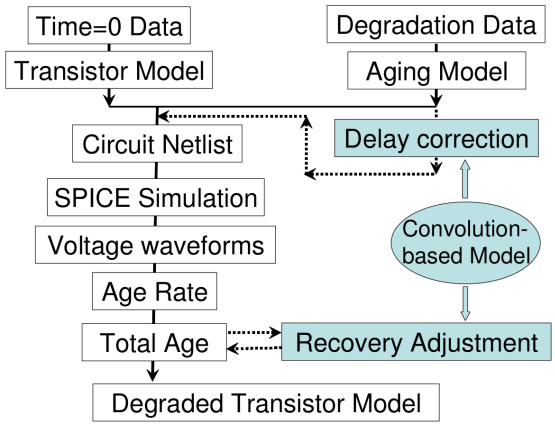


Fig. 2.11: Simulation flow, as presented in [9]

sented model is applied to RelXpert and the added steps are shown as the shaded blocks of Figure 2.11 The SPICE simulation that is driven from RelXpert generates the voltage waveforms for all transistors in the circuit. The total age computed without stress history is substituted by the convolution-based result. In addition, the corresponding delay correction is provided for the following simulation steps.

### 2.3.3 One-time Analysis with Device Individuality

As we have already mentioned, with the CMOS downscaling trend, a single device may not be still representative of all devices of the same technology. Consequently, it is crucial that a BTI approach accurately incorporates this device individuality. A notable work that aims to capture the time-dependent variability of degradation mechanisms is [23]. In this work, a "defect-centric" approach is used that is founded upon the atomistic BTI model [1]. The total $V_{th}$ degradation due to BTI is modeled from the occupancy transitions of gate oxide defects in each particular FET. That way, the time-dependent variability is taken into consideration at both the device and circuit level, and the device individuality is therefore properly emerged.

The main differentiator of [23] from other state-of-the-art atomistic BTI approaches is the methodology used for the $V_{th}$ shift due to BTI to be evaluated. Unlike other transient simulation atomistic approaches that will be presented in the following Subsection, in this work an analytical description of defects using capture/emission time (CET) maps is proposed. These CET maps describe the probability density function of capture and emission times for both high ($H$) and low ($L$) voltage levels. The occupancy probability for AC stressing is calculated for a given frequency $f$, duty factor $DF$ and stress time $t_{stress}$. The $t_{stress}$ dependency of the occupancy probability formula indicates the one-time analysis nature of this methodology.

By multiplying the original CET map with this occupancy probability map a CET-active map is constructed. This resulting map describes the distribution of the active traps after the corresponding stress waveform. By numerical integrating of the CET-active map, the

ratio of occupied traps and therefore the mean number of captured traps can be obtained. The total $\Delta V_{th}$ for this given mean number of traps $N_T$ is evaluated. The $V_{th}$ shift of each transistor is provided to a SPICE level simulator. The circuit under test is SRAM cells and the performance metric is the SNM degradation.

### 2.3.4 Transient Analysis with Device Individuality

As it was indicated at our systematic classification (Figure 2.2), this Subsection is the residence of the current thesis. Thus, any other works that could be classified here, share similar goals with our implementation. Two representative examples of a novel transient BTI simulation approach with device individuality are [10] and [1].

Those approaches are founded upon the atomistic BTI model [1]. By properly porting the stochastic nature of individual oxide defects into commercial circuit simulation tools, the proposed framework realistically monitors the occupancy transitions and their time-dependent variability impact on FET operation during the circuit simulation. Moreover, the occupancy probability formula is dependent on the adaptive time-step of every simulation phase. Thus, the transient nature of this approach is quite clear. In addition, this probability is also correlated with the imposed workload at any given time and the workload dependency is therefore properly incorporated.

More specifically, the simulation flowchart of [10] is illustrated in Figure 2.12. A control script (Perl script) is used to annotate the `netlist.cir` file with defects. The annotated list is simulated using the Cadence® Virtuoso® Spectre® commercial simulator. A proper Verilog-A model is used for the transient part of the BTI model to be implemented. Note that this work will be used as one of the reference tools for the correctness of our implementation to be verified.

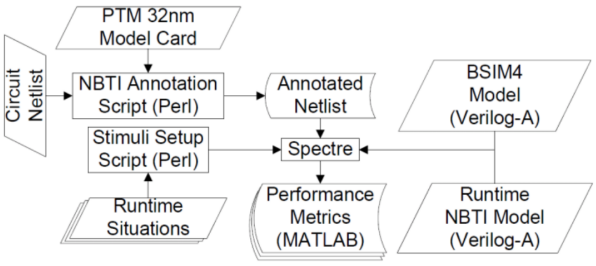


Fig. 2.12: Simulation flowchart, as presented in [10]

## 2.4 Conclusions

From the latter classification, we can make several observations on the landscape of BTI modeling. First of all, we can easily notice that in many approaches the $V_{th}$ variation is emulated with a voltage source on the gate terminal of the transistor. The degradation due to BTI is therefore precomputed before the actual circuit simulation. As a result, only the impact of a predefined BTI shift is actually simulated and the transient evolution of the BTI effect is not fully captured.

In many cases, we observe several insufficiencies that are related to the analytical methods used to represent the stress applied on a device. Besides their computational complexity during the preprocessing stage of $\Delta V_{th}$ evaluation, many of these statistical methods average out the input signal and therefore cannot accurately trace the workload dependent variability at run time. Another important observation is that in the majority of state-of-the-art works no memory of the stress history is taken into consideration between the independent subsequent stress phases. In addition, we note that the majority of these methods are limited to NBTI and models that simultanesouly include the PBTI or the Random Telegraph Noise (RTN) effects are extremely rare.

Furthermore, as we have already pointed out the uniform reliability behaviour of devices of older technologies cannot be now taken as an efficient assumption for modern downscaled devices. As a consequence, an individual device may not be representative of the way all devices of the same technology behave. In the latter point of view, the atomistic BTI model seems a better option over the RD model for the BTI effect to be properly ported to transistor models in the sub–100nm regime, such as the BSIM4.6.0 model [24].

More specifically, as far as the atomistic BTI approach with transient analysis is concerned, we can make several noteworthy observations. It is important that both the presented related works realistically capture the workload dependency under actual, irregular circuit workloads. Notice that the defects are characterized by widely distributed time scales. Thus, these approaches incorporate not only NBTI, but PBTI and RTN as well.

Nevertheless, the interface of these two frameworks is complicated because of the several

heterogeneous source layers (i.e. Perl scripts, Verilog-A BTI models), that are separated from the simulation software. As a consequence, the BTI initialization cannot take place during the setup phase of the actual simulator. Therefore, an impractical presimulation phase is necessary for each FET device to be annotated with BTI parameters. Moreover, these models are built on the top of commercial tools. Although industrial tools usually constitute an accuracy guarantee, their usage could be a constraint in cases where access to such costly software options is not available.

The proposed framework of this current work maintains all the aforentioned strenghts of the representative atomistc BTI approaches with transient analysis. At the same time, it aims to provide a straight-forward BTI-to-simulator interface. To this end, the transient part of the atomistic model is embedded in the state-of-the-art BSIM4.6.0 transistor model [24]. Thus, a computationally feasible and user-friendly simulation tool will be developed. More details on the implementation and its usage will be provided within the following Chapter.

# Chapter 3
# Tool Description

## 3.1 Introduction

The purpose of this Chapter is to describe the simulation tool where the Bias Temperature Instability (BTI) model has been incorporated. In order to enable maximum flexibility during development stages, an open source version of the SPICE class of programs was selected, namely `ngspice` [25]. `Ngspice` is a general-purpose circuit simulation framework, publicly available through [26].

The selected device that will be used as an implementation framework is the Berkeley Short-channel IGFET Model (BSIM) [27]. BSIM is a family of MOSFET transistor models that represent the behavior of devices found on an IC. The BSIM model simulates accurately a great variety of physics-based phenomena, including second-order effects such as channel-length modulation and subthreshold conduction. More specifically, the BTI model will be ported to the BSIM4.6.0 model. BSIM4.6.0 is the latest extension of BSIM that was developed to address the MOSFET physical effects into sub-100nm regime [24].

Within the following Section we will present the BSIM model and its implementation details. Therefore, the target code base involves only the BSIM functionality, rather than the functionality of `ngspice` as a whole. A complete BSIM flowchart will be presented in order to examine all the simulation steps during setup and run time. The BSIM to SPICE interface will also be described. Thereby it will become obvious where all the BTI steps should be added. All these code alterations will be described. Having addressed the above subjects, the current Chapter will be completed with a brief user manual of the developed simulation tool.

## 3.2 BSIM Flowchart

In this Section we will present the functionality of the existing BSIM model. We will inspect the behavior of BSIM as a simulator object. Hence we will have to meticulously examine how the SPICE simulator manipulates each type of object. A model is a second level construct which the simulator manipulates [28]. Functions that will create, delete or set the model parameters should be provided. In addition, instances are treated as instances of a model and thus third level objects. They also should be described by a set of functions that will create, delete, initialize them or define their behavior. We can observe that such a set of device specific routines should be implemented for both a model and an instance.

According to the available documentation [29], each device is described by a structure which contains three types of pointers: (i) Pointers to functions which provide device specific operations, (ii) pointers to parameter descriptors and (iii) pointers to functions necessary at the higher SPICE and user-interface levels. This data structure should follow certain rules and specifications. A useful advice from the related documentation is to avoid building a new device from scratch [29]. It is obviously more efficient to implement it by walking through the framework of an existing device. By following this approach all these device specifications are directly met.

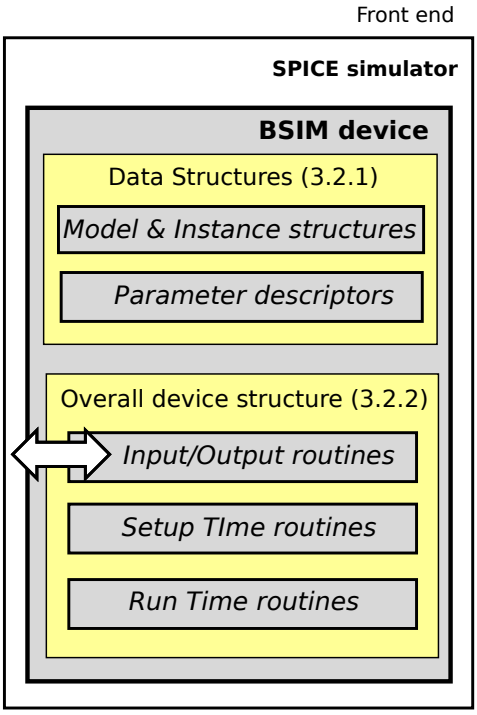

Fig. 3.1: BSIM device hierarchical approach

In the following subsections we will extensively demonstrate the BSIM device. All the declared data structures and their parameters will be presented. In addition, the functionality of basic BSIM routines will be illustrated. An indicative hierarchy of the BSIM abstraction levels is given in Figure 3.1. This hierarchical approach will be also followed at the next section where all the code alterations will be presented.

### 3.2.1 Data Structures

#### Model and Instance structures - BSIM header file

Like all other SPICE devices, BSIM should have its internal data structures defined. There are two required structures : (i) one for the device model and (ii) one for the instance. Both of them should be properly declared inside the header file and follow a certain format. The implemented header file of BSIM is `bsim4v6def.h` . Inside this code file two struct variable types are defined. These are the `BSIM4v6model` and `BSIM4v6instance` records for the device and the instance structure respectively.

In order to be connected to its instances, the model structure contains some initial pointer entries. Apart from these standard entries, the model parameters that are common among different devices should be properly declared. For example, if a number of devices are alike to have common substrate doping concentration, such a model parameter should be declared inside the `BSIM4v6model` record. The instance structure also follows a similar format. There are its specific entries followed by all the needed instance parameters.

#### Parameter descriptors - `b4v6.c` code file

The implementor should provide an adequate description of all the declared parameters. Hence the next pair of structures that is complementary to the the aforementioned pair is the parameter descriptor arrays. These arrays are properly defined inside the `b4v6.c` file. For each included parameter, they provide information needed by the simulator routines. For example, the data type of a variable is set and also it is defined whether it is an input or an output variable.

### 3.2.2 Overall device structure - `bsim4v6init.c` file

The next stage is to examine the overall device structure. This structure contains information required from the front-end interface and pointers to all the device routines. When a certain operation needs to be performed on the device, this overall structure identifies the

specific subroutine to be called. There are lot of different functions which perform certain tasks. For example, functions are provided to set parameters, to delete a model from a circuit or to initialize an instance. We should point out that not all these routines will be extensively presented in this thesis.

Both in terms of modeling and behavior, we observe that our implemented mechanism is responsible only for threshold voltage fluctuations. This means that we should focus on how SPICE computes the $V_{th}$ value. We will try to locate all the simulation steps where the threshold voltage is computed during setup and at run time. Therefore we will examine only the corresponding routines and code files.

### BSIM to SPICE Interface - Input/Output routines

A crucial functionality of any implemented model is the ability to interact with the higher SPICE and user-interface levels. Consequently, there should be a simple interface that will permit the developer to handle variables to and from the device. The BSIM to SPICE interface consists of 4 simple input and output routines.

First of all, the input routines are used by the frond end to communicate input parameters to the device. The code files `b4v6par.c` and `b4v6mpar.c` handle input values for an instance and a model parameter respectively. Users can fully define all the model parameters through the model card. The `b4v6mpar.c` function takes the model card values from the input parser and sets the appropriate fields in the per model data structure. The `b4v6par.c` functionality is analogous to `b4v6mpar.c`, but handles values for instance parameters. The user can directly set an instance parameter inside the circuit datafile. For example, they can define the MOS channel length at the same line where the device is declared.

The pair of output routines provides a complementary behavior. These routines are used by the simulator to obtain data from the device. The code files `b4v6ask.c` and `b4v6mask.c` handle output values for an instance and a model parameter respectively. They both assign the asked values to the proper field inside the `IFvalue`. `IFvalue` is the structure used to pass values to and from the simulator.

### Setup time routines

The function that performs the first step of preparing a device for simulation is the setup function. The corresponding BSIM code file is `b4v6set.c`. All the device parameters are initialized to their given or their default values. Therefore the threshold voltage is properly set during this stage. All the executed steps are shown in the upper part of Figure 3.2.

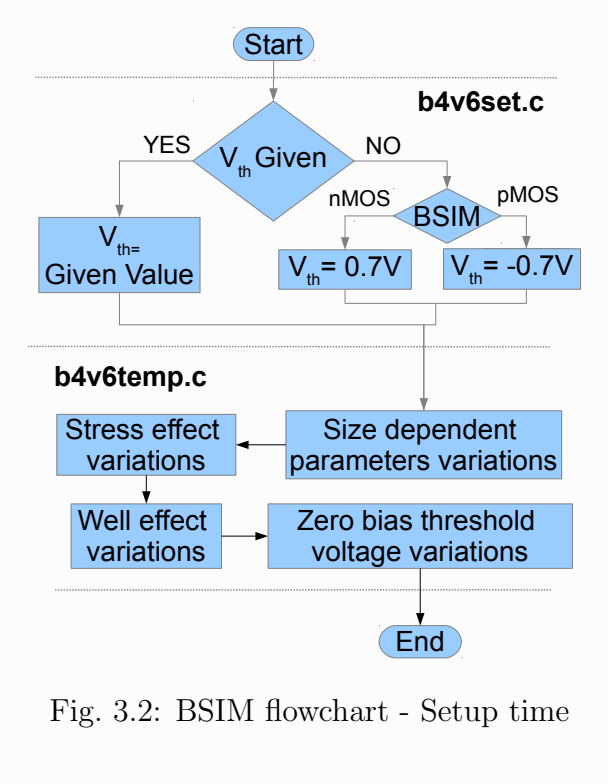


Fig. 3.2: BSIM flowchart - Setup time

The following stage that completes the device preprocessing is the temperature routine. The given `b4v6temp.c` code file is executed. Different MOSFET physical effects are initialized and computed. In the lower part of Figure 3.2 we summarize all the steps that are related to threshold voltage variations.

### Run time routines

After the aforementioned initial steps, the dc or transient analysis iterations follow. The `b4v6ld.c` code file is the device routine that is responsible for evaluating all instances at each iteration. Therefore this routine is the most frequently called function and it is crucial that it is as efficient as possible. All the executed steps for $V_{th}$ calculation at run time are shown in Figure 3.3. After the `b4v6ld.c` code file has been executed, the threshold voltage is updated to its newest value.

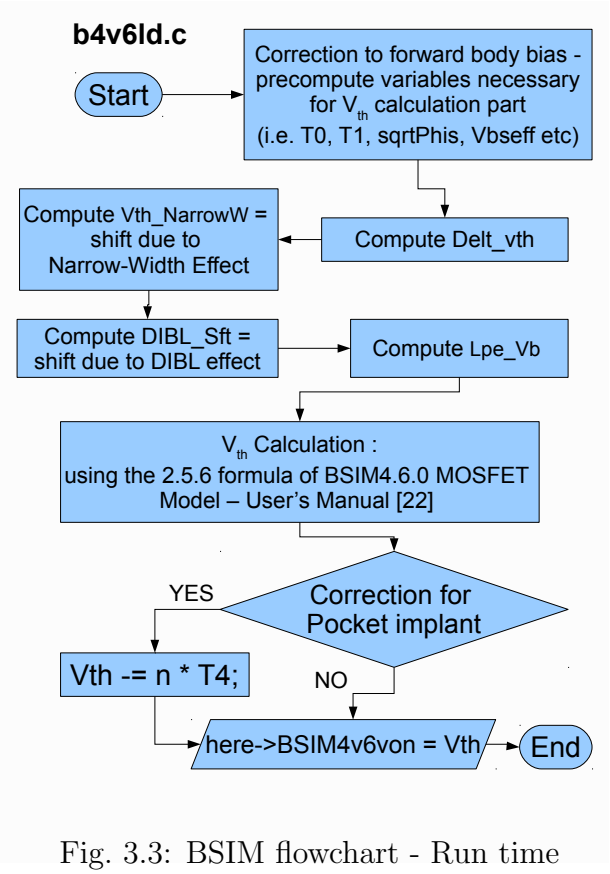


Fig. 3.3: BSIM flowchart - Run time

## 3.3 Alterations to the Source Code

In the previous section we presented the BSIM device as a simulator object. All its data structures and its routines were illustrated. Thus, all the threshold voltage computation steps both during setup and at run time have been revealed. With this detailed code examination, we are able to locate all these lines where further code alterations should be added, in order to enable BTI support. We will follow the same hierarchical approach as in the previous Section. All these code alterations that will implement the BTI mechanism (cyan blocks in Figure 3.4) will be presented.

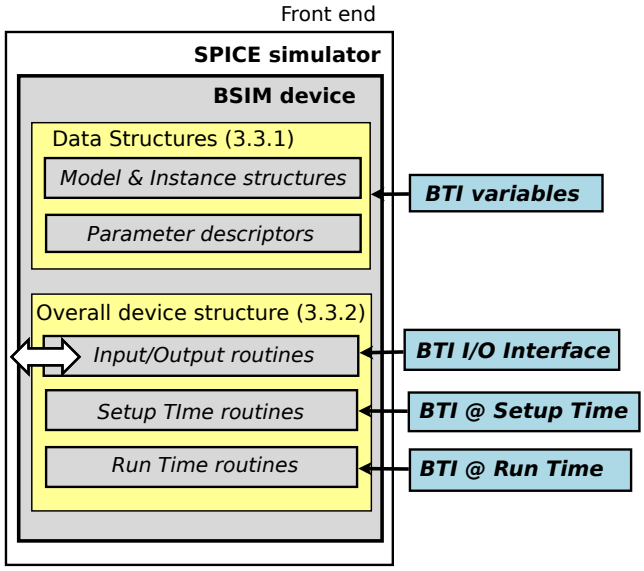


Fig. 3.4: BSIM flowchart - BTI alterations

### 3.3.1 Data Structures

#### Model and Instance structures - BSIM header file

As we have already mentioned, both Model and Instance structures should be properly declared inside the header file and follow a certain format. Using the existing BSIM header file, this asked format is satisfied. The only remaining alteration is the declaration of the proper BTI variables. Therefore we will start with a brief description of all the parameters that are needed for the BTI model. It is crucial that we make a correct discrimination between the per instance and the per model BTI variables.

The per model data include the parameters that are common within the different devices of the model. These will be predefined from the user inside the model card file. The BTI variables that will be independent of any instance are the parameters of the statistical distributions from which we draw to initialize gate stack defects (their number, their time

constants and the associated $\Delta V_{th}$). Hence, these variables are added at the `BSIM4v6model` structure.

On the other hand, we can easily observe that each instance is enhanced with a random number of traps. Each trap has different threshold voltage impact and time constants. These parameters of each trap follow specific distributions[10]. Hence all these data should be placed in the `BSIM4v6instance` data structure. In addition, a BTI flag variable will be used to indicate whether the BTI mechanism is ignored or not. A defect variable will be also defined to indicate whether a trap is occupied or not.

#### Parameter descriptors - `b4v6.c` code file

All the aforementioned BTI variables should be also included in the descriptor arrays at the `b4v6.c` code file. Therefore inside the `BSIM4v6mPTable` structure we added the data type and a short description for each one of the four BTI model parameters. We also followed the same steps for all the BTI instance parameters inside the `BSIM4v6pTable`.

### 3.3.2 Overall device structure

The next implementation stage is to fill in the overall device structure. Using the existing BSIM routines all the asked structure rules are satisfied. Our coding alterations will address three important functionalities: (i) The BTI variable handling through Input/Output routines, (ii) the BTI model initialization during setup time and (iii) the $V_{th}$ fluctuations due to BTI at run time. Therefore we will examine only the corresponding routines and their source code files.

#### BSIM to SPICE Interface - Input/Output routines

With all the BTI variables declared, we should now investigate all possible I/O simulation requirements. We will start with the fairly simple case of the BTI model parameters. The distribution parameters for interface trap instantiation will be only used for the BTI ini-

tialization during setup time. Consequently there is no need for them to be included in the model output routine (`b4v6mask.c` file). We should only add the appropriate fields inside the `b4v6mpar.c` code file.

As we have already mentioned, this function takes the model card values from the input parser. Users can fully define all the BTI model parameters through the model card file. Note that such an approach comes in direct contrast to previous works [1][10], since no pre-processing stages are required. Also the netlist syntax is perfectly retained, which enhances the user-friendly attributes of our approach.

An analogous implementation inside the `b4v6par.c` file would be more than inefficient. Usage of the input parser for the BTI instance parameters would demand that all input values be typed inside the `netlist.cir` file. Such a practice is followed in [1]. In this work a control script (Perl script) is used to annotate the `netlist.cir` file with defects, making the entire process hugely impractical for larger circuits (SRAMs, Multipliers, etc.) with an enormous number of traps to be simulated.

We will follow a completely different approach on the way the simulator will manipulate the BTI instance parameters. For any input and output requirements a specific file will be used. Users can define its exact name. The BTI output file contains the BTI instance parameters of all BSIM devices and provides a straightforward BTI-to-front-end interface. When a transient simulation is completed the final BTI instance values are saved at the BTI output file. These output data can be used as an input file for later BTI simulations. That way we perfectly decouple the netlist structure and syntax of the target circuit from the associated gate stack defect database.

**Setup time routines**

Having implemented the BTI-to-front-end interface, the following stage is the initialization of BTI variables during the setup time. For that purpose the BSIM setup routine will be used. As we have already mentioned, the `BTI.out` file can be used as an input file for later BTI simulations. When this file is present at the current simulation directory the setup

routine handles its data as BTI initial conditions. Therefore, a critical code addition was a straightforward text file parser inside the `b4v6set.c` code file that parses the `BTI.out` data and assigns them to the initial instances values.

On the contrary, when no BTI initial conditions are present the BTI parameters will be generated according to the provided model card data. The number of traps, their time constants and their threshold voltage impact follow specific distributions [10]. Hence another significant code addition was the implementation of these distributions inside the `b4v6set.c` code file.

### Run time routines

The most crucial part of our work is the proper evaluation of all BTI parameters at run time. Both in terms of modeling and behavior, our model accurately implements the simulation flowchart of the atomistic BTI model described in [10]. During every transient analysis iteration, we loop through all the defects of all BSIM devices. For each individual defect, the $P_c$ and $P_e$ probabilities are determined using the appropriate formulas found in [10]. Their computed values are compared to a random number. If the random number is found smaller than the probability, the respective occupancy transition occurs. When a defect becomes occupied, its threshold voltage impact is added to the runtime $V_{th}$ value of the device. All these execution steps are shown in Figure 3.5

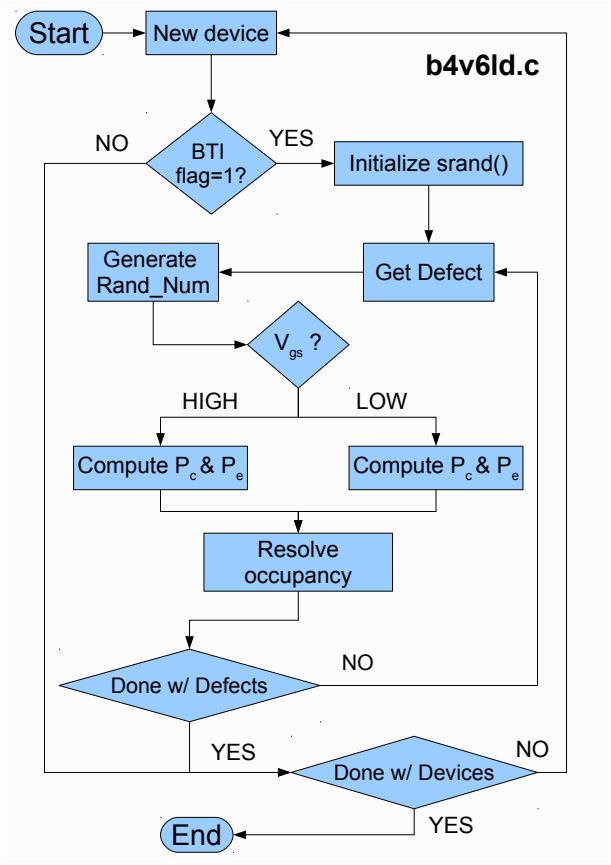


Fig. 3.5: BTI simulation flowchart - Run time

## 3.4 User Manual

In this Section we will provide a brief user manual of the implemented simulation tool. The tool is actually the LINUX version of `ngspice` along with the aforementioned code alterations. Therefore all the prerequisite steps for source compilation and installation are exactly the same as described in [25]. We will focus on the operation of the added BTI model using the command line interface of `ngspice`.

The operation will be illustrated through a simple inverter. The circuit description file for this example is provided below:

**The inverter.cir file – The netlist of a simple inverter**

```
** Subcircuits and Modelcards
.include modelcard
.include subcircuits

** DC Sources
Vvdd vdd 0 1
Vvss vss 0 0

**Input Source
Vinput input 0 pulse (0 1 5n 0.01n 0.01n 5n 10n) DC 0

**Inverter Instantiation
X1 vdd vss input output inv
C1 output 0 2f

**Simulation Definition *
.tran 10ns 10ns

** Output File Definition *
.save tran @m.x1.m0[vth]

.control
set filetype=ascii
.endc
```

Observe that we select a netlist that contains subcircuits, in order to fully illustrate the functionality of the implemented tool. Both the appropriate modelcard and subcircuits files should be provided at the current directory. More specifically, the subcircuits file used for this example is:

**The subcircuits file – The subcircuit of a simple inverter**

```
** Inverter = 2 devices
.subckt inv vdd vss in out

M0 out in vdd vdd p1 W=360n L=90n
M1 out in vss vss n1 W=180n L=90n

.ends
```

The three different utilization methods will be presented in the following Subsections.

### 3.4.1 Normal Usage

We will start with the normal case of `ngspice` utilization where the BTI model is ignored. As an actual `ngspice` distribution, our tool fully maintains all the functionalities of the original SPICE simulator. Hence the BTI model is not a limited tool but a part of a powerful and flexible simulation program. *This is an important characteristic that permits users to exploit simultaneously the BTI model with all the other available SPICE features.*

To simulate this circuit the user can simply type the following command :

```
$ ngspice -b -r rawfile -o logfile inverter.cir
```

We have used the simple command line options `-b`, `-r` and `-o` as described in [25]. The `ngspice` will run the simulation and store the output data in the `rawfile`. Comments, warnings and similar information go to `logfile`. To verify that BTI is ignored, just list the contents of the current directory by issuing the `ls` command after the simulation. Note that no BTI output file is present, unlike the `rawfile` and `logfile` files that have been properly generated.

### 3.4.2 BTI without Initial Conditions

In order to perform a BTI simulation, a specific BTI output filename should be defined. Consequently, the required step before every BTI simulation is to set the `BTI_FILENAME` environment variable. Only with that environment variable available, the BTI model will be taken into consideration during transient analysis. Therefore, we should type:

```
$ export BTI_FILENAME=BTI.out
```

By using the latter command, we have indicated the `BTI.out` string as the name of the BTI output file. We will maintain this name within the following examples. Of course, users can accordingly select other BTI output filenames for their simulations.

The first way of utilization to be described is a BTI simulation without any initial conditions given. Again, we will use the same example of a simple inverter. To run a BTI simulation without initial conditions means that a BTI output file of a previous simulation does not exist in the current directory. In our case, no `BTI.out` file should be present. As we have already mentioned, the BTI instance variables will be generated according to the provided BTI model parameters. Therefore, we add the following lines inside the model card :

```
+trapdensity = 1.0e+10
+stepmean = 0.005
+timeminimum = -8
+timemaximum = +8
```

Notice that we have added them only inside the `.model n1`. This was done to illustrate the fact that BTI will be properly not taken into consideration at the `.model p1`. Once again, to perform the simulation just type the following command :

```
$ ngspice -b -r rawfile -o logfile inverter.cir
```

An output file named `BTI.out` has now been generated. If we display its contents, we have:

```
_Model_n1_Instance_m.x1.m1_BTI= 1
ntraps= 3
defects= 1 1 0
teh= 4.978282e+02 8.086966e+04 8.041643e+03
tch= 1.113378e-08 5.513980e-06 9.136106e+07
tel= 1.506615e-08 1.181948e-06 1.327282e-05
tcl= 1.890342e+01 4.823635e-07 3.376247e+06
deltavth= 0.003507 0.001563 0.006897
_Model_p1_Instance_m.x1.m0_BTI= 0
```

Each instance of all different models is listed with its BTI flag. As we were expecting, the first BTI flag is one. This declares that BTI was taken into consideration at this device.

Therefore all the BTI instance parameters are listed below the flag variable. Time constants and threshold voltage impact of all traps are provided. The state of each trap after the BTI simulation is also given. For example, notice that the nMOS device is enhanced with 3 traps. The `defects` vector demonstrates which traps are occupied (value `1`) and which are not (value `0`). Furthermore, we can verify that BTI is ignored for the pMOS device.

An important feature is that users can access the $P_c$ and $P_e$ probabilities of the implemented BTI model. Their values that are computed at run time can be accessed and saved as easily as all the other internal BSIM4.6.0 device parameters. We can simply specify these probabilities among the parameters listed in the `.save` statement of the `inverter.cir` file. The syntax of a `.save` statement is described in [25]. *This specific syntax is not only perfectly retained but also properly expanded, in order to enhance the user-friendly attributes of our approach.*

More specifically, the default format described in [25] is used:

`@device_identifier.subcircuit_name.<subcircuit_name_nn>.device_name[parameter]`

The above general form is extended to a `@device_full_name[probability +flag]` format, where the probability can be replaced with either `pc` or `pe` to select the $P_c$ or the $P_e$ values respectively. Furthermore, the `pc`, `pe` identifiers should be accompanied by numbers that will indicate only certain traps to be accessed. For instance, in our case the `m.x1.m0` nMOS device has 3 traps. We can select only the $P_e$ values of the first and the third trap to be saved by adding the following `.save` line inside the `inverter.cir` file:

```
@m.x1.m1[pe 1 3]
```

In order to store the probabilities of all traps, we can simply replace the numbers with the `all` identifier. If the $P_c$ and $P_e$ parameters are specified in the `.save` statement, an extra appropriate BTI output file will be generated. This file will be named after the `BTI_FILENAME` environment variable, plus `.p` at the end of the word. Note that this output file has a "rawfile" format, in order to be easily handled the same way as the default `ngspice` ascii rawfile data.

The netlist of the inverter with the proper syntax of the `.save` statement is provided below:

**The inverter.cir file – Syntax of the .save statement to access $P_c$ , $P_e$ values**

```
** An simple inverter
** Subcircuits and Modelcards
.include modelcard
.include subcircuits

** DC Sources
Vvdd vdd 0 1
Vvss vss 0 0

**Input Source
Vinput input 0 pulse (0 1 5n 0.01n 0.01n 5n 10n) DC 0

**Inverter Instantiation
X1 vdd vss input output inv
C1 output 0 2f

**Simulation Definition *
.tran 10ns 10ns

** Output File Definition *
.save tran @m.x1.m0[vth] @m.x1.m1[pc all] @m.x1.m1[pe 1 3]

.control
set filetype=ascii
.endc
```

We delete the older `BTI.out` file and we perfom the same simulation as before. Notice that an extra output file named `BTI.out.p` is present in the current directory. We can verify from its contents that the selected probabilities have been stored:

```
Title:  ** Probabilities of BTI events **
Plotname:  BTI Transient Analysis
Flags:  real
No.  Variables:  6
Variables:
        0 time time
        1 Modeln1_Instancem.x1.m1_Trap1_pc value
        2 Modeln1_Instancem.x1.m1_Trap1_pe value
        3 Modeln1_Instancem.x1.m1_Trap2_pc value
        4 Modeln1_Instancem.x1.m1_Trap3_pc value
        5 Modeln1_Instancem.x1.m1_Trap3_pe value
Values:
[..]
```

Finally, it goes without saying that we can simply unset the defined BTI_FILENAME environment variable to return to the default ngspice utilization. For instance, we can simply type:

```
$ unset BTI_FILENAME
```

If we simulate again the netlist of the same inverter, we will have the extact results as described in the previous Subsection.

### 3.4.3 BTI with Initial Conditions

The third operation to be described is the BTI simulation with initial conditions. Its only difference from the previous case is that the BTI instance parameters will not be generated but they will be parsed from an existing BTI output file. Make sure that this existing file has the same name with the BTI_FILENAME environment variable. At this point, we have already completed a BTI simulation and the BTI.out file has been generated. To run a new BTI simulation with initial conditions just include this file in the current directory and type the following command :

```
$ ngspice -b -r rawfile -o logfile inverter.cir
```

If you display the contents of the newest BTI output file, the following lines should appear:

```
_Model_n1_Instance_m.x1.m1_BTI= 1
ntraps= 3
defects= 1 1 0
teh= 4.978282e+02 8.086966e+04 8.041643e+03
tch= 1.113378e-08 5.513980e-06 9.136106e+07
tel= 1.506615e-08 1.181948e-06 1.327282e-05
tcl= 1.890342e+01 4.823635e-07 3.376247e+06
deltavth= 0.003507 0.001563 0.006897
_Model_p1_Instance_m.x1.m0_BTI= 0
```

It is important to observe that time constants and threshold voltage impact of all traps maintain the same values as before. This proves that the provided BTI.out file was parsed correctly and its data were used as initial conditions.

If we had desired to select other initial conditions for BTI instance parameters, we could simply type and save a totally new BTI.out file. It is crucial that the new typed file follows EXACTLY the same format as the one demonstrated before; otherwise the BTI parser will not be able to handle the provided data correctly

Once again, it goes without saying that we can simply set a new name for the `BTI_FILENAME` environment variable to return to the case of BTI simulation without initial conditions. For instance, we can simply type:

```
$ export BTI_FILENAME=NewBTI.out
```

No initial conditions will now be provided as no BTI output file named `NewBTI.out` is present. If we simulate again the netlist of the same inverter, we will have similar results as described in the previous Subsection. Indeed, observe that an output file named `NewBTI.out` will be generated.

## 3.5 Conclusion

In this Chapter, we described the development stages of our simulation tool. An existing device model was used as the implementation guideline, instead of building a new device from scratch. The selected device model was the BSIM4.6.0 model [24]. First of all, all the BSIM parameters, routines and data stractures were presented. All the execution steps where the $V_{th}$ value is computed were illustrated. That way, we pointed out where all the BTI steps should be added. Then, we extensively presented all the code alterations for the BTI model to be implemented. Having addressed the above subjects, a brief user manual was provided. The usage of the resulted tool with representative examples was demonstrated.

# Chapter 4

# Verification Test Cases

## 4.1 Introduction

In this Chapter, we will verify the correctness of the BTI/RTN-enhanced `ngspice`. As we have already mentioned, the implemented framework is founded upon the atomistic BTI model. Further details are provided in [1]. The selected test cases should not be limited to examples that will verify only the operational accuracy of the simulation software. It is important to fully investigate the phenomenological validity of the BTI model as well.

Thus, we will properly organize the inspected test cases into three types of verification processes: (i) phenomenological and (ii) detailed verification, (iii) validation of BTI impact. Each one of these types will be presented within the three following Sections respectively. The netlist files of the circuits under test will be provided. For each test case, the simulation results will be extensively presented. All these output data will be compared to already published results. That way, the correctness of the implemented framework will be verified.

## 4.2 Phenomenological Verification

In the current Section, we will demonstrate that our simulation tool realistically captures the physical principles of the atomistic BTI model. This atomistic approach is based on the stohastic properties of gate oxide defects. The physics of defect occupancy is experimentally proven in [11]. From this work, we quote a noteworthy part of its conclusions: "This theory passes all sets of performed experimental tests. The model can thus be transferred to circuit simulators in order to simulate the effect of individual traps under AC workloads."

Consequently, an obviously straightforward way to inspect the phenomenological validity is to adopt a similar experimental setup of [11] in the context of our simulator. By successfully reproducing the experimental results, the simulation accuracy in terms of BTI modeling will be verified. More specifically, in [11] the responce of a single trap in a single selected pMOS device is studied. The device is gate stressed with an AC signal for a certain time $t_{stress}$. Afterwards, the relaxation transient follows for a relaxation time $t_{relax}$. Both the capture and emission events are observed and the results are illustrated in representative figures.

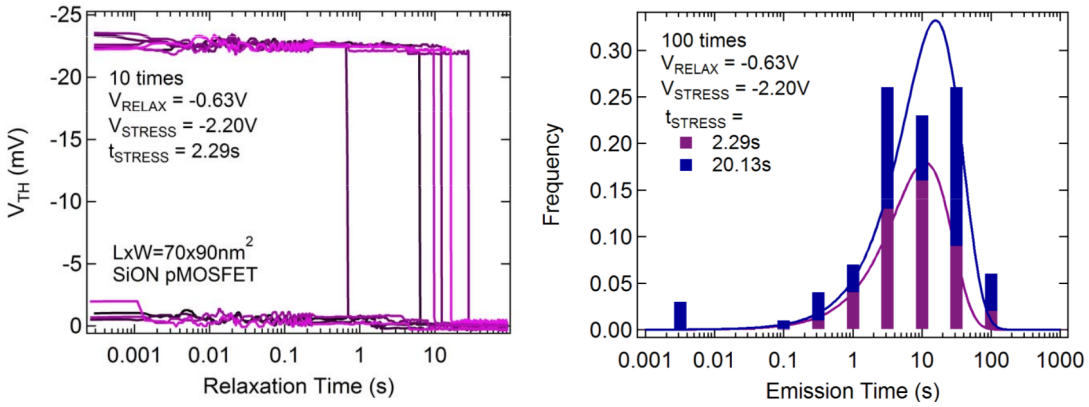


Fig. 4.1: Relaxation transient results as presented in [11]: (a) Typical $V_{TH}$ transients after DC stress and (b) Histograms of emission times extracted from 100 $V_{TH}$ relaxation transients

In the context of our test case, we intend to reproduce similar graphs as shown in Figure 4.1. Note that we focus on the relaxation transient of a nMOS devide with a single defect. Given that our implementation is not restricted to NBTI or PBTI, the choice of test case does not affect the validity of our claims. We assume that a past stress phase has been already completed and the single trap is occupied. In terms of simulation setup, this means that a prior BTI simulation has been already performed and an output file of initial conditions is used. The `BTI.out` file and the netlist of the simulated nMOS device are respectively:

**The BTI.out file – The initial conditions of the relaxation transients**

```
_Model_n1_Instance_m1_BTI= 1
ntraps= 1
defects= 1
teh= 2.000000e-09
tch= 5.000000e+10
tel= 5.000000e-04
tcl= 1.000000e+14
deltavth= 0.030000
```

**The singleDevice.cir file – The netlist of a single nMOS device**

```
** An simple nMOS device
** Modelcards
.include modelcard

** DC Sources
Vvddcell vddcell 0 1
Vvddbulk vddbulk 0 1
Vvsscell vsscell 0 0
Vvssbulk vssbulk 0 0

** Input Source
Vinput input 0 0

** nMOS device
m1 vddcell input vsscell vssbulk n1 W=180n L=90n

** Simulation Definition *
.tran 10ns 10ns

** Output File Definition *
.save tran @m1[vth]

.control
set filetype=ascii
.endc
```

We repeat the above simulation configuration for four different $t_{relax}$ durations: (i) $10nsec$, (ii) $1msec$, (iii) $1sec$ and (iv) $10sec$. For each case, the proper values of the time constants should be defined. The $\tau_{el}$ constant should be less than the $t_{relax}$ value, for the emission event to occur within the simulation duration. Here, we select the $\tau_{el}$ to be the half of $t_{relax}$ value. At the same time, we select a greater value for the $\tau_{cl}$ constant, in order not to have a new capture event right after the emission. That way, we avoid a switching activity of the single trap.

The current verification procedure can be divided into two parts. On the one hand, we demonstrate the $V_{th}$ variation for a small number of simulations, as shown in the left graph of Figure 4.1. In our case, we perform five relaxation transients. For each simulation, the emission times should be close to the characteristic mean emission time $\tau_e$[11]. In other words, the discrete steps illustrated in the $V_{th}$ transient graphs are expected to be gathered within a time interval nearby the $\tau_e$ value.

On the other hand, we record the exact time of the emission events extracted from one hundred relaxation transients. Based on the recorded values, the proper histograms are plotted, as presented in the right graph of Figure 4.1. We follow the exact histogram form as described in [11]. The emission times are binned on the logarithmic scale and the histograms are fitted with Equation 2 of [11].

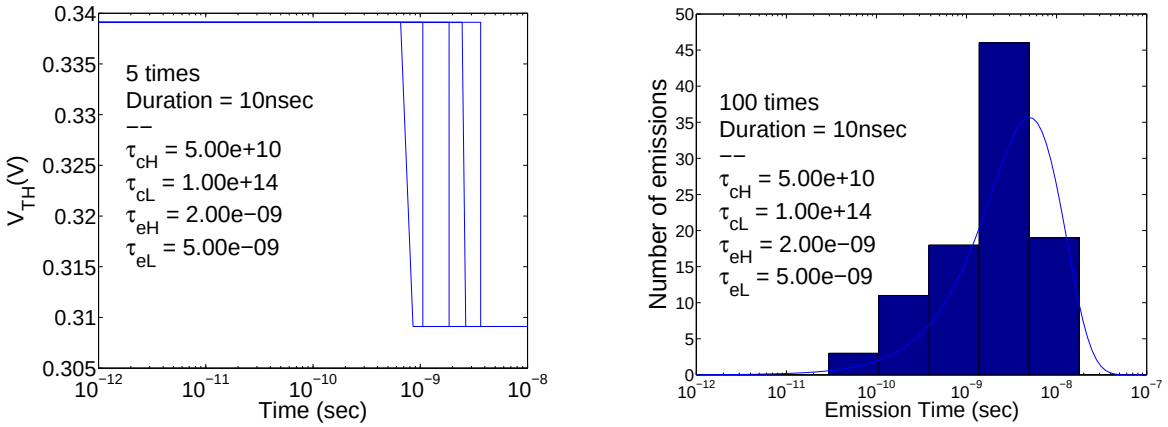


Fig. 4.2: Simulation results for $t_{relax} = 10nsec$ : (a) 5 Typical $V_{TH}$ transients and (b) Histograms of emission times extracted from 100 $V_{TH}$ relaxation transients, fitted with the equation 2 of [11]

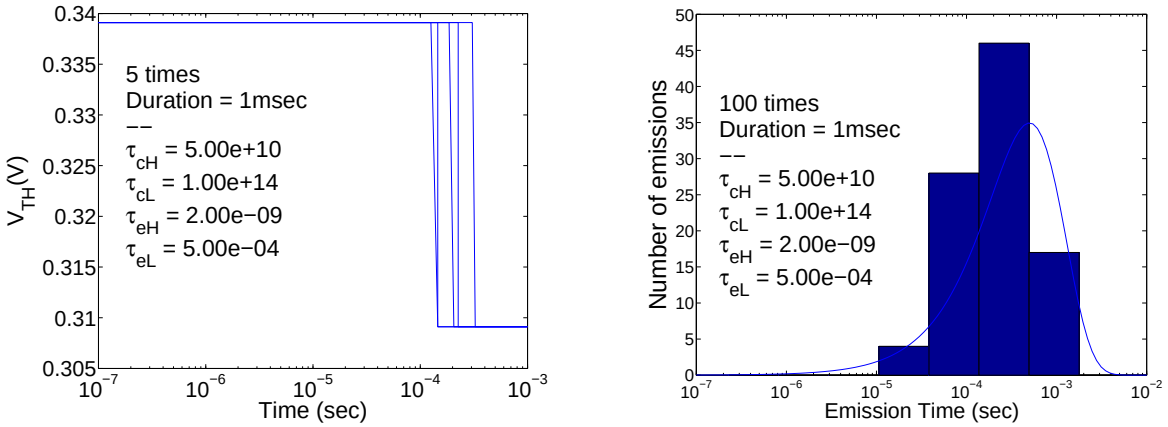


Fig. 4.3: Simulation results for $t_{relax} = 1msec$ : (a) 5 Typical $V_{TH}$ transients and (b) Histograms of emission times extracted from 100 $V_{TH}$ relaxation transients, fitted with the equation 2 of [11]

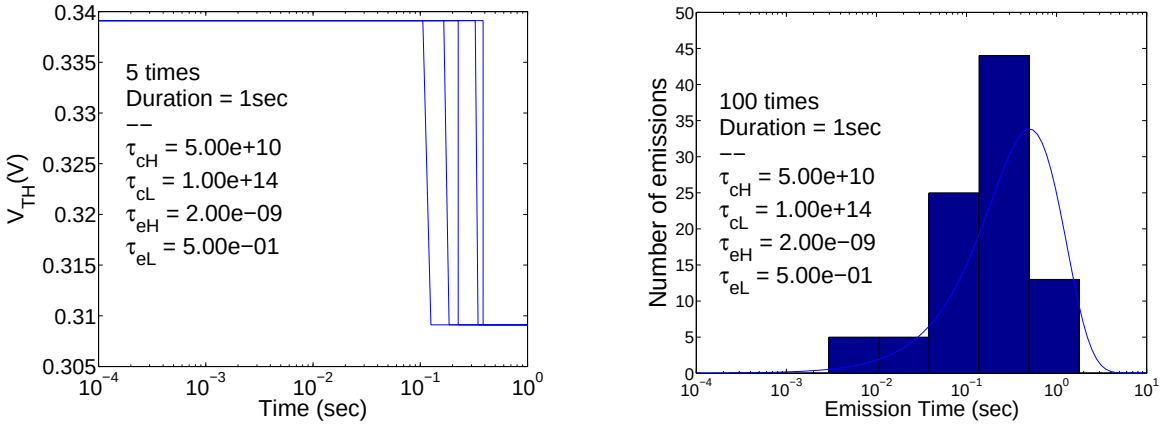


Fig. 4.4: Simulation results for $t_{relax} = 1sec$ : (a) 5 Typical $V_{TH}$ transients and (b) Histograms of emission times extracted from 100 $V_{TH}$ relaxation transients, fitted with the equation 2 of [11]

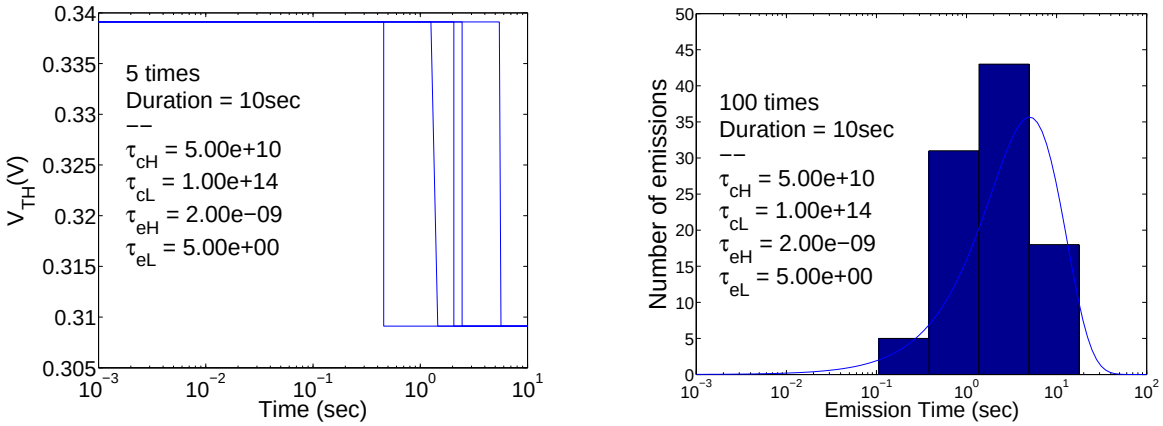


Fig. 4.5: Simulation results for $t_{relax} = 10sec$ : (a) 5 Typical $V_{TH}$ transients and (b) Histograms of emission times extracted from 100 $V_{TH}$ relaxation transients, fitted with the equation 2 of [11]

We can easily observe that the simulation output data (Figures 4.2–4.5) are consistent with the experimental results of [11] (Figure 4.1). More specifically, notice that for each different $\tau_e$ value, the emission events and therefore the $V_{th}$ discrete steps are located within a time interval near the respective emission time of the defect. In addition, we can clearly observe that the shapes of distributions sufficiently follow the equation 2 of [11] (solid blue line of Figures 4.2–4.5). Thus, the simulation tool accurately captures the behaviour of individual oxide traps.

## 4.3 Detailed Verification

We should now proceed to the second level of our verification procedure. Once again, proof of validity will be provided with comparison to already published results [1][10]. From the conclusions of [1], we quote: "The employment of industry-standard circuit simulator tools ensures correct combination of the deterministic workload-dependent component with the stochastic modeling aspect". Up to this point, we have already demonstrated that our model accurately captures the stochastic dependent nature of oxide traps. We should now verify that the time-dependent workload dependency is also realistically incorporated.

In this Section, the correctness of our implementation will be verified against the atomistic BTI model, as implemented on top of the Cadence® Virtuoso® Spectre® commercial simulator [10]. This approach has been already described in the Chapter 2 of Research Landscape. For each test case inspected here, an identical simulation will be repeated with this simulation framework. It goes without saying that, if the output data are identical to each other, a straightforward validation of correctness will be provided.

We will investigate a detailed operational example. The inverter is again selected as the circuit under test. Only NBTI will be inspected, but PBTI can be easily included, as shown in the previous Section. The metrics to be compared are the probability values $P_c$ and $P_e$, that are computed at run time. Given the mixed stochastic and workload-dependent nature of the atomistic BTI/RTN model, comparison of Pc and Pe is quite suitable, since their generation is strictly workload-dependent (and has no stochastic component). Furthermore, in order to properly investigate the workload dependency, we inspect the more realistic case of an irregural input signal. A simple Perl script is used to generate a random waveform before the simulation. The `inverter.cir` with the proper syntax of the `.save` statement and the `subcircuits` file are provided below:

**The subcircuits file – The subcircuit of a simple inverter**

```
** Inverter = 2 devices
.subckt inv vdd vss in out
M0 out in vdd vdd p1 W=360n L=90n
M1 out in vss vss n1 W=180n L=90n
.ends
```

**The inverter.cir file – The netlist of a simple inverter**

```
** Subcircuits and Modelcards
.include modelcard
.include subcircuits

** DC Sources
Vvdd vdd 0 1
Vvss vss 0 0
**Input
.include input.in

**Inverter Instantiation
X1 vdd vss input output inv
C1 output 0 2f

**Simulation Definition *
.tran 40ns 40ns

** Output File Definition *
.save tran @m.x1.m0[pc all] @m.x1.m0[pe all] @m.x1.m0[vgs] @m.x1.m0[vth]

.control
set filetype=ascii
.endc
```

Once again, we should provide the proper file that will be used as the initial conditions of the BTI simulation. Note that the pMOS device has four traps. In order to inspect the impact of both the emission and capture events, we define the first two traps to be initially free and the other two to be occupied. The `BTI.out` file in our case is:

**The BTI.out file – The initial conditions of the relaxation transients**

```
_Model_n1_Instance_m.x1.m1_BTI= 0
_Model_p1_Instance_m.x1.m0_BTI= 1
ntraps= 4
defects= 0 0 1 1
teh= 1.000000e-04 1.000000e-03 1.000000e-01 1.000000e+02
tch= 1.000000e-07 1.000000e-04 1.000000e-04 1.000000e+01
tel= 1.000000e-05 1.000000e-09 1.000000e+03 1.000000e-09
tcl= 1.000000e+05 1.000000e+03 1.000000e-03 1.000000e+08
deltavth= 0.010000 0.02000 0.00300 0.03000
```

In the next two pages, we present the simulation results of our implemented framework and of the reference tool [10] (Figures 4.6 - 4.9). A representative set of graphs is provided for each one of the four traps. More specifically, we demonstrate the probability values $P_c$ and $P_e$, that are computed at run time. In addition, the transient $V_{gs}$ signal of the pMOS device is provided.

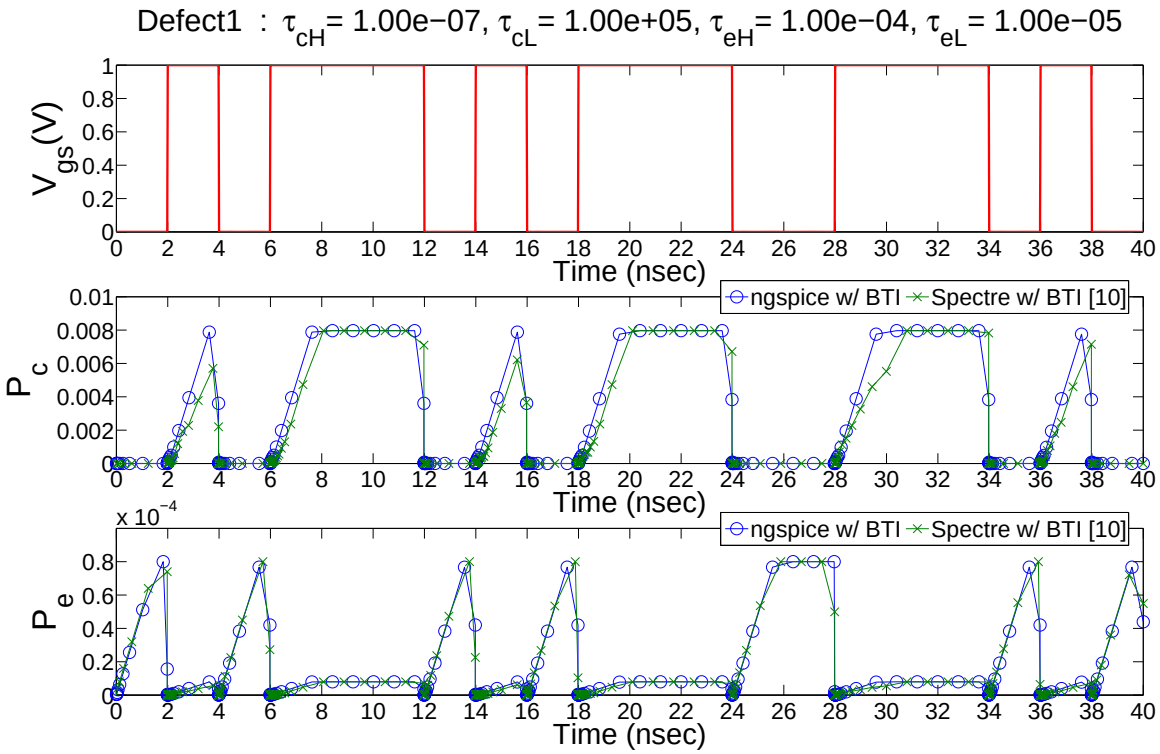


Fig. 4.6: Probabilities of Trap 1. The Runtime $P_c$ and $P_e$ values of our implemented framework and of the reference tool [10] are almost identical.

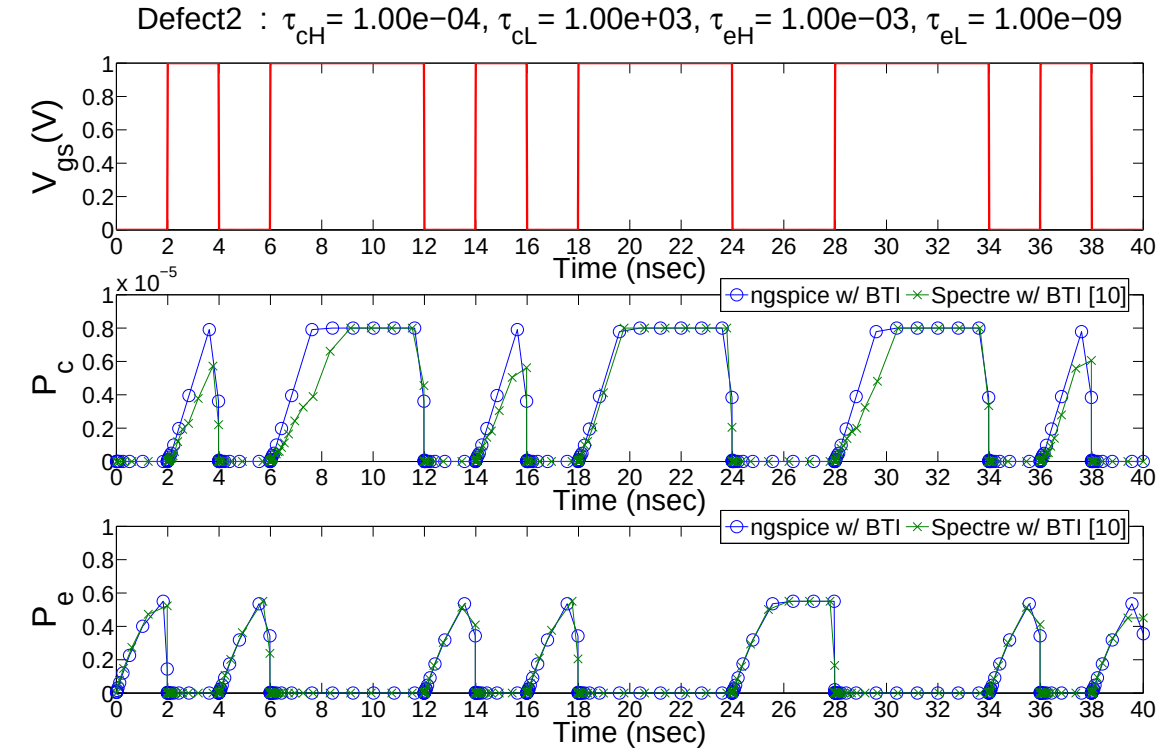


Fig. 4.7: Probabilities of Trap 2. The Runtime $P_c$ and $P_e$ values of our implemented framework and of the reference tool [10] are almost identical.

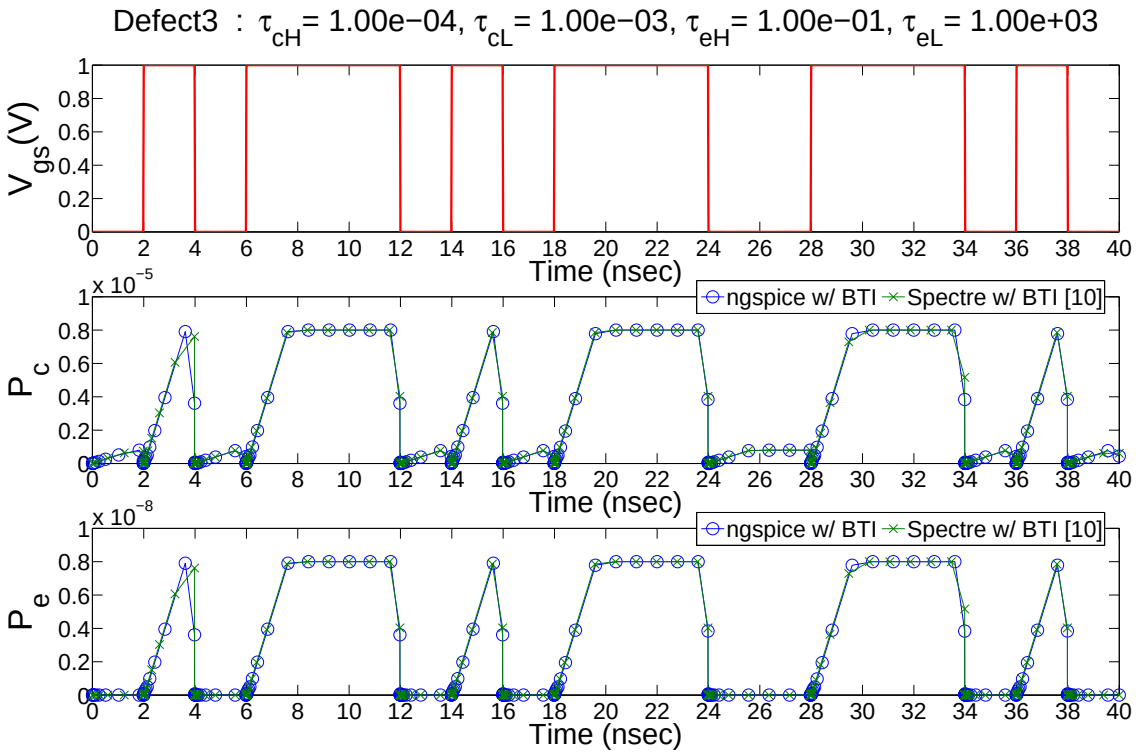


Fig. 4.8: Probabilities of Trap 3. The Runtime $P_c$ and $P_e$ values of our implemented framework and of the reference tool [10] are almost identical.

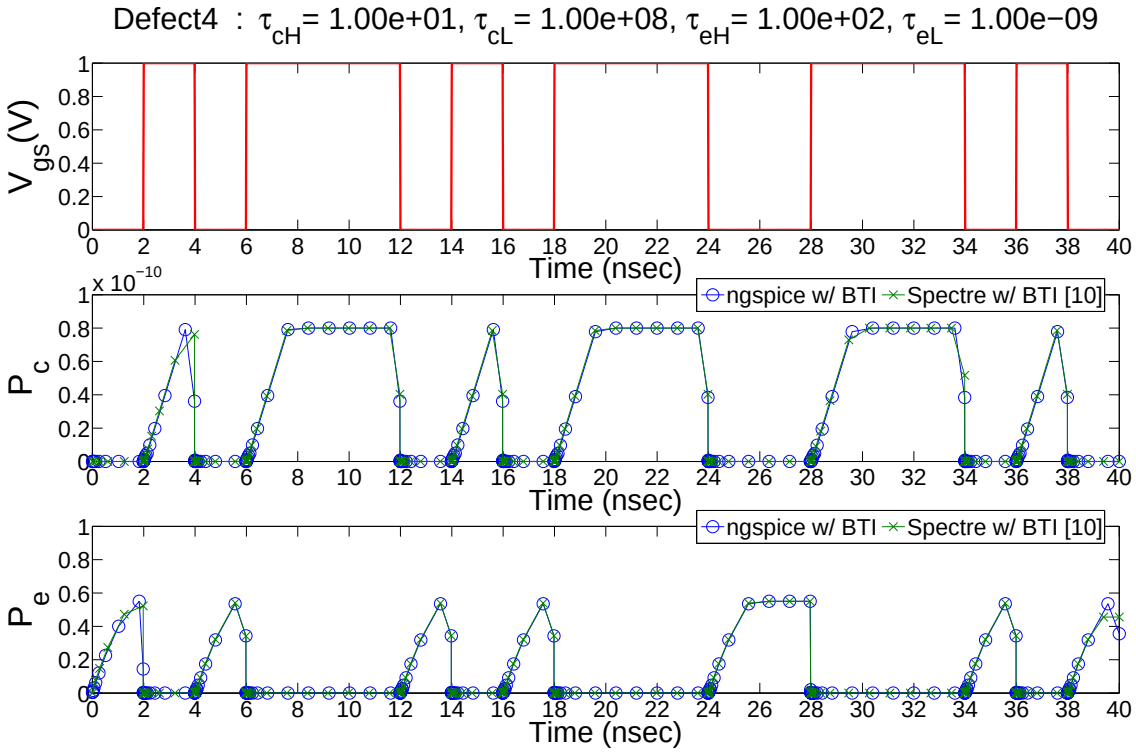


Fig. 4.9: Probabilities of Trap 4. The Runtime $P_c$ and $P_e$ values of our implemented framework and of the reference tool [10] are almost identical.

We can see that the transient output data of our simulation tool (solid blue line of Figures 4.6 - 4.9) are almost identical to the results of the reference tool [10] (solid green line). Consequently, we have an immidate verification of correctness against an established implementation of the atomistic BTI/RTN model [30]. If we further evaluate the results, we can make two major observations.

First of all, the only cause of slight divergence of few output data is the different Size-Step Algorithms of `ngspice` and Cadence® Virtuoso® Spectre®. We shall take into consideration that the probability formula [10] is time-step dependent. For this adaptive transient step to be adjusted at run time, each one of these two simulation tools uses different algorithms in terms of complexity and accuracy. Nevertheless, we can observe that only few cases of value divergence are caused because of such a different selection of transient simulation time-steps.

Moreover, we have another crucial observation related to the validity of our simulation tool. As we have already mentioned, the atomistic BTI model upon which our framework is founded, realistically captures the time-dependent workload dependency [10]. In our case, we can see that the Probability values are $V_{gs}$ dependent. That way, the defects occupancy transitions correspond to the device workload at every simulation step. Thus, we verify that the implemented tool accurately incorporates the workload dependency of degradation due to BTI/RTN.

## 4.4 Validation of BTI impact

Within the two previous Sections, we inspected our implemented simulation flowchart in terms of valid behavior and correctness. To this end, we simulated the appropriate circuit examples, i.e. a single pMOS device and an inverter. In this currenct Section, we will now expand our verification procedure to circuits that are common among integrated CMOS products. The usage of our simulation tool will be demonstrated on the case of a multiplier. That way, we will provide a complete operational example in which the BTI degradation of a representative CMOS circuit will be investigated.

For our test case, a Perl script was used for the creation of the netlist and the input signal files. The used script has as arguments the data block size of the multiplier, the duration of a time slice, the number of these time slices and the duration of slope. In our case, we inspected a multiplier with a data block size of 1 bit. We first performed a simulation with the BTI effect ignored. The delays measurements were obtained by using the `.meas` command of `ngspice` inside the netlist file.

We then performed the simulation of the same circuit, but with the BTI taken into consideration. We started with a BTI simulation without initial conditions. The BTI instance variables were generated according to the provided BTI model parameters. Inside the model card, we added :

```
+trapdensity = 1.0e+11
+stepmean = 0.020
+timeminimum = -12
+timemaximum = +12
```

After the latter BTI simulation, the proper BTI output was generated. We repeated the same simulation by using this file as initial conditions. We took into consideration only these traps that have time constants smaller or equal to $10^{-7}$. For that purpose, another script was used for the output file to be properly parsed and the traps that do not meet this criterion to be omitted. Then, a second simulation was performed. Once again, the delay measurements were obtained with the usage of the `.meas` command.

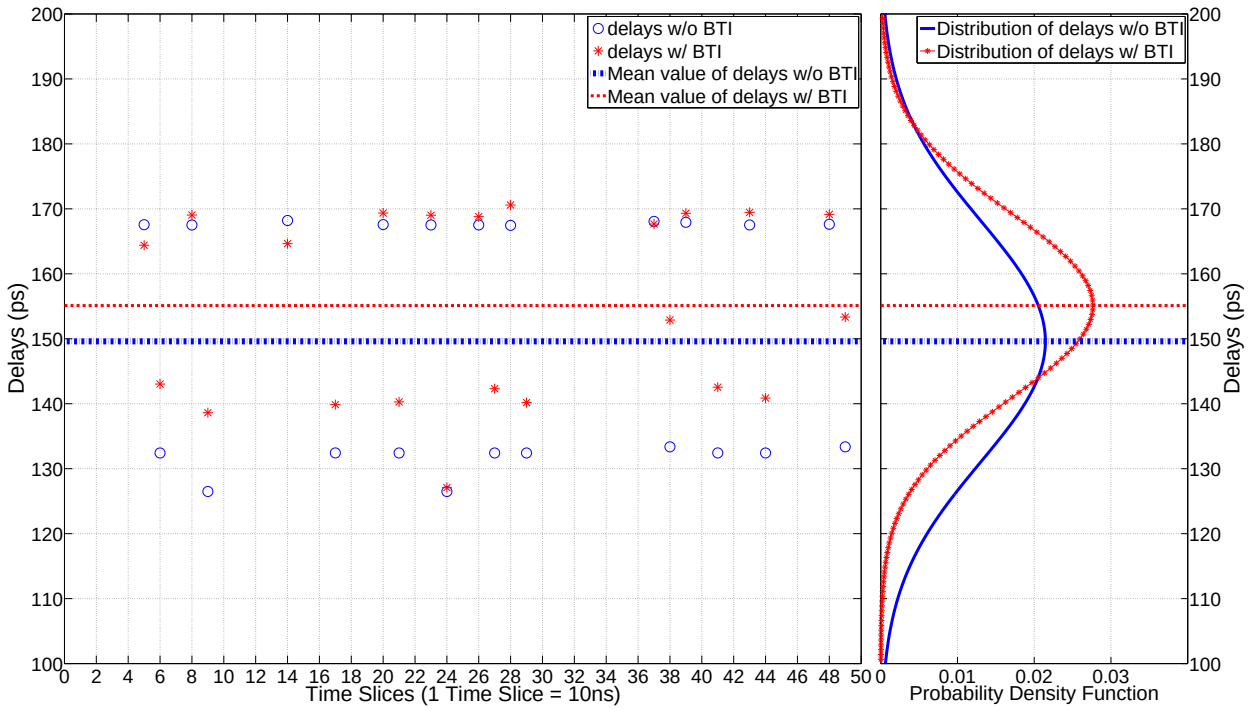


Fig. 4.10: Delay measurements of the multiplier output for $t_{simulation} = 500nsec$ : (a) Results of both simulations with and without BTI degradation and (b) Distributions extracted from these delay measurements per case.

The simulation results are presented in Figure 4.10. Note that the majority of the delays recorded during the BTI simulation (red stars of Fig. 4.10) has greater values than the respective delays without the BTI effect (blue circles). For this observation to be further illustrated, the distributions extracted from the delay measurements of both cases are presented. Thus, we can easily observe that these output data verify the expected delay degradation due to BTI. That way, the correctness of the implemented tool was further validated through an operational example of a representative CMOS circuit.

## 4.5 Conclusion

In this Chapter, we provided representative operational examples in order to investigate the validity of the implemented framework. Our tool was verified, in terms of correctness, by comparison to published results. For each inspected test case, the netlist to be simulated and the BTI initial conditions were provided. More specifically, the phenomenological verification

was achieved by performing relaxation transients of a single device with an individual trap. We then proceeded to a detailed verification of correctness, based on the Pc and Pe values of defects during a transient simulation. The simulation results were compared with the results of another simulation framework, implemented on top of the Cadence® Virtuoso® Spectre® simulator. Finally, the delay degradation due to BTI was verified with simulation of an array multiplier. In all these cases, the acquired output results were consistent with respective reference data.

# Chapter 5
# Conclusions and Future Work

## 5.1 Conclusions

As the device dimensions of modern electronics aggressively decrease, an extensive variability during the device lifetime emerges. Two mechanisms with adverse impact in threshold voltage $V_{th}$ degradation are Bias Temperature Instability (BTI) and Random Telegraph Noise (RTN). Models that realistically capture the impact of BTI/RTN effect are necessary. A representative and novel approach is the atomistic BTI model [1]. This model captures the transient behaviour of individual gate oxide defects and transient simulation approaches has been therefore enabled.

To meet lifetime reliability requirements of modern electronics, designers need simulation tools that will realistically capture BTI/RTN effects during transient simulations. Such accurate yet efficient frameworks will trigger the development of proper mitigation techniques. To contribute to this research process, the current thesis proposes a SPICE-based simulation framework founded upon the atomistic BTI model. The purpose of our work is to port the BTI/RTN model to an open source distribution of the SPICE program, namely `ngspice` [25]. More specifically, the atomistic approach is ported to a state-of-the-art transistor model in the sub100nm regime, the BSIM4.6.0 model [24].

The main differentiator in comparison to related previous works [1][10] is the inline implementation of the BTI model. The novelty of our work is that the atomistic approach is embedded in the actual BSIM4.6.0 source code. That way, the BTI initialization is performed during setup time of the `ngspice` simulator. Thus, no impractical pre-processing stage is required. Furthermore, the transient part of BTI effect is executed inside the run time `ngspice` routines. Consequently, the whole simulation procedure is transparent to

the user as no additional, heterogeneous execution stages are added at the default `ngspice` simulation flow. In addition, the netlist syntax is perfectly retained, which enhances the user-friendly attributes of our approach.

First, we presented the state of the art on the simulation of Bias Temperature Instability (BTI). The two dominants approaches, the Reaction-Diffusion (RD) model and the atomistic model, were reviewed. Then, the landscape of BTI modeling was categorized into several levels of classification. Various simulation approaches were presented and properly classified and any insufficiencies observed in the state of the art were revealed. Thereby, we pointed out that our approach is meaningful and novel for simulation purposes of modern downscaled technologies.

Second, we extensively described the implementation procedure. The selected open source SPICE distribution and its MOSFET transistor models were explored. All the executions steps where the $V_{th}$ value is computed were illustrated with representative figures. That way, we located where the BTI steps should be added. All these code alterations were also described. Then, we demonstrated the usage of the resulted tool with descriptive operational examples. A simple inverter was selected as the circuit under test. Besides the default `ngspice` usage, both the cases of BTI with and without initial conditions were inspected.

Third, we verified our implementation in terms of correctness, by comparison to already published results. Our first step was to demonstrate that our simulation tool realistically captures the physical principles of the atomistic BTI model. For the phenomenological validation, relaxation transients were performed in a single device with an individual trap. Then, detailed validation was provided against another state-of-the-art atomistic BTI model, implemented on top of the Cadence® Virtuoso® Spectre® commercial simulator [10]. In addition, we verified the impact of BTI by investigating the case of an array multiplier.

## 5.2 Recommendations for Future Work

### 5.2.1 Extension of the implemented framework to other Device Models

As described in previous Chapters, the purpose of this thesis is the creation of an accurate yet efficient, SPICE-based simulation framework that will realistically capture BTI/RTN effects during transient simulations. We expect that the resulted simulation tool will facilitate researchers to explore further mitigation techniques and develop designs that meet lifetime reliability requirements. Such an intention underlines the importance of extending the current model to other state-of-the-art device models.

A notable recommendation for future work in the extension of our implemented framework to FinFET devices. According to the latest International Technology Roadmap for Semiconductors (ITRS) [31], FinFETs have emerged as important candidates for device scaling within the following years. Recent announcements that a FinFET transistor will be used in manufacturing at the 16 nm has placed renewed emphasis on 3D device structures research. Among the near-term (2011–2018) difficult challenges presented in the ITRS 2012 update [31], we found 3D interconnect reliability aspects to be listed.

If the respective atomistic BTI model for FinFET devices is available, it will be essential to be ported to the proper device model of `ngspice`. Note that FinFETs are part of a compact model for the class of common multi-gate FETs, namely BSIM-CMG [32]. The points where the code alterations should be added will have to be identified within the BSIM-CMG model. In this case, the implementation flow presented in Subsections 3.2-3.3 will be a valuable implementation guideline.

The aforementioned addition will enable simulations in FinFET devices, while the current user-friendly simulation flowchart will be fully maintained. That way, the presented simulation tool will be further enhanced as a meaningful option for reliability simulations.

### 5.2.2 Transient BTI Simulations of Large Netlists on Multi-Processor Systems

During the classification of research landscape in Section 2.3, we presented various BTI simulation approaches. Different circuits under test were reviewed and simulated per work. We can observe that the majority of these test cases seem to have limited processing and memory overhead.

More specifically, we had cases where the $V_{th}$ shift was pre-evaluated and emulated with a voltage source on the gate terminal of each pMOS transistor. As a result, for a simple combinational circuit, the total simulation overhead by adding these sources were not that significant. On the other hand, even in cases of (i.e. SRAMs), single memory cells where simulated instead of the whole circuitry.

Nevertheless, the aggressive technology scaling evinced the need for reliability simulations not only of simple test cases, but of more sophisticated circuitries. It is, therefore, important for our simulation framework to be extended to even larger netlist sizes. To this end, an efficient way should be found in order to overcome any limitations due to simulation memory requirements. A notable potential for a novel approach is recommended here.

Our SPICE-based tool can be perfectly combined with a powerful simulation framework presented in [33]. In this work, extensive SPICE simulations on multi-processor systems were enabled through a novel approach of workload and node partitioning. A remarkable size of test cases with more than $10^6$ MOSFET devices was achieved. In all the inspected experiments, `ngspice` was used. However, as we pointed out in Section 3.4, the only difference between normal and BTI-aware usage of `ngspice` is whether the `BTI_FILENAME` is set or not. Thereby, if such an option is added on the framework of [33] and our `ngspice` version enhanced with the BTI model is then executed, then transient BTI simulations on multi-processor systems will be instantly enabled.

# References


[1] B. Kaczer, S. Mahato, V. de Almeida Camargo, M. Toledano-Luque, P. Roussel, T. Grasser, F. Catthoor, P. Dobrovolny, P. Zuber, G. Wirth, and G. Groeseneken, "Atomistic approach to variability of bias-temperature instability in circuit simulations," in *Reliability Physics Symposium (IRPS), 2011 IEEE International*, pp. XT.3.1–XT.3.5, 2011.

[2] S. Kumar, C. Kim, and S. Sapatnekar, "An Analytical Model for Negative Bias Temperature Instability," in *IEEE/ACM International Conference on Computer-Aided Design, 2006. ICCAD '06.*, pp. 493–496, 2006.

[3] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "NBTI–Aware Synthesis of Digital Circuits," *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, pp. 370–375,4–8, June 2007.

[4] A. Calimera, E. Macii, and M. Poncino, "NBTI-aware power gating for concurrent leakage and aging optimization," in *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design*, ISLPED '09, (New York, NY, USA), pp. 127–132, ACM, 2009.

[5] A. Calimera, E. Macii, and M. Poncino, "Analysis of NBTI-induced SNM degradation in power-gated SRAM cells," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 785–788, 2010.

[6] R. Vattikonda, W. Wang, and Y. Cao, "Modeling and minimization of PMOS NBTI effect for robust nanometer design," in *Design Automation Conference, 2006 43rd ACM/IEEE*, pp. 1047–1052, 2006.

[7] Seyab and S. Hamdioui, "Temperature Impact on NBTI Modeling in the Framework of Technology Scaling," vol. 0, pp. 1–10, 2010.

[8] "Reliability Simulation in Integrated Circuit Design." White Paper, Cadence Design Systems, Inc., 2003.

[9] H. Kufluoglu, V. Reddy, A. Marshall, J. Krick, T. Ragheb, C. Cirba, A. Krishnan, and C. Chancellor, "An extensive and improved circuit simulation methodology for NBTI recovery," in *Reliability Physics Symposium (IRPS), 2010 IEEE International*, pp. 670–675, 2010.

[10] D. Rodopoulos, S. B. Mahato, V. V. de Almeida Camargo, B. Kaczer, F. Catthoor, S. Cosemans, G. Groeseneken, A. Papanikolaou, and D. Soudris, "Time and Workload Dependent Device Variability in Circuit Simulations," *IC Design Technology (ICICDT), 2011 IEEE International Conference Review of Scientific Instruments*, pp. 1–4, May 2011.

[11] M. Toledano-Luque, B. Kaczer, P. Roussel, T. Grasser, G. Wirth, J. Franco, C. Vrancken, N. Horiguchi, and G. Groeseneken, "Response of a single trap to AC negative Bias Temperature stress," in *Reliability Physics Symposium (IRPS), 2011 IEEE International*, pp. 4A.2.1–4A.2.8, 2011.

[12] K. O. Jeppson and C. M. Svensson, "Negative bias stress of MOS devices at high electric fields and degradation of MNOS devices," *Journal of Applied Physics*, vol. 48, no. 5, pp. 2004–2014, 1977.

[13] B. Paul, K. Kang, H. Kufluoglu, M. Alam, and K. Roy, "Temporal Performance Degradation under NBTI: Estimation and Design for Improved Reliability of Nanoscale Circuits," in *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings*, vol. 1, pp. 1–6, 2006.

[14] Muhammad A. Alam (2005), "On the Reliability of Micro-Electronic Devices: An Introductory Lecture on Negative Bias Temperature Instability." `https://nanohub.org/resources/193`.

[15] "HSPICE® Simulation and Analysis User Guide." Synopsys, Inc., `http://www.rudraj.it/hspice_sa.pdf`.

[16] S. Mahapatra, P. Kumar, and M. Alam, "Investigation and Modeling of Interface and Bulk Trap Generation During Negative Bias Temperature Instability of p-MOSFETs," *IEEE Transactions on Electron Devices*, vol. 51, no. 9, pp. 1371–1379, 2004.

[17] M. Alam and S. Mahapatra, "A Comprehensive Model of PMOS NBTI Degradation," *Microelectronics Reliability*, vol. 45, pp. 71 – 81, 2005.

[18] A. Ricketts, J. Singh, K. Ramakrishnan, N. Vijaykrishnan, and D. Pradhan, "Investigating the impact of NBTI on different power saving cache strategies," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, pp. 592–597, 2010.

[19] Seyab and S. Hamdioui, "NBTI modeling in the framework of temperature variation," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, pp. 283–286, 2010.

[20] M. Alam, H. Kufluoglu, D. Varghese, and S. Mahapatra, "A Comprehensive Model for PMOS NBTI degradation: Recent progress," *Microelectronics Reliability*, vol. 47, pp. 853–862, 2007.

[21] H. Kufluoglu and M. Alam, "A Generalized Reaction Diffusion Model With Explicit H–$H_2$ Dynamics for Negative-Bias Temperature-Instability (NBTI) Degradation," *IEEE Transactions on Electron Devices*, vol. 54, pp. 1101–1107, May 2007.

[22] E. Maricau and G. Gielen, "Efficient Variability-Aware NBTI and Hot Carrier Circuit Reliability Analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 12, pp. 1884–1893, 2010.

[23] P. P. Weckx, B. Kaczer, M. Toledano-Luque, T. Grasser, P. Roussel, H. Kukner, F. Raghavan, Catthoor, and G. Groeseneken, "Defect-based Methodology for Workload-dependent Circuit Lifetime Projections – Application to SRAM," in *Reliability Physics Symposium (IRPS), 2013 IEEE International*, pp. 3A.4.1–3A.4.7, 2013.

[24] Dunga, M.V. et al., *BSIM4.6.0 MOSFET Model – User's Manual.* 2006.

[25] P. Nenzi and H. Vogt, *Ngspice Users Manual.* January 2013.

[26] "Ngspice simulator home page." http://ngspice.sourceforge.net/.

[27] "BSIM-CMG model." http://www-device.eecs.berkeley.edu/bsim/.

[28] T. L. Quarles, *The Front End to Simulator Interface*. April 1989.

[29] T. L. Quarles, *Adding Devices to SPICE3*. April 1989.

[30] F. CATTHOOR, B. KACZER, D. RODOPOULOS, V. VALDUGA DE ALMEIDA CAMARGO, and S. BANDHU MAHATO, "Time and workload dependent circuit simulation ," Patent EP2509011 (A1), IMEC [BE], 2012.

[31] "International Technology Roadmap for Semiconductors." http://public.itrs.net/.

[32] Sriramkumar, V. et al. , *BSIM-CMG 106.1.0, Multi-Gate MOSFET Compact Model – Technical Manual*. 2012.

[33] G. Lyras, D. Rodopoulos, A. Papanikolaou, and D. Soudris, "Hypervised transient SPICE simulations of large netlists & workloads on multi-processor systems," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, pp. 655–658, 2013.