RELIABILITY-AWARE AND VARIATION-AWARE CAD TECHNIQUES

A PHD DISSERTATION THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

SANJAY V. KUMAR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY IN ELECTRICAL ENGINEERING

Sachin S. Sapatnekar, Advisor
NOVEMBER 2009

Abstract

Technology scaling into the sub-100nm domain implies that the effects of process, voltage, and temperature variations have a resounding effect on the performance of digital circuits. An increase in complexity has resulted in challenges in design and manufacturing of these circuits, as well as guaranteeing their accurate and reliable performance. Two key challenges in present-day circuit design are to ensure the long term reliability of circuits and to accurately estimate the arrival times and margins of the various paths in the circuit during timing analysis, under the presence of variations, at all operating conditions. Bias Temperature Instability (BTI), a long-term transistor degradation mechanism has escalated into a growing threat for circuit reliability; hence its exact modeling and estimation of its effects on circuit performance degradation have become imperative. Techniques to mitigate BTI and ensure that the circuits are robust over their lifetime are also becoming vital. Similarly, the impact of process variations has prompted new areas of research, to determine the timing and power estimates of the circuit, as accurately as possible. Since the effects of variations can no longer be ignored in high-performance microprocessor design, sensitivity of timing slacks to parameter variations has become a highly desirable feature to allow designers to quantify the robustness of the circuit at any design point. Further, the effect of varying on-chip temperatures, particularly in low voltage operation, has led to inverted temperature dependence (ITD) in which the circuit delay may actually decrease with increase in temperature. This thesis addresses some of these major issues in present day design.

We propose a model to estimate the long term effects of Negative Bias Temperature Instability (NBTI). We initially present a simple model based on an infinitely thick gate-oxide in transistors, to compute the asymptotic threshold voltage of a PMOS transistor, after several cycles of stress and recovery. We then augment the model to handle the effects of finite-oxide thickness, and justify the findings of several other experimental observations, particularly during the recovery phase of NBTI action. Our model is robust and can be efficiently used in a circuit analysis setup to determine the long-term asymptotic temporal degradation of a PMOS device, over several years of operation.

In the next chapter, we use this model to quantify the effect of NBTI, as well as Positive Bias

Temperature Instability (PBTI), a dual degradation mechanism in PMOS devices, on the temporal degradation of a digital logic circuit. We provide a technique for gaging the impact of signal probability on the delay degradation numbers, and investigate the problem of determining the maximal temporal degradation of a circuit under all operating conditions. In this regard, we conclude that the amount of overestimation using a simple pessimistic worst-case model is not significantly large. We also propose a method to handle the effect of correlations, in order to obtain a more accurate estimation of the impact of aging on circuit delay degradation.

The latter part of this chapter proposes several circuit optimization techniques that can be used to ensure reliable operation of circuits, despite temporal degradation caused by BTI. We first present a procedure of combating the effects of NBTI during technology mapping in synthesis, thereby guardbanding our circuits with a minimal area and power overhead. An adaptive compensation scheme to overcome the effects of BTI through the use of adaptive body bias (ABB) over the lifetime of the circuit is explored. A combination of adaptive compensation and BTI-aware technology mapping is then used to optimally design circuits that meet the highest target frequency over their entire lifetime, and with a minimal overhead in area, average active power, and peak leakage power consumption. Lastly, we present a simple cell-flipping technique that can mitigate the impact of NBTI on the static noise margin (SNM) of SRAM cells.

In the final chapter of this thesis, we first present a framework for block based timing sensitivity analysis, where the parameters are specified as ranges - rather than statistical distributions which are hard to know in practice. The approach is validated on circuit blocks extracted from a commercial 45nm microprocessor design. While the above approach considers process and voltage variations, we also show that temperature variations, particularly in the low voltage design space can cause an anomalous behavior, i.e., the circuit delays can decrease with temperature. Thus, it is now necessary to estimate the maximum delay of the circuit, which can occur at any operating temperature, and not necessarily at a worst case corner setting. Accordingly we propose a means to efficiently compute the maximum delay of the circuit, under all operating conditions, and compare its performance with an existing pessimistic worst-case approach.

Acknowledgment

"*At times our own light goes out and is rekindled by a spark from another person.*
Each of us has cause to think with deep gratitude of those who have lighted the flame within us."

Utmost thanks are due to my advisor Prof. Sachin S. Sapatnekar. Any effort to sum in words of his mentorship, cooperation and rationale is scarcely incomplete. The problem solving skills, and the attention to detail that I have tried to acquire, as part of his tutelage, have already proved to be extremely useful in my short work-experience at Synopsys. This thesis is an amalgamation of his vast acumen, intellect, and dedication, as well as his encouragement that boosted my efforts all along. The last five years have been truly outstanding, and it is the greatest privilege to have had this association with him.

Much water has flown down the river since I first began to pen this thesis, and many a things have changed in my own life, often causing me to balance other things, amongst quality completion of the dissertation writing. Yet, during these last nine months, his cooperation has been beyond what any graduate student can hope to expect from an advisor.

Thanks are due to Prof. Chris H. Kim for several technical discussions, and most importantly for introducing me to the domain of Negative Bias Temperature Instability (NBTI) in 2005. I would also like to acknowledge Prof. Marc Riedel and Prof. Eugene Shragowitz for reviewing my thesis and giving valuable feedback. I am also grateful to my undergraduate faculty members, particularly Dr. Anu Gupta for the foundation that I received in VLSI during my formative years at BITS-Pilani.

I would like to thank the members of Mobility Platform Group, Intel Design Center, Folsom, for a wonderful learning experience as a graduate technical intern during 2005. I would also like to extend my gratitude to members of Strategic CAD Labs at Intel Hillsboro, for my second innings at Intel, as an intern during 2007. Chandramouli V. Kashyap, my mentor during the internship provided me with invaluable guidance during my stint at Intel and beyond.

A great "amount" of thanks are due to the University of Minnesota for awarding me the graduate school and the doctoral dissertation fellowships, as well as to the Semiconductor Research Corporation (SRC) for their continued support. Thanks are also due to colleagues here at the Department of Electrical Engineering for their help and friendship. Nitin and Kapil in particular deserve special mention; the former is disavowed for his amazing ability to convince me to do a PhD along with,

only to swap places soon at Intel, Folsom, while the latter for his inspiring enthusiasm, wit, wisdom, and attitude, and for introducing me to the art of technical writing using Latex.

Thanks are due to Shankar Radhakrishnan, my manager at Synopsys, for allowing me to start work at the right juncture, and for giving me the right amount of push to ensure completion of this dissertation work.

As much as I wish to acknowledge the support of my parents and Siri (especially for considering my resume), perhaps it is better done in person than in mere words.

I can no other answers make, but thanks, and thanks.

- William Shakespeare

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

Chapter 1

Introduction

With technology scaling into the nanometer regime, the impact of process, voltage, temperature, and other environmental variations on the performance of digital integrated circuits has become significant. Further, the effects of scaling, and increased on-chip operating temperature have also affected the long term reliability and the lifetime of these circuits. The accurate modeling of these variations, and accounting for their effects while estimating circuit performance, has become one of the most important challenges in the field of computer aided design.

1.1 Reliability of Digital Circuits

Recent trends in technology scaling into the sub-130nm technology have escalated the impact of Negative Bias Temperature Instability (NBTI), a transistor degradation mechanism, on the long term reliability of circuits. When a PMOS device is biased in inversion, the application of a negative bias at the gate node, causes the generation of interface traps. Interface trap sites are formed due to crystal mismatches at the Si-SiO_2 interface. During oxidation of Si, most of the tetrahedral Si atoms bond to oxygen. However, some of the atoms bond with hydrogen, leading to the formation of weak Si-H bonds. Fig. 1.1(i) shows the 3-D structure of Si at the Si-SiO_2 interface in the 111 crystal orientation. N_{it} is the site containing an unsaturated electron (crystal mismatch) leading to the formation of an interface trap. Fig. 1.1(ii) shows the Si-SiO_2 interface in 2-D along with the Si-H bonds and the interface traps.

When a PMOS transistor is biased in inversion, the holes in the channel dissociate these Si-H bonds, thereby generating interface traps (Fig. 1.1(iii)). Interface traps (interface states) are electrically active physical defects with their energy distributed between the valence and the conduction band in the Si band diagram [7]. The rate of generation of traps is accelerated by high operating temperatures. These traps cause an increase in the threshold voltage ($|V_{th}|$) of the PMOS transistor, over a large period of time. This effect is known as Negative Bias Temperature Instability (NBTI).

The impact of NBTI has notably been measured on SRAM logic cells [8], and digital circuits in [9]. The results show that in digital logic, NBTI can cause a reduction in the drain current, thereby

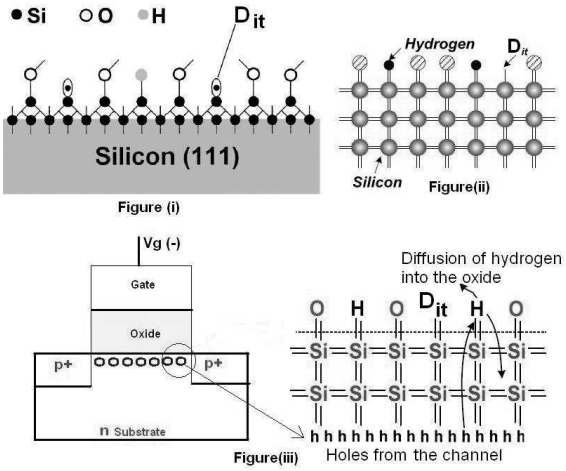


Figure 1.1: Schematic description showing the generation of interface traps when a PMOS transistor is biased in inversion. Fig. (i) shows the 3-D structure of Si at the Si-SiO_2 interface in the 111 crystal orientation. N_{it} is the site containing an unsaturated electron (crystal mismatch) leading to the formation of an interface trap. Fig. (ii) shows the Si-SiO_2 interface in 2-D along with the Si-H bonds and the interface traps. Fig. (iii) shows the dissociation of Si-H bonds by the holes when the PMOS device is biased in inversion and the diffusion of hydrogen into the oxide, thereby generating an interface trap [7].

increasing the delay of the circuits over a few years of operation. This may also cause the circuit to violate its timing specifications, thereby resulting in functional failure. In SRAM logic, NBTI can cause a decrease in the static noise margin (SNM), thereby affecting the read stability. Thus, not only is it essential to quantify the impact of NBTI on transistor performance, but it is also important to mitigate this effect through optimal circuit design techniques, or to make them robust enough to ensure reliable performance over the guaranteed lifetime.

With the use of thinner gate-oxides, the impact of NBTI has increased. To further exacerbate the issue of long-term reliability of circuits, the use of Hf-based high-k dielectrics for gate-leakage reduction has caused additional degradation in NMOS devices by a dual mechanism known as Positive Bias Temperature Instability (PBTI) [10, 11]. PBTI occurs in NMOS devices when a positive bias stress is applied across the gate oxide, i.e., when ($V_{gs} = V_{dd}$). Moreover, techniques developed to reduce NBTI can contribute to increasing PBTI. For example, if the input to an inverter is biased so that it is more likely to be at logic 1 than logic 0, the NBTI stress on the PMOS device is reduced since the V_{gs} bias becomes zero; however, this now places a bias on the NMOS device, which now

has a positive V_{gs} value.

Thus, under transistor degradation mechanisms such as BTI (bias temperature instability), it is necessary to quantify the long-term degradation of digital circuits. Circuits can then be guardbanded by adding appropriate timing margins or may be designed robustly such that they are tolerant to temporal degradation, thereby guaranteeing optimal performance over their lifetime.

1.2 On-Chip Variations

A major challenge in present day circuit design, particularly in the context of device modeling, timing and power analysis, is the impact of variations. Imperfections in the manufacturing process as well as dynamic changes in the operating conditions can cause the various parameters that influence circuit performance, to be perturbed from their expected values, (i.e., the values that were used during presilicon circuit design and estimation). This has led to a significant deviation between the delay (and leakage power) numbers computed by a tool through simulations, as opposed to their actual numbers as seen on silicon. The impact of these variations have assumed vast proportions, due to technology scaling, and decreasing feature sizes, making it extremely hard to predict the exact postsilicon delays of the circuit, with the highest degree of accuracy and confidence. This has opened up newer challenges in timing analysis.

The sources of these variations can broadly be categorized as [12]:

1. Process variations: These arise due to imperfections in the fabrication process, and manifest themselves as a deviation in the values of parameters such as channel length L_e, device width W, oxide thickness d_{ox}, dopant concentration N_a, interconnect thickness and width variations, etc. These variations can be perceived to be "one-time" in nature, causing a change in the circuit delay and power values during fabrication, and mostly remain invariant over the entire lifetime of the circuit. These variations can be further classified into intra-die variations or inter-die variations, depending on whether they have impacted all devices on the chip uniformly or whether their impact is local to a specific region on the die.

2. Environmental variations: These arise due to changes in the operating conditions of the chip, such as changes in supply voltage or temperature. Unlike process variations, that are "one-

time", these are "run-time" in nature, and impact the dynamic performance of the circuit.

The accurate modeling of these variations during timing analysis is essential to determine their impact on the performance of the circuit.

1.3 Outline of the Thesis

The content of this dissertation can be classified into three broad chapters:

1.3.1 Negative Bias Temperature Instability Modeling

In order to quantify the impact of NBTI on the temporal delay of digital circuits, it is vital to develop a transistor degradation model that can capture the shift in the various parameters that affect the circuit delay. Much work in the area of NBTI has been active within the communities of device and reliability physics. However, with its increasing impact, a CAD framework for managing the NBTI degradation at the circuit level is essential.

In this regard, Alam [13] presented an analytical model based on a Reaction-Diffusion (R-D) [14, 15] framework to determine the number of interface traps and the threshold voltage degradation as a function of time. The work in [13] shows that an alternating stress-relaxation pattern (AC stress) produces lower degradation as compared with a DC stress input, thereby implying that the long term temporal degradation depends on the nature of stress applied to the PMOS device. The authors in [8, 9] present a numerical simulation-based model using the R-D framework, and determine the change in the frequency of an NBTI stressed ring oscillator. However, an accurate estimation of the effects of NBTI on the delay of a combinational circuit requires a comprehensive model that can determine the long term asymptotic degradation of a transistor after several years of operation under different conditions.

As part of this thesis, in Chapter 2, we develop the first analytical model for simulating NBTI over multiple cycles of stress and relaxation. Our work uses the classical Reaction-Diffusion (R-D) model to analytically capture the physical effects of both the stress and relaxation phases of NBTI action. It must be noted that although the R-D paradigm has been used to develop models in [13, 16, 17] that physically explain the effects of NBTI, all of these models have been restricted to a single

stress cycle, or a single stress cycle followed by a single relaxation cycle. The subsequent phases of stress and relaxation are handled in [13] by numerically solving the R-D model equations. In contrast, our work builds a comprehensive analytical model, over *multiple* stress/relaxation phases, and also leads to a concrete proof of the frequency independence property of NBTI.

While the R-D theory [14, 15] has commonly been used to model NBTI, leading to various long-term models for circuit degradation [13, 18–20], alternative views among researchers exist, particularly about the inability of the R-D model to explain some key phenomena, as detailed in [5, 21–24]. This has led to models such as [21, 25–29], as well as efforts to resolve the controversy between the R-D model theory and the hole trapping theory [30–33]. While this area is still under active research, the domain of our work is restricted to NBTI modeling based on the R-D theory.

This thesis compares these existing models for predicting the long term effects of aging on circuit reliability within the R-D framework [13, 18–20], and finds that these models do not successfully explain the experimentally observed results. In this regard, we first sketch an outline for the basic requirements of any NBTI model, based on observations from a wide realm of experimental data. Further, most of these models assume that the oxide thickness (d_{ox}) is infinite, which is practically not valid in sub-65nm technologies, where d_{ox} is of the order of a nanometer. Hence a model must consider the effect of interface trap generation and recombination in polysilicon. Numerical simulations are also performed to illustrate the drawbacks of existing models based on the R-D theory, and to highlight the importance of considering the effect of finite oxide thickness.

Accordingly, we propose an R-D based analytical model for NBTI that does not consider the oxide to be infinitely thick. The model provides an expression for asymptotically computing the NBTI-induced threshold voltage degradation of a transistor as a function of time, for any arbitrary stress and relaxation pattern. The results show that this model can resolve several inconsistencies, noted with the reaction-diffusion theory for NBTI generation and recombination, as observed in [5, 21–23]. Further, we can also explain the widely distinct experimentally observed results in [5, 34, 35].

1.3.2 BTI Aware Analysis and Optimization of Circuits

NBTI manifests itself as an increase in V_{th} of the devices, which causes the device drain currents to decrease, thereby increasing the gate delays. While Chapter 2 presents a model that captures the V_{th} of a transistor as a function of time of operation, at the circuit level, it is necessary to build a mechanism to quantize the effect of "aging" on the delay of the various critical paths in the circuit. In this regard, in Chapter 3, we first develop a method to use the NBTI model to compute the asymptotic shift in the V_{th} of every transistor in the circuit after several years of operation, at the end of its lifetime. Chapter 3 then presents a method to determine the impact of NBTI, as well as PBTI, on the performance of digital circuits. We consider the dependence of the interface trap generation on the signal probabilities of the various nodes in the circuit. We also discuss the problem of determining an accurate timing guardband over the delay of the circuit under all operating conditions, based on a previously proposed "worst-case" method [36, 37]. Our results indicate that the "worst-case" method does indeed provide a reasonably accurate means of estimating the impact of BTI on the shift in timing numbers, i.e., the pessimism incurred by this method is within 5% of the nominal delays of the circuits.

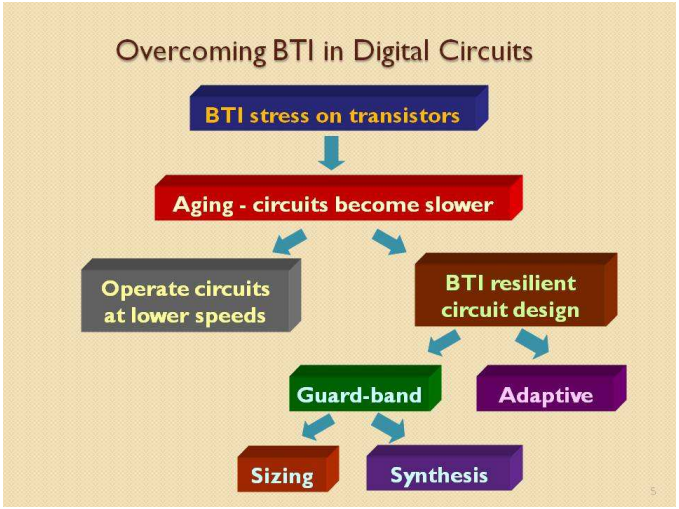


Figure 1.2: BTI resilient circuit design techniques [1].

While it is imperative to quantize the aging in circuits under the impact of BTI, it is also essential to overcome this effect, thereby guaranteeing reliable operation and the highest performance

of the circuit, over its entire lifetime. In this regard, Fig. 1.2 [1] shows the two classes of BTI resilient circuit design solutions, as applied to digital combinational logic. The work in [36, 38] presents a sizing-based solution to determine the optimal sizes of the transistors (or the gates) that can guarantee reliable operation over the lifetime of the circuit. Further, the work uses a model for NBTI that ignores the recovery in threshold voltage when the negative bias stress applied is relaxed. In Chapter 3, we alternatively propose a synthesis-based technique using our transistor threshold voltage degradation model derived in Chapter 2. We demonstrate that incorporating the effects of NBTI into technology mapping, as well as using a model for NBTI that considers the recovery effect produces a superior solution, as compared with sizing.

While sizing and synthesis both offer design-time solutions to BTI, where the circuit is suitably redesigned to account for temporal degradation over its lifetime, these techniques also result in a higher area and power overhead over a nominally designed circuit. Further while BTI causes the threshold voltage of transistors to increase, it also reduces the subthreshold leakage with time, thereby providing opportunities for power-performance trade-offs. Accordingly, we investigate the effectiveness of an adaptive technique to counter aging in digital circuits. We propose a guard-banding technique using a combination of adaptive body bias (ABB) and adaptive supply voltage (ASV), as well as BTI-aware synthesis, to recover the performance of an aged circuit, and compare its merits over previous approaches.

Lastly, Chapter 3 also presents an overview of the impact of NBTI on SRAM cells. We note that NBTI significantly affects the static noise margin (SNM), which is a measure of its read stability. We use the analytical model derived in Chapter 2 to quantize the threshold voltage degradation on the PMOS devices, and thereby the SNM degradation of the SRAM cell.

While the application of a continuous negative bias to the gate of the PMOS transistor degrades its temporal performance, the removal of the bias helps anneal some of the interface traps generated, leading to a partial recovery of the threshold voltage. We utilize this phenomenon to develop a novel solution to overcome the effect of NBTI by flipping the contents of the SRAM cell periodically. This ensures that the PMOS devices are subjected to periods of alternate stress and relaxation (as opposed to continuous stress) allowing dynamic recovery of threshold voltage. We present results obtained through simulations based on the R-D model, which indicate that about 30% of read

stability (measured in terms of SNM) can be restored through cell flipping. Hardware and software implementations for this methodology are also discussed.

The contents of this dissertation have been published/submitted for review and publication in [18, 39–46].

1.3.3 Timing Analysis under Process, Voltage, and Thermal Variations

Static timing analysis plays a prominent role in ensuring a fast and accurate estimate of the timing of the circuit, and has become an indispensable feature in circuit design over the last few decades. Recent trends in technology scaling such as process, voltage, and temperature (PVT) variations have led to new areas of research in this domain, in order to guarantee optimal and reliable performance of digital circuits. In the final chapter of the thesis, we focus on two different approaches, one of which is geared at die-to-die [12] process and voltage variations, while the other is targeted toward temperature variations.

During circuit design, variability in manufacturing and operating conditions must be accounted for during timing analysis, to verify the timing of a circuit under all operating conditions, before committing it to manufacturing. Two existing classes of timing analysis techniques to handle the effect of variations include statistical static timing analysis (SSTA) and multicorner static timing analysis. SSTA models parameters as random variables and is based on the assumption that the distributions and correlations of the varying parameters are known *a priori* [12, 47, 48], etc. On the other hand, multicorner timing analysis models these parameters as uncertain variables and attempts to verify the timing at a carefully chosen set of "corner" settings.

While linear time techniques using linear delay models and Gaussian distributions for process parameters were proposed to perform a statistical timing analysis, with technology scaling and the increasing complexity of design and manufacturing, the effect of variations on delays has increasingly become nonlinear. Although several attempts have been made to address these issues, SSTA works on the fundamental assumption that the knowledge of the distributions and correlations on the varying parameters is available. However, this information is extremely hard to be obtained in practice; what is known and easy to obtain is a set of bounds or ranges of the varying parameters.

Multicorner static timing analysis on the other hand does not require the knowledge of the

actual distributions of the varying parameters. A subset of "corners", each representing a setting of the varying parameters, is carefully chosen, and timing analysis is performed at these corners to verify that the circuit meets the performance constraints. However, the number of corners required to guarantee timing closure under all operating conditions increases exponentially with the number of varying parameters. Further, such corner-based techniques lack information about the sensitivity of the circuit delays to small perturbations in the process parameters, which is of greatest utility in enabling the designers to fix the right set of paths.

Thus, noting the limitations of existing classes of timing analysis methodologies, we propose a **parameterized timing analysis** framework as part of this thesis, in Chapter 4. Three key features of parameterized timing analysis are:

- No assumptions on the distributions of the varying parameters are made. In other words, we only require bounds on the varying parameters.
- Inaccuracies due to approximations in the $\max$ computation, as is the case with SSTA, are eliminated by developing a framework that preserves the correctness of the $\max$ computation.
- The framework enables an **exact** computation of the arrival times and slacks at every setting of the varying parameters, thereby implying that multicorner timing analysis is a subset of this framework.

The framework is verified on commercial 45nm (Intel) microprocessor blocks, and has proven to be a viable alternative to current heuristic-based techniques in quantifying the robustness of the design at any design point, and in prioritizing on the right set of paths to fix.

While the effects of process and voltage variations have previously been addressed using statistical timing analysis, and as part of our thesis using a novel parameterized timing analysis framework, temperature variations pose yet another challenge to achieving timing closure. In particular, timing analysis typically assumes that the cell delay decreases with increasing temperature. However, this assumption breaks down at lower voltage operation due to the opposite effects of the mobility of carriers and threshold voltage of transistors on the cell delays, often causing the delays to decrease with temperature. This phenomenon is known as inverted temperature dependence (ITD) [49], or positive temperature dependence [50, 51]. A major consequence of ITD is that it makes it nontrivial

to determine the temperature that results in the delays being maximized (or minimized). Our proposed parameterized timing analysis framework, as described above, assumed linear variations of the delay with temperature. Therefore, it cannot model temperature variations due to the fact that the delay can now vary nonmonotonically with temperature. Corner-based techniques must accordingly be modified to consider the fact that the temperature that maximizes the delay of the circuit can occur anywhere within the bounds given by $[T_{\min}, T_{\max}]$. Accordingly, Chapter 4 of this thesis also presents a framework to handle mixed temperature dependence in timing analysis. The delay variation of a library gate-timing arc as a function of temperature is modeled, using enumerated and quadratic models. Temperatures of blocks on a chip are then determined subject to spatial, thermal, and power constraints, such that the delays of the critical paths are maximized. This approach is then compared with a pessimistic "worst-case" approach that fails to account for such constraints, and merely determines the maximal delay of each gate on the critical path, across all temperatures.

Chapter 2

NBTI Modeling

In this chapter, we present the detailed analysis of an analytical model for Negative Bias Temperature Instability (NBTI), and also determine the impact of signal probabilities (SP) of the various nodes in the circuit on its delay degradation numbers. Section 2.1 provides a detailed description of the Reaction-Diffusion (R-D) model that explains the NBTI action. The R-D model is used to derive a multicycle model that can estimate the number of interface traps as a function of the time of stress (and recovery). A simplified expression to determine the threshold voltage degradation of a PMOS transistor is also developed. Frequency independence of interface trap generation is also proved analytically for the first time, in Section 2.2.

While the above work provides a simple means to quantify the impact of NBTI on digital circuit degradation, the limitations of the multicycle model in Section 2.1 are detailed in Section 2.4. In this regard, we first sketch an outline for the basic requirements of any NBTI model in Section 2.4.2, based on observations from a wide realm of experimental data. Further, most of these models assume that the oxide thickness (d_{ox}) is infinite, which is particularly not valid in sub-65nm technologies, where d_{ox} is of the order of a nanometer.

Accordingly, we propose an R-D based model for NBTI that does not consider the oxide to be infinitely thick. Section 2.5 describes the modifications to the R-D model equations for the infinite d_{ox} case in Section 2.1.1, and presents a solution to the first stress phase, or the DC stress case of NBTI action. In Section 2.6, we outline a numerical simulation framework for the first stress and recovery phases, thereby showing the origin for some of the key drawbacks of the R-D based model in [13], as well as highlighting the role of finite oxide thickness in long term recovery. Section 2.7 then provides a detailed derivation of the model for the first recovery phase. Simulation results and comparison with experimental data are shown in Section 2.8. We use the stress and recovery models derived for a single stress and relaxation phase, and extend this to a multicycle framework in Section 2.9.

The results show that the model can resolve several inconsistencies, noted with the reaction-diffusion theory for NBTI generation and recombination, as observed in [5, 21–23]. Further, the model can also explain the widely distinct experimentally observed results in [5, 34, 35]. It must be

noted that although a two-region model has been proposed in [19, 20], our work not only improves upon the drawbacks in [20], but also provides a satisfying explanation for using such a model. Besides the actual analytical modeling and the framework for estimating the degradation of digital circuits, our contribution also involves providing a better understanding of the empirical constant ξ, as used in [13], and has been misinterpreted as being universal.

2.1 An Analytical Multicycle Model for NBTI

The Reaction-Diffusion model was first used in [14] to physically explain the mechanism of negative bias stress (NBS) in p-channel MOS memory transistors, based on the activation energy of electrochemical reactions. Several years later, a detailed mathematical solution to the R-D model was presented by [15], considering an infinite oxide thickness case, as well as a finite oxide thickness case, in which polysilicon was assumed to be a perfect absorber. Subsequently, [13, 16, 17, 52] have used the R-D model to describe the NBTI effect in present day PMOS devices. The analytical model for NBTI in [13] by Alam, provides a simple means to estimate the number of interface traps for a single stress phase, followed by a relaxation phase only, under the assumption of infinite oxide thickness. A numerical simulation based model is used to extend the results to a multicycle simulation.

However, this work presents an accurate analytical model that is simple and can allow the simulation of NBTI effects in a computationally efficient manner. Further, our model captures the recently observed shift in time dependence of NBTI from $t^{\frac{1}{4}}$ to $t^{\frac{1}{6}}$ [52]. Unlike [37], we consider the fact that the threshold voltage can recover back partially during annealing, and incorporate these into the calculations. We also mathematically prove (for the first time), the observed phenomenon of frequency independence for periodic square waveforms [2, 13, 52–55] and use this to efficiently compute the total amount of degradation of threshold voltage for a PMOS transistor based on the amount of time it is stressed and relaxed.

In this section, we describe the framework of the Reaction-Diffusion (R-D) model, used to develop an analytical model for NBTI action. The R-D model is solved assuming that alternate periods of stress and relaxation, each of equal duration τ, are applied to the gate of a PMOS device, whose source and bulk are tied to V_{dd}, as shown in Fig. 2.1. It must be noted that the derivation

is valid, with minor changes in the limits of integration, for any arbitrary sequence of stress and relaxation. However, since the special case of a square wave-like sequence of "alternating" stress and relaxation (also called AC stress in the NBTI literature) is frequently used in experimentation, we consider this case.

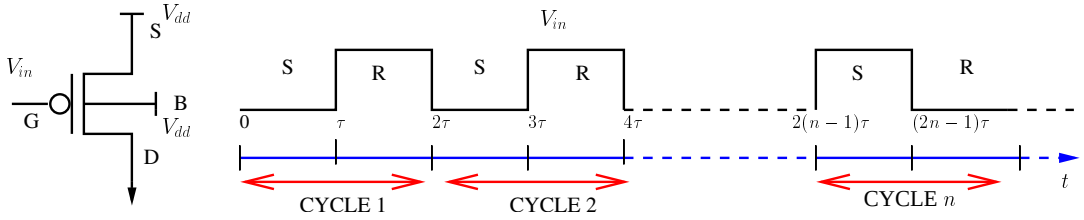


Figure 2.1: Input waveform applied to the gate of the PMOS transistor to simulate alternate stress (S) and relaxation (R) phases of equal duration τ.

2.1.1 Reaction-Diffusion (R-D) Model

The R-D model is used to annotate the process of interface trap generation and hydrogen diffusion, which is governed by the following chemical equations:

$$\begin{aligned} \mathrm{Si-H} + \mathrm{h}^{+} &\rightarrow \mathrm{Si}^{+} + \mathrm{H} \\ \mathrm{H} + \mathrm{H} &\rightarrow \mathrm{H}_2 \end{aligned} \tag{2.1}$$

where the holes in the channel interact with the weak Si-H bonds, thereby releasing neutral hydrogen atoms, and leaving behind interface traps. Hydrogen atoms combine to form hydrogen molecules, which diffuse into the oxide.

According to the R-D model, the rate of generation of interface traps initially depends on the rate of dissociation of the Si-H bonds (which is controlled by the forward rate constant, k_f) and the local self-annealing process (which is governed by the reverse rate constant, k_r). This constitutes the **reaction phase** in the R-D model. Thus, we have:

$$\frac{dN_{IT}}{dt} = k_f[N_0 - N_{IT}] - k_r N_{IT} N_H^0 \tag{2.2}$$

where N_{IT} is the number of interface traps, N_0 is the maximum density of Si-H bonds and N_H^0 is the density of hydrogen atoms at the substrate-oxide interface. After sufficient trap generation, the rate of generation of traps is limited by the diffusion of hydrogen molecules[1]. The rate of growth of interface traps is controlled by the diffusion of hydrogen molecules away from the surface based on the equation:

$$\frac{dN_{IT}}{dt} = \phi_{N_{H_2}} \tag{2.3}$$

where $\phi_{N_{H_2}}$ is the flow of diffusion of H_2 from the interface to the oxide. At the interface, it follows the equation:

$$\frac{dN_{IT}}{dt} = -D_{ox}\frac{dN_{H_2}}{dx} \tag{2.4}$$

while the diffusion of hydrogen into the oxide is given by:

$$\frac{dN_{H_2}}{dt} = D_{ox}\frac{d^2N_{H_2}}{dx^2} \tag{2.5}$$

where N_{H_2} is the concentration of hydrogen molecules at a distance x from the interface at time t, (while $N_{H_2}^0$, at the substrate-oxide interface)[2], and D_{ox} is the diffusion coefficient. This constitutes the **diffusion phase** in the R-D model. In order to find a coupling relation between N_H^0 in the reaction-phase equation in (2.2) and $N_{H_2}^0$ in the diffusion-phase equation, we use the mass action law:

$$N_{H_2}^0 = k_H (N_H^0)^2 \tag{2.6}$$

since two hydrogen atoms can combine to form a hydrogen molecule with the rate constant k_H [52, 56].

2.1.2 Solution to the Reaction Phase

During the initial reaction phase, the concentration of hydrogen atoms and interface traps are both very low, and there is virtually no reverse reaction. Hence, the number of interface traps increases

[1]Initial works assumed diffusion of hydrogen atoms, although it is now widely conjectured that hydrogen molecular diffusion occurs [2, 3, 52].

[2]We will represent $N_{H_2}^0(t)$ and $N_H^0(t)$ as $N_{H_2}^0$ and N_H^0, respectively, except in cases where the the value of t is not obvious within the context.

with time linearly as:

$$N_{IT}(t) = k_f N_0 t \tag{2.7}$$

The linear dependence of N_{IT} on time t correlates with results from numerical simulations in [2,56]. This process lasts for a very short time (around 1ms). Gradually, the process of interface trap generation begins to slow down due to the increasing concentration of hydrogen molecules, and the reverse reaction. The process then attains a quasi-equilibrium [57], and subsequently becomes diffusion limited.

Fig. 2.2 shows results from our numerical simulation setup (described later on in Section 2.6), showing the three regimes namely:

1. Reaction phase which lasts less than a millisecond, during which N_{IT} increases linearly with time, as seen from Fig. 2.2.

2. Quasi-equilibrium phase during which the interface trap count does not increase.

3. Rate-limiting diffusion phase during which the mechanism is diffusion limited.

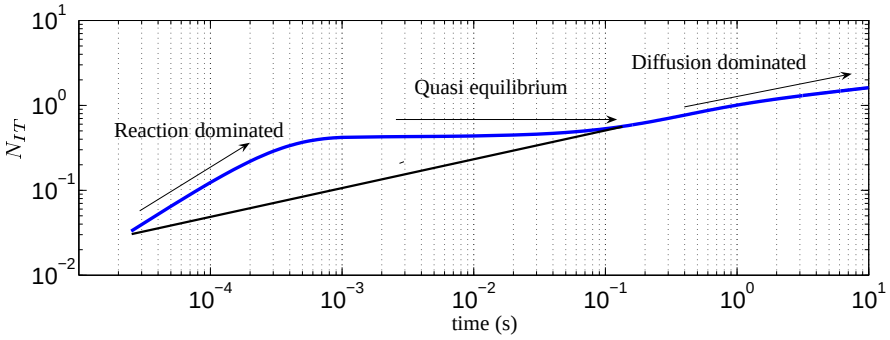


Figure 2.2: Results of numerical simulation showing the three regimes of interface trap generation, during the DC stress phase.

The reaction phase is ignored in the final model, for reasons that will become apparent at the end of Section 2.1.4.

2.1.3 Diffusion Phase

During this phase, the diffusion of hydrogen molecules becomes the rate limiting factor. Since the number of interface traps now grows rather slowly with time, the left hand side in (2.2) is approximated as zero. The initial density of Si-H bonds is larger than the number of interface traps that are generated, so that $N_0 - N_{IT} \approx N_0$. This leads to the following approximation for the reaction equation:

$$\frac{k_f N_0}{k_r} \approx N_{IT} N_H^0 \tag{2.8}$$

We now solve the R-D model for the specific case of a square waveform, as shown in Fig. 2.1, applied to the gate of a PMOS device whose source is at V_{dd}. Four different cases arise in the model, namely the first stress phase, which occurs from time $t = 0$ to τ, ($t = 0$ corresponds to the time at which stress is first applied, and the device is unstressed for all $t < 0$), the first relaxation phase which occurs from time τ to 2τ, the second and subsequent stress phases, and the second and subsequent relaxation phases. The varying physical mechanisms and the boundary conditions in each of these phases, as well as the dependence on history effects require them to handled separately. Our analytical solution for the first stress and relaxation phases is largely consistent with [13,16,17]. The extension of the analytical model to the subsequent stress/relaxation phases, is an entirely new contribution of this work.

2.1.4 First Stress Phase

The first stress phase occurs from time $t = 0$s to τ, as indicated in Fig. 2.1. During this stage, the PMOS device is under negative bias stress, and hence, generation of interface traps occurs. The first stress phase occurs from time $t = 0$s to τ, as indicated in Fig. 2.1. During this stage, the PMOS device is under negative bias stress, and hence, generation of interface traps occurs.

The number of interface traps increases with time rapidly initially, as given by (2.7), before reaching quasi-equilibrium, and eventually the mechanism becomes diffusion-limited. At this point, the rate of generation of hydrogen is rather slow, and therefore, diffusion within the oxide, described by (2.5), can be approximated as:

$$D_{ox} \frac{d^2 N_{H_2}^x(t)}{dx^2} = 0 \tag{2.9}$$

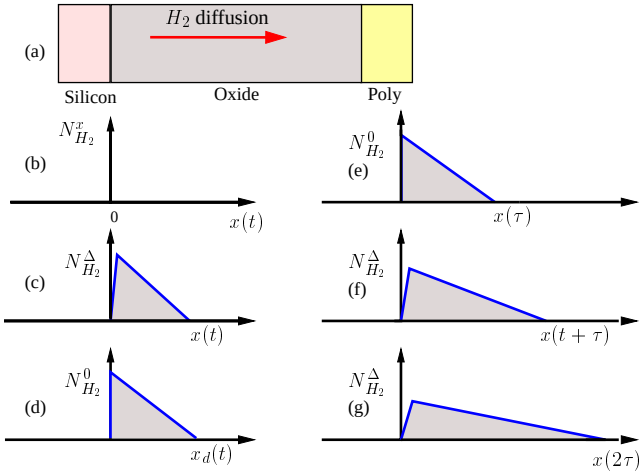


Figure 2.3: Diffusion front for the first cycle: (a) shows the cross section of the PMOS transistor: $x > 0$ in the direction of the oxide. (b) shows the front at time $t = 0$, and the hydrogen concentration is 0. (c)-(e) show the front during the first stress phase. (f) shows the front during the first recovery phase, indicating the decrease in the peak of the hydrogen density at the interface from $N^0_{H_2}$ to $N^\Delta_{H_2}$. (g) shows the front at the end of first recovery phase.

This implies that $N^x_{H_2}(t)$ is an affine function of x, where x is the extent to which the front has diffused at a given time t. The diffusion front can be approximated as shown in Fig. 2.3, which plots the diffusion front at various time points, during the diffusion process. The concentration of hydrogen molecules is highest ($N^\Delta_{H_2}$), at some Δ, close to the interface, where the traps are generated, and gradually decreases as hydrogen diffuses into the oxide, as illustrated in Fig. 2.3(c). This can be easily approximated as a right angled triangle as shown in Fig. 2.3(d), where the hydrogen concentration at the interface is $N^0_{H_2}$, and is zero at a point known as the *diffusion front*, which we will denote as $x_d(t)$: this is the extent to which the diffusing species has penetrated the oxide at time t[3]. Therefore, we have

$$\frac{dN_{H_2}}{dx} = -\frac{N^0_{H_2}}{x_d(t)} \tag{2.10}$$

$$\text{and } N^x_{H_2}(t) = N^0_{H_2} - \left[\frac{N^0_{H_2}}{x_d(t)}\right] x \tag{2.11}$$

[3]This is consistent with the right half of Fig 4a in [13]: the curve there looks (deceptively) more rounded, but this is because the y-axis is on a log scale, and on a linear y-axis, the triangle is a reasonable assumption. The reason why the hydrogen concentration is 0 at $x = 0$, is given by the Reaction-Diffusion equation.

Due to the one-one correspondence between the interface traps and the H_2 species, the total density of interface traps must equal the total density of hydrogen in the oxide. Therefore,

$$N_{IT}(t) = \int_{x=0}^{x=x_d(t)} N_{H_2}^x(t)dx \tag{2.12}$$

The value of the above integral is simply the area of the triangle enclosed by the diffusion front in Fig. 2.3(d). Thus, we have:

$$N_{IT}(t) = \frac{1}{2} N_{H_2}^0 x_d(t) \tag{2.13}$$

The above equations can be expressed equivalently in terms of N_H^0 using (2.6) and the fact that the number of hydrogen atoms is twice the number of hydrogen molecules. Hence,

$$\begin{aligned} N_{IT}(t) &= 2\int_{x=0}^{x=x_d(t)} k_H \left((N_H^0)^2 - \left[\frac{(N_H^0)^2}{x_d(t)} \right] x \right) dx \\ &= k_H (N_H^0)^2 x_d(t) \end{aligned} \tag{2.14}$$

The approximation comes about because the reaction rate is fast enough that uncombined N_H^0 are sparse: this is supported by the fact that practically, diffusion is seen to be due to H_2 and not H. The above equation relates the number of interface traps to the number of hydrogen species at the interface. We may now substitute (2.14) in the LHS of (2.4), and (2.10) in the RHS of (2.4), and further use (2.6) to obtain:

$$\begin{aligned} k_H (N_H^0)^2 \frac{dx_d(t)}{dt} &= D_{ox} \frac{k_H (N_H^0)^2}{x_d(t)} \\ \text{i.e., } x_d(t) dx_d(t) &= D_{ox} dt \end{aligned} \tag{2.15}$$

Integrating this, we obtain:

$$x_d(t) = \sqrt{2 D_{ox} t} \tag{2.16}$$

and using this in (2.14), we get:

$$N_{IT}(t) = k_H (N_H^0)^2 \sqrt{2 D_{ox} t} \tag{2.17}$$

Finally, we substitute the above relation in (2.8) to obtain:

$$N_{IT}(t) = \left(\frac{k_f N_0 \sqrt{k_H}}{k_r}\right)^{\frac{2}{3}} (2D_{ox}t)^{\frac{1}{6}} = k_{IT}\,(2D_{ox}t)^{\frac{1}{6}} \tag{2.18}$$

where $k_{IT} = \left(\frac{k_f N_0 \sqrt{k_H}}{k_r}\right)^{\frac{2}{3}}$.

At $t = \tau$, $N_{IT}(\tau) = k_{IT}(2D_{ox}\tau)^{\frac{1}{6}}$, and the diffusion front is as shown in Fig. 2.3(e).

It must be noted that we ignore the reaction phase equation given by (2.7), which captures the rapid initial rise in the number of interface traps. Fig. 2.2 shows the extrapolated shape of the curve (using dotted lines) from a numerical simulation, for the case where the reaction phase is ignored in the model, and merely the diffusion phase is considered. The results show that ignoring the reaction and equilibrium phases leads to an underestimation in N_{IT} initially, as shown in Fig. 2.2. However, the mechanism is clearly diffusion limited, and we are interested in determining the impact of NBTI after a few years of operation. Hence, an underestimation in the number of interface traps for up to 1s does not affect the overall accuracy of the model, or the long-term shape of the N_{IT} curve.

2.1.5 First Recovery Phase

Let us model the events at the interface as a superposition of two effects: "forward" diffusion, away from the interface, and "reverse" diffusion, toward the interface; the latter anneals the interface traps. During this condition, the diffusion of existing species continues as $x(t+\tau)$ and the peak of the diffusion front decreases from $N^0_{H_2}$ to some $N^{\Delta}_{H_2}$, as shown in Fig. 2.3(f)-(g). The reason for the reduction in the peak of the hydrogen species can be explained as follows: Due to the annealing of traps (also known as *backward diffusion*), some of the Si-H bonds are formed again, causing a decrease in the density of H_2 species, and thereby, a net reduction in the height of the triangle. Further, the net area under the triangle (= number of interface traps), shown by the shaded region in Fig. 2.3(f), is lower than that in Fig. 2.3(e). Since hydrogen continues to diffuse into the oxide, the base of the triangle in Fig. 2.3(f) is longer than that in Fig. 2.3(e), implying that the peak of the front at the interface must reduce.

We may think of the hydrogen concentration as a triangle that is almost right angled: at time $(\tau + t)$, it goes from 0 at $x = 0$, to $N^{\Delta}_H(\tau + t)$ at $x = \Delta$, for some small value of Δ, and then to 0

again at $x = x(\tau + t)$.

Using the same notation as [13], page 3:

$$\frac{dN_{IT}}{dt} \approx 0 = -k_r(N_{IT}^0 - N_{IT}^*)(N_H^0 - N_H^*) \tag{2.19}$$

Since the residual number of interface traps, $(N_{IT}^0 - N_{IT}^*)$, is significantly larger than zero, it must mean that the residual hydrogen concentration at the interface, $(N_H^0 - N_H^*)$ must be near-zero.

Denoting the number of annealed traps as $N_{IT}^*(\tau+t)$, we can express the net number of interface traps during the relaxation phase as the original number of traps, minus the number of annealed traps:

$$N_{IT}(\tau + t) = N_{IT}(\tau) - N_{IT}^*(\tau + t) \tag{2.20}$$

The number of interface traps annealed due to backward diffusion [13] can be expressed as:

$$N_{IT}^*(\tau + t) = \frac{1}{2} N_{H_2}^{\Delta} \sqrt{\xi 2 D_{ox} t} \tag{2.21}$$

Intuitively, this can be considered to be equivalent to a triangle whose height is given by $N_{H_2}^{\Delta}$ and the backward diffusion front beginning at time τ is given by some $x^*(t) = \sqrt{\xi 2 D_{ox} t}$, where ξ indicates the nature of backward diffusion [13] (two-sided or one-sided). Theoretically, $\xi = 0.5$ for double-sided diffusion, where as empirically, its value is reported to be around 0.58. Based on the argument in the previous section, the total number of interface traps is given by the area enclosed under the triangle in Fig. 2.3(f) and can be easily approximated (since $\Delta \approx 0$) as:

$$N_{IT}(\tau + t) \approx \frac{1}{2} N_{H_2}^{\Delta}(\tau + t) x(\tau + t) \tag{2.22}$$

From (2.21) and (2.22), we have

$$N_{IT}^*(\tau + t) = \frac{N_{IT}(\tau + t)}{x(\tau + t)} \sqrt{\xi 2 D_{ox} t} \tag{2.23}$$

From (2.20) and (2.23), we have

$$N_{IT}(\tau + t) = N_{IT}(\tau) - \frac{N_{IT}(\tau + t)}{x(\tau + t)}\sqrt{\xi 2 D_{ox} t} \tag{2.24}$$

Substituting for $x(t + \tau)$ from (2.16), and re-arranging some terms, we get

$$N_{IT}(\tau + t) = \frac{N_{IT}(\tau)}{1 + \sqrt{\frac{\xi t}{\tau + t}}} \tag{2.25}$$

For small values of t, for the case of double sided diffusion, where $\xi = 0.5$, using the approximation $\frac{1}{1+x} \approx 1 - x$, this matches the formula in [13]. However, for $t = \tau$, this gives $N_{IT}(2\tau) = \frac{2}{3}N_{IT}(\tau)$ instead of $N_{IT}(2\tau) = \frac{1}{2}N_{IT}(\tau)$, as in [13]. This value is consistent with Fig. 20 of [2], which shows the value of N_{IT} going down to about $\frac{2}{3}$ of its peak value during recovery. Further, at $t = \tau$, the approximation used in [13] is not valid, and hence we use (2.25) for modeling the recovery phase.

2.1.6 Second and Subsequent Stress Phases

For the second and subsequent stress phases, it is important to take into account the boundary conditions that result from the previous stress and relaxation phases. For example, if the first stress phase is much longer than the first recovery phase, there is practically no recovery, and a second stress phase is virtually equivalent to applying a continuous stress: as we will see, our derivation will provide correct solutions under this and other cases.

At the end of the first recovery phase, the total number of interface traps is $N_{IT}(2\tau) = \frac{2}{3}N_{IT}(\tau)$. In addition, new traps are generated under stress. During the recovery phase, the peak value, $N_{H_2}^{\Delta}$ is reduced from $N_{H_2}^{0}(\tau)$ (Figs. 2.3(e) and 2.4(b)), but this rapidly recovers back, during the next stress phase. The recovery phase effectively sets back the degradation phase to an earlier time, t_{eff}, as the point that satisfies the equation:

$$N_{IT}(2\tau) = \frac{2}{3}k_{IT}(2D_{ox}\tau)^{\frac{1}{6}} = k_{IT}(2D_{ox}t_{\text{eff}})^{\frac{1}{6}} \tag{2.26}$$

It is easy to see that the solution to this is $t_{\text{eff}} = (\frac{2}{3})^6\tau$. This corresponds to a new *effective location*

of the diffusion front, x_{eff} (Fig. 2.4(c)), given by:

$$x_{\text{eff}}(2\tau) = \left(\frac{2}{3}\right)^3 \sqrt{2D\tau} \tag{2.27}$$

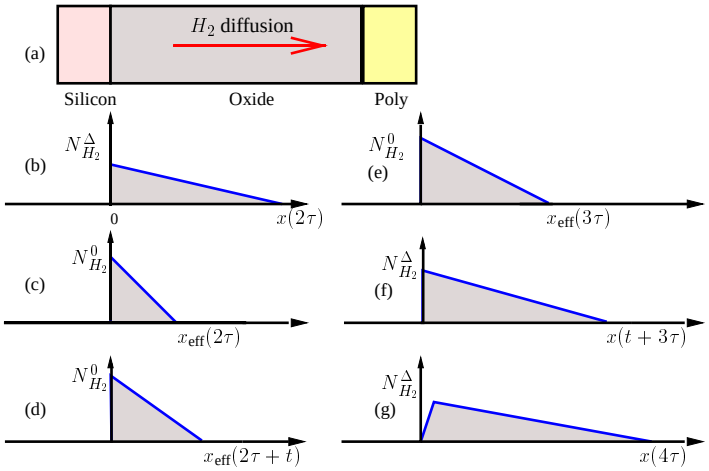


Figure 2.4: Diffusion front for the second cycle of stress and recovery: (a) shows the cross section of the PMOS transistor. x is positive for diffusion occurring into the oxide. (b) indicates the front at the end of the first relaxation phase. (c) shows the equivalent front at time $t = 2\tau$, where the effective diffusion front is at $x_{\text{eff}}(2\tau)$, and the hydrogen concentration calculated accordingly. (d)-(e) show the front during the second stress phase. (f) shows the front during the second relaxation phase, indicating the decrease of hydrogen concentration at the interface from $N^0_{H_2}$ to $N^{\Delta}_{H_2}$. The tip of the diffusion front is now at some $x(3\tau + t)$. (g) shows the front at the end of second relaxation phase.

Diffusion now continues beyond this point as shown in (Fig. 2.4(d)-(e)) and the location of the diffusion front is found by integrating (2.15) from time 2τ to $2\tau + t$, given the initial condition $x_d(2\tau) = x_{\text{eff}}(2\tau)$:

$$x_d(2\tau + t) = \sqrt{2D_{ox}t + x^2_{\text{eff}}(2\tau)^2} \tag{2.28}$$

Therefore, from (2.27) and (2.28),

$$x_d(2\tau + t) = 2\sqrt{2D_{ox}\left(t + \left(\frac{2}{3}\right)^6 \tau\right)} \tag{2.29}$$

and

$$\begin{aligned} N_{IT}(2\tau + t) &= k_H (N_H^0)^2 x_d(2\tau + t) \\ &= (N_H^0)^2 \sqrt{2D_{ox}\left(t + \left(\frac{2}{3}\right)^6 \tau\right)} \end{aligned} \tag{2.30}$$

A similar analysis as in Section 2.1.4 can be applied to substitute this in (2.8) to obtain

$$\begin{aligned} N_{IT}(2\tau + t) &= k_{IT}(2D_{ox}\tau)^{\frac{1}{6}} \left(\frac{t}{\tau} + \left(\frac{2}{3}\right)^6\right)^{\frac{1}{6}} \\ &= \left(\frac{t}{\tau} + \left(\frac{2}{3}\right)^6\right)^{\frac{1}{6}} N_{IT}(\tau) \end{aligned} \tag{2.31}$$

At $t = \tau$, we have $N_{IT}(3\tau) = (\frac{793}{729})^{\frac{1}{6}} N_{IT}(\tau)$. Defining s_k as the "scaling factor," relative to $N_{IT}(\tau)$ at time $k\tau$, (i.e., $s_k = \frac{N_{IT}(k\tau)}{N_{IT}(\tau)}$), we have $s_3 = (\frac{793}{729})^{\frac{1}{6}}$. The derivation for the subsequent stress phases is elucidated in Appendix A. The total number of interface traps during the $(k+1)^{th}$ stress phase, from time $2k\tau$ to $(2k+1)\tau$, is given by:

$$N_{IT}(2k\tau + t) = \left(\frac{t}{\tau} + \left(\frac{N_{IT}(2k\tau)}{N_{IT}(\tau)}\right)^6\right)^{\frac{1}{6}} N_{IT}(\tau) \tag{2.32}$$

$$N_{IT}((2k+1)\tau) = \left(1 + s_{2k}^6\right)^{\frac{1}{6}} N_{IT}(\tau) \tag{2.33}$$

Using the s_k notation as defined above, (i.e., $s_{2k+1} = \frac{N_{IT}((2k+1)\tau)}{N_{IT}(\tau)}$), we can write:

$$s_{2k+1} = \left(1 + s_{2k}^6\right)^{\frac{1}{6}} \tag{2.34}$$

[2] As a sanity check, it is rather simple to see that if the first recovery phase were replaced by a continuation of the stress, then $x_{\text{eff}}(2\tau)$ would be $2\sqrt{D_{ox}\tau}$, leading to the expected result, $x_d(2\tau + t) = \sqrt{2D_{ox}(2\tau + t)}$.

2.1.7 Second and Subsequent Recovery Phases

The analysis of the second and subsequent recovery phases is very similar to the first recovery phase, since the hydrogen front diffuses in an identical manner as that in the first recovery phase (Figs. 2.3 (f)-(g) and 2.4 (f)-(g)). The detailed derivation for this case is presented in Appendix A. Thus, we can determine the number of interface traps during recovery, in the k^{th} cycle, (i.e., time $(2k-1)\tau$ to $2k\tau$) using the equation:

$$N_{IT}((2k-1)\tau + t) = \frac{N_{IT}((2k-1)\tau) + N_{IT}((2k-2)\tau)\sqrt{\frac{\xi t}{\tau + t}}}{1 + \sqrt{\frac{\xi t}{\tau + t}}} \tag{2.35}$$

At $t = \tau$, we have

$$N_{IT}(2k\tau) = \frac{2}{3}N_{IT}((2k-1)\tau) + \frac{1}{3}N_{IT}(2(k-1)\tau) \tag{2.36}$$

Using the s_k notation as defined in the previous section, (i.e., $s_{2k} = \frac{N_{IT}(2k\tau)}{N_{IT}(\tau)}$), we can write:

$$s_{2k} = \frac{2}{3}s_{2k-1} + \frac{1}{3}s_{2k-2} \tag{2.37}$$

The analytical solution of the R-D model for the first two cycles is plotted in Fig. 2.5. The curve marked as "DC stress" corresponds to the case where the device is stressed continuously, while "AC stress" corresponds to the case, as shown in Fig. 2, i.e., stress and relaxation applied alternatively. Expectedly, the values for the AC stress case are lower than that for the DC stress case, due to recovery. Further, the simulation values are compared with the experimental data shown in circles (obtained from Fig. 20 of [2]). Our simulation results for the AC stress case, match with the experimental results, thereby validating the correctness of our analytical solution. Although the exact shape of the waveform may not perfectly correlate with the experimental results, the values at the end of the stress and the relaxation phases closely match the experimental data, and hence captures the overall transient NBTI-action effectively.

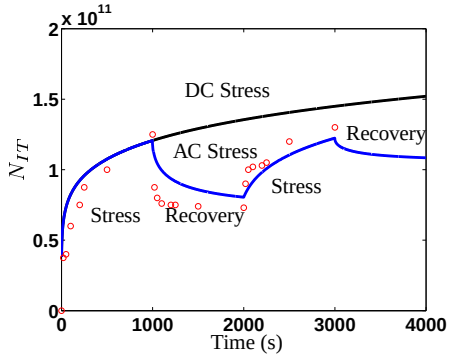


Figure 2.5: Analytical solution to the R-D model for the first two cycles and comparison with experimental data from [2].

2.1.8 Overall Expression

In general, we may write $s_k = \frac{N_{IT}(k\tau)}{N_{IT}(\tau)}$, for even or odd values of k, (i.e., for the alternate stress or recovery phases of equal duration τ) as:

$$s_k = \begin{cases} 0 & k = 0 \\ 1 & k = 1 \\ (1 + s_{k-1}^6)^{\frac{1}{6}} & k > 1, k \text{ odd} \\ \frac{2}{3}s_{k-1} + \frac{1}{3}s_{k-2} & k > 1, k \text{ even} \end{cases} \tag{2.38}$$

These values can be plotted as shown in Fig. 2.6(a). Since the data is plotted for a large number of cycles, the stress and relaxation transients are not easy to discern, and the AC stress curve appears to be smooth.

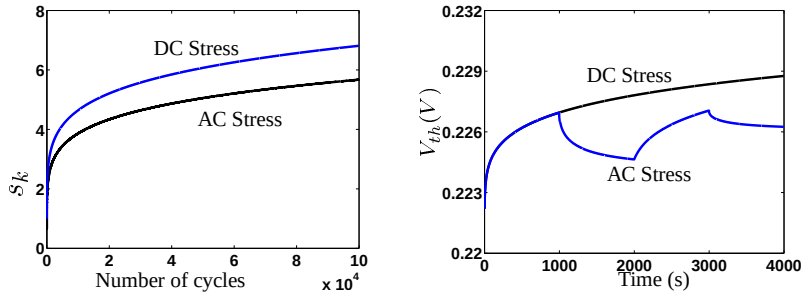
2.1.9 Threshold Voltage Degradation

The generation of interface traps due to NBTI causes a shift in the transistor threshold voltage, given by [37] as,

$$\Delta V_{th} = -\frac{(m+1)qN_{IT}}{C_{ox}} \tag{2.39}$$

where m is a measure of the additional V_{th} degradation caused due to mobility degradation [8]. Thus, the effect of N_{IT} on V_{th} is linear in nature. The threshold voltage is plotted for the first four

phases (first and second stress and relaxation phases) of an AC stress case, and also for the DC stress case in Fig. 2.6(b). The shape of the plot is similar to that seen in [8, 13, 55, 58], all of which use numerical solutions to estimate the extent of V_{th} shift.



(a) Plot showing s_k for DC and AC stress.

(b) V_{th} shift for DC and AC stress.

Figure 2.6: Simulation results showing N_{IT} and V_{th} for DC stress and AC stress.

While the analysis of the R-D model has been derived using the $t^{\frac{1}{6}}$ case of neutral H_2 species diffusion, a similar analysis has been carried out for the $t^{\frac{1}{4}}$ case of neutral H diffusion, and the key results are shown in Appendix B.

2.2 Frequency Independence

An important requirement of any physical model for NBTI [13] is that it should be able to explain the phenomenon of frequency independence, observed over a wide range of frequencies [2, 13, 52–55]. Frequency independence implies that for any two periodic waveforms with frequencies f_1 and f_2, and the same duty cycle, the number of interface traps generated at any time t, such that $t \gg \frac{1}{f_1}$ and $t \gg \frac{1}{f_2}$, is the same.

The concept of frequency independence has been demonstrated through experiments using periodic square waveforms of different frequencies. Although, the exact range of frequencies over which this phenomenon holds good is still not very clear, some form of frequency independence is widely observed in the 1Hz - 1MHz range [13, 59] and has recently been shown to exist over the entire range of 1Hz - 2GHz in [60]. We first verify this numerically for three frequencies (0.1Hz,

1Hz, and 10Hz) by measuring the threshold voltage (equivalent to number of interface traps) after 1000s. The values are computed using the expressions derived for s_k, from (2.38). The extent of V_{th} degradation for the three curves after 1000s, plotted in Fig. 2.7 is the same, thereby demonstrating the ability of our model to demonstrate frequency independence.

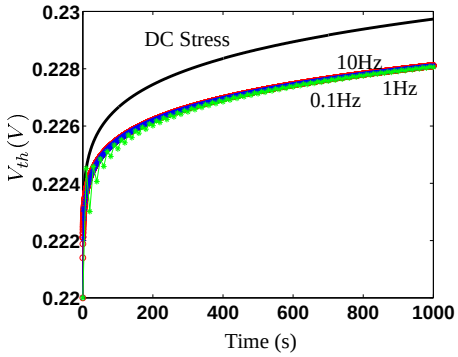


Figure 2.7: Curve showing V_{th} degradation for DC case and 3 different AC frequencies (10Hz, 1Hz, and 0.1Hz).

We now provide a mathematical proof for the special class of periodic waveforms, namely square waveforms which are widely used in NBTI-experiments. Consider two square waveforms A and B, as shown in Fig. 2.8. Let the time period (T) of A be denoted as τ_1, while the time period of B is τ_2. The duty cycle of both these waveforms is 50%, (i.e., on time = off time = $\frac{T}{2}$). Let these waveforms be applied to the PMOS device separately, up to a certain time t, such that t is a large integral multiple of τ_1 and τ_2. In other words, $t = k_1\tau_1 = k_2\tau_2$, k_1 and k_2 being sufficiently large. Further, let $\tau_1 < \tau_2$, and hence $k_1 > k_2$. Let the number of interface traps after time t for the two cases be denoted as $N_{IT}(\tau_1 k_1)$ and $N_{IT}(\tau_2 k_2)$, respectively. By "frequency independence", we imply:

$$N_{IT}(t) = N_{IT}(\tau_1 k_1) = N_{IT}(\tau_2 k_2) \tag{2.40}$$

The above equation can be proved (2.40) mathematically using the expressions for s_k, developed

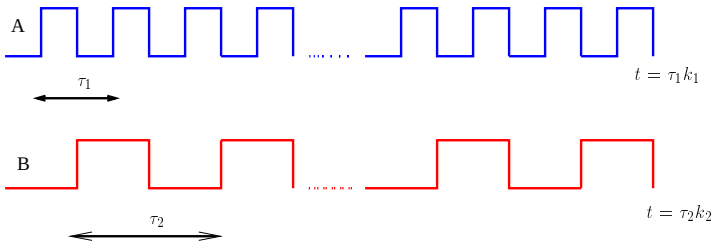


Figure 2.8: Figure showing two square waveforms with different time periods but the same duty cycle (50%).

in the previous section. From (2.38), we have:

$$\begin{aligned} s_k^6 &= s_{k-1}^6 + 1 \\ s_{k-1} &= \frac{2}{3} s_{k-2} + \frac{1}{3} s_{k_1-3} \\ s_{k-2}^6 &= s_{k-3}^6 + 1 \end{aligned} \tag{2.41}$$

The above set of equations can be simplified to obtain a relation for large values of k as:

$$s_k^6 \approx s_{k-2}^6 + \frac{2}{3} \tag{2.42}$$

The proof of the above approximation is shown in Appendix C. From (2.42), for some $m < k$, we can write

$$s_k^6 \approx s_m^6 + \frac{2}{3}\frac{(k-m)}{2}. \tag{2.43}$$

Choosing k as k_1 or k_2, and $m = 0$, and using the fact that $s_0 = 0$, we can write $s_{k_1}^6 \approx \frac{k_1}{3}$, and $s_{k_2}^6 \approx \frac{k_2}{3}$, which implies:

$$\left(\frac{s_{k_1}}{s_{k_2}}\right)^6 \approx \frac{k_1}{k_2} \tag{2.44}$$

Now, we can write N_{IT} as:

$$N_{IT}(t_{0_1} k_1) = s_{k_1} N_{IT}(t_{0_1}) = s_{k_1} k_{IT} (2 D_{ox} t_{0_1})^{\frac{1}{6}} \tag{2.45}$$

$$N_{IT}(t_{0_2} k_2) = s_{k_2} N_{IT}(t_{0_2}) = s_{k_2} k_{IT} (2 D_{ox} t_{0_2})^{\frac{1}{6}} \tag{2.46}$$

and therefore,

$$\frac{N_{IT}(t_{0_1}k_1)}{N_{IT}(t_{0_2}k_2)} = \frac{s_{k_1}}{s_{k_2}} \left(\frac{t_{0_1}}{t_{0_2}}\right)^{\frac{1}{6}} \tag{2.47}$$

Using (2.42), the above equation can be re-written as:

$$\frac{N_{IT}(t_{0_1}k_1)}{N_{IT}(t_{0_2}k_2)} \approx \left(\frac{k_1}{k_2}\right)^{\frac{1}{6}} \left(\frac{t_{0_1}}{t_{0_2}}\right)^{\frac{1}{6}} \tag{2.48}$$

Since $t_{0_1}k_1 = t_{0_2}k_2$, we have

$$N_{IT}(t_{0_1}k_1) = N_{IT}(t_{0_2}k_2) = N_{IT}(t).$$

Thus, the phenomenon of frequency independence is proved for square waveforms.

Although we have proved frequency independence only for the special case of square waveforms here, the same is also true for rectangular waveforms, where the time of stress and the time of relaxation in any given cycle may not be the same. The proof works in a similar manner, and is omitted due to space constraints. The result of this proof however, is used in the next section to efficiently simulate the impact of NBTI on digital circuits.

2.3 Signal Probability and Activity Factor Based Solution to the R-D Model (SPAF method)

While the impact of NBTI has been analyzed for a square wave, such waveforms are rarely seen in digital circuits, due to the random distribution of node probabilities. We now present a method to estimate the NBTI-induced V_{th} degradation of a PMOS device, when a random aperiodic waveform is applied to its gate. This method, which we call the SPAF (Signal Probability and Activity Factor) method, is based on *signal probability* (SP), i.e., the probability that the signal is at logic high, and *activity factor* (AF), i.e., the probability that a node toggles every phase, with respect to a reference clock. We first explain the underlying idea behind the SPAF method using simple square waveforms.

2.3.1 AF Independence and SP Dependence of Trap Generation

In this subsection, we show that the extent of trap generation is independent of the activity factor (AF) of the signal and depends on the signal probability only (SP). We first elucidate this using two periodic square waveforms of different frequencies.

Let us consider two square waveforms of frequencies, say f_1 =1Hz and f_2 = 100Hz. Comparing the two waveforms with a reference clock of f = 100Hz, the activity factors (AF) and signal probabilities (SP) of these waveforms over a period of time can be computed as: AF_1 = 0.01, AF_2 = 1, SP_1 = 0.5, and SP_2 = 0.5. Using (2.40), it can be seen that the interface trap density, calculated at some large time t, is the same for both these waveforms. Hence, we conclude that the amount of trap generation for any two waveforms with identical signal probabilities is independent of their activity factors. Intuitively, this is true because for the signal with a higher activity factor, although fewer traps are generated in each cycle, there are larger number of cycles, thereby leading to the same number of interface traps. However, the dependence of the trap generation on the signal probability (for two waveforms with equal time period) is rather obvious, since higher on-times imply larger amounts of stress, and lower amounts of recovery. As seen in Fig. 2.9, four waveforms with the same frequency and different stress duty cycles[4] (0.25, 0.5, 0.75, and 1) are considered and the amount of trap generation is plotted for 1000 cycles. Expectedly, there is considerable difference in the final values of the four curves.

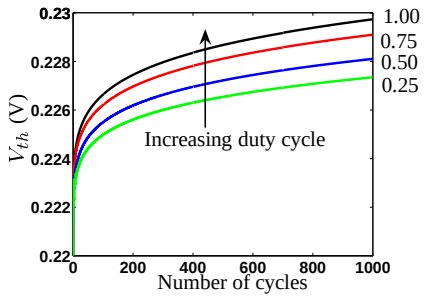


Figure 2.9: Signal probability dependence of trap generation shown for four waveforms of equal frequency but varying off-time duty cycles.

[4]Since a PMOS device is under stress when the signal is a logic zero, and signal probability is defined as the probability that the signal is a logic high, stress duty cycle = 1-SP.

2.3.2 SPAF Method

The SPAF method helps convert a random aperiodic signal to an equivalent waveform based on its statistical information (SP). Such a transformation is essential in order to perform temporal simulation of circuits, since the exact input waveforms cannot be determined. Since the interface trap density is independent of the AF of a signal, the crux of the SPAF method is to statistically compute only the signal probabilities of any given node over a large period of time (say ≈ 10000 cycles). The SP information can now be used to compute an equivalent rectangular pulse with the same SP value. Such a transformation is equivalent to representing a 100Hz square waveform by a 1Hz square waveform for instance, both of which have the same SP. The validity of this transformation is based on the frequency independence property proved in the previous section, since both these waveforms generate the same number of interface traps. The reconstructed waveform is now assumed to repeat periodically, and the total number of interface traps generated by this periodic signal is computed by simply modifying the expressions derived for s_k, outlined in Section 2.1.8.

Let the total time period of this new waveform be k cycles, such that it is low for m cycles and high for $k - m$ cycles (Fig. 2.10). Let n indicate the number of such periods, each of duration k. We can determine the number of interface traps generated, using (2.32) and (2.35), for the stress and the relaxation phases respectively. Using the s_k notation, these can be simplified as:

$$s_{nk+i} = \begin{cases} (i + s_{kn}^6)^{\frac{1}{6}} & nk < i \leq nk + m \\ \frac{s_{nk+m} + s_{nk}\left(\frac{i}{2(m+i)}\right)^{0.5}}{1+\left(\frac{i}{2(m+i)}\right)^{0.5}} & nk + m < i \leq (n+1)k \end{cases} \tag{2.50}$$

It must be noted that (2.38) can be obtained as a special case of (2.50) by using $k = 2$ and $m = 1$.

The SPAF method of equivalent waveform construction is verified for two random waveforms of SP = 0.25 and SP = 0.75. It must be noted that the stress probability of the PMOS transistor is equal to (1-SP), since the PMOS device is under stress when the signal is a logic zero. The samples are accumulated over 10000 cycles. Equivalent rectangular pulse waveforms with $k = 100$ cycles, and $n = 100$ cycles are reconstructed. The total number of interface traps are calculated at the end of 10000 cycles, using (2.50). The results, plotted in Fig. 2.11, show the same amount of trap generation for the actual waveform (Monte-Carlo simulation) and the SPAF method based

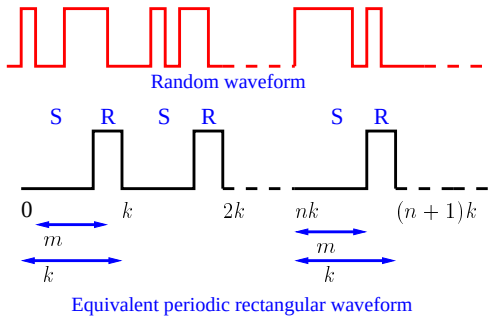


Figure 2.10: SPAF method to convert a random waveform to a periodic rectangular waveform with equivalent stress duty cycle.

solution. Thus, the SPAF method of equivalent waveform construction is validated. The SPAF method can reduce a random aperiodic waveform to an equivalent simple periodic waveform, and hence, can decrease the amount of computation, without loss of accuracy. This helps is performing quick simulations in a computationally efficient manner, as described in the next section.

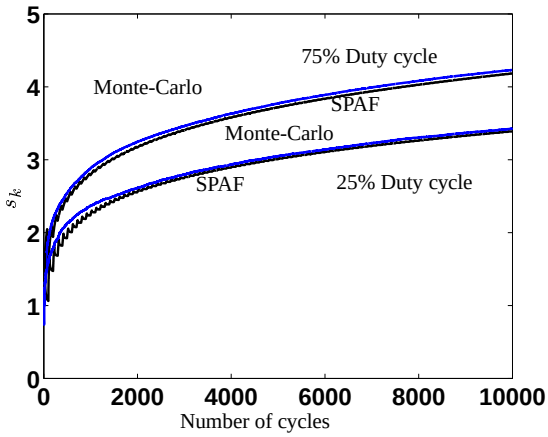


Figure 2.11: Simulations for 25% and 75% duty cycle waveforms showing the validity of the SPAF method.

The SPAF method is used in Section 3.1 during BTI-aware timing analysis, to determine the impact of NBTI on the asymptotic transistor V_{th} degradation, during high frequency operation.

2.4 Limitations

The analytical model for NBTI described in the previous section does not provide a good fit with experimental data, particularly during the recovery stage, as can be seen from Fig. 2.5. Experiments in [5, 21–23, 34, 35] show rapid recovery in V_{th} during the initial few seconds of recovery, as well as a higher fractional recovery (than the 33% as seen in Fig. 2.5, and 50% as predicted by the analytical model in [13]). The amount of fractional recovery is shown to depend on the thickness of the oxide, in [3], while the model presented in Section 2.1 is based on the assumption that the oxide thickness is infinite (or equivalently that there is no difference in the diffusion constant of hydrogen in the oxide or in polysilicon). However, the authors in [3] show that is significantly higher in the oxide than in polysilicon, thereby implying that a model for NBTI must consider the impact of finite oxide thickness.

Subsequent analytical models for NBTI have tried to address the issue of obtaining a better fit with experimental data, particularly during the beginning of the recovery phase, as well as capturing the impact of finite oxide thickness. In particular, the work in [20, 61, 62] attempts to incorporate the effects of finite oxide thickness, and the differing rates of diffusion of H_2 in oxide, and poly, and thereby provides a comprehensive multicycle model. The work in [20] concurs with our work in Section 2.2 in showing frequency independence analytically. The model provides an excellent fit with experimental data from [3], and shows more recovery for a higher d_{ox} value, which is consistent with experimental observations in [3].

However, the value of ξ in the model in [20] is deemed to be universal, and this can lead to unexpected results as follows. For instance, the recovery phase of the model in [20] for the d_{ox} = 1.2nm case is examined, for a single stress phase of 10000s, followed by continuous recovery for a long period of time. It is expected that the amount of recovery must continue to increase, with time, leading to near complete recovery at infinite time [63]. However, an evaluation of the model shows that the recovery curve reaches a minimum at around 40000s, and continues to increase beyond that time. A similar behavior is seen for the d_{ox} = 2.2nm case, with the minimum occurring at around 20000s, and the deviation from the minimum value is larger here. This may lead to unexpected behavior, and the minimum may shift toward a lower time point, for lower stress periods, and higher oxide thicknesses.

The work in [64] considers two separate models for dynamic and static NBTI stress, with the dynamic stress model showing lower amount of threshold voltage degradation after 1/10 years of operation, due to the effect of recovery. The V_{th} degradation expectedly increases as a function of input duty cycle, and the shape of the curve is similar to that in Fig. 3.2. However, the model has a severe limitation in that the V_{th} value for input duty cycle = 1.0 is much lower than the corresponding value for static NBTI: since dynamic NBTI with a 100% stress is equal to static NBTI, it is expected that the two values match, and that the dynamic NBTI curve converge with the static NBTI value. However, this drawback represents an underestimation in the impact of dynamic NBTI particularly at transistor degradation probabilities in the range of [0.9-1.0]. Subsequent, savings obtained with IVC (input vector control) as shown later on in [64], is partly due to this inherent discrepancy in the modeling itself, as opposed to the novelty of the scheme.

2.4.1 A Note on OTFM and UFM Techniques and Validity of the R-D Theory

Two current state-of-the-art techniques to measure the impact of NBTI on V_{th} during recovery include OTFM (On-the-Fly Measurement) which estimates ΔV_{th} by measuring $|\frac{\Delta I_d}{I_{d_0}}|$, and UFM (Ultra-Fast V_{th} Measurement) which estimates the intrinsic NBTI and V_{th} degradation directly. UFM-based techniques, which can measure the V_{th} degradation during the recovery phase, within 1μs after removal of the stress, have been employed in [22, 23]. Experimental results show that there is a uniform recovery of V_{th} during the relaxation phase, with an almost identical amount of fractional recovery in every decade. Subsequently, [21, 23] show results comparing the large differences between an R-D theory-based model for recovery, and the experimental data, suggesting that the R-D mechanism does not provide a satisfactory explanation for the physical action during recovery. Further, [21] explains the various drawbacks of the R-D theory-based analytical model proposed by Alam in [13], such as:

1. 50% recovery in V_{th} predicted after τ seconds of recovery, following τ seconds of stress, irrespective of the value of τ, whereas experimental results show a dependence on τ, particularly with smaller values of τ producing larger fractional recovery.

2. Numerical simulations of the R-D model predict 100% recovery, whereas [13] predicts only around 75% recovery, as $t \rightarrow \infty$.

3. Poor fit during the beginning of the recovery phase ($t \ll \tau$), and for $t \gg \tau$.

The authors in [21] hence propose a dispersive transport based model for trap generation and recovery. Further, the works in [25, 28, 29, 65] support a bulk trapping-detrapping based model, instead of a reaction-diffusion based model. However, [32] distinguishes the gate dielectrics into two types (Type I and Type II) depending on whether they are PNO (plasma nitrided oxides) or TNO (thermal nitrided oxides), and explains the discrepancy between the bulk trapping and the R-D models, for each of these types. Recently, [66] highlights the differences between an OTFM and a UFM-based technique for analyzing the impact of NBTI. The above work also shows that the R-D theory is consistent with the experimental results obtained using OTFM techniques, and the log-like recovery (equal recovery in every decade) observed in [22, 23] is consistent with a UFM-based technique. The authors in [66] also state that the log-based recovery of V_{th} observed in [23] is due to the inappropriate usage of the quasi-state relationship:

$$\Delta V_{th} = \frac{q\Delta N_{IT}}{C_{ox}} \tag{2.51}$$

to ultrafast transient conditions. Further [66] explains the drawbacks in using a UFM-based technique, and strongly supports the validity of the reaction-diffusion theory for predicting the impact of NBTI correctly.

2.4.2 Guidelines for an NBTI Model

Based on the drawbacks identified from existing NBTI models, as well as observations from several publications such as [5, 23, 27], we present some key guidelines for an NBTI model as follows:

1. The model must predict that the number of interface traps increases rapidly with time initially, as explained in [52,67], and asymptotically lead to a $N_{IT}(t) \propto t^{\frac{1}{6}}$ relationship, (assuming that the diffusing species are neutral hydrogen molecules), as experimentally observed in [2, 3, 9].

2. The model must be able to capture the "fast initial recovery phase" that is of the order of a second [5], during which recovery is higher.

3. The model must predict higher fractional recovery for a PMOS device with a larger d_{ox}, for the same duration of stress, as observed in [3]. This is because, a larger d_{ox} implies a larger number of fast diffusing hydrogen molecules in the oxide, and hence implies higher amounts of annealing.

4. For an AC stress case where the stress duration is equal to the relaxation time period, the model must predict larger fractional recovery, with lower stress times [5]. Previous works using an NBTI model [13,19] and numerical solutions of the model in [21,23] all predict 50% recovery, when the ratio of the relaxation time to the stress time is equal to one, irrespective of the actual duration of the stress time.

5. The model must predict some form of frequency independence, i.e., the number of interface traps generated must approximately be the same asymptotically, irrespective of the frequency of operation. Although, the exact range of frequencies over which this phenomenon holds good is still not very clear, some form of frequency independence is widely observed in the 1Hz - 1MHz range [13, 59] and has recently been shown to exist over the entire range of 1Hz - 2GHz in [60].

Accordingly, we propose an R-D based model for NBTI that does not consider the oxide to be infinitely thick. The results show that the model can resolve several inconsistencies, noted with the reaction-diffusion theory for NBTI generation and recombination, as observed in [5, 21–23]. Further, the model can also explain the widely distinct experimentally observed results in [5, 34, 35]. It must be noted that our model is presented under the above assumption that the R-D theory provides a valid and satisfying explanation for interface trap generation and recombination. Our work seeks seeks to provide a better understanding of the R-D mechanism, thereby improving upon the drawbacks in previous (R-D based) works, as listed in the beginning of this section. Further, it must be noted that our goal is to build a modeling mechanism for NBTI action that can be used to predict the impact on the timing degradation of digital circuits after several years of operation. Hence, a fast asymptotically accurate model, as opposed to a slow cycle accurate model that requires extensive numerical simulations, is of utmost utility.

2.5 Finite Oxide Based Model for NBTI Action

While Section 2.1.1 describes the R-D model equations for the infinite oxide case, the derivation is suitably modified here, to incorporate the fact that the diffusion constant of hydrogen in the oxide (D_{ox}) is higher than that in polysilicon (D_p). Using Fick's second law of diffusion, the rate of change in concentration of the hydrogen molecules inside the oxide is given by:

$$\frac{dN_{H_2}}{dt} = D_{ox}\frac{d^2N_{H_2}}{dx^2} \text{ for } 0 < x \leq d_{ox} \tag{2.52}$$

as described in (2.5) in Section 2.1.1. Similarly, the rate of change in concentration of the hydrogen molecules inside poly is given by:

$$\frac{dN_{H_2}}{dt} = D_p\frac{d^2N_{H_2}}{dx^2} \text{ for } x > d_{ox} \tag{2.53}$$

The derivation for the reaction phase is identical to that described in Section 2.1.1. We now rederive the model for the case where a square waveform whose stress and relaxation times are equal to τ, as shown in Fig. 2.1, is applied to the PMOS device.

The first stress phase occurs from time t = 0s to τ, as indicated in Fig. 2.1. During this stage, the PMOS device is under negative bias stress, and hence, generation of interface traps occurs. The stress phase consists of two components, namely diffusion in oxide and diffusion in polysilicon, leading to two analytical expressions, respectively.

Fig. 2.12 shows the evolution of the diffusion front at different time stamps. Unlike the triangular front in Fig. 2.3 for the infinitely thick oxide case, the diffusion front becomes a quadrilateral in the oxide, followed by a triangle in polysilicon, as described in Fig. 2.12.

The derivation for the number of interface traps for diffusion in oxide is presented in Section 2.1.4. Thus, we have:

$$N_{IT}(t) = k_{IT}\left(2D_{ox}t\right)^{\frac{1}{6}} \tag{2.54}$$

where $k_{IT} = \left(\frac{k_f N_0\sqrt{k_H}}{k_r}\right)^{\frac{2}{3}}$, as shown in (2.18). The above equation is valid until the tip of the diffusion front has reached the oxide-poly interface, as shown in Fig. 2.12(d). The time at which

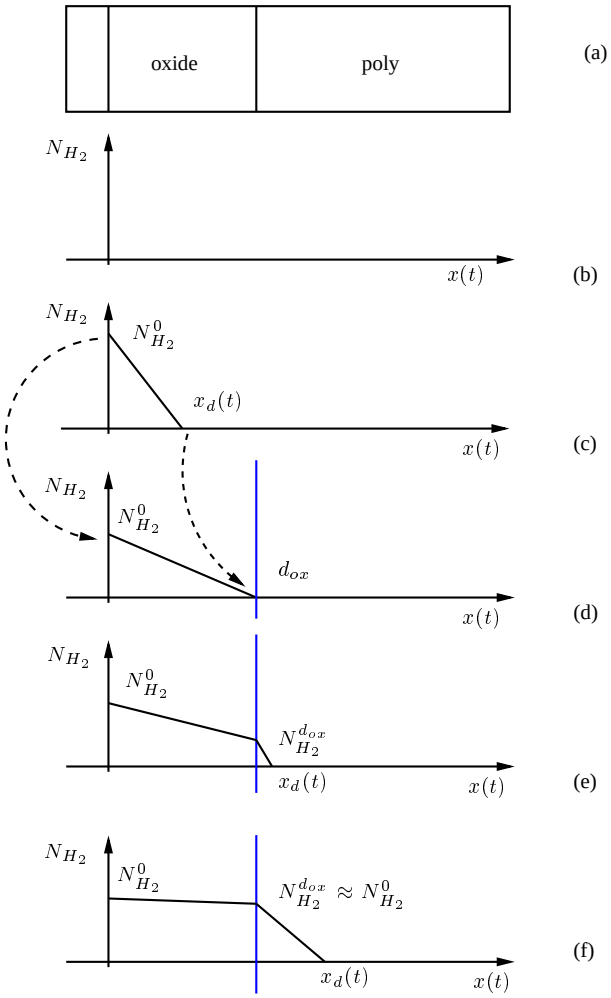


Figure 2.12: Diffusion front for the first stress phase: (a) shows the cross section of the PMOS transistor: $x > 0$ denotes the direction of the oxide-poly. (b) shows the front at time $t = 0$, and the hydrogen concentration is 0. (c)-(f) show the front during the first stress phase. (c) shows the triangular approximation of the diffusion front in the oxide, with the peak denoted by $N^0_{H_2}$, while the tip of the front is at $x_d(t)$. (d) shows the front at the oxide-poly boundary, i.e., when $x_d(t)$ = d_{ox}, and the subsequent decrease in the peak concentration. (e) shows the front extending into poly, while (f) shows that since $D_{ox} > D_p$, the front can be approximated as a rectangle in oxide, followed by a triangle in poly, i.e., $N^{d_{ox}}_{H_2} \approx N^0_{H_2}$.

this occurs is denoted by t_1, and can be computed by substituting $x_d(t) = d_{ox}$ in (2.16) to obtain:

$$t_1 = \frac{{d_{ox}}^2}{2D_{ox}} \tag{2.55}$$

Typically, t_1 is of the order of a second for current technologies, considering the values of the oxide thickness, and D_{ox}. The number of interface traps for the first stress phase can thus be expressed as:

$$N_{IT}(t, 0 < t \leq t_1) = k_{IT} x_d(t)^{\frac{1}{3}} \tag{2.56}$$

where $x_d(t) = \sqrt{2D_{ox}t}$.

Assuming that τ is greater than t_1 (the case where $\tau < t_1$ is handled later), the diffusion front moves into polysilicon as well, as shown in Fig. 2.12(e), although the diffusion coefficient for H_2 in poly (denoted as D_p), is lower than that in the oxide [3]. The rate of change in concentration of the hydrogen molecules inside poly is given by:

$$\frac{dN_{H_2}}{dt} = D_p \frac{d^2 N_{H_2}}{dx^2} \text{ for } x > d_{ox} \tag{2.57}$$

which is similar to the equation for oxide in (2.5). Assuming steady state diffusion, as in the case with the oxide (2.9) in Section 2.1.4, the above expression can also be approximated as:

$$D_p \frac{d^2 N^x_{H_2}(t)}{dx^2} = 0 \tag{2.58}$$

implying that the diffusion front in poly is also linear. The diffusion front assumes a quadrilateral shape inside the oxide, followed by a triangle in poly, with the tip of the diffusion front being at some $x_d(t) > d_{ox}$. Hence, we have:

$$\begin{aligned} \phi_{N_{H_2}} &= -D_p \frac{dN_{H_2}}{dx} \\ \frac{dN_{H_2}}{dx} &= \frac{N^{d_{ox}}_{H_2}}{x_d - d_{ox}} \end{aligned} \tag{2.59}$$

for $x > d_{ox}$ and

$$N^x_{H_2}, x > d_{ox} = N^{d_{ox}}_{H_2} - \frac{N^{d_{ox}}_{H_2}}{x_d - d_{ox}}(x - d_{ox}) \tag{2.60}$$

For large values of t, i.e., $t \gg t_1$, the shape of the plot can be approximated as a rectangle in oxide, followed by a triangle in poly, since the oxide thickness is of the order of a few angstroms, and $D_{ox} > D_p$. We verify this analytically by computing $N_{H_2}^{d_{ox}}$ as a function of $N_{H_2}^0$, as follows:

The number of interface traps is equal to the integral from (2.12), which is equal to the area under the curve in Fig. 2.12(e), as follows:

$$N_{IT}(t, t > t_1) = \left[d_{ox} \left(N_{H_2}^0 + N_{H_2}^{d_{ox}} \right) + N_{H_2}^{d_{ox}} \left(x_d - d_{ox} \right) \right] \tag{2.61}$$

where $N_{H_2}^{d_{ox}}$ is the hydrogen molecular concentration at the oxide. Differentiating, with respect to time, and ignoring the $\frac{dN_{H_2}^{d_{ox}}}{dt}$ component, since $N_{H_2}^{d_{ox}}$ is a slowly decreasing function of time[5], we have:

$$\frac{dN_{IT}}{dt} \approx N_{H_2}^{d_{ox}} \frac{dx_d}{dt} \tag{2.62}$$

From (2.59) and (2.62), we have:

$$(x_d - d_{ox})dx = D_p dt \tag{2.63}$$

Integrating, and using initial conditions, i.e., $x_d(t_1) = d_{ox}$, we have:

$$x_d = d_{ox} + \sqrt{2D_p(t - t_1)} \tag{2.64}$$

We now use the diffusion equation in (2.3) to compute the value of $N_{H_2}^{d_{ox}}$ in (2.61). Along the oxide-poly interface, the outgoing flux from the oxide is equal to the incoming flux into poly. Therefore, we have:

$$\phi_{N_{H_2}}^{d_{ox}} = D_{ox} \frac{dN_{H_2}}{dx} = D_p \frac{dN_{H_2}}{dx} \tag{2.65}$$

at $x = d_{ox}$. Since, N_{H_2} is a linear function of x, we have, at the interface:

$$D_{ox} \frac{\left(N_{H_2}^0 - N_{H_2}^{d_{ox}} \right)}{d_{ox}} = D_p \frac{N_{H_2}^{d_{ox}}}{x_d - d_{ox}} \tag{2.66}$$

[5]The value of $N_{H_2}^{d_{ox}}$ and $N_{H_2}^0$ are determined by the rate of generation of interface traps at the surface (increases as $\sim t^{\frac{1}{6}}$), and the rate of diffusion of hydrogen molecules at the tip of the diffusion front (decreases as $\sim \sqrt{t}$), causing N_{H_2} to be a slowly decreasing function of time.

Substituting and simplifying, we have:

$$\begin{aligned} N_{H_2}^{d_{ox}} &= N_{H_2}^0 \left[\frac{D_{ox}\sqrt{2D_p(t-t_1)}}{D_{ox}\sqrt{2D_p(t-t_1)} + D_p d_{ox}} \right] \\ &= N_{H_2}^0 f(t) \text{ for brevity} \end{aligned} \tag{2.67}$$

It is easy to see that for $t \gg t_1$, the value of $N_{H_2}^{d_{ox}}$ almost becomes equal to $N_{H_2}^0$. The diffusion front is shown in Fig. 2.12(f) for this case. The front almost becomes a rectangle in the oxide followed by a right angled triangle in poly. Using (2.67) in (2.61), we have:

$$N_{IT}(t, t_1 < t < \tau) = \left[d_{ox} N_{H_2}^0 (1 + f(t)) + N_{H_2}^0 \sqrt{2D_p(t-t_1)} f(t) \right] \tag{2.68}$$

Lumping the terms in (2.68), we have:

$$x_{\text{equiv}}(t, t_1 < t \leq \tau) = d_{ox}(1 + f(t)) + \sqrt{2D_p(t-t_1)} f(t) \tag{2.69}$$

where x_{equiv} represents the tip of an equivalent triangular front that has the same area. This step is performed such that the expression resembles the form in (2.56). Thus, we have the final expression:

$$\begin{aligned} N_{IT}(t, 0 < t \leq t_1) &= k_{IT}(2D_{ox}t)^{\frac{1}{6}} \\ N_{IT}(t, t_1 < t \leq \tau) &= k_{IT}\left[d_{ox}(1 + f(t)) + \sqrt{2D_p(t-t_1)} f(t) \right]^{\frac{1}{3}} \end{aligned} \tag{2.70}$$

$$f(t) = \left[\frac{D_{ox}\sqrt{2D_p(t-t_1)}}{D_{ox}\sqrt{2D_p(t-t_1)} + D_p d_{ox}} \right] \approx 1 \text{ for } t > t_1 \tag{2.71}$$

where the first equation accounts for diffusion in the oxide leading to a rapid stress phase, followed by the second equation which involves diffusion in poly, and therefore, a slower stress phase.

Using the above equations, it is easy to obtain an analytical expression for the number of interface traps for the static NBTI stress case, or the DC stress case as follows:

$$\begin{aligned} N_{IT_{DC}}(t, 0 < t \leq t_1) &= k_{IT}(2D_{ox}t)^{\frac{1}{6}} \\ N_{IT_{DC}}(t, t > t_1) &= k_{IT}\left[d_{ox}(1 + f(t)) + \sqrt{2D_p(t-t_1)} f(t) \right]^{\frac{1}{3}} \end{aligned} \tag{2.72}$$

Simulation results for the DC stress case, using the above model are shown in Section 2.8 - Fig. 2.23.

2.6 Numerical Simulation for the First Stress and Recovery Phases

Before deriving an analytical model for the first recovery phase as shown in Fig. 2.1, we present a detailed numerical analysis and solution to this case. This section aims to identify the origin of the drawbacks of the recovery modeling in [13], and argues that these are not necessarily a limitation of the R-D mechanism itself, as contended in [21]. Accordingly, a modified R-D model for recovery, based on the model in [13] is developed in Section 2.7. It must be noted that numerical simulation is only used to aid the reader in understanding the development of the actual mathematical model for the recovery phase. The employment of such a numerical simulation-based model is prohibitively computationally intensive, particularly in a multicycle framework to estimate the asymptotic impact of NBTI on transistor threshold voltage after three years ($\approx 10^{17}$ cycles at a frequency of 1GHz) of operation.

We present a numerical solution framework for the R-D model equations, described in Section 2.1.1. We provide an in-depth analysis of the recovery modeling in [13], and show that the value of the back-diffusion coefficient $\xi = 0.5$, as used in [13] is not universal, and ξ is actually based on curve-fitting. We argue that the poor fit between the analytical model in [13] and measured data is partly due to the misinterpretation of the value of ξ as being universal, and not the R-D model itself.

We then explore the impact of using a two-region model considering the finite thickness of the gate-oxide, and a higher value of the diffusion constant in oxide, as compared with poly [3]. We show simulation results using this finite-oxide thickness-based model for NBTI recovery, and argue that the model further helps eliminate the previously encountered limitations in using the R-D theory based models.

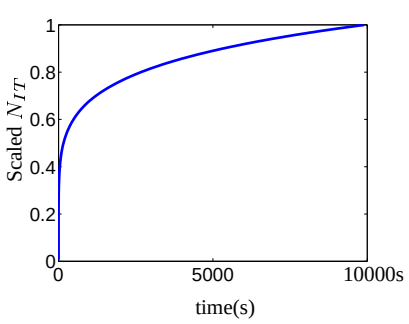
2.6.1 Simulation Setup

A backward-Euler numerical solver based on [68] is implemented with adaptive time stepping, using k_f = 4.66s^{-1}, k_r = 4.48e-9cm^{3}s^{-1}, k_H = 1.4e-3s^{-1}, N_0 = 5e12cm^2, and D_{ox} = 4e-17cm^{2}s^{-1}. It must be noted that the exact values do not influence the time-dependencies [56]. A minimum

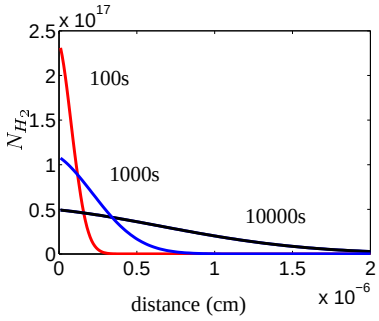
step-size of 1e-4s is used for the simulations. We assume that there is a one-one correspondence between ΔV_{th} and N_{IT} for each of the cases, and that the y-axis, which denotes the normalized N_{IT} values (marked as "Scaled N_{IT}" in the figures), may also be interpreted as the normalized V_{th} values. The results are shown in the following subsections:

2.6.2 DC Stress

We first present the simple case of applying a DC stress on the PMOS transistors for 10000s. Fig. 2.13(a) shows the growth of N_{IT} with time, while Fig. 2.13(b) shows the evolution of the diffusion front with time, for t = [100s, 1000s, 10000s]. The tip of the diffusion front grows as $\sqrt{t}$ and the peak concentration decreases, while N_{IT} increases asymptotically as $\propto t^{\frac{1}{6}}$. Both results are consistent with the findings of the analytical model, detailed in Section 2.1.4.



(a) N_{IT} for stress phase.



(b) Evolution of H_2 diffusion front at 100s, 1000s, and 10000s.

Figure 2.13: Trap generation and H_2 diffusion for DC stress.

2.6.3 Effect of Stopping Stress

Fig. 2.14 shows the evolution of the diffusion front where stress was applied until time τ = 10000s, followed by diffusion of existing hydrogen molecules for time $t > \tau$. The results show that the peak concentration of hydrogen at the interface reduces, whereas the tip of the diffusion front continues to grow as $\sqrt{t}$. The shape of the diffusion front, and the decrease in $N^0_{H_2}$ for $t > \tau$ is obvious since

there is no further generation of interface traps, and the increase in the base of the triangular front must be accompanied by a decrease in its height.

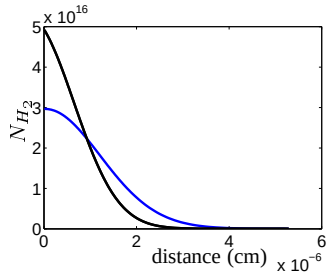


Figure 2.14: Evolution of diffusion front for 10000 seconds of stress followed by diffusion of existing species: upper curve shows the front after τ = 10000s, while the lower curve plots the case where diffusion of existing species occurs after 10000s of stress, with a lowering of the peak concentration, and widening of the tip of the diffusion front $x_d(t)$.

Recovery is modeled as a superposition of two mechanisms:

1. Continued diffusion of existing hydrogen molecules away from the interface.

2. Annealing of interface traps, and backward diffusion of hydrogen molecules near the interface.

Thus, we have:

$$N_{IT}(t, t > \tau) = N_{IT}(\tau) - N^*_{IT}(t) \tag{2.73}$$

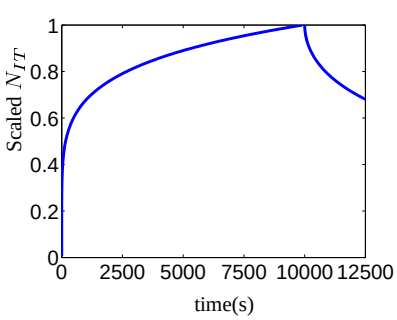
where N^*_{IT} is the annealed component.

In the absence of annealing, i.e., if k_r = 0 (along with k_f = 0) during the recovery phase, the profile of hydrogen molecular diffusion must be as shown in Fig. 2.14. Hence, the area under both curves in Fig. 2.14 is the same, and is given by:

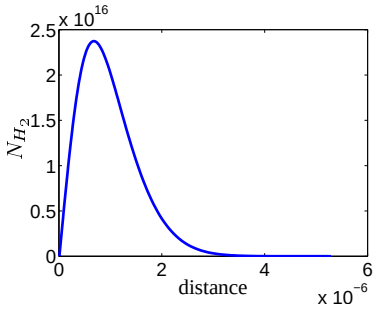
$$N_{IT}(t, t \geq \tau) \propto x_d(t) N^0_{H_2}(t) = x_d(\tau) N^0_{H_2}(\tau) \tag{2.74}$$

2.6.4 Impact of Annealing

In order to determine the impact of annealing, we first simulate the case where 10000s of stress followed by 2500s of recovery is applied to the PMOS device. Fig. 2.15(a) shows the decrease



(a) N_{IT} for stress phase.



(b) Evolution of H_2 diffusion front.

Figure 2.15: Trap generation for AC stress case: 10000s of stress followed by 2500s of recovery.

in N_{IT} beyond 10000s, with $N_{IT}(1.25\tau$ = 12500) = $0.673N_{IT}(\tau$ = 10000), whereas Fig. 2.15(b) shows the diffusion front, where there is annealing close to the interface. The peak concentration point moves away from the interface, unlike the diffusion curves in the stress phase, which resemble a right angled triangle. However, the tip of the diffusion front continues to grow further into the oxide.

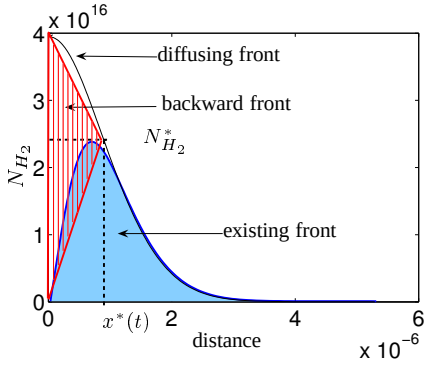


Figure 2.16: Diffusion fronts during recovery.

Fig. 2.16 shows the diffusion front after 2500s of recovery (Fig. 2.15(b)), superimposed on the diffusion front for the case where the device is stressed for 10000s, followed by continued diffusion (without annealing) for the remaining 2500s, as explained in Section 2.6.3. The area under the black curve, denoted as **diffusing front** represents $N_{IT}(\tau)$, as explained in (2.74), whereas the area under

the shaded curve (in blue) is $N_{IT}(t > \tau)$, and is denoted as the **existing front**. In Fig. 2.16, the region under the triangular shape filled with (red) vertical lines, denoted as **backward front** indicates the number of interface traps annealed, given by N^*_{IT}. Assuming that all fronts are triangular, which is reasonably accurate based on Fig. 2.16, we can write[6]:

$$\begin{aligned} N_{IT}(\tau) &= N_{H_2}(x=0,t,t>\tau)\sqrt{2D(t+\tau)} \\ N^*_{IT}(t) &= N_{H_2}(x=0,t,t>\tau)x^*(t) \\ N_{IT}(t,t>\tau) &= N_{H_2}(x^*(t),t)\sqrt{2D(t+\tau)} \\ N_{IT}(t,t>\tau) &= N_{IT}(\tau) - N^*_{IT}(t) \end{aligned} \tag{2.75}$$

where $x^*(t)$ is the point at which the diffusion front during the recovery phase reaches its peak. Unlike the figures in Fig. 2.3, where we assume that the peak value occurs at $\Delta \approx 0$, i.e., close to the Si-SiO_2 interface, $x^*(t)$ grows with time, i.e., the peak point moves away from the interface, due to forward diffusion of existing hydrogen species.

Fig. 2.17(a) shows the case for τ seconds of stress followed by τ seconds of recovery, where τ = 10000s (as this case is widely used to compare the performance of an analytical model, as well as to demonstrate experimental results). The shape of the fronts indicate that the number of interface traps can be expressed as a difference in the area of the two triangles between the diffusing front, and the backward front, as shown in Fig. 2.16, and derived in (2.75). We now derive the analytical modeling in [13] using (2.75).

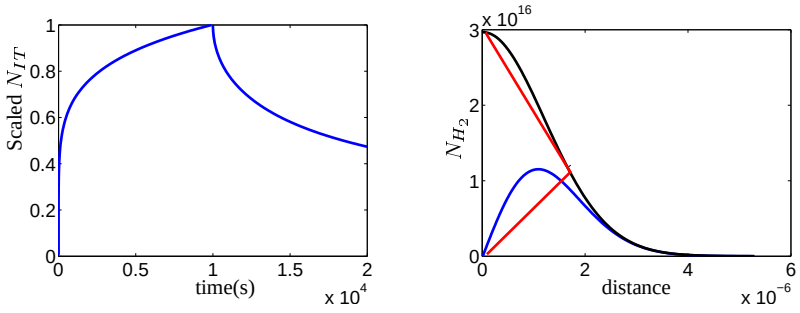
Numerical simulations plotted in Fig. 2.17(a), for this case show that:

$$N_{IT}(2\tau) = 0.47 N_{IT}(\tau) \tag{2.76}$$

From Figs. 2.15(b) and 2.17(b), we can see that $x^*(t) \propto \sqrt{t}$, and can be written as:

$$x^*(t) = \sqrt{\xi \times 2Dt} \tag{2.77}$$

[6]Number of interface traps N_{IT} is equal to twice the area under the N_{H_2} curve, from (2.12).



(a) N_{IT} for stress-relaxation phases.

(b) Evolution of H_2 diffusion front.

Figure 2.17: Trap generation for AC stress case - 10000s of stress followed by 10000s of recovery.

where ξ is the curve-fitting parameter whose value must be determined. Using the above relation in (2.75), we have:

$$\begin{aligned} N_{IT}(\tau) &= N_{H_2}(x=0,t,t>\tau)\sqrt{2D(t+\tau)} \\ N^*_{IT}(t) &= N_{H_2}(x=0,t,t>\tau)\sqrt{2\xi Dt} \\ N_{IT}(t,t>\tau) &= N_{IT}(\tau) - N^*_{IT}(t) \\ &= N_{IT}(\tau) - \frac{N_{IT}(\tau)\sqrt{2\xi Dt}}{\sqrt{2D(t+\tau)}} \\ &= N_{IT}(\tau)\left[1-\sqrt{\frac{\xi t}{t+\tau}}\right] \end{aligned} \tag{2.78}$$

which is the equation for recovery in [13]. Substituting the value of $N_{IT}(2\tau)$ from (2.76) in (2.78), we have:

$$0.47N_{IT}(\tau) = N_{IT}(\tau)\left[1-\sqrt{\frac{\xi\tau}{\tau+\tau}}\right] \tag{2.79}$$

from which, we obtain ξ = 0.58, which is the theoretical value of ξ for double sided diffusion, as stated in [13]. However, for simplicity, a fixed value of ξ = 0.5 is used, which results in $N_{IT}(2\tau) = 0.5N_{IT}(\tau)$.

We now compare the values of the analytical model for recovery using (2.78) and the results from numerical simulations, for different values of t, with a fixed value of ξ = 0.58. Table 2.1 shows

the values of $\frac{N_{IT}(t+\tau)}{N_{IT}(\tau)}$, i.e., the fractional recovery numbers during the relaxation phase, computed using numerical simulations, and using the analytical model from (2.78) with ξ = 0.58, for different values of t, where τ = 10000s. The last column of Table 2.1 recomputes ξ from (2.78) by substituting

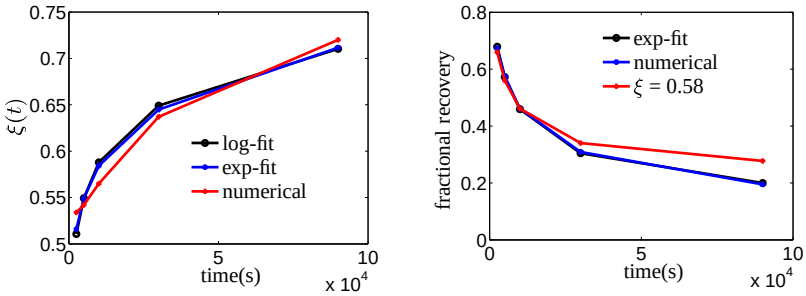
Table 2.1: Comparison between fractional recovery numbers obtained through numerical simulations and analytical model.

time (t)	analytical	numerical	new value of ξ
2500	0.659	0.673	0.534
5000	0.560	0.575	0.542
10000	0.468	0.468	0.580
30000	0.340	0.309	0.637

the value of $N_{IT}(t+\tau)$ for each case. The results show that ξ is not a constant, and increases with t. However, for $t < \tau$, the difference between numerical and analytical results using ξ = 0.58 is not large. Thus, the discrepancy between numerical simulation results and analytical modeling for the recovery phase, for large values of t, is clearly attributed to the use of a fixed value of ξ, based on curve fitting at one time stamp $t = \tau$. This discrepancy can be resolved by using a curve-fitted expression for ξ, as shown in Fig. 2.18. Two sample curve fitted expressions and their accuracies are shown in Fig. 2.18(a), while the corrected model is plotted in Fig. 2.18(b) along with numerical data, as well as the case where ξ = 0.58 is used. The results indicate that with a time varying ξ, a good fit between numerical and analytical results can be obtained. Such a modified analytical solution from an R-D theory based on [13], with a time varying ξ does indeed converge well with numerical simulation results. It must be noted that the curve fitted expression for ξ in Fig. 2.18 is one of many choices, and is merely shown to illustrate the usage of a time-varying model for ξ[7].

Simulation results also show that the value of ξ depends on τ, as well, particularly for smaller values of τ. Hence, any comparison of recovery models with the R-D theory based analytical model expression of [13] must be performed using the appropriate value of ξ.

[7]Both the curve-fitted expressions in Fig. 2.18 do not guarantee that ξ converges to 1, as $t \rightarrow \infty$, and may require to be further modified for the case of a single stress phase followed by recovery of the device for infinite time, thereby resetting it to be equivalent to an original unstressed device. However, these expressions are merely shown to illustrate the fact that ξ is a function of t, and is not a constant.



(a) Curve-fit for ξ.

(b) Fractional recovery using time-varying curve-fitted ξ.

Figure 2.18: Curve-fitted expressions for time varying ξ using an exponential relation ($\xi = 0.5843\left(\frac{t}{\tau}\right)^{.0897}$) and a log relation ($\xi = .0557log\left(\frac{t}{\tau}\right) + 0.5879$).

2.6.5 Finite Oxide Thickness

In this section, we propose to account for further discrepancies between the findings from a numerical or an analytical model and experimental data, such as:

1. Experimental results for a single stress phase followed by a single recovery phase show more than 80% recovery in [5] for τ = 1000s, around 60% recovery in [3] for τ = 10000s, and 50% recovery in [13] for τ = 1000s, for devices with an oxide thickness of 1.2nm-1.3nm.

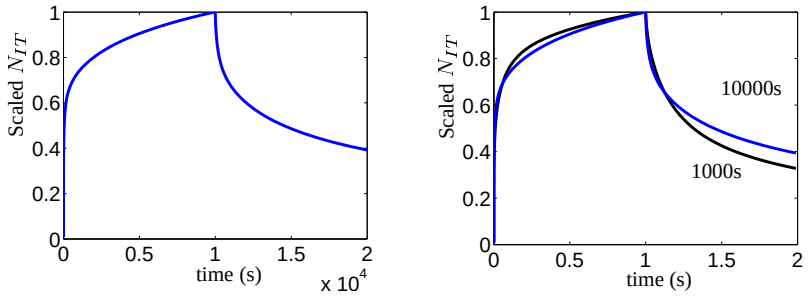
2. Larger fractional recovery for the same value of τ for a higher oxide thickness is seen in [3].

3. Rapid decrease in V_{th} at the beginning of the recovery phase [5], implying a $logt$ behavior for recovery, where equal recovery is observed in every decade [21–23][8].

Accordingly, a finite oxide thickness-based two-step model is contended since the diffusion constant of hydrogen in oxide is larger than that in polysilicon ($D_{ox} > D_p$). Although the exact values of D_{ox} and D_p are still widely debated [3], their relative ratio influences the shape of the N_{IT} curve. We perform numerical simulations, using our setup, as described in Section 2.6.1 for a case where d_{ox} = 1.3nm. Additional boundary conditions at the oxide-poly interface are added to the

[8]It must be noted that [66] has attributed this behavior to an inaccurate way of estimating the impact of NBTI by using UFM techniques.

numerical simulation setup used for the infinitely thick oxide case, in Section 2.6.1. D_p is assumed to be $0.25D_{ox}$. Fig. 2.19(a) which plots the simulation results shows that there is approximately 60% recovery after τ seconds of recovery for τ = 10000s, as opposed to Fig. 2.17(a) which shows 50% recovery.



(a) Finite oxide thickness-based model.

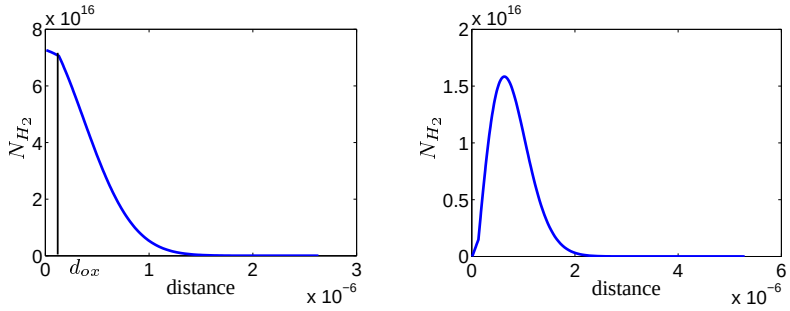
(b) Fractional recovery for different values of τ.

Figure 2.19: Validation of finite oxide thickness-based model.

Fig. 2.19(b) shows the model for the case of τ = 10000s, and τ = 1000s, with higher fractional recovery for the 1000s case, since more H_2 is contained in the oxide, and rapidly diffuses back to the interface. Unlike the infinite oxide thickness case, which would have incorrectly predicted a fractional recovery of ≈50% for both τ = 10000s, and τ = 1000s, higher fractional recovery is seen with lower values of τ. The shape of the diffusion profile at the end of the first stress and recovery phases for the case of τ = 10000s, and D_{ox} = 4 D_p are shown in Fig. 2.20. Fig. 2.20(a) shows the diffusion of N_{H_2} at the end of the stress phase, with the rectangular shaped front in the oxide, followed by a triangular front in poly. The diffusion profile for recovery in Fig. 2.20(b) indicates that the fraction of the hydrogen molecules contained in the oxide quickly diffuses backwards during recovery.

Thus, it is clear that a two-region based model for recovery with differing diffusion constants for oxide and poly is necessary to model the recovery phase of NBTI action. Accordingly, we also use two curve fitting constants ξ_1 and ξ_2, for the backward diffusing fronts in oxide and poly, respectively, and determine the values of these constants to match the experimental results. The

development of the analytical model for recovery is detailed in the next section.



(a) Diffusion front for stress phase.

(b) Diffusion front at the end of the recovery phase.

Figure 2.20: Diffusion front considering finite oxide thickness.

2.7 Model for the First Recovery Phase

During the recovery phase, the stress applied to the PMOS device is released, as shown in Fig. 2.1. Some of the hydrogen molecules recombine with Si^+ species, to form Si-H bonds, thereby annealing some of the existing traps. Since the rate of diffusion of hydrogen molecules in the oxide is greater than that in poly, rapid annealing of traps occurs in the oxide, followed by a slow annealing in polysilicon. Accordingly, we have two stages of recovery in each relaxation phase, that are modeled separately:

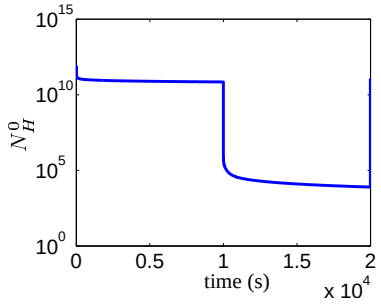


Figure 2.21: Hydrogen concentration at the oxide-substrate interface during the first stress and recovery phases, showing the rapid decrease in N_H^0 at the beginning of the recovery phase.

2.7.1 Recovery in Oxide

Recovery in oxide consists of two sub-phases, namely, a reaction phase and a diffusion phase. During the reaction phase, we have from (2.2):

$$\frac{dN_{IT}}{dt} = -k_r N_{IT} N_H^0 \tag{2.80}$$

where k_f is zero since there is no trap generation. The hydrogen concentration decreases exponentially during the beginning of the recovery phase, as shown in Fig. 2.21. A decrease in the concentration of interface traps occurs during this process. However, the reaction phase lasts only a few milliseconds, as seen from the simulation results. As the hydrogen concentration remains almost constant, diffusion becomes the dominant physical mechanism. During this diffusion phase, annealing of interface traps near the interface, followed by back-diffusion of existing hydrogen molecular species in the oxide occurs. For simplicity in modeling, we combine the reaction phase and the diffusion phase into a single stage of modeling as follows:

We model the rapid annealing of interface traps inside the oxide, which occurs from time τ to $\tau + t_2$, where t_2 is the time at which annealing proceeds into poly. Let us model the events at the interface as a superposition of two effects: "forward" diffusion, away from the interface, and "reverse" diffusion, toward the interface; the latter anneals the interface traps, as explained in Section 2.6.4. During this condition, the diffusion of existing species continues as $x(t+\tau) \propto \sqrt{2D_p(t+\tau)}$ inside poly, while the peak of the diffusion front decreases from $N_{H_2}^0$ to $N_{H_2}^{\Delta}$, as shown in Fig. 2.22(c), for some $\Delta \leq d_{ox}$.

We may approximate the hydrogen concentration in the oxide as being a triangle plus a quadrilateral: at time $(\tau + t)$, it goes from 0 at $x = 0$, to $N_{H_2}^{\Delta}(\tau + t)$ at $x = \Delta$, for some $\Delta \leq d_{ox}$. The hydrogen molecular concentration follows a right angled triangle profile in poly, since there is no effect of annealing here yet, with the concentration being $N_{H_2}^{d_{ox}} \approx N_{H_2}^{\Delta}(\tau + t)$ at the oxide-poly interface, and decreasing to 0 again at $x_d(\tau + t)$. During this phase, the rate of decrease of interface traps can be assumed to be low, and is hence approximated as 0. Using the same notation as [13], page 3, we have:

$$\frac{dN_{IT}}{dt} \approx 0 = -k_r (N_{IT}^0 - N_{IT}^*)(N_H^0 - N_H^*) \tag{2.81}$$

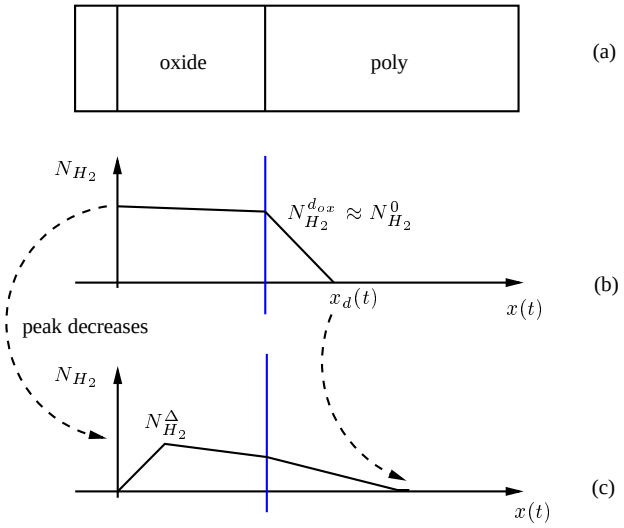


Figure 2.22: Diffusion front for the first recovery phase: (a) shows the cross section of the PMOS transistor, (b) shows the front at time τ, i.e., at the end of the first stress phase, while (c) shows the front at time $\tau + t$, into the first recovery phase.

Since the residual number of interface traps, $(N_{IT}^0 - N_{IT}^*)$, is significantly larger than zero, it must mean that the residual hydrogen concentration at the interface, $(N_H^0 - N_H^*)$ must be near-zero. Denoting the number of annealed traps as $N_{IT}^*(\tau + t)$, we can express the net number of interface traps during the relaxation phase as the original number of traps, minus the number of annealed traps:

$$N_{IT}(\tau + t) = N_{IT}(\tau) - N_{IT}^*(\tau + t) \tag{2.82}$$

The number of interface traps annealed due to backward diffusion [13] can be expressed as:

$$N_{IT}^*(\tau + t) = N_{H_2}^{\Delta} \sqrt{\xi_1 \times 2D_{ox}t} \tag{2.83}$$

Intuitively, this can be considered to be equivalent to a triangle whose height is given by $N_{H_2}^{\Delta}$, and the backward diffusion front beginning at time τ is given from [13] as:

$$x^*(t) = \sqrt{2\xi_1 D_{ox}t} \tag{2.84}$$

ξ_1 is a parameter that captures the effect of two-sided diffusion, and its original value is of the order

of ≈ 0.58 [13]. However, in order to account for the exponential decrease in the interface trap concentration during the reaction phase of the first recovery phase, using a single analytical model, ξ_1 is set to a large number, and its exact value is determined through curve fitting.

Based on the argument in the previous section, the total number of interface traps is given by the area enclosed under the quadrilateral plus the triangles in Fig. 2.22(c) as:

$$N_{IT}(t+\tau) \approx N_{H_2}^{\Delta}\left(2d_{ox} - \Delta + \sqrt{2D_p(t+\tau)}\right) \tag{2.85}$$

where Δ, i.e., the location of the peak concentration of hydrogen molecules during recovery (follows the dynamics of the diffusion front for stress phase, and hence from (2.16)) increases with time as:

$$\Delta = \sqrt{2D_{ox}t} \tag{2.86}$$

The tip of the diffusion front $x_d(t)$, computed from (2.69), is approximately at:

$$x_d(t) = d_{ox} + \sqrt{2D_p(t+\tau)} \tag{2.87}$$

Solving for $N_{H_2}^{\Delta}$ in (2.85), we have:

$$N_{H_2}^{*} = \frac{N_{IT}(t+\tau)}{2d_{ox} - \sqrt{2D_{ox}t} + \sqrt{2D_p(t+\tau)}} \tag{2.88}$$

Since, the number of interface traps is given by the difference between the number of traps at τ, and the number of traps annealed, we have:

$$N_{IT}(t+\tau) = N_{IT}(\tau) - N_{IT}^{*}(t+\tau) \tag{2.89}$$

Substituting for $N_{IT}^{*}(t+\tau)$, and simplifying, we have:

$$N_{IT}(t+\tau) = N_{IT}(\tau) - N_{IT}(t+\tau)g(t) \tag{2.90}$$

$$\text{where for brevity, } g(t) = \left[\frac{\sqrt{2\xi_1 D_{ox}t}}{2d_{ox} - \sqrt{2D_{ox}t} + \sqrt{2D_p(t+\tau)}}\right] \tag{2.91}$$

Simplifying, we have:

$$N_{IT}(t+\tau, 0 < t \leq t_2) = \frac{N_{IT}(\tau)}{1+g(t)} \tag{2.92}$$

This process continues until time t_2, when the back-diffusion front has reached the oxide-poly interface.

2.7.2 Slow Recovery in Poly

If recovery continues beyond time t_2, the back-diffusion front now enters poly, where its growth is slower, in comparison with that in the oxide ($\equiv$ to the diffusion front during the first stress phase in Fig. 2.3). Hence, during this phase, the rate of annealing of interface traps reduces. However, by this time, since the oxide is almost completely annealed, only a slow recovery in poly occurs. The diffusion front in poly is triangular, and its peak moves further away from the oxide-poly interface as being proportional to $\sqrt{\xi_2 t}$ where ξ_2 is the curve fitting parameter. The mechanism is similar to recovery for the case of an infinitely thick oxide. Hence, the model derived in Section 2.6.4 for the infinite oxide case, can be used here. Thus, we have:

$$N_{IT}(t+\tau, t > t_2) = N_{IT}(\tau + t_2)\left[1 - \sqrt{\frac{\xi_2(t-t_2)}{t+\tau}}\right] \tag{2.93}$$

for time $\tau + t_2$ to 2τ, where ξ_2 is the curve fitting factor. It must be noted that due to the difference in the coefficients of oxide and poly, and the slow progression of the back-diffusion front in poly, the value of ξ is less than 0.58, and is of the order of around 0.125 for $t < t_0$[9]. Thus, the two-step model for annealing consists of a quick annealing stage where the number of interface traps decreases rapidly in the first few milliseconds to about a second, followed by a slow decrease over the remaining time period.

The model proposed can thus also account for rapid recovery during the beginning of the relaxation stage, due to mechanisms not attributed to a reaction-diffusion process, using the curve fitted value of ξ_1. The authors in [66] argue that the rapid decrease in V_{th} at the beginning of the recovery phase, that does not correspond to a simultaneous decrease in N_{IT}, is an incorrect manifestation of

[9]A time varying ξ_2, as deemed necessary in Section 2.6.4 is used to model the impact of a single stress phase, followed by long periods of recovery, in the plots (Fig. 2.27) shown later on, in Section 2.8.4.

the UFV technique used to measure recovery in PMOS devices. While it is not clear what the actual physical mechanism is, in nanometer scale PMOS devices during actual circuit operation, the use of a curve-fitted ξ_1 helps fit better the results of the model with experimental data, while still adhering to the basic guidelines of the R-D theory.

2.7.3 Complete Set of Equations for First Stress and Relaxation Phase

The equations for the first stress and relaxation phase can be summarized as follows:

$$\begin{aligned} N_{IT}(t, 0 < t \leq t_1) &= k_{IT}(2D_{ox}t)^{\frac{1}{6}} \\ N_{IT}(t, t_1 < t \leq \tau) &= k_{IT}\left[d_{ox}(1+f(t)) + \sqrt{2D_p(t-t_1)}f(t)\right]^{\frac{1}{3}} \\ N_{IT}(t+\tau, 0 < t \leq t_2) &= \frac{N_{IT}(\tau)}{1+g(t)} \\ N_{IT}(t+\tau, t_2 < t \leq \tau) &= N_{IT}(\tau+t_2)\left[1 - \sqrt{\frac{\xi_2(t-t_2)}{t+\tau}}\right] \end{aligned} \tag{2.94}$$

2.8 Simulation Results and Comparison with Experimental Data

In this section, we compare the results of our model with the requirements outlined in Section 2.4.2.

2.8.1 DC Stress

The plot for a DC stress case, for a PMOS transistor with d_{ox} = 1.2nm is obtained using (2.72), and is shown in Fig. 2.23. The plot consists of three significant phases:

1. The initial phase of $t <$ 0.1s, during which the reaction phase is dominant. It must be noted that this phase has been not been explicitly modeled in (2.94), and (2.56) is used for $t \geq 0$, as has been explained in the end of Section 2.1.4, using Fig. 2.2.

2. The transient phase of 0.1s $\leq t <$ 10s, during which the process is dominated by diffusion in the oxide.

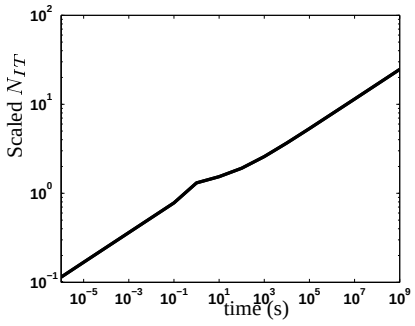
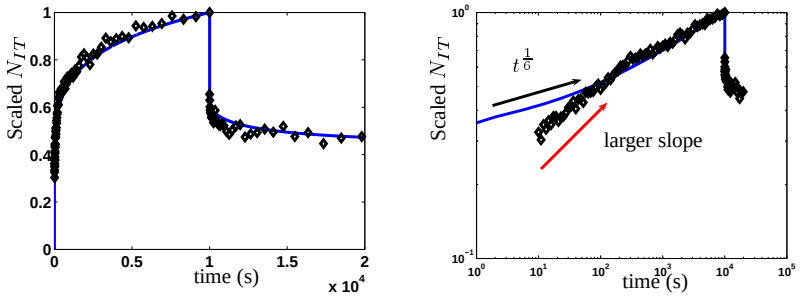


Figure 2.23: Plot of DC stress for d_{ox} = 1.2nm. The curve plots the normalized interface trap values for k_{IT} = 1.

3. The final phase, for large values of t, over which the mechanism is dominated by diffusion in poly.

It follows from the shape of the log-log plot in Fig. 2.23, that as t increases, the number of interface traps asymptotically approaches a $t^{\frac{1}{6}}$ relationship, which satisfies the first guideline outlined at the end of Section 2.4.2. It must be noted that in the analytical model for DC stress, and hence the plots in Fig. 2.23, we ignore the reaction and the quasi equilibrium phases of interface trap generation, for reasons already explained in Section 2.1.2.



(a) Linear Scale (b) Log Scale

Figure 2.24: Plot of first stress and recovery phases for τ = 10000s, and d_{ox} = 1.2nm, with experimental data from [3, 4], shown in $\diamond$.

2.8.2 AC Stress

The plot in Fig. 2.24(a) shows the simulation results for the number of interface traps generated for a single stress phase, followed by a relaxation phase, each of duration τ = 10000s, using (2.94), for a PMOS device whose oxide thickness (d_{ox}) is 1.2nm. These match the values used in the experimental setup from [3]. The values of ξ_1 and ξ_2 are chosen based on curve fitting, with $\xi_1 >> \xi_2$. The results of our simulation are shown in Fig. 2.24(a). The curve shows a good fit with experimental data from [3, 4]. The accurate fit with experimental data, particularly during the recovery phase, satisfies the second requirement outlined in Section 2.4.2.

Recent publications [22,23] have motivated the plotting of stress and relaxation data, on a semi-log scale, to compare the accuracy of the fit, over the broad spectrum of time constants. The fit with experimental data from [3] on a semi-log scale is shown in Fig. 2.24(b). The fit for our model is not very accurate, during the beginning of the stress phase, and our model shows a higher exponent as opposed to experimental data. Recently published works [32, 33, 56] have shown that this is nevertheless consistent with a H_2 diffusion based R-D model, and attribute this discrepancy in short-term measurements to the assumption that H-to-H_2 conversion is extremely fast, which may not be realistic [56]. A detailed analysis of the H$\leftrightarrow$$H_2$ conversion has been incorporated into an analytical model recently by [32], and the fit of the model with the experimental data indeed verifies that this is true. The shape of the plots from [56] are similar to that shown in Fig. 2.24(b), with measurement data showing an initial slope of $t^{\frac{1}{3}}$, whereas the R-D model solution using only H_2 diffusion predicts a $t^{\frac{1}{6}}$ behavior. However, for the purposes of circuit delay degradation estimation and optimization, we are more concerned about long term effects of aging after a few years of circuit operation under various conditions, rather than actual cycle accurate values. In this context, the accuracy of the plot toward the end of the stress phase, and the asymptotic fit is more important, since this governs the shape of the next recovery phase, and the subsequent stress phases.

2.8.3 Effect of Thicker Oxides

Experimental results have shown that as the oxide thickness increases, greater amount of recovery is expected. We verify this by simulating the case of d_{ox} = 2.2nm, and τ = 10000s. While the d_{ox} = 1.2nm case showed ≈60% recovery after τ seconds of relaxation, we expect a higher fractional

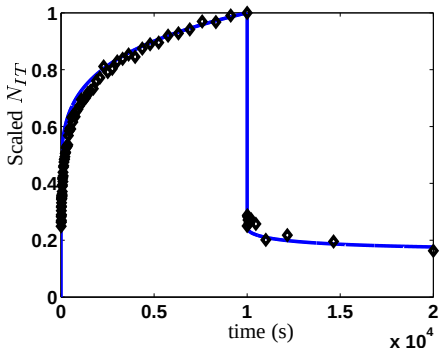


Figure 2.25: Plot of first stress and recovery phases for τ = 10000s, and d_{ox} = 2.2nm, with experimental data from [3, 4], shown in $\diamond$, on a linear scale.

recovery for this case, since more N_{H_2} is contained in the oxide, and hence diffuses back faster. The results are shown in Fig. 2.25, and expectedly there is 80% recovery after τ seconds of relaxation. The results match well with experimental data from [3], thereby satisfying the third requirement in Section 2.4.2.

2.8.4 Effect of Lower Stress Times on the Amount of Recovery

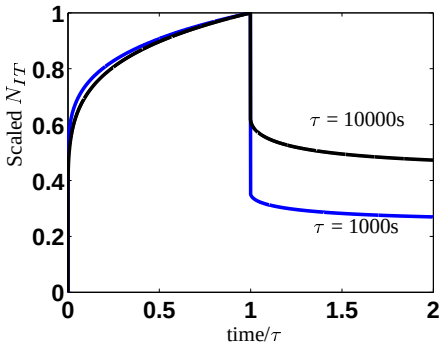


Figure 2.26: Plot of the first stress and recovery phase for τ = 10000s, and τ = 1000s, showing the effect of reduced stress times.

Previous solutions to the R-D model ignored the effect of finite oxide thickness, and the difference in the diffusion rates in polysilicon and the oxide. Hence, these results always showed 50% recovery, when the ratio of recovery time to stress time was one, independent of the stress time.

However, experimental results [5] show that a higher fractional V_{th} recovery is observed for lower stress times. We verify this by plotting the results for the case of d_{ox} = 1.2nm, with stress times of 10000s and 1000s, respectively, in Fig. 2.26. Further, we also use compare the results of our model with experimental data from [5], for the case of a single stress phase followed by variable amounts of recovery, for different values of τ. Fig. 2.27 shows the case where a single stress phase was followed by 100 seconds of recovery for four cases of stress times: 1000s, 100s, 10s, and 1s, respectively, for a 1.3nm oxide case.

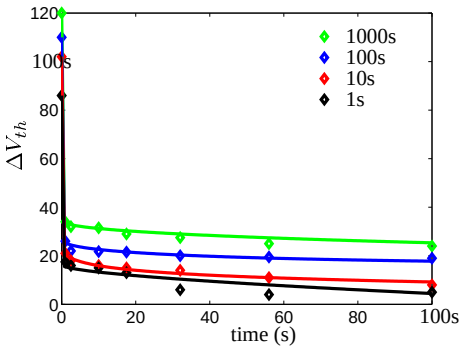


Figure 2.27: Experimental data from [5] compared with model results to demonstrate the effect of reducing τ.

Fig. 2.27 indicates that our two-stage model for recovery with two sets of curve fitted constants ξ_1 and ξ_2 provide a reasonably accurate fit with the experimental results. Some key findings are:

1. The plots in [69] shows results where a 1000s of NBTI stress on a PMOS device with an oxide thickness of 1.3nm causes a threshold voltage shift of 30mV. Subsequent recovery causes an approximate 50% reduction in the amount of V_{th} degradation. However, the results in [5] show approximately 120mV increase in V_{th} with 1000s of stress, and a large amount of recovery as well, after 100s of relaxation.

2. The curve fitted value of ξ_1 is largest for the case of 1000s of stress, and decreases with a reduction in the value of τ.

3. A single value of ξ_2 suffices for the τ = 1000s and τ = 100s cases of stress, followed by 100s of recovery, (since t = 100s is $\leq \tau$ for these two cases). However, a curve fitted expression

for ξ_2 of the form $\xi_{2_0}(\frac{t}{\tau})^{\alpha}$ is used for the cases of 10s and 1s of stress followed by continuous recovery for 100s, since $t >> \tau$.

4. The value of ξ_2 decreases with τ, as well. This can be explained as follows. For the case of 1000s of stress, the tip of the diffusion front is well into the polysilicon region, implying that the base of the triangular diffusion front is large, and its height relatively narrower. Hence, with 100s of recovery, the back diffusion front moves deeper into the poly region with its narrower height - as compared with the 100s case, implying a larger ξ_2 for a larger τ.

Thus, our model satisfies the guidelines outlined in Section 2.4.2 (the last observation about frequency independence is deferred to Section 2.9.4), and provides reasonably accurate fits with experimental data. We now present the extension of our single cycle model, to a multicycle operation, i.e., we calculate the number of interface traps for any k^{th} stress or relaxation phase, assuming the input pattern in Fig. 2.1.

2.9 Extension for Multicycle and High-frequency Operation

2.9.1 Second Stress and Relaxation Phase

For the second stress phase, we use boundary conditions at time 2τ, to determine the tip of the effective diffusion front. We solve for $x_{\text{eff}}(2\tau)$ by assuming an equivalent front which has diffused from time 0 to 2τ, and has the same interface trap concentration as $N_{IT}(2\tau)$:

$$N_{IT}(2\tau) = k_{IT} x_{\text{eff}}(2\tau)^{\frac{1}{3}} \tag{2.95}$$

The integral for $x_d(t)$ from (2.15) is now solved with the limits modified, to obtain:

$$x_d(t) = \sqrt{x_{\text{eff}}(2\tau)^2 + 2D_{ox}t} \tag{2.96}$$

instead of (2.16). This equation can be used in (2.56) to estimate the rapid increase in interface traps due to diffusion inside the oxide for the second stress phase as follows:

$$N_{IT}(t + 2\tau, 0 < t \leq t_1) = k_{IT}[2D_{ox}t + x_{\text{eff}}(2\tau)^2]^{\frac{1}{6}} \tag{2.97}$$

This process continues until time t_1, beyond which diffusion occurs in poly. Diffusion inside poly can be computed using the method outlined in the previous section, and the number of interface traps is approximated as:

$$N_{IT}(t+2\tau, t_1 < t \leq \tau) = k_{IT}\left[\sqrt{((1+f(t))d_{ox})^2 + x_{\text{eff}}(2\tau)^2} + f(t)\sqrt{2D_p(t-t_1)}\right]^{\frac{1}{3}} \quad (2.98)$$

for time 2τ + t_1 to 3τ. For large values of t, $f(t) \approx 1$. Hence, we can approximate the above expression as:

$$N_{IT}(t+2\tau, t_1 < t \leq \tau) \quad = \quad k_{IT}\left[\sqrt{2{d_{ox}}^2 + x_{\text{eff}}(2\tau)^2} + \sqrt{2D_p(t-t_1)}\right]^{\frac{1}{3}} \quad (2.99)$$

As a sanity check, setting $x_{\text{eff}}(2\tau)$ in (2.97), we obtain:

$$N_{IT}(t) = k_{IT}[2D_{ox}t]^{\frac{1}{6}} \quad (2.100)$$

which is the equation for the interface trap generation inside the oxide for the first stress phase, from (2.18). Similarly, setting $t = t_1$ and therefore $f(t) = 0$ in (2.98), we have:

$$\begin{aligned} N_{IT}(t_1+2\tau) &= k_{IT}[d_{ox}^2 + x_{\text{eff}}(2\tau)^2]^{\frac{1}{6}} \\ &= k_{IT}[2D_{ox}t + x_{\text{eff}}(2\tau)^2]^{\frac{1}{6}} \end{aligned} \quad (2.101)$$

which is the equation for interface trap generation inside the oxide during the stress phase, from (2.97).

Recovery modeling for the second relaxation phase is similar to that in the first relaxation phase. We assume that by this time, the diffusion front has recovered to its original shape of almost a rectangle in the oxide, followed by a triangle in poly (Fig. 2.3(f)). The above assumption has been verified through numerical simulations to be valid for large values of τ >1s. Accordingly, the front for the second recovery phase is similar to that in (2.87) and (2.69), and is given by:

$$x_d(3\tau) \approx d_{ox} + \sqrt{2D_p(3\tau)} \quad (2.102)$$

During the second recovery phase, the tip of the existing front is away from the interface, and hence grows as:

$$x_d(t+3\tau) = d_{ox} + \sqrt{2D_p(3\tau+t)} \tag{2.103}$$

However, rapid annealing occurs near the interface, causing a decrease in the number of interface traps. Accordingly, we have the equation:

$$N_{IT}(t+3\tau, 0 < t \leq t_2) = \frac{N_{IT}(3\tau)}{1+g'(t)} \tag{2.104}$$

$$\text{where } g'(t) = \left[\frac{\xi_1\sqrt{2D_{ox}t}}{2d_{ox} - \sqrt{2D_{ox}t} + \sqrt{2D_p(t+3\tau)}}\right] \tag{2.105}$$

for time 3τ to $3\tau + t_2$, which is similar to the expression for the first recovery phase, in the oxide, given by (2.92). Modeling for the slow recovery phase is similar to that derived in the previous section, and the final expression is given by:

$$N_{IT}(t+3\tau, t_2 < t \leq \tau) = N_{IT}(3\tau+t_2)\left[1 - \sqrt{\frac{\xi_2(t-t_2)}{t+3\tau}}\right] \tag{2.106}$$

for time $3\tau + t_1$ to 4τ. The plot for the first two stress and relaxation phases is shown in Fig. 2.28.

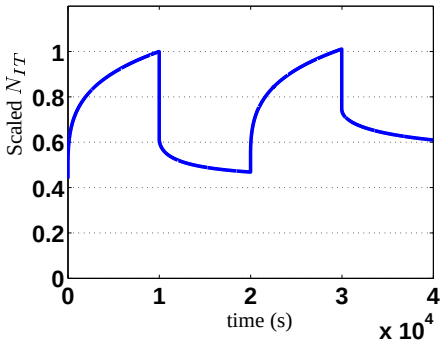


Figure 2.28: Plot of the first two stress and recovery phases for τ = 10000s, and d_{ox} = 1.2nm.

The figure shows the number of interface traps rapidly increasing during the beginning of the second stress phase, because of rapid dissociation of the Si-H bonds, which is consistent with the results in [13]. Recovery during the second relaxation phase is expectedly less than that during the

first relaxation phase, since the peak concentration has now decreased, due to further diffusion of hydrogen molecules into the poly region.

2.9.2 Comparison with Experimental Results

We also compare the results of our multicycle model with some published experimental results. Fig. 2.9.2 shows the model results for the first stress phase, first recovery phase, as well as the second stress phase for a 1.3nm oxide thickness case, and τ = 1000s. Experimental results from [69] for this case indicate a 50% recovery after τ seconds of relaxation. Fig. 2.9.2 shows that the fit is reasonably accurate.

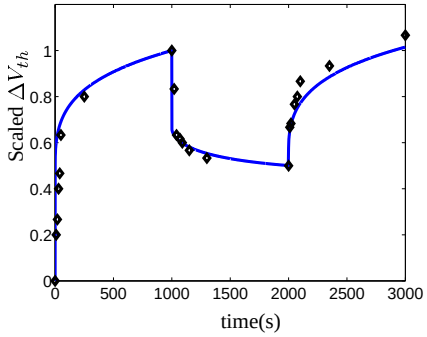


Figure 2.29: Comparison of experimental data and model results for subsequent stress and relaxation phases.

2.9.3 Final Simplified Model and Range of Operation

For a multicycle periodic operation, where an AC stress is applied on the PMOS device, with stress time being the same as relaxation time, both being equal to τ, as shown in Fig. 2.1, we obtain the following expressions for the $(n+1)^{th}$ cycle, consisting of stress from time $2n\tau$ to $(2n+1)\tau$, and relaxation from time $(2n+1)\tau$ to $2(n+1)\tau$, respectively:

Stress Phase:

$$
\begin{aligned}
N_{IT}(2n\tau + t, 0 < t \leq t_1) &= k_{IT}\left[\left(\frac{N_{IT}(2n\tau)}{k_{IT}}\right)^6 + 2D_{ox}t\right]^{\frac{1}{6}} \\
N_{IT}(2n\tau + t, t_1 < t \leq \tau) &= k_{IT}\left[\sqrt{\left(\frac{N_{IT}(2n\tau)}{k_{IT}}\right)^6 + (2d_{ox})^2} + \sqrt{2D_p(t-t_1)}\right]^{\frac{1}{3}}
\end{aligned} \tag{2.107}
$$

Relaxation Phase:

$$
\begin{aligned}
N_{IT}((2n+1)\tau + t, 0 < t \leq t_2) &= \frac{N_{IT}((2n+1)\tau)}{1 + h_1(t)} \\
N_{IT}((2n+1)\tau + t, t_2 < t \leq \tau) &= N_{IT}((2n+1)\tau + t_2)\,[1 - h_2(t)]
\end{aligned}
$$

$$
\begin{aligned}
\text{where } h_1(t) &= \left[\frac{\sqrt{\xi_1 \times 2D_{ox}t}}{2d_{ox} - \sqrt{2D_{ox}t} + \sqrt{2D_p(t + (2n+1)\tau)}}\right] \\
h_2(t) &= \left[\sqrt{\frac{\xi_2(t - t_2)}{t + (2n+1)\tau}}\right]
\end{aligned} \tag{2.108}
$$

The above model is valid for $\tau > t_1$ and $\tau > t_2$, i.e., for $\tau >$ 1s. Simulation results using this model for τ = 10000s, for 10 years of operation are shown in Fig. 2.30. The results show that the number of traps produced by AC stress is about 0.7 times that produced by a DC stress. The shape of the curves also indicates that the asymptotic slopes of the two stress cases are the same. This is suggestive of the fact that AC stress can be modeled as a linear function of DC stress, for long term estimates, as explained in Section 3.1.

2.9.4 NBTI Model for High Frequency Operation

For high frequency operation, the above multicycle model cannot be used, due to the underlying assumptions about the shape of the diffusion front, and the various approximations made during the course of the derivations. However, for a 1GHz frequency operation, it is computationally infeasible to compute the interface trap concentration on a cycle accurate basis for 10 years of operation, amounting to $\approx 10^{17}$ cycles, either using analytical models or through simulations. Hence we seek

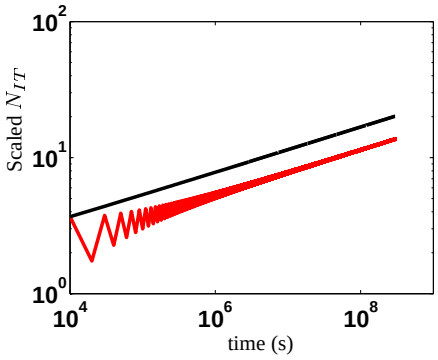


Figure 2.30: Plot showing interface trap generation for τ = 10000s for AC and DC stress cases up to 10 years of operation, on a log-log scale.

transformations of high frequency waveforms into extremely low frequency waveforms (say, of the order of $\leq$ 1Hz), thereby obtaining tractable and fairly accurate asymptotic estimates with a large speed-up. In this regard, we explore a key property of the dynamics of interface trap generation, namely, frequency independence.

Experimental results have shown that the number of interface traps, measured after a large duration of time is approximately the same irrespective of the actual frequency of the input AC waveform being applied [13,20,52,53], implying identical asymptotic N_{IT} estimates. This property is known as **frequency independence**. Although several differing experimental results have been observed, recent experiments have shown that this holds good over the 1Hz-1GHz bandwidth [60], which seconds the analytical findings in [20]. However, as we move closer to DC, some form of frequency dependence is expected. We verify this phenomenon by plotting the number of interface traps up to 10^6s for five different τ values differing by an order each, ranging from 1s to 10000s. The values are compared with the DC case as well, and the plots are shown in Fig. 2.31. The results show that with increasing τ, the N_{IT} curves tend to become closer. Hence, for τ = 1s, some form of frequency independence can be assumed to hold good asymptotically.

Thus, on the basis of experimental data from [60], and the trend seen in Fig. 2.31, we conclude that the interface trap count determined for τ = 1s, asymptotically equals the number for a case where τ = 1ns, over t_{life}, where t_{life} is the lifetime of the circuit, and is assumed to be 10 years of

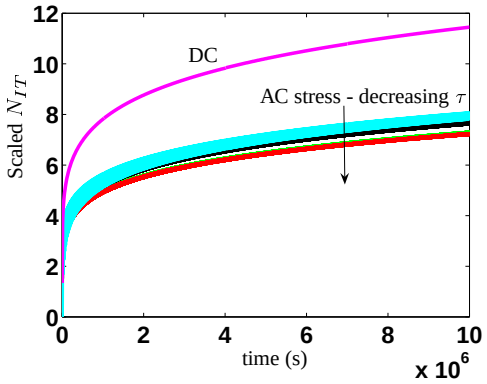


Figure 2.31: Plot showing interface trap generation for different time periods, along with the DC stress case, to demonstrate frequency independence.

operation:

$$N_{IT}(t = t_{\text{life}}, \tau = 1\text{s}) \approx N_{IT}(t = t_{\text{life}}, \tau = 1\text{ns}) \tag{2.109}$$

Thus, we can use our multicycle model derived in the previous subsection, with τ = 1s, to estimate the impact of NBTI on gigascale circuits.

Chapter 3

BTI-Aware Analysis and Optimization of Circuits

As seen from the results in Chapter 2, the impact of NBTI on the PMOS transistor depends on the sequence of stress and relaxation applied to the gate. Since a digital circuit consists of millions of nodes with differing signal probabilities and activity factors, asymmetric levels of degradation are experienced by various timing paths. The exact amount of degradation must be determined using a model that estimates the amount of NBTI-induced shift in the various parameters of the circuit that affect the delay. This metric can then be used to optimize circuits, such that they remain reliable over the desired lifetime of operation, despite temporal degradation.

We begin this chapter with a timing analysis framework that estimates the end-of-lifetime delays of a circuit, taking into account the temporal degradation of the various devices along the paths. In Section 3.1, we first show how the multicycle model developed in the previous chapter can be used to asymptotically determine the end-of-lifetime threshold voltage degradation of every transistor for varying stress-relaxation duty cycles. Accordingly, a lookup table of V_{th} versus transistor degradation probability (DP) is constructed. Section 3.1 then presents a BTI-aware timing analysis framework, where we consider NMOS degradation due to PBTI as well as PMOS degradation due to NBTI. We use degradation probabilities for every transistor in the circuit netlist, and map them to V_{th} shifts, based on the V_{th}-DP lookup table. Libraries are characterized to determine the sensitivity of the gate delays to V_{th}, and subsequently, the new arrival times at the primary outputs of the circuit are computed. Section 3.2 also addresses the limitation of obtaining a more accurate estimate of temporal circuit delays by considering signal correlations. We show results in Section 3.3.1 on 45nm benchmark circuits indicating that the extent of degradation in delay is strongly dependent on the state probability of the various nodes in the circuit. Accordingly, a method is proposed in Section 3.4 to determine the maximal temporal degradation of the circuit under all operating conditions. Finally, we explore the impact of input vector control (IVC) on the amount of temporal degradation, in Section 3.5. Our results indicate that using optimal state assignment to the primary inputs of a circuit, a reduction of up to 50% in the degradation of the delays of the critical paths can be achieved.

In the latter part of this chapter, we present several circuit optimization techniques to mitigate

NBTI and to ensure reliable performance over the lifetime of the circuit. Section 3.6 incorporates the effects of NBTI into technology mapping during synthesis, and a method to build NBTI tolerant circuits is presented. Section 3.7 then describes an adaptive approach that can ensure that the circuit operates reliably over its lifetime. An adaptive guardbanding technique using Adaptive Body Bias (ABB) and Adaptive Supply Voltage (ASV), that maintains optimal performance is presented, and is compared with one-time design-fix techniques such as sizing, and synthesis for BTI. A hybrid approach that optimally combines the advantages of the adaptive and synthesis approaches, thereby ensuring maximal performance with a minimal power overhead is presented in Section 3.11. Lastly, Section 3.13 is aimed at determining the impact of NBTI on SRAM cells. In particular, we investigate the degradation in the static noise margin (SNM) of SRAM cells, which is a critical measure of its read stability. A solution to recover the amount of SNM degradation is proposed in Section 3.14.

3.1 A Framework for Estimating the Shift in V_{th} of Every Transistor in the Circuit

In this section, we begin with a framework for using the NBTI model developed in the previous chapter, to estimate the temporal degradation of the threshold voltage of every transistor in the circuit over 10 years of operation. We use the method described in [70], where the authors claim that AC NBTI can be represented as being asymptotically equal to some α times DC NBTI, where α represents the ratio between the number of interface traps for the AC and DC stress cases:

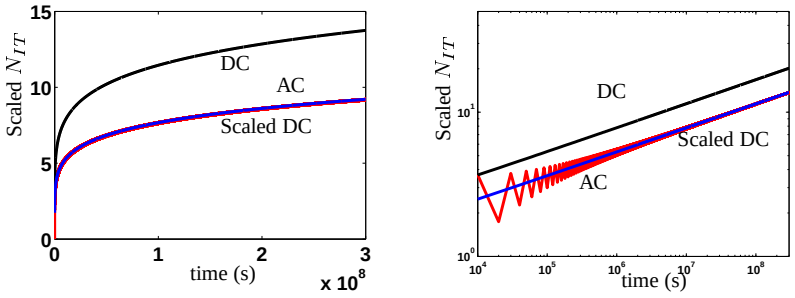
$$\alpha = \frac{N_{IT_{AC}}(t = t_{\text{life}} = N\tau)}{N_{IT_{DC}}(t, t = t_{\text{life}})} \tag{3.1}$$

where N denotes the number of half cycles, each of duration τ, in 10 years of operation. Accordingly, AC stress can be approximated as:

$$N_{IT_{AC}}(t, \tau < 1s) \equiv \alpha N_{IT_{DC}}(t) \tag{3.2}$$

where $N_{IT_{AC}}(t)$ is the number of interface traps due to AC stress, and $N_{IT_{DC}}(t)$, that due to DC stress, at time t. We verify this method graphically by plotting the actual AC waveform and the

scaled DC waveform, where α is the ratio of the number of interface traps computed after 10 years of operation, for τ = 10000s, in Figs. 3.1(a) and (b). A good fit in the linear plot (Fig. 3.1(a)) guarantees correct estimates, for the circuit lifetime, ranging over the 1 year-10 year period.



(a) Linear scale (b) Log scale

Figure 3.1: Plot showing AC stress represented as an equivalent scaled DC stress. The two curves almost perfectly overlap.

Although, the equivalent DC stress model may not provide an exact upper bound, especially over the first few stress and relaxation phases, and may not show the exact transient response initially, the overall fit is fairly accurate for asymptotic NBTI estimates, over a period of time, as large as 10 years, as seen from Fig. 3.1(b). Since reliability estimates do not require cycle accurate behavior of the number of interface traps, the scaled DC model is simple and sufficient.

The above method in conjunction with frequency independence can be used to estimate the number of interface traps as follows:

1. Convert the high frequency waveforms to equivalent 1Hz waveforms, by using the SPAF method outlined in Section 2.3.

2. Calculate the number of interface traps up to 10 years of operation, for the 1Hz square waveform, and the DC waveform using the model.

3. Compute the value of α, and use the scaled DC model as an approximate temporal estimate of the number of interface traps, at various time stamps.

4. Repeat this method for waveforms of different duty cycles, and compute the value of α in

each case, to obtain a simple look-up table of α versus signal probability (such that ΔV_{th} for each signal probability = some α times the ΔV_{th} for DC stress), as described in Section 3.1, or even a smooth curve fitting-based model, as desired.

5. Compute the number of interface traps and the V_{th} degradation at any desired time stamp, for any signal probability, using this scaled DC model.

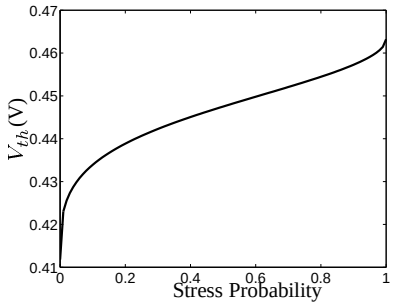


Figure 3.2: PMOS V_{th}, after three years of aging, as a function of the probability that the transistor is stressed.

Since N_{IT} is linearly proportional to V_{th}, experimental results can be used to compute this ratio, and the N_{IT} numbers can accordingly be converted to V_{th} values. We present a generic framework in our work, and hence, simply work with normalized N_{IT} values. A plot of V_{th} versus the probability that a PMOS device is stressed, computed using the method outlined above, is shown in Fig. 3.2. The figure shows an initial steep rise, since N_{IT} and ΔV_{th} are $\propto t^{\frac{1}{6}}$. A lookup table built using this figure can then be used to determine the sensitivity of gate delays to temporal degradation caused by aging, and thereby shifts in timing numbers can be estimated, as described in the next section.

3.2 BTI-Aware Timing Analysis

In this section, we present a timing analysis framework that computes the temporal delay of the circuit considering PMOS and NMOS V_{th} degradation due to NBTI and PBTI, respectively. While the exact physical modeling for PBTI is currently being actively researched [10, 11, 71, 72], we simply assume that the mechanism is identical to that of NBTI, but the absolute magnitude of V_{th}

degradation is less than that induced by NBTI, in order to demonstrate the impact. Hence, a similar plot can be obtained for NMOS transistor degradation, assuming an identical physical mechanism.

We first begin with a gate-level model for determining the shift in the delay due to BTI, on a NOR gate, whose schematic is shown in Fig. 3.3. The analysis for other inverting CMOS gates can be performed similarly. The delay (rising or falling) from an input to the output is modeled as a sum of the nominal timing-arc delay D_0, plus the additional delay due to the degraded V_{th} values of the individual transistors. Approximating the change in the delay using a linearized model with the first order sensitivities of the delay with respect to the V_{th} of each transistor in the gate, we have:

$$D = D_0 + \Sigma_{i=1}^{n} \left(\frac{dD}{dV_{th_i}} \Delta V_{th}(\text{DP}_i) \right) \tag{3.3}$$

where $\frac{dD}{dV_{th_i}}$ for each transistor M_i can be computed through spice simulations, and $\Delta V_{th}(\text{DP}_i)$ can be computed using a look-up table derived from Fig. 3.2, based on the actual degradation probability (DP). Typically, for a rising transition, the $\frac{dD}{dV_{th}}$ terms of the PMOS transistors lying along the input falling-output rising paths are strongly negative, while those for the NMOS transistors are negligible or weakly positive, (and vice versa for a falling transition). It must be noted that the transistor degradation probability, while closely related, need not be equal to the actual signal probability (SP) of the input connected to the gate node, particularly in the case of a stack, as shown in Fig. 3.3.

While the upper PMOS transistor connected to the supply net degrades whenever the input is low (DP_2 = p(I_2=0) = 1-SP_2, where SP is the probability that a signal is high), the lower transistor degrades only when both the inputs are low, i.e., DP_1 = p(I_1=0|I_2=0) = f(SP_1,SP_2), and is equal to (1-SP_1)(1-SP_2), only when the inputs are (functionally and temporally) independent of each other. The maximal value of DP_1 is equal to $\min$((1-SP_1),(1-SP_2)) when the rising periods of the signals are exactly aligned, as shown in Fig. 3.3(b), while the minimal value is given by $\max$(0,(1-SP_1)+(1-SP_2)-1), when their falling periods are maximally skewed apart, as shown in Fig. 3.3(c).

$$\begin{aligned} \max(0, \overline{\text{SP}_1} + \overline{\text{SP}_2} - 1) &\leq \text{DP}_1 \leq \min(\overline{\text{SP}_1}, \overline{\text{SP}_2}) \\ \overline{\text{SP}_1} &= 1 - \text{SP}_1 \\ \overline{\text{SP}_2} &= 1 - \text{SP}_2 \end{aligned} \tag{3.4}$$

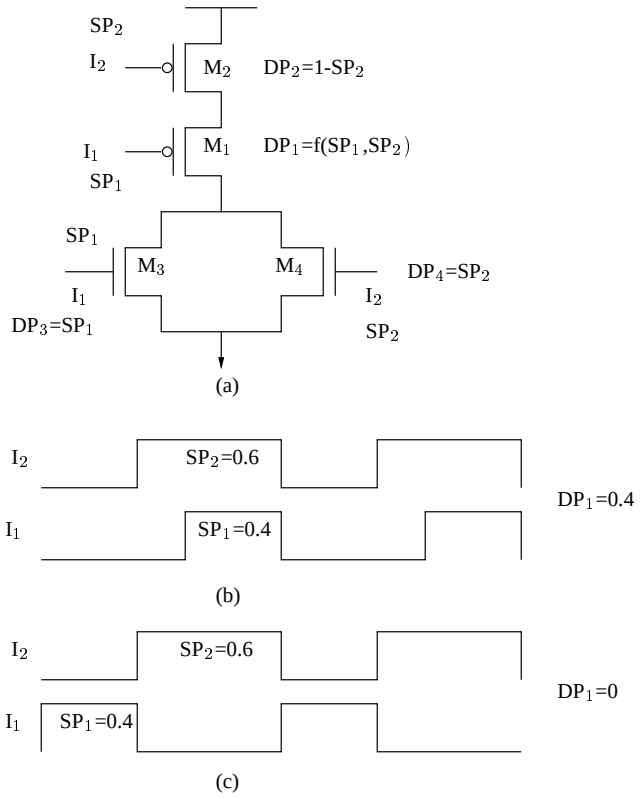


Figure 3.3: Transistor degradation probabilities in a NOR gate - (a): Schematic of two-input NOR gate showing the relation between signal probability (SP) and transistor degradation probability (DP). (b): Signal alignment for I_1 and I_2 such that DP_1 is maximized. (c): Signals I_1 and I_2 aligned such that DP_1 is minimized.

For the case shown in Fig. 3.3(b-c), the degradation probability of the stacked PMOS transistor M_1, i.e., DP_1 computed using the above equation, can vary between 0 and 0.4, depending on the exact correlation between the two signals. Thus, it is vital to make this distinction between the

signal probability of the nodes, and the degradation probabilities of the transistors, particularly to handle the effect of temporal signal correlations. Ignoring this distinction between the nodal signal probability and the transistor degradation probability, can cause erroneous estimations for a stacked network of transistors.

In order to determine the signal activity based degradation probability of every transistor in the circuit, a suitable set of "stress-test" vectors are applied to the benchmark circuits, for a specified number of cycles. A logic simulator is used in every cycle to determine the probability that every transistor is degraded. The test vectors are then assumed to repeat over the entire lifetime of the circuit operation, thereby allowing us to use these numbers as estimated DP values of the transistors over the lifetime of t_{life} seconds of operation. A look-up table listing the V_{th} for each DP, based on Fig. 3.2 is then used to determine the amount of ΔV_{th} for each transistor, and the new delay of each gate is determined, using (3.3). The framework is summarized in Algorithm 1. The above framework can be easily implemented by modifying a logic/switch level simulator to compute the degradation probabilities of the transistors instead of signal probabilities of nodes (for power estimation), based on the vector of primary inputs in every cycle.

Algorithm 1 BTI-aware timing analysis.

1: During library characterization, determine the $\frac{dD}{V_{th}}$ terms for each transistor in all gates in the standard cell library, for each timing arc.
2: Apply a fixed number of logic vectors from the "BTI-stress test" on the circuit under test, and compute the degradation probability (DP) values of the transistors. The impact of correlations is handled by using a Monte-Carlo based simulation technique [73], or probabilistic analysis using BDDs, while the distinction between nodal SP and transistor DP is made by directly determining whether each transistor is under BTI stress or not (based on the logic states of the gate and source devices), as opposed to incorrect SP-based models such as those in [40].
3: Assume these numbers to be the estimated DP values of the transistors during actual circuit operation, over their lifetime.
4: Perform static timing analysis on the circuit to compute the new arrival times, by using the BTI-aware model for mapping DP into ΔV_{th} using a look-up table, and using (3.3) to compute the gate delays.

In order to determine the impact of BTI on the delay degradation of a circuit, ISCAS85 and LGSYNTH93 benchmarks are synthesized on a 45nm based library. 1000 random test patterns for each benchmark circuit tested. Algorithm 1 is applied for each test pattern and the degradation probabilities of all transistors in the design, as well as the delay of the circuit after three years of

aging assuming those estimated degradation probabilities are computed. The method is repeated for each of the 1000 random test patterns. The results are shown for a representative LGSYNTH93 benchmark "alu2" in Fig. 3.4. The results show the nominal delay, the worst-case delay (described

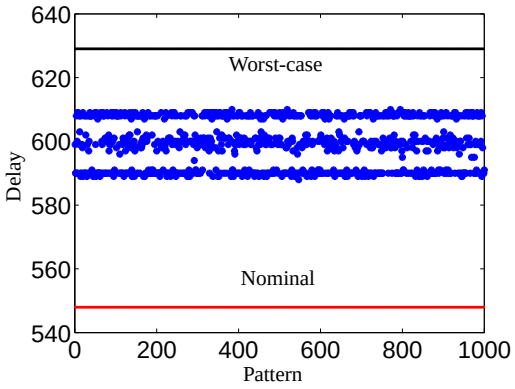


Figure 3.4: Delay of benchmark circuit "alu2" for different test patterns. Each point represents a set of BTI-test vectors applied for n cycles, and assumed to repeat periodically over t_{life}.

later on in Section 3.3.1), and the delay after three years of degradation considering both NBTI and PBTI for each of the 1000 test patterns. The results indicate that BTI increases the end of lifetime delay of the circuit by up to $\approx 15\%$.

3.3 Bounds on the Delay Degradation of the Circuit

While the method detailed in Algorithm 1 uses a set of test vectors over a limited number of cycles of operation, the performance degradation of the circuit depends on the actual sequence of input patterns applied over the entire lifetime of the circuit, and can vary with the applications run, the workload conditions, usage of the chip, etc. While the previous section presented an overview of performing timing analysis using a predetermined set of test vectors, and estimated the transistor degradation probability numbers, it is virtually impossible to determine *a priori*, the exact set of test vectors that would be applied to a digital circuit over its lifetime. Hence, in order to guardband the circuit for aging-induced performance degradation, it is crucial to compute the maximal delay degradation of the circuit under all operating conditions, and to design the circuit reliably, accordingly. Fig. 3.4 showed the spread in the delay of an LGSYNTH93 benchmark circuit "alu2", for

1000 such randomly chosen set of test vectors. The results indicate a large spread in the end of lifetime delay of the circuit, and hence, an upper bound on this number must be computed, so that the circuit can be designed accordingly.

3.3.1 Worst-case Delay of the Circuit

The work in [36] proposes a worst-case method to determine an upper bound on the delay degradation of the circuit. Under the worst-case method (for NBTI), all PMOS transistors are assumed to be under DC stress over their entire lifetime. Hence, the V_{th} of all these PMOS devices shifts by a constant amount, whereas the V_{th} of NMOS transistors remains unaffected. The new end-of-lifetime delay of the circuit is then estimated using a simple static timing analysis, with the degraded delays for each gate in the standard cell library, instead of their nominal delay numbers. It can be intuitively seen that since NBTI affects only the PMOS devices, assuming maximal degradation of every PMOS device along the critical path in the circuit implies that this method provides an upper bound on the maximal delay of the circuit, under any set of actual input vectors.

Considering NMOS degradation due to PBTI, the worst-case method can be extended by now assuming that under this case, all PMOS transistors are under DC stress with their inputs being at logic 0, while the NMOS devices are under DC stress with their inputs being at a logic high. Maximal degradation of PMOS devices along gates undergoing a rising transition, while maximal aging of NMOS transistors along those gates in the critical paths whose outputs have a falling transition is assumed. In effect, the gate delays in (3.3) are computed by assuming that:

- The degradation probability DP is 0 for a transistor whose $\frac{dD}{dV_{th}}$ term is negative.
- If $\frac{dD}{dV_{th}}$ is positive, then we assume that DP = 1 for that transistor.

The above assumption ensures that the method is guaranteed-pessimistic, and therefore represents the “worst-case” delay degradation of the circuit.

However, structural correlations and the logic functions of the gates, may lead to a case where for instance, the input of two particular PMOS devices cannot simultaneously be equal to a logic 0, which is the condition for DC stress. Fig. 3.5 shows an example of such a case, for a toy circuit with a reconvergent fanout, and implementing the logic function $y = ab$. Worst-case aging on

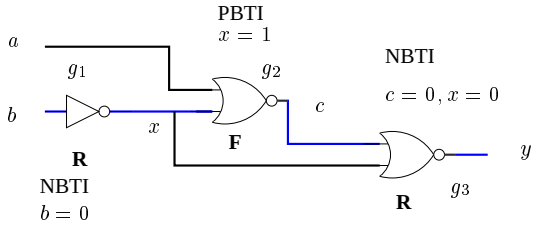


Figure 3.5: Circuit with "worst-case" delay path not sensitizable.

this circuit requires the signals x and b to both be logic zero, for maximal degradation on gates g_3 and g_1, respectively, which is infeasible. It must be noted that such circuit structures are quite common in digital circuits, particularly due to subfunction sharing, during the structuring phase of logic optimization. Further, considering maximal NBTI as well as PBTI, it can be seen that this case may require the input of a gate to simultaneously be a zero and a one (signal x in Fig. 3.5 for instance), to cause maximal DC stress on the NMOS or PMOS transistors on the gates along the critical path, to realize the assumptions of the worst-case method, as marked in the figure. Hence, the delays computed by the worst-case method may not necessarily be a tight upper bound on the maximal impact of aging. Accordingly, we propose to investigate the amount of pessimism in using this method as a measure of the maximal delay degradation of the circuit.

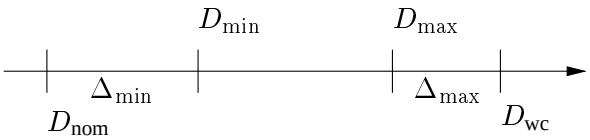


Figure 3.6: Bounds on the delay degradation.

Fig. 3.6 depicts the goal of our work, and outlines the problem statement. The nominal delay of the circuit, i.e., the delay of the circuit at birth, without any timing degradation due to aging is denoted as D_{nom}, while D_{wc} denotes the end-of-lifetime delay of the circuit, considering the worst-case method. Our aim is to determine the maximal delay D_{max}, and to quantize the difference between D_{max} and D_{wc}, denoted as Δ_{max}. This enables us to determine the amount of pessimism introduced by the worst-case method, thereby reducing the timing guardbands accordingly.

Clearly, since the circuit is intended to perform some functionality, transistors in the circuit undergo some amount of degradation and recovery, resulting in an increase in their V_{th} values. Thus,

there exists a minimal degradation value $> D_{\text{nom}}$, and this number is denoted as $D_{\min}$. It must be noted that D_{nom} and D_{wc} are loose lower and upper bounds, respectively, on $D_{\min}$ and $D_{\max}$. The utility of BTI-aware delay minimization and the $D_{\min}$ numbers is elaborated in Section 3.4.4.

3.3.2 Circuits without Reconvergent Fanouts

We begin our analysis with circuits that do not consist of a reconvergent fanout. Shown in Fig. 3.7 is a chain of even number ($2n$ for some n) of inverters, assumed to be of equal size, and each having a nominal rise and fall delay D_0. Let α be the SP at the primary input. Consequently, the degradation probability of the PMOS transistors on the odd numbered gates is given by (1-α), while that on the NMOS transistors is α, and vice versa for the even numbered gates. Let us assume that NBTI only affects the rising transition of a gate, and PBTI, the falling transition. Let, the Δ delay on a gate be expressed as:

$$\begin{aligned} \Delta D_R &= \frac{dD}{dV_{thP}} \Delta V_{thP}(\alpha) \\ \Delta D_F &= \frac{dD}{dV_{thN}} \Delta V_{thN}(\alpha) \end{aligned} \tag{3.5}$$

where the rising delay D_R depends on the PMOS degradation only, while the falling delay D_F depends on the NMOS degradation only[1]. ΔV_{thN}(MOS) and ΔV_{thP}(MOS) are functions of the signal probability of the primary input, α. Further, since the PMOS (NMOS) degradation due to NBTI (PBTI) is maximum when the input signal probability is zero (one) and minimum when the signal probability is one (zero), let us assume a simple relationship[2] of the form:

$$\begin{aligned} \Delta V_{thP} &= K_P(1-x)^m \\ \Delta V_{thN} &= K_N x^m \end{aligned} \tag{3.6}$$

[1]This assumption is reasonably accurate based on our simulation results.

[2]The exact nature of this relationship does not affect the final result. It is only necessary that ΔV_{thP} monotonically decrease with x, and ΔV_{thN} increase monotonically with x. Clearly, from the shape of the curve in Fig. 3.2, this is true.

for an inverter whose input SP is x, and m is some number > 0, depending on the shape of the V_{th}-SP curve, while K_N and K_P are some constants. The maximal output fall delay D_F (rise delay D_R) is given by the nominal delay of every gate, plus the Δ delay terms due to NBTI on the odd (even) numbered gates, and PBTI on the even (odd) numbered gates.

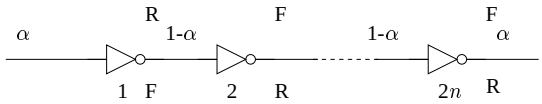


Figure 3.7: A chain of inverters, numbered 1, . . . , $2n$. R and F indicate a rising and falling transition at the output, respectively.

Thus, we have:

$$\begin{aligned} D_R &= 2nD_0 + \Sigma_{i=1}^{n} K_N \frac{dD}{dV_{thN_{2i-1}}} \alpha^m + \Sigma_{i=1}^{n} K_P \frac{dD}{dV_{thP_{2i}}} \alpha^m \\ D_F &= 2nD_0 + \Sigma_{i=1}^{n} K_P \frac{dD}{dV_{thP_{2i-1}}} \overline{\alpha}^m + \Sigma_{i=1}^{n} K_N \frac{dD}{dV_{thN_{2i}}} \overline{\alpha}^m \\ \overline{\alpha} &= 1 - \alpha \\ D &= \max(D_R, D_F) \end{aligned} \tag{3.7}$$

If the signal probability at the primary input is assumed to vary between 0 and 1, the above function ($D = \max(D_R, D_F)$) is maximized at α = 0, or α = 1, implying that a DC stress does indeed produce the worst-case degradation on the circuit. Since alternate gates along a critical path undergo either a rising or a falling transition, with NBTI only affecting the pull-up and PBTI only the pull-down network, it is possible for a circuit in operation to experience DC stress, or often close to DC stress along some of its paths, depending on the value of α. While the above analysis was performed on a simple chain of inverters, this can be extended to an arbitrary set of inverting gates along a path, as long as there is no reconvergent fanout.

3.3.3 Impact of Reconvergent Fanouts

Fig. 3.5 shows that reconvergent fanouts can lead to infeasible conditions on the signal states, for the circuit delay due to aging to be equal to the worst-case value. For this particular case, the signal state of b affects the degradation of the three gates, and the transistors along the critical paths in each

of these gates, differently. For instance, we require b to be equal to a logic 0 for the inverter (gate g_1) to have a maximal shift in its rising delay, and the NOR gate g_2 to have a maximal degradation for the falling transition, (since $b = 0 \Rightarrow x = 1$, and hence DC stress on the NMOS transistor of gate g_2), as marked in Fig. 3.5. However, for the rising delay of the NOR gate g_3 to be affected, we require both its inputs to be a logic 0 - implying that $x = 0$, and $b = 1$. Thus, by setting SP(b) to be 0 or 1 or any other intermediate value (for a suitable value of SP(a)), the maximal delay of the circuit under aging is still not equal to the value computed by the worst-case method. Fig. 3.8 shows the delay of the critical path for the circuit in Fig. 3.5 as a function of the signal probability of b, for the case where SP(a) = 1[3], since SP(a) = 1 maximizes the delay of the critical path. The delay increases as the signal probability of B decreases (from 1), due to higher NBTI on the PMOS device in g_1 (rising transition of g_1 lies along the critical path), as well as higher PBTI on the NMOS device in g_2 (falling transition of g_2 lies along the critical path). However, this causes a reduction in the stress times on the PMOS transistor in g_3; hence the delay on the rising transition of gate g_3 and the total arrival time along the critical path subsequently decreases. The delay is maximized at SP(b)≈0.5, but the value is clearly less than the "worst-case" number, as can be seen from Fig. 3.8.

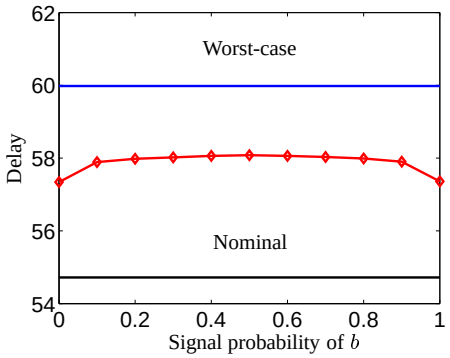


Figure 3.8: Delay of the circuit in Fig. 3.5 as a function of signal probability of node b.

Thus, it is clear that reconvergent fanouts in circuits can cause the "worst-case" delay of the circuit (computed using the method described in Section 3.3.1), to be a pessimistic estimate of the

[3]Since a is assumed to be under a steady state, the degradation probabilities of the stacked transistors can be easily computed, without explicitly considering temporal correlations between signals a and b based on their values in each cycle.

maximal degradation of the circuit under aging. Besides, if the input signals can be guaranteed to be bounded within a subset of switching probabilities ($0.1 \leq$ SP ≤ 0.9 for instance, instead of the full range of [0,1]), it is obvious that transistors are not under DC stress, and therefore recover partially; this leads to a case where the maximal delay degradation of the circuit under any feasible signal activity over the lifetime is still less than the worst-case number. An optimization framework that can compute a tight upper bound on the maximal delay degradation of the circuit is therefore essential. Such a framework can be utilized in multiple analysis and optimization scenarios as:

- Compute a tighter upper bound on the delay degradation of the circuit under aging, and compare this metric with the worst-case value.
- Compute the effect of tighter bounds on the signal probabilities of the primary inputs; ranges of [0,1] represent the unbounded case, while [0.5,0.5] (with uncorrelated inputs) represents the case used in [18, 19] etc, to demonstrate the impact of NBTI on circuit degradation.
- Modify the optimization framework to compute the minimal delay degradation of the circuit under aging. This enables us to quantify the gap between the maximal and minimal degradation numbers, and can accordingly guide us toward a sleep-state input vector control (IVC) scheme for aging-aware delay optimization.
- The vectors that maximize the degradation of a circuit can be used to generate a "stress-test" pattern, that can be applied to the circuit during testing for measuring the delay of the circuit.

3.4 BTI-Aware Delay Optimization

The optimization problem is formulated as follows:

$$\begin{aligned} &\text{Maximize } D(p_1, \ldots, p_n) \text{ s.t.} \\ &0 \leq p_i \leq 1, i = 1, \ldots, n \end{aligned} \tag{3.8}$$

where D denotes the delay of the circuit, n the number of its primary inputs, and p_i the signal probability (SP) of the i^{th} primary input (PI) node. The p_i terms are continuous variables, and are

assumed to be independent of one another, i.e., they have no temporal correlations. The impact of correlation on these numbers is discussed later on, in this section.

The objective function in (3.8) can also be modified as a delay minimization problem, subject to the signal probability constraints. This can be used to perform standby optimization, by determining whether the signals must be parked at logic high or low, such that the BTI-induced aging effect is minimized.

3.4.1 Optimization Techniques

It can be seen from Section 3.3.2 that if the worst-case delay critical path does not consist of a reconvergent fanout, the primary inputs can be assigned signal states of 0, or 1, (implying signal probabilities of 0, or 1, accordingly), so that the maximal delay is equal to the highest possible delay degradation number, i.e., the worst-case delay of the circuit. However, the problem of determining whether such a feasible assignment exists is equivalent to the problem of determining whether a satisfiable assignment to the inputs exist such that the output of the critical path is either a 0 (or a 1), depending on whether the last gate along the critical path has a falling transition (or rising transition), and is therefore NP hard. The proof of this hypothesis is shown in Appendix D.

In the presence of reconvergent fanouts, and circuit structures leading to infeasible SP requirements (as shown in Fig. 3.5 for instance), the solution that maximizes the delay depends on the sensitivities of the delays to the degradation probability of the transistors, lying along the critical path. Evidently, the solution space for this case is larger than the case where the critical path does not have a reconvergent fanout, since the optimal solution is no longer guaranteed to be either a 0 or a 1 for each p_i, in (3.8).

Accordingly, we present two heuristics, in order to determine the maximal delay of the circuit, and seek to provide a reasonably accurate solution in linear time (linear in terms of the number of primary inputs). Since the heuristics do not explore the complete search space, the resulting numbers only provide a lower bound on the maximal delay degradation numbers (and an upper bound on the minimal delay degradation numbers), as explained further on, in Section 3.5.

3.4.2 Branch and Bound-based Heuristic

In this procedure, random test vectors are generated such that the SP of all PI nodes are initially arranged to be 0.5, and the degradation probabilities of all transistors are computed. The delay of the circuit is calculated with the SP of each primary node set to a fixed set of k values in [0,1]. The SP that results in the worst-case delay is determined, and this value is used for that primary input in the subsequent iterations, (instead of the initial value of 0.5). The SP of each input that maximizes the delay of the circuit is determined accordingly[4].

Algorithm 2 Branch and bound-based heuristic.

1: $\{n$ iterations: one for each PI$\}$
2: **for** Pin i = G.First-PI; i; i = G.Next-PI **do**
3: $\{$Initial assignment$\}$
4: Generate test vectors such that p_i = 0.5.
5: **end for**
6: **for** Pin i = G.First-PI; i; i = G.Next-PI **do**
7: $\{k$ iterations for each PI$\}$
8: **for** p_i = 0; $p_i += \frac{1}{k}$; $p_i \leq 1$ **do**
9: $\{$Use Algorithm 1 to determine the delay numbers.$\}$
10: Compute degradation probabilities of all transistors in the circuit using SP(PI$_i$) = p_i, and either the initially assigned value or the optimally determined p_i^* from previous iterations, for the remaining inputs.
11: Perform static timing analysis on G and compute the delay of the circuit.
12: $\{$Maintain the solution for p_i that maximizes the delay.$\}$
13: if $D > D_{\max}$, $D_{\max} = D$ and $p_i^* = p_i$.
14: **end for**
15: $\{$Use this value of p_i in successive iterations.$\}$
16: Set $p_i = p_i^*$
17: **if** i = G.Last-PI **then**
18: $\{$Leaf node in the tree has been reached.$\}$
19: Return maximum delay $D_{\max}$.
20: **end if**
21: **end for**

The above method is equivalent to a branch and bound technique, where only one downward traversal is performed, and the decision to continue along the left-most (p_i = 0) or the right-most

[4]It must be noted that the final solution depends on the ordering of the variables (and there exists a certain ordering for which the solution computed using this method is equivalent to the true optimal solution). However, it is computationally intensive to determine such an optimal ordering, and may require significantly large number of iterations.

(p_i = 1) or any other intermediate child node (along the remaining k-2 nodes) is based on the computed bounds on the delay, so far. The above method is similar to the first heuristic used in [74] for static state leakage reduction. The pseudocode is shown in Algorithm 2. The run-time of this technique is of the order $O(nkt_{\text{STA}})$, for n primary inputs, and k values for each SP, where t_{STA} is the time for one STA run.

3.4.3 Sensitivity-based Heuristic

In the second approach, we sensitize the dependence of the delay with respect to the SP of each PI node. The SP of each PI that locally maximizes the delay of the circuit, under the condition that the remaining inputs are assigned their p_i terms such that the optimal solution is realized, is determined. This method is implemented as follows:

The SP of a primary input node whose impact on the delay of the circuit that we are interested in determining, is assumed to vary in [0,1]. The degradation probabilities of all other transistors in the circuit are assumed to be equal to 1, while those lying on the fanout cone of this primary input are determined accordingly, based on this SP value. k different iterations are performed, for each SP value of this primary input, and the SP that leads to the maximum delay of the circuit is determined. The above procedure is repeated for each primary input node independently, and the local solution that maximizes the delay, subject to all other input nodes assigned values corresponding to the "worst-case" is determined. One final timing analysis run is performed with these optimal SP values for each of the primary inputs, to determine the maximum delay of the circuit. The algorithm is shown in Algorithm 3.

Clearly, in the absence of reconvergent fanouts, if the delay of the circuit depends on the signal probability of some i^{th} input node, this method can help us identify the optimal value of p_i, by determining whether the delay D is monotonically increasing or decreasing, based on $\frac{dD}{dp_i}$. For critical paths that have reconvergent fanouts, this enables us to determine the value that still locally maximizes the delay. Variants to this method are obtained by setting the degradation probability of all other transistors in line 3 of Algorithm 3 to 0, instead of 1. While the case shown in the algorithm (with the initial setting of DP = 1) begins with the "worst-case" assignment, and gradually relaxes the delay values, thereby providing a bound from the outside, (the case where DP = 0 for all

Algorithm 3 Sensitivity-based heuristic.

1: $\{n$ iterations: one for each PI$\}$
2: **for** Pin i = G.First-PI; i; i = G.Next-PI **do**
3: Assign maximum DP (DP = 1) to all transistors except the ones along the fanout cone of primary input i.
4: **for** p_i = 0; $p_i += \frac{1}{k}$; $p_i \leq 1$ **do**
5: Determine DP of transistors connected to PI i using the value of SP(PI$_i$) = p_i.
6: Perform static timing analysis on G.
7: Compute the delay of the circuit.
8: Maintain the solution for p_i that maximizes the delay.
9: if $D > D_{\max}$, $D_{\max} = D$ and $p_i^* = p_i$.
10: **end for**
11: Final SP of PI: p_i = p_i^*.
12: **end for**
13: Perform static timing analysis on G using final SP values for all PI nodes.
14: Return maximum delay of the circuit.

transistors initially) starts with the nominal delay values, and hence provides bounds on the delay from the inside. The above algorithm requires k STA (static timing analysis) runs for each input nodes, followed by a final STA run where the SP of each input node i is assigned p_i^* accordingly. Thus, the total run time is also of the order $O(nkt_{\text{STA}})$.

Results of using these two heuristic algorithms on benchmark circuits are shown in the next section. It must be noted that the solution produced by either algorithms represents a lower limit on the maximal delay of the circuit, since the two approaches are not guaranteed to yield globally optimal solutions to the maximization problem in (3.8), in $O(nkt_{\text{STA}})$ time. We discuss the implications of this on our conclusions later on in Section 3.5.

3.4.4 Optimal Solution for Delay Minimization

For the case of the chain of inverters in Fig. 3.7, i.e., a circuit whose critical path does not consist of a reconvergent fanout, the maximal delay degradation of the circuit is equal to the worst-case number, and occurs when the inputs are assigned a signal probability of either 0, or 1, accordingly. However, the SP value that leads to a minimal delay degradation depends on the exact gates along the critical paths, and the sensitivities of the delays with respect to the degradation probabilities of the transistors. We illustrate simulation results for the case of the chain of even number of inverters

in Fig. 3.7. The circuit consists of two paths, i.e., the output falling and the output rising path. Without loss of generality, let as assume that the rising path has a higher nominal delay than the falling path, after maximal aging under worst-case conditions. The sensitivity of the path delays with respect to the signal probability of the primary input is shown in Fig. 3.9. As the SP of the primary input increases, the NMOS (PMOS) devices along the odd (even) numbered gates undergo higher degradation, while the corresponding PMOS (NMOS) devices are under higher recovery. Hence, the delay of the output rising path increases, while that of the output falling path decreases.

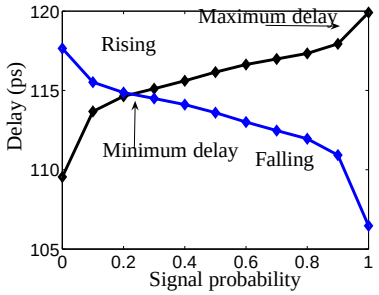


Figure 3.9: Delay of a chain of inverters along the output rising and falling transitions.

Table 3.1: Minimum and maximum delays for different benchmarks.

Bench mark	Nominal Delay (ps)	Min-Delay		Max-Delay		Worst-case		Difference
		Delay %	Increase (ps)	Delay %	Increase (ps)	Delay %	Increase (ps)	%
vda	173	190	10%	199	15%	199	15%	5%
i9	306	322	5%	350	14%	350	14%	9%
i8	322	340	6%	366	14%	366	14%	8%
t481	362	379	5%	408	13%	408	13%	8%
C880	385	405	5%	442	15%	443	15%	10%
C432	520	557	7%	587	13%	591	14%	6%
C7552	539	584	8%	614	14%	622	15%	6%
alu2	548	588	7%	616	12%	629	15%	5%
C5315	651	697	7%	725	11%	748	15%	4%
C3540	669	720	8%	743	11%	763	14%	3%
C6288	1959	2112	8%	2203	12%	2232	14%	5%
Average			7%		13%		14%	6%

The exact shape of the delay versus SP curve in Fig. 3.9 depends on the sequence of gates and

their sizes along the path, and may be nonlinear in general. From the figure, it can be seen that while the SP value that maximizes the delay of such paths is guaranteed to occur at a corner setting, i.e., at either SP = 0 or SP = 1, the same is not necessarily true for the *minimal* delay degradation after aging for t_{life} seconds. The exact SP value that minimizes the extent of aging depends on the relative shapes of the two paths, and can occur at an intermediate setting as well. Therefore, we use the two heuristics in Section 3.4 to determine the minimal delay of the circuit after aging, with minor modifications to maintain the minimal delay solution, instead of the maximal solution. It must be noted that the nominal delay of the circuit, i.e., the delay of the circuit at $t = 0$s without considering any aging serves as a loose lower bound on this number. Simulation results for minimal delays under aging are shown in the next section.

3.5 Simulation Results

In this section, we show simulation results in Table 3.1 for the minimal and maximal delay numbers under temporal degradation of circuits due to aging. Results are shown for the ISCAS85 and LGSYNTH93 benchmarks synthesized over a PTM [75] based 45nm library. The nominal delays of the benchmarks, shown in Table 3.1 are the delays, at $t = 0$s, when there is so temporal degradation, while the "worst-case" delays are the numbers computed based on the worst-case method described in Section 3.3.1. The heuristics described in Section 3.4.2 and 3.4.3 are applied, and the maximum delays obtained across these methods are listed in the table. Similarly, the heuristics are modified to compute the minimum delays. As explained in the previous section, these numbers represent a lower bound on the maximum delays and an upper bound on the minimum delays, due to the performance by the heuristics.

3.5.1 Worst-Case Delay Numbers

The results indicate that the gap between the "worst-case" and the maximal delays is extremely small (of the order of 1%). In cases where there is no difference, the critical path does not consist of a reconvergent fanout, and hence the "worst-case" critical path can be sensitized by a feasible SP assignment to the primary inputs. For circuits where the maximal delay is less than the "worst-case" delay, the critical path consisted of reconvergent fanouts, thereby placing infeasible SP requirements

on some of the primary inputs, to maximize the impact of aging along all gates along the path. However, the difference between the "worst-case" and the actual delays under aging of such gates is extremely minimal, and hence does not affect the overall number significantly. For instance, along the most-critical path of the benchmark circuit "alu2", consisting of 44 gates, only five gates had a reconvergent fanout leading to an infeasible SP assignment, for the gate delays computed using our method to be equal to that computed using the "worst-case" method. Further, since the heuristics provide a **lower bound** on the maximal delays of the circuit, as explained in the previous section, this implies that the gap between the "maximal delays" and the "worst-case" delays is **at most** 1%. Hence, the worst-case delay can be used as a measure of the maximal delay degradation of the circuit, without incurring a huge overhead in estimation. In other words, the accuracy of the heuristics is sufficient enough to enable us to conclude that the "worst-case" delay numbers can be used to quantify the impact of aging on the end of lifetime delay degradation of a circuit, without leading to a large overestimation.

3.5.2 Input Vector Control

The last column in the table also compares the difference between the maximal and minimal delays of the circuits after three years of temporal degradation. The results indicate that the best-case can reduce the amount of temporal degradation by up to 50%. Thus, an input vector control (IVC) scheme that arranges for the signal probabilities of the primary inputs to be close to the value that results in minimal BTI-induced degradation can be employed. Since BTI causes the leakage of the circuit to decrease temporally (and reach values lower than the nominally budgeted value), as shown later on in Fig. 3.15(b), existing leakage minimizing sleep state optimization algorithms can be modified to perform BTI-aware sleep state assignment during standby mode through IVC, without violating the leakage constraints.

While the results in Table 3.1 are obtained assuming that the input signal probabilities can vary fractionally in [0,1], the algorithms can be modified to allow only states of 0,1 for the primary inputs (and all other signals), as may be deemed necessary in a standby mode setting. Under this setup, the minimal amount of temporal degradation due to BTI is of the order of 8% (instead of 7%, as shown in the last row of Table 3.1). Thus, the method still has potential benefits, and can help reduce the

temporal degradation of circuits. The optimal solution obtained from Algorithms 2-3 can be used to park the inputs under logic 0 or logic 1 accordingly, thereby minimizing the BTI effect on the gates along the critical paths.

3.5.3 Impact of Bounded Signal Probabilities

Lastly, we also perform experiments where the signal probability of primary inputs was restricted to lie between tighter bounds, instead of [0,1]. This reflects a scenario where additional information is known about the state of the signal, based on the microarchitecture, or through simulations, and that can be used to restrict the SP values of the inputs to lie within tighter bounds. Heuristics presented in the previous section are modified accordingly, with the inner for loop in Algorithms 2-3 running over a reduced range instead of [0,1]. Table 3.2 shows the delays for the benchmarks, for the maximum delays after three years of aging, for different bounds on the primary inputs: [0,1], [0.1,0.9], [0.2,0.8], [0.3,0.7], [0.4,0.6], and [0.5, 0.5], and the ratio of the maximal delay numbers to the worst-case estimated delay of the circuit.

The results indicate that the impact of bounded primary input signal probabilities, on the gap between the maximal delay of the circuit and the worst-case delay of the circuit is **minimal**, with the largest gap of at most 5% occurring when the primary inputs are assumed to have a signal probability of exactly 0.5. This is due to the fact that signal probabilities in a circuit block are skewed in nature due to the depth of logic, and the sensitivity of the SP of an internal node to the SP of the primary inputs is extremely small. Thus, even under the presence of bounded signal probabilities, the worst-case method can be used to estimate the delay degradation of a circuit due to aging, without leading to a large amount of overestimation.

Table 3.2: Impact of bounds on primary input signal probabilities on the gap between maximal and worst-case delays.

Bench mark	Nominal Delay (ps)	Worst Case (ps)	Maximal Delay											
			[0-1]		[0.1-0.9]		[0.2-0.8]		[0.3-0.7]		[0.4-0.6]		[0.5-0.5]	
			Delay (ps)	Ratio	Delay (ps)	Ratio	Delay (ps)	Ratio	Delay (ps)	Ratio	Delay (ps)	Ratio	Delay (ps)	Ratio
vda	173	199	199	1.00	196	0.98	196	0.98	195	0.98	194	0.97	194	0.97
i9	306	350	350	1.00	344	0.98	341	0.97	338	0.97	335	0.96	334	0.95
i8	322	366	366	1.00	360	0.98	357	0.98	355	0.97	352	0.96	351	0.96
t481	362	408	408	1.00	398	0.98	393	0.96	388	0.95	386	0.95	383	0.94
C880	385	443	442	1.00	427	0.96	424	0.96	421	0.95	418	0.94	414	0.93
C432	520	591	587	0.99	583	0.99	570	0.96	568	0.96	566	0.96	564	0.95
C7552	539	622	614	0.99	610	0.98	596	0.96	594	0.95	591	0.95	589	0.95
alu2	548	629	616	0.98	610	0.97	603	0.96	602	0.96	601	0.96	600	0.95
C5315	651	748	725	0.97	718	0.96	716	0.96	714	0.95	713	0.95	712	0.95
C3540	669	763	743	0.97	744	0.98	737	0.97	735	0.96	732	0.96	730	0.96
C6288	1959	2232	2203	0.99	2145	0.96	2127	0.95	2125	0.95	2123	0.95	2121	0.95
Average				0.99		0.97		0.96		0.96		0.96		0.95

3.6 NBTI-Aware Technology Mapping of Digital Circuits

The results in [18] demonstrate the impact of NBTI on the temporal degradation of circuits (at the 65nm technology node). NBTI can cause the delay of the circuit to increase by as much as around 10%, at the end of the lifetime of the circuit. The impact is seen to increase with technology scaling into the 45nm node, particularly with the use of high-k dielectrics resulting in additional degradation along the NMOS transistors due to PBTI, as shown in Section 3.2. Thus, it is imperative to design circuits accounting for this effect, so that they remain reliable over the desired lifetime. Accordingly, a technology mapping technique that incorporates the NBTI stress and recovery effects, in order to ensure optimal performance of the circuit, during its entire lifetime, is presented.

3.6.1 Previous Work and Limitations

A general solution to maintaining optimal performance under the influence of NBTI has been to reduce the delay of the critical paths through the use of gate sizing [19, 36]. The work in [36] formulates a nonlinear optimization problem to determine the optimal set of gate sizes required to ensure that the circuit runs at its delay specification, after 10 years of operation. The work is based on a model for NBTI, that ignores the effect of recovery, in computing the threshold voltage degradation. The model cumulatively adds the time for which the gates are stressed during their entire lifetime, and estimates the threshold voltage degradation, assuming that the gates are continuously stressed for that duration. Hence, their results show that the increase in the circuit area is rather weakly dependent on the signal probabilities of the nodes, and assuming that all gates in the circuit are always NBTI affected (worst-case design) does not significantly affect the final solution. The authors consider the gate sizes to be continuous, and show that an increase in area of about 8.7%, as compared to a design that ignores NBTI effects, is required to meet the target delay.

However, we find the conclusion that the delay is independent of signal probability does not hold, under a model that captures the healing of NBTI, on removal of the applied stress. This happens frequently in a circuit: for example, when the input signal to a CMOS inverter changes from logic 0 to logic 1, the V_{gs} stress is relaxed from $-V_{dd}$ to zero. The recovery in threshold voltage on removing the applied stress, can be explained by physical mechanisms related to annealing of interface traps, and reformation of Si-H bonds. Experiments in [2, 13], and subsequently the models

in [19, 20] as well as the work in Section 2.1, have shown that considering the effect of annealing and recovery has a significant bearing on the overall NBTI impact.

3.6.2 Problem Formulation

We observe that the above idea can be readily used in other transforms, such as technology mapping, by replacing the nominal value of the delays of the gates in the standard cell library, with the delay under worst-case NBTI. The target frequency is given to the synthesis tool, and technology mapping can be performed using these NBTI-affected library cells to produce a circuit, which is structurally different from that obtained using the sizing algorithm in [36], but is functionally equivalent, and meets the timing.

Further, the work in [36] merely computes the fractional increase in the area of each gate on an existing design, (implying a library consisting of infinitely many sizes of each gate), to counter the temporal degradation caused by NBTI. Instead, our work relies on integrating the effects of NBTI stress and recovery into circuit design at a much earlier stage, i.e., during technology mapping. Since a digital circuit consists of millions of nodes with differing signal probabilities, it is essential to estimate the delay of the gates in the library based on their input node signal probabilities, and use these delay values during technology mapping. Accordingly, our work proposes an approach to modify the process of technology mapping, based on the signal probability of the nodes in the circuit. The SP values of the primary inputs are assumed to be known, based on RTL-level simulations or statistical estimates. The SP values at every other node in the subject graph are calculated accordingly, and this information is used to choose the best gate to meet the timing at each node.

3.6.3 NBTI-Aware Technology Mapping

In this subsection, we describe the process of technology mapping using the smallest ISCAS85 benchmark C17, as an example. The benchmark consists of five inputs i0, i1, i2, i3, and i4. There are two primary outputs, y9 and y10. The logic function computed by this circuit is given as follows:

$$\begin{aligned} \mathrm{y9} &= (\mathrm{i1} + \mathrm{i4}) \cdot (\overline{\mathrm{i2}} + \overline{\mathrm{i3}}) \\ \mathrm{y10} &= \mathrm{i0} \cdot \mathrm{i2} + \mathrm{i1} \cdot (\overline{\mathrm{i2}} + \overline{\mathrm{i3}}) \end{aligned} \tag{3.9}$$

The subject graph for C17 is obtained through SiS [76], using a NAND2-NOT representation, is as shown in Fig. 3.10. Technology mapping is then performed, using a 65nm node [75] based standard cell library, consisting of 10 NOT gates, 6 NAND2 gates, 6 NOR2 gates, 5 NAND3 gates, 3 NOR3 gates, 3 AOI22 gates, 3 AOI12 gates, 3 OAI22 gates, and 3 OAI12 gates of varying sizes. A large library set consisting of different gates of varying sizes is chosen, to provide the synthesis tool with different options to implement a given logic cone. The delay of each of these gates in the library is precharacterized as a function of their input signal probabilities, at each of the original data points. The overhead in precharacterizing the library depends on the original number of corners at which the delay was characterized, and can be reduced by using linear models with respect to the signal probabilities, which provide a reasonably accurate fit, as can be observed from [40].

During technology mapping, the logic cone for each node can be implemented in various structurally different ways, to realize the same functionality. The mapping tool computes the area and the delay of each of these realizable structures, using data from the standard cell library, and performs a "best match" search over the candidate gates, based on the optimization constraints (minimum area, minimum delay, a linear combination of both, etc.). The best match is retained and the corresponding input binding is preserved. This procedure is repeated over all nodes in the covering phase, in a primary output to primary input order, such that the fan-out nodes for a particular node are synthesized before mapping the node itself. This step is followed by global area recovery, and fanout-optimization, during which the gate sizes are altered. A buffer insertion step is also performed to further optimize the circuit (This step does not change the SP values of the existing nodes.). The NBTI-aware optimization strategy is itself independent of the exact synthesis methodology. Three different objectives are used to synthesize the circuits, namely:

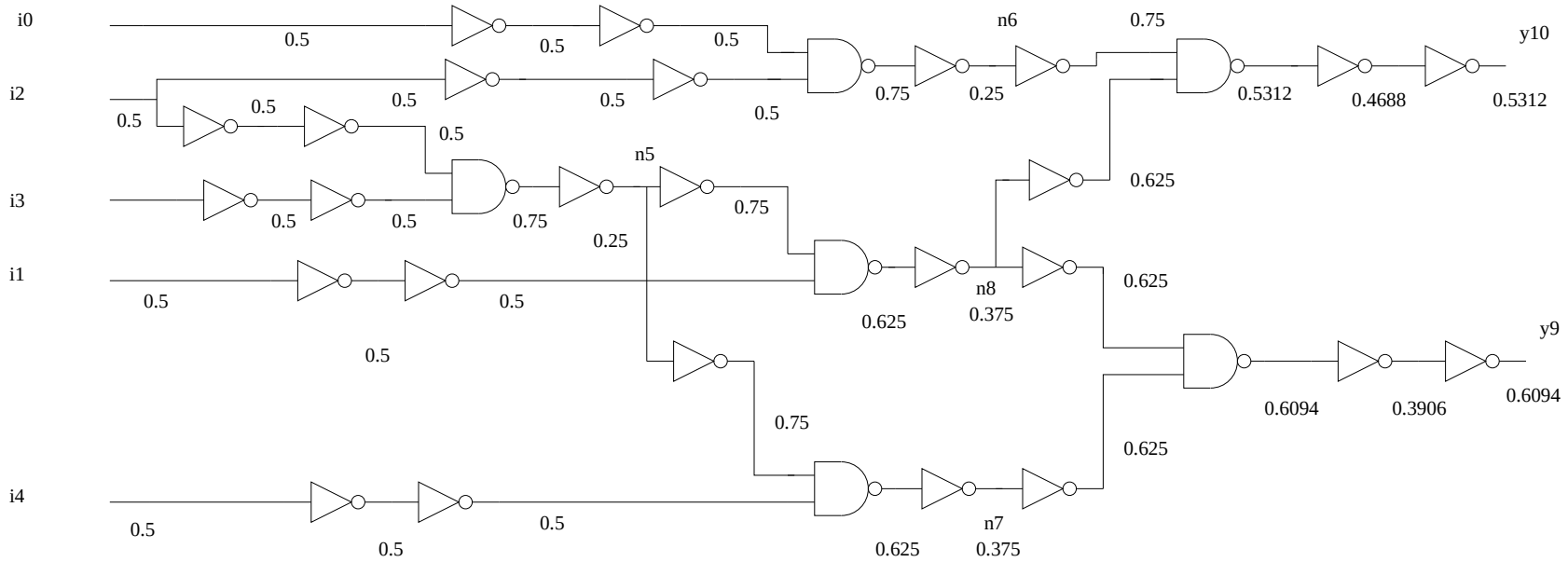


Figure 3.10: Subject graph for C17 using a NAND2-NOT representation.

- **Nominal Synthesis**: Technology mapping is realized using the nominal timing library, where the delay of each gate at time $t = 0$ is used. It must be noted that circuits designed using this method fail well before their lifetime, due to the NBTI induced temporal delay degradation.
- **Worst-case NBTI Synthesis**: Technology mapping is performed using the NBTI-affected timing library, where the worst-case delay of each gate computed after 10 years of continuous NBTI stress is used, instead of the nominal delay values.
- **SP based Synthesis**: Technology mapping is performed using the SP information at each node to choose the gate with the least area overhead to meet the timing requirement. The delay of the gates in the library as a function of SP, is precomputed in the form of linear best fit-curves.

The input parameters to the synthesis tool and the results of technology mapping, for each of these three cases for a given target delay, are described below, using C17 as the test circuit. The target delay is chosen as 70ps for this case.

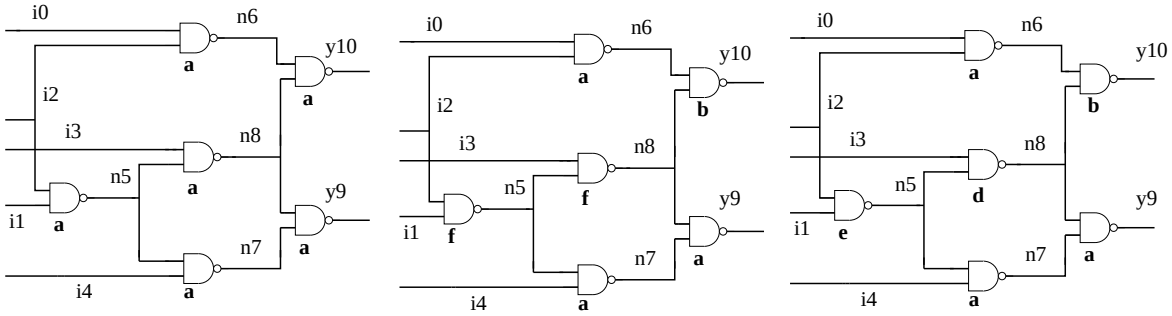


Figure 3.11: Results of technology mapping for C17 benchmark: (a) shows the result for nominal synthesis, which results in a circuit that fails with aging, while (b) shows the result for worst-case NBTI synthesis, and (c) indicates the result for SP based synthesis (Gate sizes increase as a,b,c,d,e,f).

Nominal Synthesis

This corresponds to the case where the nominal delay of each gate is used during technology mapping. The final result of the synthesizer is shown in Fig. 3.11(a). The area of the circuit, computed

as the sum of the widths of all transistors, is 7.4μm, and all the gates used in the circuit for mapping are minimum sized (of size "a"). Note that "a" represents the minimum size for a particular gate and "b", "c", "d", etc. represent gates of higher sizes. The nominal synthesis method is considered for comparison purposes only, since circuits designed using this scheme are not NBTI-tolerant and fail to meet the timing specifications after a certain period of time, due to the temporal degradation caused by NBTI.

Worst-case NBTI Synthesis

In this case, the rising delays of the gates in the library are replaced by their NBTI-affected value, after 10 years of continuous stress, while the falling delays remain unaltered. SPICE simulations are performed with the V_{th} value corresponding to constant DC stress in the V_{th}-SP look-up table, (i.e., V_{th} = -0.456V), and the corresponding delays are used in the timing library, instead of the nominal values. Expectedly, larger sized gates must be used to meet the timing required, resulting in higher area as compared with the nominal case. The mapped circuit for C17 is shown in Fig. 3.11(b). The size of each gate is shown in the figure, and it is evident that the gates along the critical path must be appropriately sized to meet the timing constraints. The final area of the circuit is 11.6μm.

SP based Synthesis

For the SP based synthesis, it is vital to propagate the SP information across all nodes based on the logic function being realized. This step is performed on the subject graph in SiS, which consists of a NAND2-NOT based decomposition of the circuit, as shown in Fig. 3.10. The SP for the primary inputs are assigned initial values, determined by RTL level simulations and statistical estimates (0.5 in our case), and these values are propagated along the various nodes in the subject graph in a PI-PO (primary input-primary output) order, as depicted in Fig. 3.10. The SP value at each node is marked in the figure. During technology-mapping, the SP values of the nodes are passed to a function that determines the delay of the various logic structures that realize the logic cone. The best-delay match is subsequently obtained and the corresponding gate that has the minimum NBTI-impact is chosen. The above step is repeated globally, until all nodes have been mapped to their best matches. The final mapped circuit for C17 is shown in Fig. 3.11(c). The area for this case is 9.8μm. The SP based

synthesis requires 15% lower area as compared with the worst-case NBTI synthesis, for a particular target delay.

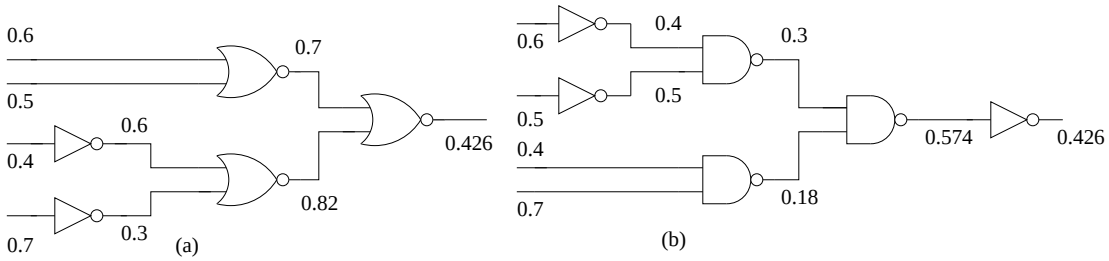


Figure 3.12: Two different implementations for a logic function, showing the stress probabilities of the nodes in each case.

Another advantage of SP-based synthesis is that it can help push nodes with large stress probabilities (small signal probabilities since transistors degrade when signals are at logic zero) inside the gates. Fig. 3.12 shows such an example where two different implementations realize the same functionality. The probability of each node being at logic zero (equal to (1-SP)) for the two implementations is also shown in the figure. In Fig. 3.12(a), nodes have higher degradation probabilities whereas in Fig. 3.12(b), the NAND-NOT based implementation pushes such nodes inside the gate, thereby minimizing the extent of temporal degradation due to BTI.

The experimental results for the SP based and worst-case synthesis methods, obtained over different ISCAS85 and LGSYNTH benchmarks, are presented in the next subsection.

3.6.4 Results

This subsection presents the results of technology mapping using the 65nm technology node [75] based library, for the worst-case NBTI, and SP based synthesis methods. The results are shown for some ISCAS85 and LGSYNTH benchmarks in Table 3.3. The target delay for each benchmark is set such that it lies in the region of the area-delay curve where the percentage change in the area is comparable with the percentage change in the delay, thereby providing scope for optimization. The area (calculated as the sum of the transistor widths of all gates) and the sum of active and leakage powers, are reported for each circuit. The columns titled “Savings” estimate the amount of area or power that can be recovered using the SP based synthesis as against using the worst case

NBTI synthesis. Most benchmarks result in better area and power when synthesized using the SP based method, as opposed to the worst-case NBTI synthesis. Although, technology mapping was performed to obtain a circuit with minimal area, the objective function can be modified to minimize the active power, leakage, etc. The table shows an average of 10% recovery in area and an average of 12% savings in power (active + leakage) for the benchmarks, due to significant reduction in the total device size, and capacitance. The results indicate that considering the SP values during technology mapping has a significant bearing on the circuit generated during logic synthesis.

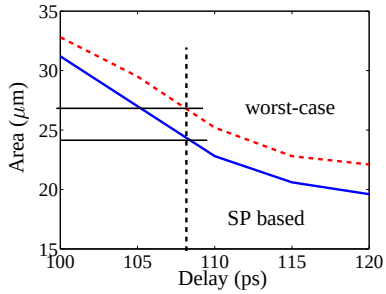


Figure 3.13: Area-delay curve for the benchmark b1.

The area versus delay curve for varying target delay values is shown for the SP based and worst-case NBTI synthesis methods for the LGSYNTH benchmark b1 in Fig. 3.6.4. It must be noted that the figure only shows the central linear region of the area-delay curve, where the percentage change in area is comparable with the percentage change in the delay. The target delay of the circuits is chosen to lie in this region, since efficient area-delay trade-offs can be achieved here.

Beyond this region, either the area or the delay overhead is large, thereby leading to a suboptimal design. The upper curve represents data for the worst-case NBTI synthesis, while the lower curve corresponds to SP based synthesis. In this region, the area of the SP based synthesis method is less than the area of the worst-case NBTI library based method by about 10%, for any target delay. Accordingly for a target delay of 108ps, 11% area savings can be obtained as seen from the figure.

In order to obtain a comparison of the reliability of the circuits synthesized using the three methods, namely nominal, worst-case NBTI, and SP based synthesis, timing simulations are performed on each of the three synthesized circuits on all benchmarks, at various time stamps. The threshold voltage at each time time stamp is computed, and the gate delays are characterized to obtain a

Table 3.3: Results of technology mapping for ISCAS85 and LGSYNTH benchmarks.

Benchmark	Target Delay (ps)	Worst-case NBTI Synthesis		SP based Synthesis			
		Area (μm)	Power (μW)	Area (μm)	% Savings	Power (μW)	% Savings
C17	70	11.3	0.8	9.8	12%	0.7	10%
C432	790	594.3	57.2	548.4	8%	52.7	8%
C499	648	1192.9	57.1	1075.7	10%	52.2	9%
C880	610	636.2	121.7	588.5	7%	107.8	11%
C1355	735	1282.2	122.0	1051.8	18%	99.2	19%
C1908	860	1234.6	122.5	1191.7	3%	117.2	4%
C2670	765	1347.1	127.7	1337.9	1%	127.5	0%
C3540	1100	2569.8	256.4	2057.4	20%	206.2	20%
C6288	3200	4356.2	448.0	3817.5	28%	387.4	14%
C7552	990	4009.9	409.0	3858.4	4%	394.2	4%
majority	110	19.2	1.6	16.4	14%	1.2	25%
b1	108	27.1	2.8	24.0	11%	2.2	23%
decod	151	143.4	11.9	118.9	17%	9.2	22%
cordic	297	162.9	13.1	152.1	7%	12.6	4%
alu2	923	760.2	74.3	691.3	9%	65.5	12%
apex6	365	1080.9	98.0	1044.2	3%	90.2	8%
des	620	8738.4	891.0	8657.1	1%	866.0	3%
alu4	940	1498.6	149.0	1302.1	13%	126.2	15%
too_large	545	1582.1	153.4	1511.0	4%	140.7	8%
vda	480	2088.0	243.1	1966.7	6%	222.6	8%
Average					10%		12%

library that corresponds to the NBTI induced degradation on the standard cells, at the given time stamp. The SP of all primary inputs are assigned to be 0.5, and the SP values at the intermediate nodes are calculated through Monte Carlo simulations, based on the method in [73]. Accordingly, the arrival times at the primary output nodes at different time stamps are computed. The results for C432 are shown in Fig. 3.6.4. The top curve shows the results for the nominal case, while the bottom curve shows the results for the worst-case NBTI synthesis case, and the middle curve shows the results for the circuit designed using SP based synthesis method. The results show that the delay of the benchmarks increases with time logarithmically, and the three curves are almost parallel to one another, implying that they all have the same asymptotic time dependence of $t^{\frac{1}{6}}$.

Since the target delay for C432 is desired to be 790ps, we assume that the circuits are no longer functional if the arrival time exceeds the target delay. Although the area of the circuit synthesized using the nominal case is less than that using the SP based synthesis method, the circuit becomes dysfunctional after 4×10^4s, ($\approx$ half a day) rendering it practically useless, whereas the circuit synthesized using the SP based method can sustain timing degradation up to 10 years. The circuit synthesized using the worst-case NBTI synthesis method is reliable for over 10 years, but this method overestimates the extent of temporal degradation, and hence leads to a design that requires a higher than necessary area and power. Thus, using the SP based synthesis method leads to an optimized circuit that minimizes the area and power overhead to ensure enhanced reliability up to 10 years.

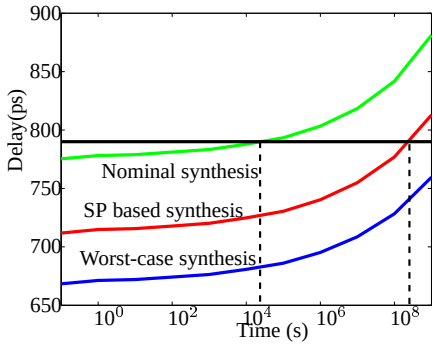


Figure 3.14: Figure showing temporal degradation of C432 due to NBTI.

3.7 Adaptive Techniques for Overcoming Performance Degradation due to Aging

Previous approaches to guardbanding a circuit and ensuring optimal performance over its lifetime, such as sizing [36, 77], and synthesis [40][5] can be classified as "one-time" solutions that add appropriate guardbands at design time. These methods are generally formulated as optimization problems of the type:

$$\begin{aligned} &\text{Minimize } \alpha \text{ Area } + \beta \text{ Power} \\ &\text{s.t. } D(0 \leq t \leq t_{\text{life}}) \leq D_{\text{spec}}, \end{aligned} \tag{3.10}$$

where α and β are weights associated with the area and the power objectives, respectively, while D_{spec} is the specified target delay that must be met at all times, up to the lifetime of the circuit, t_{life}.

Under the framework of (3.10), both the synthesis and sizing optimizations lead to an increase in area and power, as compared with a nominally designed circuit that is built to meet the specification only at birth, and not necessarily over its entire life. The work in [40] argues that synthesis can lead to area and power savings, as compared with sizing optimizations. However, guardbanding (through sizing or synthesis) is performed during design time, and is a one-time fixed amount of padding added into the circuit in the form of gates with a higher drive strength. Inevitably, this results in large positive slacks during the initial stages of operation of the circuit, and therefore, larger-than-necessary area and power overheads, in comparison with a circuit designed to exactly meet the specifications throughout its lifetime.

We also note that while BTI effects cause the transistor threshold voltages to increase, resulting in larger delays, higher V_{th} also implies lower subthreshold leakage ($I_{\text{sub}} \propto e^{\frac{-V_{th}}{mkT}}$). Therefore, both NBTI and PBTI cause the leakage of the circuit to decrease with time, thereby providing the opportunity to trade off this slack in leakage to restore the lost performance. Adaptive Body Bias (ABB) [78] provides an attractive solution to explore leakage-performance trade-offs. Forward body

[5]The work in [40] uses a BTI-aware delay model during the technology mapping phase of synthesis, and the circuit is mapped to a library such that its delay at the end of its lifetime, t_{life}, meets the specifications. In the context of this paper, we will refer to this approach as the "synthesis approach".

bias (FBB) can be used to speed up a circuit [79], by reducing the V_{th}, thereby using up the available slack in the leakage budget. Further, the amount of FBB can be determined adaptively, based on the exact temporal degradation of the circuit, and requisite amounts of body bias can be applied to exactly meet the target specifications under all conditions.

The main advantage of a body bias scheme is that the performance can be recovered with a minimal increase in the area overhead as compared with "one-time" approaches such as sizing and synthesis. While [80] demonstrated that ABB could be used to allow the circuit to recover from voltage and temperature variations as well as aging, we believe our work is the first solution to take advantage of the reduction in leakage due to bias temperature instability (BTI). We demonstrate how ABB can be used to maintain the performance of the circuit over its lifetime, by determining the appropriate PMOS and NMOS body bias values (and supply voltages) at all times. We use a look-up table whose entries consist of the optimal body bias and supply voltages, indexed by the cumulative time of BTI stress on the circuit.

Accordingly, we first propose an optimization algorithm to compute the entries of the look-up table, such that the delay specifications of the circuit are met throughout its lifetime and the power overhead is minimized. In contrast with the significant area cost for the synthesis-based method, the area overhead using this approach is limited to the look-up tables, body bias generation, and body bias routing networks and associated control circuitry, and is therefore minimal, while the power overhead is similar to that incurred by synthesis. Thus, we show that the adaptive compensation of circuit delay degradation due to aging provides a viable alternative to "one-time" fix techniques such as BTI-aware synthesis.

In the second approach, we propose an alternative hybrid formulation that combines adaptive techniques with synthesis. This iterative method first performs a power-constrained delay minimization through the application of FBB. This optimization recovers some amount of the performance degradation caused by aging by using the power slack that is created as the circuit ages. However, since this power-constrained optimization is not guaranteed to meet the delay specifications, technology mapping is used next to resynthesize the circuit to meet tighter timing specifications at birth. Using a new power specification, the iteration continues through alternate steps of FBB optimization and resynthesis until the timing specification is met. Our simulation results indicate that by

combining the merits of the adaptive and synthesis-based approaches, the resulting circuit meets the performance constraints at all times, with only a minimal expense in the area and power.

3.8 Background and Problem Formulation

We begin this section by determining the impact of NBTI on the delay and leakage of digital circuits. We then explore the potential of FBB to achieve power-performance trade-offs, and accordingly formulate an optimization problem.

3.8.1 Impact of BTI on Delay and Leakage

At the transistor level, the reaction-diffusion (R-D) framework [13, 14] has widely been used to determine the long-term impact of NBTI on the threshold voltage degradation of a PMOS device. Accordingly, the V_{th} degradation for a PMOS transistor under DC stress increases asymptotically with time, t, as $\Delta V_{th}(t) \propto t^{\frac{1}{6}}$ [18, 20, 32, 43]. We also use a PBTI model where the degradation mechanism is similar to NBTI, but the magnitude of V_{th} degradation is lower. Specifically, in our simulations, the ΔV_{th} for a PMOS device after 10^8seconds ($\approx$ 3 years) of DC stress is $\approx$ 50mV, while that for an NMOS device is $\approx$ 30mV. The corresponding nominal values of the threshold voltages, based on PTM 45nm model files [75], are -411.8mV for a PMOS device and 466mV for an NMOS device. Since NBTI affects the V_{th} of PMOS devices, it alters the rising delay of a gate. Similarly, PBTI, which affects NMOS transistors, changes the falling delay of a gate.

At the gate level, we derive models for the delay and the leakage as functions of the transistor threshold voltages. We assume the worst-case degradation [36] model for all gates in the circuit, for reasons that will become apparent in Section 3.9. The delay and leakage numbers for the degraded circuit are computed through SPICE simulations, at T = 105°C, at different times. Since BTI is enhanced with temperature, the library gates are characterized at the maximum operating temperature of the chip, assumed to be T = 105°C.

The results from the above SPICE simulations are curve-fitted to obtain models for the delay

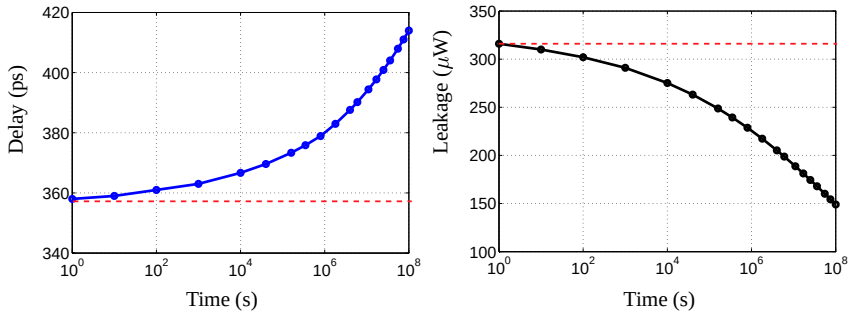
and leakage as a function of the transistor threshold voltages. The gate delay, D, is modeled as:

$$D(t) = D_0 + \sum_{\text{all transistors } i} \frac{\partial D}{\partial V_{th_i}} \Delta V_{th_i}(t) \tag{3.11}$$

where the sensitivity terms, $\partial D \partial V_{th}$, for each of the transistors in the gate, along the input-output path, are determined through a linear least-squares curve-fit. This first order sensitivity-based model is accurate, and has an average error of 1% in comparison with the simulation results, within the ranges of V_{th} degradation caused by BTI. Similarly, a model for leakage, L, can be developed as:

$$\log L(t) = \log L_0 + \sum_{\text{all transistors } i} \frac{\partial L}{\partial V_{th_i}} \Delta V_{th_i}(t) \tag{3.12}$$

Note that the $\Delta V_{th}(t), D_0,$ and L_0 values are functions of the supply voltage, V_{dd}. The leakage numbers are experimentally verified to have an average error of 5% with respect to the SPICE simulated values.



(a) (b)

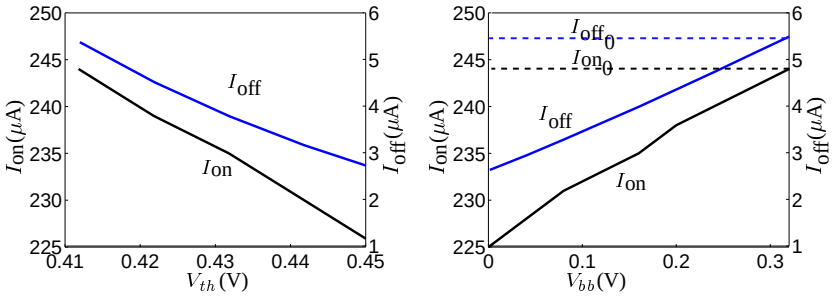
Figure 3.15: The impact of BTI on (a) the delay and (b) the leakage of the LGSYNTH93 benchmark, "des," as a function of time.

At the circuit level, Fig. 3.15 shows the impact of BTI on the delay and leakage of an LGSYNTH93 benchmark "des" as a function of time. The delay and leakage of the uncompensated circuit at $t = 0$, are shown by flat dotted lines on each plot. The results indicate that the delay degrades by around 14%, whereas the subthreshold leakage reduces by around 50%, after three years of operation. We ignore the contribution of gate leakage current here, since neither BTI nor FBB impacts the gate

leakage. Further, with the use of high-k dielectrics, gate leakage has been reduced by several orders of magnitude, making it negligible in comparison with the subthreshold and junction leakages.

3.8.2 Recovery in Performance using FBB

At first glance, one may imagine that by returning the threshold voltage to its original value, FBB could be used to fully recover any degradation in the PMOS/NMOS transistor threshold voltage, bringing the I_{on} and I_{off} values of the device to their original levels, thereby completely restoring its performance and leakage characteristics, as depicted in Fig. 3.16(b). The drain current I_{ds} for a PMOS transistor is plotted in Fig. 3.16(a) for two distinct cases, i.e., I_{on} when $V_{gs} = -V_{dd}$, and I_{off} when $V_{gs} = 0$, using different scales. Fig. 3.16(a) plots the currents as a function of PMOS V_{th}, showing the reduction in the currents due to aging. Fig. 3.16(b) shows the increase in the on and off currents with the amount of forward body bias (V_{bb}) applied, computed when the transistor is maximally aged. When a FBB of 0.32V is applied, this effectively sets V_{th} to V_{th_0}, and hence $I_{\text{on}} = I_{\text{on}_0}$, and $I_{\text{off}} = I_{\text{off}_0}$, where I_{on_0} and I_{off_0} are the nominal values. The change in junction capacitance and the subthreshold slope is assumed to be negligible within the ranges of the FBB voltages considered in this framework, based on the results in [79], and [81], respectively.



(a) Decrease in I_{ds} with increase in V_{th} due to NBTI.

(b) Increase in I_{ds} with FBB.

Figure 3.16: I_{on} and I_{off} characteristics (shown using different scales) for a PMOS device with NBTI and FBB.

However, on closer examination, it is apparent that this is not the case, due to the effect of the substrate junction leakage. The results of applying FBB on a temporally degraded inverter (after

three years of constant continuous stress on all the transistors) are shown in Fig. 3.17. Fig. 3.17(a) shows the average delay[6] of the inverter, measured as $\frac{1}{2}(\tau_R + \tau_F)$, where τ_R and τ_F are the output rise and fall times, respectively, plotted against the body bias voltage V_{bb}. Here, we apply an equal V_{bb} to all devices. The value of the delay at zero body bias represents the delay of the aged circuit. The horizontal dotted line represents the delay specification, and after three years of maximal aging, the circuit clearly violates this requirement. At this point, the application of a V_{bb} of $\approx$ 0.3V can restore the delay of the inverter to its original value.

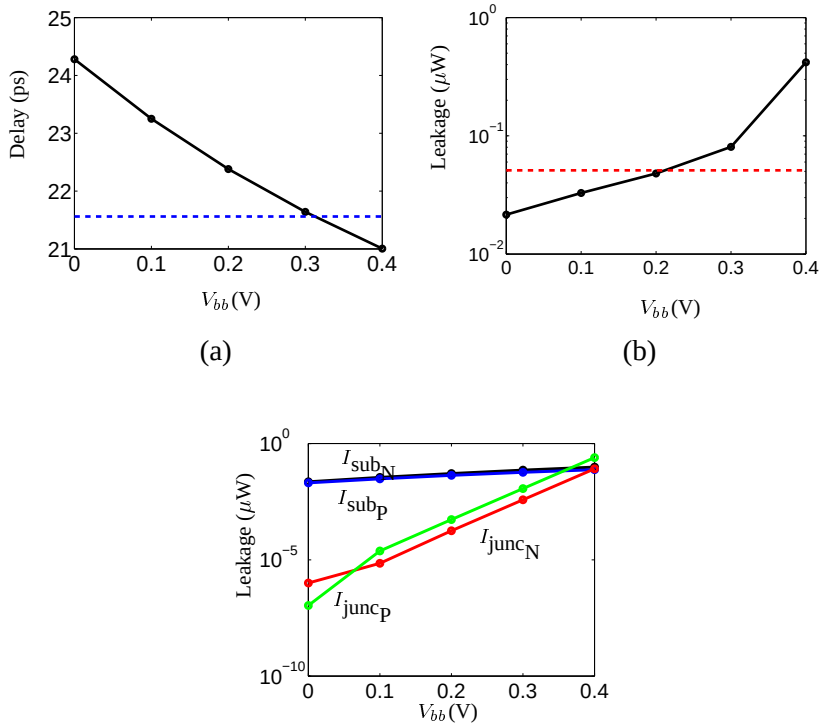


Figure 3.17: The impact of applying FBB to a degraded inverter at $t = t_{\text{life}}$ on its (a) delay (b) leakage. Leakage subcomponents are shown in (c).

Fig. 3.17(b) plots the corresponding total leakage power, consisting of the sum of the subthreshold leakage and the substrate junction leakage, under maximal V_{th} degradation of both the NMOS

[6]The trend of average of rise and fall delays mimics that of a path delay, since alternate stages of logic in a path (consisting of all inverting gates) undergo rising and falling transitions, respectively.

and the PMOS transistors, The leakage computed at $t = 0$, i.e., with ΔV_{th} = 0, shown by the horizontal dotted line, is chosen as the leakage budget: after three years of aging, the leakage value (shown at zero body bias) falls below this budget. The figure shows that with the application of V_{bb}, the leakage rises, and exceeds the budget at around 0.2V. In particular, the exponential increase in substrate junction leakages with FBB leads to a sharp increase in the leakage beyond a certain point. This is due to the exponential increase in substrate junction leakages with forward body bias, as shown in Fig. 3.17(c), which plots the individual components of leakage power, namely the subthreshold and junction leakages for the NMOS and PMOS devices, denoted as I_{sub} and I_{junc}, respectively. We ignore the contribution of gate leakage current to the leakage power overhead, since BTI and FBB both do not cause any impact on gate leakage. Further, with the use of high-k dielectrics, gate leakage has been reduced by several orders of magnitude, making it negligible in comparison with the subthreshold and junction leakages.

From Figs. 3.17(a) and 3.17(b), it can be inferred that a complete recovery in the delay degradation of the circuit could cause the leakage current to exceed its nominal value. Simulation results indicate that our benchmark circuits require FBB of the order of (0.3-0.4V), which leads to a large increase in the power dissipation, and can potentially exceed the available budget.

In other words, the sole use of ABB (FBB) to restore fully the performance of the circuit results in a substantial power overhead, particularly as we approach the lifetime of the circuit, where large values of FBB are necessary. The use of ASV in combination with ABB has been demonstrated to be more effective than using ABB individually [82]. Hence, we propose our first method, termed the "adaptive approach," that applies ASV in conjunction with ABB to minimize the total power overhead, while meeting the delay constraints throughout the circuit lifetime.

As we will see from the results in Section 4.9.2, while the adaptive approach provides area savings in comparison with the synthesis approach, the maximal power dissipation overhead is significant. Although ASV, when used in combination with ABB, tempers the exponential increase in junction leakages with FBB, the corresponding increases in V_{dd} cause the subthreshold leakage to increase exponentially, while the active power increases quadratically. Further, the amount of threshold voltage degradation has a second order dependence on the supply voltage, with larger V_{dd} leading to higher ΔV_{th} [19]. Our second approach further reduces the power dissipation by

combining the merits of the adaptive and synthesis approaches, thereby trading off area with power. In particular, it supplements the use of ABB with synthesis, instead of ASV as in the adaptive approach, yielding improved trade-offs. We refer to this as the "hybrid approach."

3.8.3 Adaptive Approach

Under the adaptive approach, the optimal choice of the values of the NMOS body bias voltage, v_{bn}, the PMOS body bias voltage, v_{bp}, and the supply voltage, V_{dd}, to meet the performance constraint is such that the total power dissipation at all times is minimized. An optimization problem may be formulated as follows:

$$\begin{aligned} \text{Minimize } P_{\text{act}}(t, V_{dd}) &+ P_{\text{lkg}}(t, v_{bn}, v_{bp}, V_{dd}) \\ \text{s.t. } D(t, v_{bn}, v_{bp}, V_{dd}) &\leq D_{\text{spec}} \\ 0 \leq t &\leq t_{\text{life}}, \end{aligned} \tag{3.13}$$

where P_{act} and P_{lkg} are the weighted active and leakage (subthreshold + junction leakage) power values, respectively, while D_{spec} is the timing specification that must be met at all times. It can be intuitively seen that a solution to the optimization problem in (3.13) attempts to maintain the circuit delays to be as close to (but still lower than) the specification as possible, since any further reduction in delay using ABB/ASV is accompanied by a corresponding increase in the active and leakage power dissipation.

3.8.4 Hybrid Approach

The hybrid approach uses a combination of adaptive methods and presilicon synthesis to optimize the circuit for aging effects. The use of ASV results in a quadratic increase in the active power; in contrast, at reasonable design points, synthesis can provide delay improvements with subquadratic increases in the power dissipation. Therefore, the hybrid adaptive approach is restricted to the use of ABB only, at the nominal V_{dd} value.

The hybrid approach employs synthesis and ABB in an iterative loop, tightly controlling the power increase in each step. For the ABB assignment step of the loop, the optimization formulation

in (3.13) is recast within a power envelope, as a problem of delay minimization subject to power constraints.

$$\begin{aligned} \text{Minimize} \quad & \max_{t \in [0, t_{\text{life}}]} D(t, v_{bn}, v_{bp}) \\ \text{s.t.} \quad & P_{\text{lkg}}(t, v_{bn}, v_{bp}) \leq P_{\text{lkg}}(t = 0) \\ & 0 \leq t \leq t_{\text{life}}, \end{aligned} \tag{3.14}$$

where $P_{\text{lkg}}(t = 0)$ denotes the leakage power budget. This budget is taken to be the peak leakage of the uncompensated circuit, i.e., its leakage at $t = 0$. Note that this effectively bounds the total power dissipation of the circuit to its value at $t = 0$, since the above optimization has a negligible effect on the active power dissipation.

The solution to the above optimization problem reduces the delay of the circuit under power constraints, but does not guarantee that the delays will be lower than D_{spec}. If this is the case, in a second step, the circuit is resynthesized to meet a heuristically chosen delay specification, tighter than D_{spec}, at $t = 0$. The iteration continues until the optimization in (3.14) can guarantee that the compensated circuit meets D_{spec} over its entire lifetime.

3.9 Control System for Adaptive Compensation

In this section, we investigate how an adaptive control system can be implemented to guardband circuits against aging. Prior work in this area can be summarized as follows. A look-up-table-based approach that precomputes and stores the optimal ABB/ASV/frequency values, to compensate for droop and temperature variations, is presented in [80]. An alternative approach [78, 79] uses a replica of the critical path to measure and counter the effects of within-die and die-to-die variations. Techniques for sensor design have been addressed in [34, 35], which propose high-resolution on-chip sensors for capturing the effects of aging.

However, with increasing levels of intra-die variations, critical path replica-based test circuits require a large number of critical paths to provide an f_{max} distribution that is identical to the original circuit, leading to an area overhead. Further, the critical paths in a circuit can dynamically change, based on the relative temporal degradation of the potentially critical paths. Adding every potentially

critical path from the original circuit into the critical path replica may cause the test circuit to become extremely large. Apart from a high area overhead, such a large test circuit may incur its own variations that may be different from those in the original circuit.

Owing to these drawbacks, we propose the use of a look-up-table-based implementation to determine the actual v_{bn}, v_{bp}, and V_{dd} values that must be applied to the circuit to compensate for aging. The entries in the look-up table are indexed by the total time for which the circuit has been in operation. This time can be tracked by a software routine, with $t = 0$ representing the beginning of the lifetime of the circuit after burn-in, testing, and binning. The degradation in delays due to accelerated stresses at high temperature during burn-in are accounted for in determining $D(t = 0)$, by adding an additional timing guardband. This software control enables the system to determine the total time for which the circuit has been operational.

The look-up table method requires the critical paths and the temporal delay degradation of the circuit to be known beforehand, to determine the entries of the table. It is impossible to determine, *a priori*, the exact temporal degradation of a circuit, since this depends on the stress patterns, which in turn depend on the percentage of time various circuit nodes are at logic levels 0 and 1. This percentage depends on the profile of computations executed by the circuit, and cannot be captured accurately by, for example, an average probabilistic analysis. The only guaranteed-pessimistic measure for BTI stress uses the worst-case degradation of the circuit. The method in [36] presents such a method, considering the impact of NBTI only, and determines the worst-case scenario by assuming maximal DC stress on every PMOS transistor. The idea can be extended to include maximal impact of PBTI on the NMOS transistors, as well, to compute the maximal degradation of the most critical path in the circuit. The worst-case method to estimate the maximal delay degradation after t seconds of aging is computationally efficient, is input-vector-independent, and requires a single timing analysis run based on the degraded NMOS and PMOS V_{th} values at t. Due to the fact that this is guaranteed-pessimistic over all modes of circuit operation, the set of v_{bn}, v_{bp}, and V_{dd} values in (3.13), determined using this number as a measure of $D(t)$ in this formulation, is guaranteed to ensure that the circuit meets the delay specification under all operating conditions.

The next sections describe the algorithms for the adaptive and the hybrid approaches to counter the impact of BTI. In Section 3.10, we first outline an algorithm for the adaptive approach to com-

pute the optimal tuple entries in the look-up table at different times. We then investigate how further area-power trade-offs can be achieved using the hybrid approach in Section 3.11, and describe the implementation.

3.10 Optimal ABB/ASV Computation for the Adaptive Approach

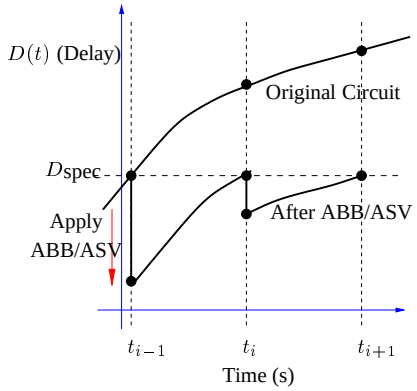


Figure 3.18: A plot of the delay of the nominally-designed circuit, without adaptation, as a function of time, showing degradation due to BTI effects, and a schematic of our compensation scheme using ABB/ASV at three consecutive compensation time points, t_{i-1}, t_i, and t_{i+1}, showing the delay of the compensated circuit as a function of time.

We will begin by pictorially illustrating the idea of the adaptive approach. Fig. 3.18 shows the temporally degraded delay, $D(t)$, of the original circuit without ABB/ASV, where the delay monotonically increases with time, and violates D_{spec} for some $t \geq t_{i-1}$. The figure shows how ABB/ASV may be applied at a time t_{i-1}, to ensure that the delay degradation during the interval $[t_{i-1}, t_i]$ does not cause the circuit delay to exceed the specifications. The delay of the circuit immediately after applying ABB/ASV, based on the look-up table values at t_{i-1}, is denoted as $D(t_{i-1})$, and is guaranteed to always be less than D_{spec}. Similarly, $D(t_i-)$ is the delay of the circuit just before applying ABB/ASV at t_i, and this typically touches D_{spec}. Considering the cumulative temporal degradation at $t = t_{i-1}$, the impact of ABB/ASV applied at that time point,

and the temporal degradation due to BTI over $[t_{i-1}, t_i]$, we have:

$$\begin{aligned} D(t_{i-1}) < D(t_{i-1}-) &\leq D_{\text{spec}} \\ D(t_{i-1}) < D(t_i-) &\leq D_{\text{spec}} \end{aligned} \tag{3.15}$$

At every compensation time point t_{i-1}, the amount of adaptation required is dependent on the delay degradation up to the next compensation time point t_i, and follows the shape of the figure in Fig. 3.18. In this figure, if no compensation is applied to the circuit, the delay during the interval $[t_{i-1}, t_i]$ will be above D_{spec}. To ensure that the delay meets specifications during this period, we apply a compensation at time t_{i-1}, whose magnitude is determined by the following result.

Theorem 1: Under small perturbations to the threshold voltage due to aging, let $D(t)$ be the delay of the aged circuit at any time t, and assume that under a specific compensation, $D(t_i-) > D_{\text{spec}}$ just prior to compensation time t_i. To bring $D(t_i-)$ to be under the specification, the value of $D(t_{i-1})$ can be adjusted, through compensation, to

$$D'(t_{i-1}) = D_{\text{spec}} \left(\frac{D(t_{i-1})}{D(t_i-)} \right) \tag{3.16}$$

Proof: For a MOS device, $V_{th} \propto t^{\frac{1}{6}}$, where the proportionality constant is different for NMOS and PMOS transistors. If we consider the effect of aging from time t_{i-1} to t_i, for a specific transistor type,

$$\frac{V_{th}(t_i)}{V_{th}(t_{i-1})} = \left(\frac{t_i}{t_{i-1}} \right)^{\frac{1}{6}} \tag{3.17}$$

Since the perturbations to V_{th} over this interval are small (by assumption), the delay of each gate can reasonably be assumed to vary linearly with V_{th}, as defined by a first-order Taylor series approximation. Therefore, the delay of each gate changes by a multiplicative factor, given by the right hand side of (3.17), implying that the delay of the circuit also changes by the same multiplicative factor. In other words, if the delay at time t_{i-1} is changed to $D'(t_{i-1})$

$$\frac{D'(t_i-)}{D'(t_{i-1})} = \frac{D(t_i-)}{D(t_{i-1})} \tag{3.18}$$

Since our goal is to set $D'(t_i-) = D_{\text{spec}}$, the result follows immediately. □

The adaptive strategy is developed at design time using the scheme shown in Algorithm 4, which shows the pseudocode for computing the optimal ABB/ASV values as the circuit ages. The algorithm begins by determining the amount of ABB/ASV that must be applied at the beginning of the lifetime of the circuit (after burn-in, testing, and binning), denoted by $t_0 = 0$, to compensate for aging until the first time t_1. This can be computed by determining the amount of change in the threshold voltage until t_1 (denoted as $\Delta V_{th}[t_0, t_1]$), and performing an STA run, to determine $D(t_1)$, as shown on lines 6-7 of the algorithm. The target delay after applying ABB/ASV is then computed, as shown on line 9, by applying the scaling factor from Theorem 1 to D_{spec}. As expected, $D(t_0) < D_{\text{spec}}$. Line 11 uses an enumeration scheme, based on the method described in [83], to determine the optimal ABB/ASV that must be applied at time t_0. Line 14 computes the delay of the circuit just prior to time t_1, i.e., $D(t_1-)$, which is less than D_{spec}. The method is repeated for successive values of t_i, and the look-up table entries are computed.

Algorithm 4 Adaptive approach: enumeration.

1: Determine the nominal ($t_0 = 0$) delay and power (active and leakage). By assumption, $D(t_0-) \leq D_{\text{spec}}$.
2: **for** $t_i = t_0, \ldots, t_{n-1}$ **do**
3: Set $V_{dd}(t_i) = V_{dd}(t_{i-1})$ if $i > 0$; else set $V_{dd}(t_i)$ to the nominal V_{dd} value.
4: **repeat**
5: Set $V = V_{dd}(t_i)$.
6: Compute $\Delta V_{th}[t_i, t_{i+1}]$ due to BTI assuming that in this interval, $V_{dd} = V$, and determine $V_{th}(t_{i+1})$.
7: Using static timing analysis (STA), determine $D(t_{i+1}-)$, the delay due to BTI just prior to time t_{i+1}.
8: {*Use $D(t_{i+1}-)$ to determine the target delay at t_i, upon the application of ABB/ASV.*}
9: Set $D(t_i) = D_{\text{spec}} \left(\frac{D(t_i)}{D(t_{i+1}-)} \right)$.
10: {*Determine ABB/ASV values to be applied at time t_i to meet $D(t_i)$.*}
11: Use an enumeration scheme, similar to [83], to solve the optimization problem in (3.13), i.e., determine (v_{bn}, v_{bp}, V_{dd}) for the interval $[t_i, t_{i+1}]$, such that the delay requirement $D(t_i)$ is met, and power is minimized.
12: **until** ($V_{dd}(t_i) == V$)
13: {*Compute the delay at the end of the interval.*}
14: Compute $D(t_{i+1}-)$ = $D(t_i)$ + temporal degradation over [t_i,t_{i+1}]. At this point, $D(t_{i+1}-) \leq D_{\text{spec}}$.
15: **end for**
16: Return the (v_{bn}, v_{bp}, V_{dd}) tuples at all times $t_0, \ldots, t_{n-1}$.

It should be pointed out that there is a second-order dependence between the level of V_{th} degra-

dation and V_{dd} [19]. The value of V_{dd} in the solution at t_i depends on the delay degradation over $[t_i, t_{i+1}]$, which in turn depends on the degradation in V_{th} during this interval, which is a function of the V_{dd} value at time t_i. Hence, an iterative approach is employed, as illustrated by the repeat loop.

The choice of the compensation time points depends on several factors. While we would like to continuously apply the requisite amount of compensation at all times, so as to just meet the performance constraints while minimizing the power overhead, in practice, the circuit can only be compensated at a finite number of time points, n. The number of compensating times chosen, (i.e., the size of the look-up table) and their specific values is limited by the following factors:

- *The resolution in generating the body-bias and supply voltages*: A large number of body bias and supply voltages require a sophisticated network of voltage generators and dividers, adding to the area and power overheads.

- *The minimum change in delay over* $[t_i, t_{i+1}]$*, subject to modeling errors*: Since the delay model has some inaccuracies, a control system with a large number of compensatory points, where the delay over a pair of such successive times changes very marginally, may lead to inaccurate computations, due to modeling errors[7].

- *The resolution of mapping each delay to a unique (*v_{bn}, v_{bp}, V_{dd}*) tuple*: Since there is a fixed discretization in the values of each element of this tuple, each compensation step will reduce the delay by a quantum, and finer-grained delay compensation is not possible.

Section 3.12.3 explores the impact of the number of compensating points chosen on the temporal profiles of the delay and power of the circuit.

3.11 Implementation of the Hybrid Approach

While the adaptive framework provides considerable savings in area as compared with synthesis, the power overhead over the original circuit can still be appreciably large, as will be shown in Section 3.12. This is due to the fact that the reduction in delay through FBB is obtained at the

[7] In this work, we select our times such that the delay changes by at least 1% in each interval.

expense of an exponential increase in leakage power, as seen in Fig. 3.17, while an improvement in performance through ASV also results in an exponential increase in leakage power ($I_{\text{off}} \propto e^{V_{dd}}$), as well as a quadratic increase in active power.

On the other hand, technology mapping can map the circuit to use gates with different functionalities and/or drive strengths. The use of this technique has empirically been seen to provide significant performance gains with low area and power overheads, for reasonable delay specifications. Hence, a combination of this synthesis technique with ABB has the potential to provide improved results.

Accordingly, we propose a hybrid approach to design reliable circuits. An iterative approach is followed during design, alternating between the ABB assignment and technology mapping phases, to ensure that the final design is reliable, and has minimal power and area overheads. The algorithm consists of two distinct phases, namely the adaptive compensation phase involving an optimization formulation subject to power constraints, and the resynthesis phase, involving technology mapping to meet a tighter design specification. Algorithm 5 describes the steps involved in this approach.

The algorithm begins with the adaptive compensation phase, where the ABB optimization formulation from (3.14) is solved. Lines 3-9 modify the framework of Algorithm 4 to compute the optimal (v_{bn}, v_{bp}) values at different time points, instead of the optimal (v_{bn}, v_{bp}, V_{dd}) tuple, such that the delay is minimized without violating the leakage power constraints. If the delay of the circuit throughout its lifetime[8] is less than the specification D_{spec}, then the optimization ends and the optimal (v_{bn}, v_{bp}) entries are used to populate the look-up table, as shown in lines 10-11.

However, if the delays are higher than D_{spec}, the circuit is technology-mapped to tighter design constraints, as shown in line 16. As a first order measure, the specification of the circuit is lowered from D_{spec} to $D_{\text{spec}} \times \frac{D_{\text{spec}}}{\max_{i=1}^{n} D(t_i-)}$, where $\max_{i=1}^{n} D(t_i-)$ is the maximum delay of the circuit over its lifetime, under the adaptive compensation scheme. If the leakage power of the circuit exceeds its budget value, the nominal value of the leakage power is updated, and this new value is used in (3.14) for $P_{\text{lkg}}(t = 0)$, as shown in line 17, and adaptive compensation is now repeated on this modified circuit. The process of adaptive compensation (lines 3-9) and technology-mapping for a tighter target delay (lines 13-16) is performed in an iterative manner, until the circuit delays converge, and

[8]Practically, this involves checking the values $D(t_i)$ only at each of the compensation times, t_i.

the timing specifications are met at all times. In practice, only a few iterations are necessary before the delay converges, as seen from our experiments.

As we will demonstrate shortly, our experimental results indicate that this approach provides savings in area as compared with the synthesis approach, and dissipates lower power in comparison with the adaptive approach.

Algorithm 5 Hybrid approach - iterative adaptive compensation and technology mapping.

1: Determine the original delay and leakage power at $t_0 = 0$. By assumption, $D(t_0-) \leq D_{\text{spec}}$.
2: Assume the leakage power at $t = 0$ to be the leakage budget during adaptive compensation phase of optimization.
3: {***Adaptive Phase***}
4: **for** $t_i = t_0, \ldots, t_{n-1}$ **do**
5: Compute $\Delta V_{th}[t_i, t_{i+1}]$ due to BTI at the nominal V_{dd}, and determine $V_{th}(t_{i+1})$.
6: Perform STA to determine the delay, $D(t_{i+1}-)$, due to BTI, just prior to time t_{i+1}, and determine the leakage power, $P_{\text{lkg}}(t_{i+1})$.
7: Use an enumeration scheme, similar to [83], to solve the optimization formulation in (3.14), i.e., to determine (v_{bn}, v_{bp}) so as to minimize the delay, $D(t_i)$, while staying within the leakage budget from line 2.
8: Determine the delay before applying FBB at the next time point, i.e., $D(t_{i+1}-)$.
9: **end for**
10: **if** all delays are $\leq D_{\text{spec}}$ **then**
11: The optimization has converged; output the computed FBB values to the look-up table.
12: **else**
13: {***Resynthesis Phase***}
14: Identify the highest $D(t_i-)$ and set $D_{\text{spec}} = D_{\text{spec}} \times \left[\frac{D_{\text{spec}}}{\max_{i=1}^{n} D(t_i-)}\right]$ to reduce D_{spec}.
15: {*Tighten the delay specification for synthesis to ensure that after aging and subsequent adaptive compensation,* $D(0 \leq t \leq t_{life}) \leq D_{spec}$.}
16: Perform technology mapping to resynthesize the circuit under the tighter delay specification, at $t = 0$.
17: If leakage power of this new circuit at $t = 0$ is greater than the original leakage budget computed in line 2, increase the budget accordingly.
18: Repeat from line 2.
19: **end if**

3.12 Experimental Results

We now present the results of applying our compensation scheme to circuits in the ISCAS85, LGSYNTH93, and ITC99 benchmark suites, synthesized on a 45nm [75]-based library. The body

bias voltage is altered in increments of 50mV, while increments of 30mV are used for the supply voltage.

3.12.1 Results on a Sample Benchmark Circuit

We present detailed experimental results on a representative LGSYNTH93 benchmark, "des," whose delay and leakage variations under BTI, without ABB/ASV compensation, were shown in Fig. 3.15.

Look-up Table Entries

Table 3.4: Look-up table entries for the LGSYNTH93 benchmark, "des," using the adaptive and hybrid approaches.

	Adaptive Approach						Hybrid Approach				
Time $\times 10^8$s	v_{bn} (mV)	v_{bp} (mV)	V_{dd} (V)	Delay (ps)	P_{act} (μW)	P_{lkg} (μW)	v_{bn} (mV)	v_{bp} (mV)	Delay (ps)	P_{act} (μW)	P_{lkg} (μW)
Nominal	0	0	1.00	355	641	327	0	0	355	641	327
0.0000	**0**	**50**	**1.03**	341	680	416	**0**	**0**	330	643	333
0.0001	**0**	**50**	**1.03**	341	680	346	**50**	**50**	334	643	332
0.0004	**0**	**100**	**1.03**	351	680	362	**0**	**100**	337	643	320
0.0016	**50**	**100**	**1.03**	351	680	369	**0**	**150**	338	643	333
0.0035	**0**	**50**	**1.06**	352	721	344	**0**	**150**	340	643	320
0.0080	**50**	**50**	**1.06**	351	721	357	**50**	**150**	339	643	329
0.0180	**50**	**100**	**1.06**	351	721	368	**50**	**150**	342	643	312
0.0400	**100**	**100**	**1.06**	352	721	377	**50**	**200**	343	643	328
0.0600	**0**	**100**	**1.09**	351	762	353	**50**	**200**	345	643	318
0.1100	**50**	**100**	**1.09**	351	762	360	**100**	**200**	343	643	326
0.1700	**100**	**200**	**1.06**	352	720	398	**100**	**200**	345	643	322
0.2500	**50**	**150**	**1.09**	352	762	362	**150**	**200**	343	643	328
0.3600	**50**	**200**	**1.09**	351	762	388	**150**	**200**	346	643	316
0.5500	**100**	**200**	**1.09**	351	762	396	**100**	**250**	351	643	325
0.7500	**50**	**150**	**1.12**	352	804	359	**100**	**250**	353	643	314
1.0000				355	804	350			355	643	305

Table 3.4 shows the entries of the look-up table that encodes the compensation scheme, and the delay, active, and leakage power numbers for the adaptive and the hybrid approaches. The circuit is compensated at different times, as shown in the first column of Table 3.4, up to its t_{life} of 10^8s. The

time-entries in the look-up table are chosen such that the increase in delay over any successive time-interval is uniform, and that the circuit is uniformly compensated for degradation, over its entire lifetime. A large starting value of t = 10^4s, is chosen for the adaptive approaches, since the BTI model for estimating the delay degradation of the circuit in Algorithm 4 is asymptotically accurate. Further discussion on the optimality of the selection of the number of time-stamps (n) to compensate the circuit, and its impact on the temporal delay-power curves is deferred to Section 3.12.3.

The remaining columns of Table 3.4 show the details of the compensation scheme. Columns 2-7 correspond to the adaptive approach, and show, for each compensation time, the (v_{bn}, v_{bp}, V_{dd}) tuples computed by Algorithm 4, the final delay after applying ABB/ASV, and the active and leakage power values. Columns 8-12 show the results for the hybrid approach and display, respectively, the optimal (v_{bn}, v_{bp}) pair, and the delay, active power, and leakage power at each compensation time. The first four columns of the table, (bold-faced, with a gray background), denote the actual entries that would be encoded into the look-up table for the adaptive approach, while the first, eighth, and the ninth columns denote the entries of the look-up table for the hybrid approach. The column, "Delay," denotes the delay of the circuit $D(t_i)$, at the given compensation time t_i, immediately after applying ABB/ASV values from the table.

The results indicate that the target delay is met at all time points, up to t_{life} = 10^8s, using both the approaches. The amount of compensation increases with time, as the circuit degrades due to BTI. With the adaptive approach, which optimizes the power under fixed delay constraints, a combination of ABB and ASV is used to counter the effects of aging, on the original design, whose delay and power values are shown in the row labeled "Nominal." The active and leakage power values vary as a function of time, depending on the optimal solution chosen at each time point. As explained in Fig. 3.18, the circuit is compensated for aging right from the first time period [0,t_1], by applying ABB/ASV at time $t = 0$. Hence, the delay of the circuit at $t = 0$ in the look-up table is less than D_{spec}. The leakage power decreases temporally due to increase in V_{th} caused by BTI, but increases with ABB/ASV, and in our scheme, it is seen to exceed the nominal leakage. For the hybrid approach, which uses a combination of ABB and synthesis, the circuit at $t = 0$ achieves its delay reduction purely through synthesis. It can be seen that the area overhead of synthesis in this case is low: as compared to the nominal case, the active power increases by 0.3% and the

leakage power by 1.8%. The results indicate that the power numbers using the hybrid approach are significantly lower than that using the adaptive approach.

Comparison of Transient Power and Delay Numbers

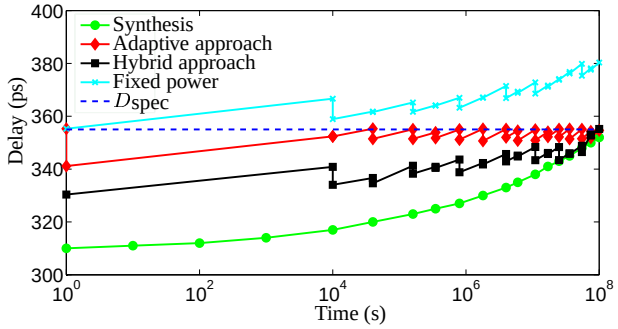
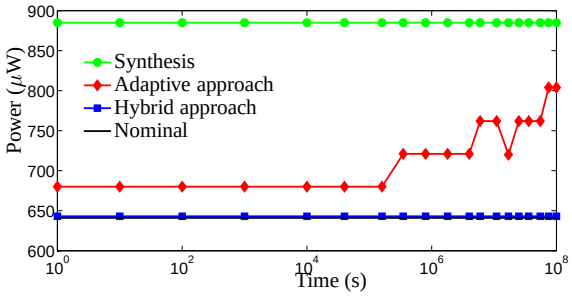


Figure 3.19: Temporal delay of benchmark "des" using different approaches.

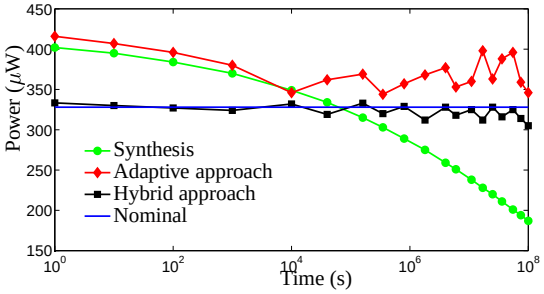
The temporal variation in the delay of "des" is shown in Fig. 3.19. The delay of the circuit, as a function of time, is shown for:

- the adaptive method from Section 3.8.3, where the delay can be seen to always be close to D_{spec}.
- the synthesis-based method from [40], where worst-case BTI-based library gate delays were used during technology mapping to synthesize the circuit: in this case, the delay increases monotonically with time.
- the fixed power case, corresponding to the results of solving the optimization problem in (3.14), where the delay is minimized through ABB under a power budget, set to the power at $t = 0$: this curve does not satisfy the delay specification.
- the hybrid method from Section 3.8.4, which satisfies the delay specification throughout its lifetime, and essentially corresponds to finding a power specification for a fixed power curve that meets D_{spec} at the end of the circuit lifetime. In this case, the power specification implies that the circuit is mapped to meet a delay specification of 330ps at $t = 0$.

All methods were targeted to meet the same delay specification, $D_{\text{spec}} = 355$ps, throughout the circuit lifetime and this value is shown by a horizontal line in the figure. This delay corresponds to the nominal delay of the original circuit at $t = 0$.



(a) Active power



(b) Leakage power

Figure 3.20: Temporal active and leakage power values of "des" using the various approaches.

Figs. 3.20(a) and (b), respectively, compare the values of the active and leakage power for the three approaches (adaptive, hybrid, and synthesis). The horizontal line marked "Nominal" represents the power dissipation of the original circuit at $t = 0$. Since the synthesis approach performs technology mapping for a tighter delay specification at birth, leading to a large area, as compared with the nominal design, the active power for the synthesis approach is constant over the lifetime of the circuit. For the adaptive approach, the supply voltage generally (but not always) increases gradually with time, as shown in Table 3.4. Correspondingly, the active power increases almost monotonically, as shown in Fig. 3.20(a). One exception to the monotonicity of V_{dd}, as seen from

Table 3.4, is at t = 0.17$\times 10^8$s, where the optimal (v_{bn}, v_{bp}, V_{dd}) tuple leads to a decrease in V_{dd} accompanied by a larger increase in (v_{bn}, v_{bp}), with respect to the solution at the previous time point, hence causing the active power to decrease temporally, as seen in the figure. The figure indicates that the maximum active power dissipated using the adaptive approach is less than that for the synthesis-based design.

Table 3.5: Area and power overhead comparison for adaptive and hybrid approaches.

	Original design				Adaptive approach		Hybrid approach					Synthesis		
Bench-mark	D_{spec}	$\frac{\Delta D(t_{\text{life}})}{D_{\text{spec}}}$	Nominal ($t = 0$s)		Overhead		Delay		Overhead			Overhead		
	(ps)	%	P_{lkg_0} (μW)	P_{act_0} (μW)	Max P_{lkg} %	Max P_{act} %	$(t = 0)$s (ps)	Reduction	Area %	Max P_{lkg} %	Max P_{act} %	Area %	Max P_{lkg} %	Max P_{act} %
b14	1078	14%	426	775	14%	26%	1027	0.95	2%	2%	1%	19%	16%	17%
b15	902	13%	781	1384	26%	19%	839	0.93	4%	4%	4%	16%	15%	18%
b17	1255	15%	3242	1790	26%	23%	1177	0.94	1%	1%	1%	22%	26%	22%
b20	1125	16%	1745	919	19%	31%	1058	0.94	2%	2%	3%	17%	18%	17%
C2670	510	15%	52	94	29%	26%	487	0.96	2%	3%	2%	32%	15%	19%
C3540	769	14%	74	136	30%	25%	724	0.95	2%	3%	4%	32%	38%	37%
C5315	729	15%	114	208	29%	19%	697	0.96	1%	1%	1%	14%	25%	18%
C6288	2190	14%	264	182	13%	28%	2110	0.96	7%	8%	5%	57%	63%	48%
C7552	616	15%	190	337	29%	19%	592	0.96	1%	1%	1%	18%	15%	19%
dalu	560	12%	227	127	12%	28%	535	0.96	1%	1%	1%	19%	19%	27%
des	355	15%	327	641	27%	25%	332	0.93	1%	3%	2%	35%	28%	38%
i8	840	17%	157	305	26%	25%	789	0.94	1%	3%	3%	18%	44%	71%
i10	830	14%	152	307	32%	25%	787	0.95	2%	2%	3%	21%	28%	26%
t481	368	14%	201	572	16%	26%	345	0.94	2%	3%	3%	38%	30%	39%
Average		15%			23%	25%		0.95	2%	3%	2%	26%	27%	30%

Similarly, Fig. 3.20(b) compares the leakage of the various approaches over the lifetime of the circuit, with respect to its nominal value. The leakage of the synthesis-based circuit is highest at $t = 0$ (when there is no BTI), but monotonically decreases with time. In contrast, the adaptive approach tries to adaptively recover performance, at the expense of increased power. The corresponding overhead implies that the leakage power for this method increases beyond its nominal value. Note that the leakage for the adaptive circuit at $t = 0$ is also greater than the nominal value, since some amount of ABB/ASV is applied to the circuit to guardband against temporal degradation during $[0, t_1]$, as shown in Fig. 3.18. The maximum leakage power (at $t = 0$) at any time point using our approach is almost identical to that using the synthesis method, as seen from Fig. 3.20(b).

For the hybrid approach, which uses a combination of synthesis and adaptive compensation, the results provide improvements over these two methods, used separately. As shown in Fig. 3.20(a) and (b), respectively, the active and leakage power at $t = 0$ increase very minimally (by less than 2%), as compared with the corresponding values for the original circuit, due to an increase in the area of the circuit during resynthesis. Subsequent adaptive compensation over the lifetime of the circuit is performed under fixed power constraints to ensure that the power never exceeds its value at $t = 0$. Hence, the curves for the overall leakage and active power, as functions of time, are closest to their corresponding budgets.

To illustrate how the hybrid approach works, let us consider the LGSYNTH93 benchmark "des". The nominal delay of the circuit is 355ps, and the leakage power at $t = 0$ is 327μW. We begin to apply the hybrid approach described in lines 4–9 in Algorithm 5 on "des". Using the fixed-power optimization formulation in (3.14), at $t = t_{\text{life}}$, the delay of the circuit reduces only to 380ps (from 415ps) without violating the leakage power budget of 327μW, and the active power budget of 641μW. Hence, in order to meet the final target delay, the circuit is resynthesized by setting D_{spec} to $\frac{355\times355}{380}$ = 332ps. Technology mapping is performed again, and the resulting circuit now has a 1% higher area overhead, 2% higher leakage overhead, and 3% higher leakage power overhead. We reapply the fixed-power optimization algorithm on this modified circuit, and the lifetime delay of the circuit subject to the new leakage power constraint of 327$\times$1.02μW is 354ps, which still meets the desired target. Thus using a combination of adaptive compensation and resynthesis, the circuit is optimally designed.

3.12.2 Area and Power Trade-offs

In this section, we compare the trade-offs in area and power for various approaches proposed in this paper, for the five largest benchmark circuits from ISCAS85 and LGSYNTH93 suites, as well as some large ITC99 benchmarks. Table 3.5 presents the area savings and the maximal power overhead of the adaptive and hybrid approaches, in comparison with the synthesis method. The column D_{spec}, is the delay of the original circuit at $t = 0$. The active (denoted as P_{act_0}) and leakage (tabulated as P_{lkg_0}) power values of the uncompensated circuit shown in the table denote their maximal numbers over the lifetime operation of the circuit. The column $\frac{\Delta D(t_{\text{life}})}{D_{\text{spec}}}$ denotes the percentage increase in delay due to maximal BTI after t_{life} (10^8s) seconds of stress. The percentage increase in the **maximum** leakage and active power values dissipated over the time interval [0,t_{life}], and the overhead in area, over the original design, are shown in the table, for the three approaches.

Table 3.5 indicates that the synthesis approach has a large average area overhead of 26%. However, the area overhead of the adaptive approach is restricted to the look-up tables, voltage generators for the additional supply voltages, and the body-bias voltages, and is therefore significantly smaller. The work in [78] has shown that this overhead is within 2-3% of the area of the original design. Thus, the adaptive approach provides significant area savings as compared with synthesis.

During optimization using the hybrid approach, the resynthesis (technology mapping) phase causes an increase in the area of the circuit, since the circuit is remapped to tighter specifications. The column "Reduction" in Table 3.5 indicates that using the hybrid approach, the target delay (at $t = 0$) during the technology mapping phase is only 5% lower than the nominal delay of the circuit, whereas the target delay (at $t = 0$) using BTI-aware synthesis is $\approx$ 15% less than the nominal delay. Expectedly, this small decrease in delay of $\approx$ 5% can be obtained with a marginal penalty in area (average value of the order of around 2%) for most circuits[9]. Hence, this overhead in area is extremely small, particularly when compared with that using synthesis.

The power numbers shown in the table indicate that while the adaptive and synthesis approaches have large power overheads, the power overhead using the hybrid approach is extremely small, with an average increase in active and leakage powers of the order of around 2-3%, over the wide range

[9]In reality, the area overhead is slightly higher, since the overhead in creating wells for body biasing, the look-up tables, and additional control circuitry must be considered. Nevertheless, the area overhead is still lower than that using synthesis.

of benchmarks tested. Thus, by combining the advantages of adaptive compensation and BTI-aware synthesis, we obtain an optimal final design whose area overhead is lower than that of the synthesis based approach, while the power overhead is lower than that of the adaptive approach, for the same delay specifications.

3.12.3 Optimal Selection of Look-up Table Entries

For the adaptive approach, the size of the look-up table (the number of entries) can be chosen according to various criteria, as discussed in Section 3.10. In this section, we investigate the impact of the size of the look-up table on the power and delay of the compensated circuit, using the adaptive approach. Accordingly, we perform simulations where the circuit is compensated at eight time points, instead of the 15 times chosen in Table 3.4. The compensation time points correspond to alternate entries from the look-up table in Table 3.4, and the corresponding (v_{bn}, v_{bp}, V_{dd}) tuples, found using Algorithm 4, are shown in Table 3.6.

Table 3.6: Look-up table entries for "des" using a coarse-grained adaptive compensation with fewer time entries.

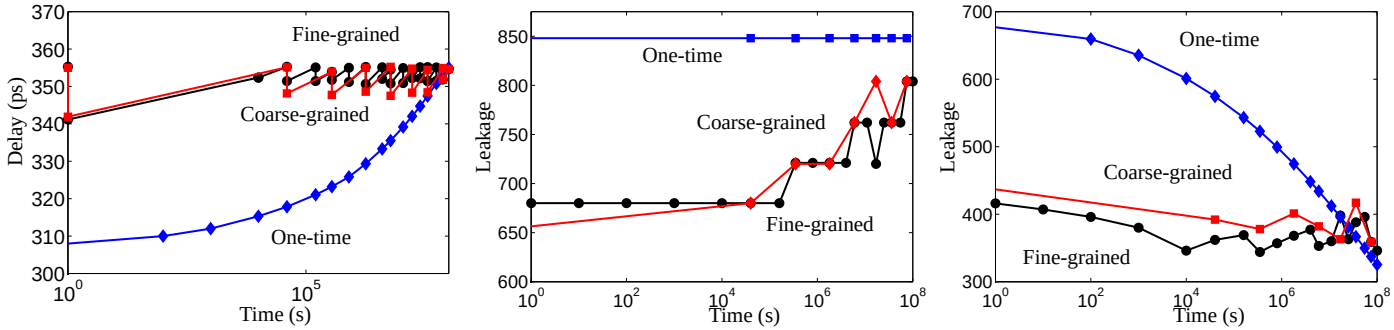
Time $\times 10^8$s	v_{bn} (mV)	v_{bp} (mV)	V_{dd} (V)	Delay (ps)	P_{act} (μW)	P_{lkg} (μW)
0.0000	**50**	**150**	**1.00**	342	641	466
0.0004	**50**	**100**	**1.03**	348	680	391
0.0035	**0**	**100**	**1.06**	348	720	378
0.0180	**100**	**100**	**1.06**	349	720	401
0.0600	**50**	**100**	**1.09**	348	762	381
0.1700	**0**	**100**	**1.12**	348	804	363
0.3600	**100**	**200**	**1.09**	348	762	417
0.7500	**50**	**150**	**1.12**	352	804	358
1.0000				355	804	340

As expected, the results indicate that the delay of the circuit is still met at all times, but the optimal (v_{bn}, v_{bp}, V_{dd}) tuples, and the corresponding delay and power values, are different from the corresponding values in Table 3.4. We compare these values by plotting the delay and power as functions of time in Fig. 3.21. We refer to the adaptive approach with 15 entries in the look-up table as the "Fine-grained" method, and that with eight entries as the "Coarse-grained" method. We also consider an extreme coarse-grained approach, where ABB/ASV is only applied at $t = 0$, to ensure

that the circuit meets its delay specifications over its lifetime: this can be considered as a look-up table with only one entry, at $t = 0$, and is referred to as the "One-time" approach. Fig. 3.21(a) shows the delays for all three of these approaches as a function of time.

By design, all methods meet the delay specification over the circuit lifetime, but as the granularity becomes coarser, the variation in circuit delay over time becomes larger, since the incremental delay degradation in each interval is higher, requiring larger changes to the (v_{bn}, v_{bp}, V_{dd}) tuple at each compensation time point, leading to larger swings for $D(t_i)$ below D_{spec}.

The active and leakage power profiles for the three cases are shown in Fig. 3.21(b), and Fig. 3.21(c), respectively. These trends show that the peak power dissipation of the circuit over its lifetime, for both the active and leakage power, increase as the granularity becomes coarser. The fine-grained approach used in our work, with 15 compensation time points, therefore satisfies the requirements laid out in Section 3.10, while maintaining a small overhead in terms of the circuitry required for its implementation.



(a) Delay (b) Active power (c) Leakage power

Figure 3.21: Temporal delay, active and leakage power of “des” for the adaptive approach, showing the impact of the number of entries in the look-up table.

3.13 Impact of NBTI on SRAM and Design for Reliability

While the temporal degradation of static CMOS circuits can be offset by transistor sizing [36] or other methods during design (to account for the decrease in the drain current of the PMOS devices due to NBTI), memory circuits pose a much greater challenge. Area-speed trade-off solutions do not work efficiently for SRAM arrays since area is a much greater concern in memory design as compared to digital CMOS or analog circuit design. Previous literature that dealt with the effect of NBTI on SRAM cells, such as [9] and [84], measured the extent of degradation of SNM due to NBTI with respect to a reduction in V_{dd}. However, our work, the first of its kind, focuses on the temporal SNM degradation of SRAM cells due to NBTI. We first present simulation results that show the impact of NBTI on an SRAM device as shown in Fig. 3.22, due to the temporal degradation of PMOS devices.

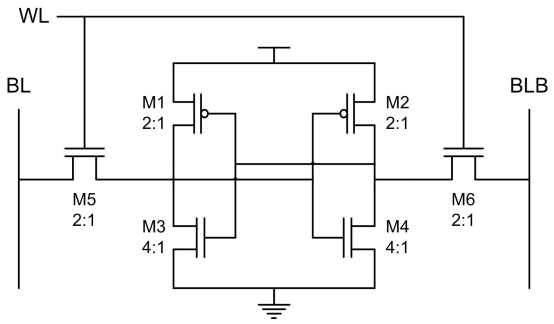


Figure 3.22: Six transistor SRAM cell.

3.13.1 Impact of NBTI on SRAM Cells

We perform transistor level simulations on the SRAM cell shown in Fig. 3.22 using BPTM 70nm and 100nm models [85], using the NBTI model from Section 2.1. The three main parameters analyzed are read-delay, write-delay and static noise margin (SNM) which is a measure of read stability. The results are tabulated in Table 3.7 for both 100nm and 70nm devices.

It can be seen from the tables that the read delay is virtually unaffected, the write delay improves marginally, while the SNM of the SRAM cell decreases due to NBTI. The SNM degradation as a function of $|V_{th}|$ is plotted in Fig. 3.23(a)-(b) for the 100nm and 70nm devices respectively. It can

Table 3.7: Performance degradation (after $10^6 s$) for a 100nm SRAM cell ($V_{th} = -0.303V$) and 70nm SRAM cell ($V_{th} = -0.22V$) with $V_{dd} = 1V$ at $T = 110^\circ C$.

	100nm cell		70nm cell	
Parameter	nominal	NBTI affected	nominal	NBTI affected
Read delay	187.6ps	187.6ps	186.6ps	186.1ps
Write delay	45.69ps	45.31ps	40.95ps	40.68ps
SNM	0.1278V	0.1254V	0.1007V	0.0987V

be seen from Table 3.8 that NBTI increases the threshold voltage, which leads to a gradual reduction in the SNM. This can lead to read stability issues and can potentially cause failures.

Table 3.8: SNM degradation for 100nm and 70nm SRAM cells with $V_{dd} = 1V$ at $T = 110^\circ C$.

	100nm cell		70nm cell	
time (s)	V_{th} (V)	SNM (V)	V_{th} (V)	SNM (V)
0	-0.303	0.1278	-0.22	0.1007
1	-0.3039	0.1277	-0.2207	0.1007
10^1	-0.3047	0.1277	-0.2212	0.1007
10^2	-0.3060	0.1277	-0.2222	0.1007
10^3	-0.3083	0.1277	-0.2239	0.1007
10^4	-0.3125	0.1273	-0.2270	0.1005
10^5	-0.3200	0.1267	-0.2324	0.0999
10^6	-0.3333	0.1254	-0.2420	0.0987
10^7	-0.3569	0.1227	-0.2591	0.0964
10^8	-0.3988	0.1174	-0.2896	0.0915

3.13.2 Recovering Static Noise Margin in SRAM Cells

As seen in the previous chapter, the generation of interface traps due to negative bias is also accompanied by a process of annealing of these traps when the negative bias applied at the gate is removed Thus, if the voltage applied at the gate of the PMOS device is regularly switched, dynamic recovery of threshold voltage occurs and thereby significant amount of performance can be recovered. This concept of performance recovery due to the application of periodic stress and relaxation on the gate of the PMOS device can be used to improve the SNM of the SRAM cell. Due to the topology of SRAM cells, one of the PMOS transistors is always turned on while the other one is turned off. Only one of the PMOS transistors is affected by NBTI if the cell contents are not modified. Hence,

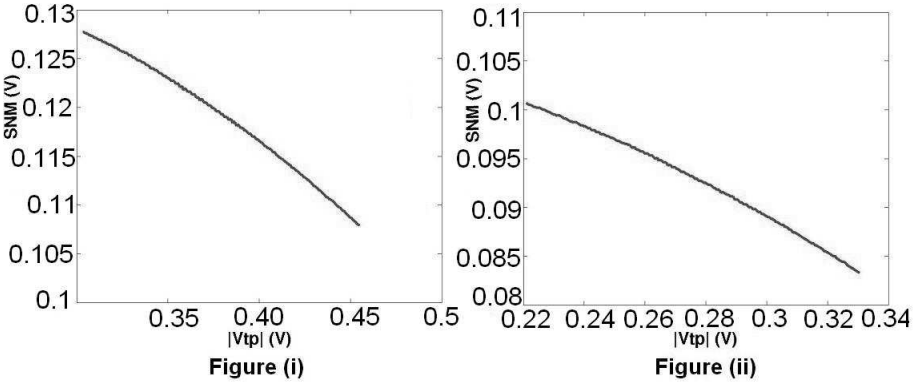


Figure 3.23: SNM versus $|V_{th}|$ for a SRAM cell simulated at V_{dd}=1.0V and $T = 110^{\circ}C$ using (i)100nm and (ii)70nm BPTM technology.

Table 3.9: Static noise margin for the SRAM cell.

	$t = 0$	$t = 10^8$ seconds				
		Non-cell flipping		Cell flipping		
device	SNM	SNM	Δ SNM	SNM	Δ SNM	Recovery
100nm	0.1278	0.1174	-0.0104	0.1205	-0.0703	30%
70nm	0.1007	0.0915	-0.0092	0.0944	-0.0063	29%

if we periodically flip the contents of the cell, the effect can be balanced out. This idea is similar in theory to applying AC stress on the PMOS transistors as opposed to DC stress, as seen in Fig. 2.30.

As can be seen from Table 3.8, NBTI is a fairly slow mechanism and the amount of degradation in SNM is noticeable only after 10^5 seconds ($\approx$ 1.16 days). Hence, it is adequate to flip the contents of the cell at a frequency of once a day. We hereby present simulation results for our SRAM cell, assuming a flipping rate of 10^5 seconds. Using the NBTI model in Section 2.1, the V_{th} values, can be calculated as a function of time. A plot of $|V_{th}|$ versus time is shown in Fig. 3.24(i)-(ii) for the 100nm and 70nm cells respectively. The V_{th} values are calculated at different time intervals and a look up table of SNM versus V_{th} is built by simulating the SRAM at each value of V_{th}. The SNM can also be plotted as a function of time and the plot for both the cell flipping and non-cell flipping case is shown in Fig. 3.25(i)-(ii) for 100nm and 70nm devices respectively. It can be seen from Table 3.9 that cell flipping (at an interval of 10^5 seconds) reduces the amount of SNM degradation by 30% for both 100nm and 70nm devices after 10^8 seconds (≈ 3 years).

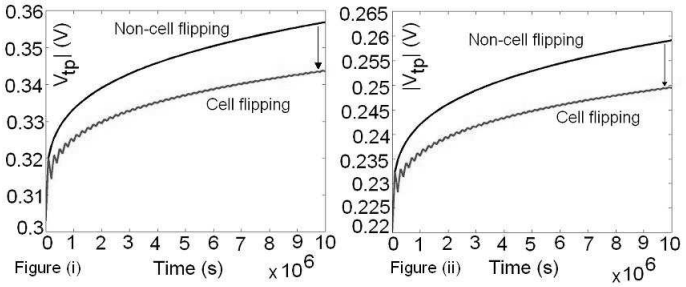


Figure 3.24: $|V_{th}|$ versus t for periodic stress and relaxation with $\tau = 10^5$ seconds for (i) 100nm and (ii) 70nm SRAM cell.

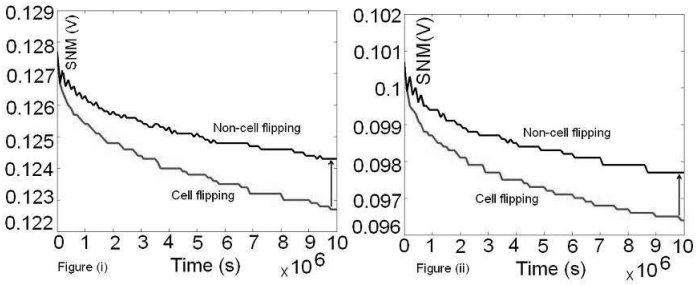


Figure 3.25: SNM versus time for cell flipping and non-cell flipping case for (i) 100nm and (ii) 70nm cell.

3.14 Implementation of Cell Flipping in SRAM Arrays

In this section, we provide an overview of the implementation of cell flipping in SRAM arrays, which are typically present in cache blocks in a processor. The two main aspects of SRAM cell flipping are the ability to flip the contents of all cells periodically, and the ability to read and write data correctly during normal course of operation, and these are explained in the subsections below.

3.14.1 Periodic Flipping of SRAM Cells

Flipping the contents of all cells every 10^5 seconds can be performed either through software or hardware.

Software Approach:

In the software approach, a subroutine is written in the system to interrupt the normal operation of the processor every 10^5 seconds. The subroutine runs from the first addressable location of the SRAM array till the last location and generates an incremental address. The data from the array is first read into the processor registers. This data is inverted and written back to the same address. The address is then incremented and the loop runs till all the contents are flipped. The advantage of this approach is that it has zero hardware overhead and can be programmed into existing processors by writing a subroutine that runs every 10^5 seconds. However, for large caches (say L3 caches) which are far away from the processor, access to a single location may take about 100 clock cycles. To read the data from every address, invert it and write back takes more than 200 clock cycles per location. It is unrealistic to run this subroutine over the entire cache since typically L3 caches are often more than a few megabytes in size.

Hardware Approach:

In the hardware approach, the SRAM array is embedded with additional hardware and control signals as shown in Fig. 3.26. A one-bit control signal (Flip) with two mutually exclusive states to indicate cache data access available to the processor and cache data being flipped is used. During a normal data access in the cache, the address is placed on the address bus and is sent to the ad-

dress decoder through S2 and the corresponding word lines on S3 are activated. In case of a read operation, the data is read from the SRAM array (S4), and the output of the sense amplifiers is the final read-data sent to the processor. Writes proceed in a similar manner except that the write-data is placed on the write-data bus (S7) and is written into the cache. For cell-flipping, the read-data after the sense amplifiers (S6) is inverted and the negated data (S5) is multiplexed with the actual processor write-data. A counter capable of generating consecutive addresses (S1) is designed such that the successive word lines are activated. The data from the bit lines are read, inverted, placed on the write-data bus and are written back to the same location through S7. The counter clock cycle is equal to the read access time, plus the write access time, plus the overhead in inverting the data and placing it on the write-data bus. This process is continued until all addresses are accessed and the entire data inside the cache is flipped.

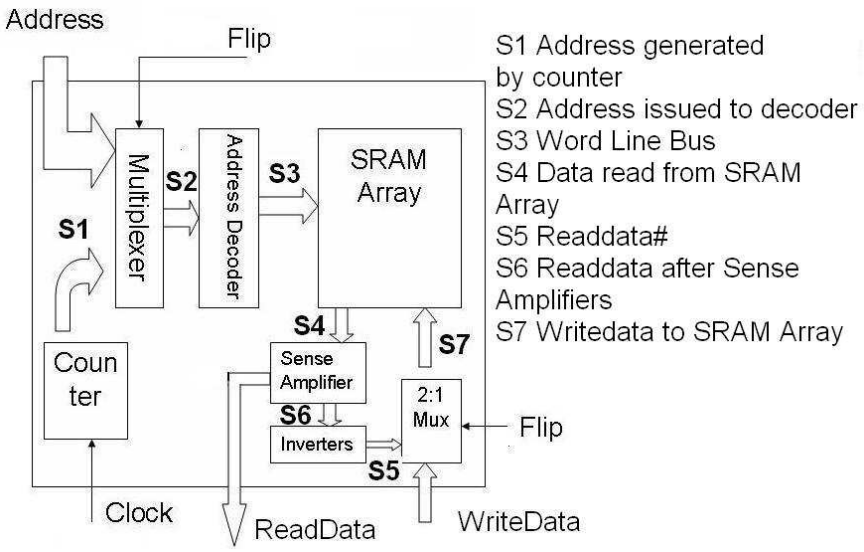


Figure 3.26: Hardware implementation showing the additional hardware and control signals to be added to SRAM arrays for periodic cell flipping.

The hardware approach is much faster compared to the software approach since the data is flipped locally using inverters and written back through the multiplexers. The total time of flipping depends on the size of the cache only and not its relative distance from the processor, thereby providing maximum benefit for L3 caches. Typically, in a processor system, this flipping operation can be performed when the processor enters *standby* mode. This ensures that the normal processor

operation is not affected and also helps better maintain cache coherence since the data is not being modified by the processor when it is being flipped internally. Further, since the cache is not accessed by the processor during standby mode, the impact of NBTI is most significant since the cache data remains unaltered for long periods of time (due to continuous stress on PMOS transistors). Hence, maximum savings are obtained during prolonged standby mode of operation. However, if the processor does not enter standby mode exactly at the time desired (say every 10^5 seconds), this flipping mechanism can be performed as soon as the processor enters standby the next time. Further, the exact time interval of flipping can also be adjusted based on the feasibility of implementation. The operating system can be scheduled to perform this task on an everyday (every 0.864×10^5 seconds) basis rather than every 10^5 seconds along with other periodic tasks that run everyday. (Simulations showed that flipping the cells every 0.864×10^5s instead of 10^5s gave only a 0.2mV improvement in SNM which implies that the cell flipping mechanism is almost insensitive to small changes in the flipping interval.)

3.14.2 Read and Write Mechanism Modification for Flipped SRAM Cells

Modifications to the existing read and write mechanism are necessary since the data present inside the cache is in its inverted state on alternate days. There exist two approaches to ensure that the data is read and written correctly namely software and hardware approaches.

Software Approach:

In the software solution, when the processor reads the data on alternate days (days when the contents of the SRAM are flipped), it must flip the contents after it is read. Hence, every read instruction that fetches data from the cache must be accompanied by a succeeding INVERT instruction to invert the contents of the read-data bus and interpret it correctly (on alternate days). Similarly, when the data is being written into the SRAM blocks, on alternate days, the inverted data must be written. Hence every write instruction is required to be preceded with an INVERT instruction.

Maintaining zero hardware overhead in the software approach ensures that the read and write access times are unaffected. This technique can be implemented in existing systems by modifying the compiler. However, this method requires the insertion of an additional instruction before every

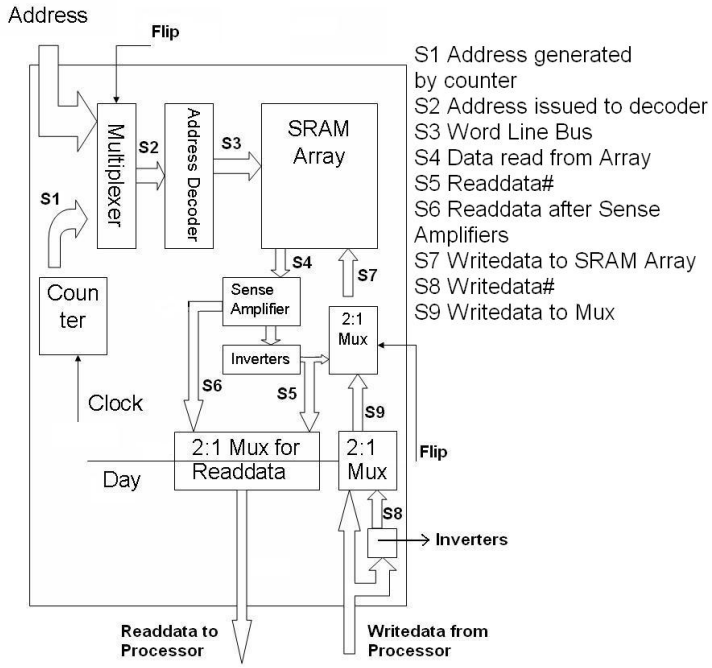


Figure 3.27: Hardware implementation to ensure correct data read and write in cell flipping caches.

write and after every read instruction. The savings in hardware delays can potentially be offset by the additional time it takes to execute these INVERT instructions. Further, in the case of direct data transfer from, say, the L3 to the L2 cache, the data that is transferred to the cache may be incorrect (if true data is written on alternate days instead of inverted data on alternate days) in the absence of efficient synchronization between the cache data transfer controller and the processor.

Hardware Approach:

In the hardware solution, the SRAM array is equipped with additional circuit to ensure that the correct data feeds into the data-bus. This is achieved with the use of additional hardware and control signals as shown in Fig. 3.27.

The read-data after the sense amplifiers (S6) is inverted to get read-data# (S5). If the hardware approach shown in Fig. 3.26 is used for flipping the contents, then this signal is already available. The read-data (S6) and read-data# (S5) signals are then multiplexed using a control signal (Day) which indicates the current state of data (true or inverter) in the SRAM arrays. The final data that

comes to the processor is the true data irrespective of the day or state of the cache. Similarly, for writing data into the cache, the processor always sends the true data on the write data bus. Internally this data is inverted, and write-data and write-data# (S8) are both fed to a multiplexer which is similar to the read-mux (controlled by the Day signal). The output of the write-mux (S9) is multiplexed with read-data# (S5) so as to ensure that inverted data is written during cache data flipping and processor data is written during normal course of operation.

This scheme can be adopted for any type of cache and does not require compiler modification to read and write data. The data that is fed into and out of the cache is always the true data and hence inter-cache data transfer overriding the processor is easily possible. However, the presence of a multiplexer and an inverter on the read and write critical paths affects the access time for read and write. This may be significant for small cache blocks which are close to the processor.

It must be noted that the above analysis does not take into account the activity factor in the caches and the intrinsic healing effect due to flipping of data during processor writes or replacements. While generating a model that reflects the operation of the processor and cache blocks, over a period of time as large as three years is seemingly impractical, it can still be argued that the above model rather pessimistically estimates the impact of NBTI on caches (8-9% degradation after 3 years) by not considering the internal cell flipping during normal process operation. Secondly, it must also be noted that intrinsic processor writes may also affect the external cell flipping recovery process causing non-uniform stress and relaxation phases on the PMOS devices in the SRAM cell. Nevertheless, it can be argued that, although caches may be written into frequently, since majority of the data stored in the caches is either 1 or 0, not every bit in every block of the cache is flipped during processor writes/replacements. Further, if certain cache blocks are written into extremely frequently (say at the rate of every 10-100 cycles), the impact of NBTI on these is almost zero, since there is no chance for interface trap build-up, thereby requiring no recovery measures. Hence, the above scheme provides a good performance metric as a baseline measure of the performance recovery obtainable using cache flipping mechanism. The rate of flipping can also be varied in accordance with the activity of the cache blocks. (L1 caches can flip at a much faster rate as compared to say L2 or L3 caches).

Chapter 4

Timing Analysis under Process, Voltage, and Temperature Variations

In this chapter of the thesis, we consider two aspects of variation-aware timing analysis: one dealing with nondeterministic parameterized variables, and the other with a nonmonotonous variation in delays with respect to temperature. While variation-aware analysis using statistical models have been researched extensively over the few years, such methods require a prior knowledge of the exact distribution of the varying parameters, In the absence of such data provided by the foundry, approximate distributions are assumed, and timing analysis is performed to obtain a cumulative distribution function (cdf) of the signal arrival times, of the various timing critical paths in the circuit.

This work presents an alternative solution to the problem, one that neither assumes any form of distribution on the varying parameters, nor does it require us to handle the complexities during propagation of signal arrival times using such probabilistic density functions, (which lead to approximations, or inaccuracies). Instead our method relies on the lower and upper bounds of these varying parameters, which are easier to obtain, and aims to compute the timing at any setting of the varying parameters through a single timing-graph traversal. This timing analysis framework, known as parameterized timing analysis, bridges the limitations of both multicorner STA (which often is criticized as not scalable in terms of the varying parameters, and as being pessimistic and risky in nature [86] due to the subset of the entire parameter space considered), and statistical STA (which is hindered by the inaccuracies during the MAX computation, and the necessity for the exact distributions of the varying parameters, which may be hard to obtain).

Section 4.1 presents a framework for block-based timing analysis, where the parameters are specified as ranges - rather than statistical distributions which are hard to know in practice. This approach is accurate at all values of the parameters within the specified bounds, and not just at the worst-case corner. This allows the designers to quantify the robustness of the design at any design point. This approach is validated on circuit blocks extracted from a commercial 45nm microprocessor, and the results are detailed in Section 4.5.

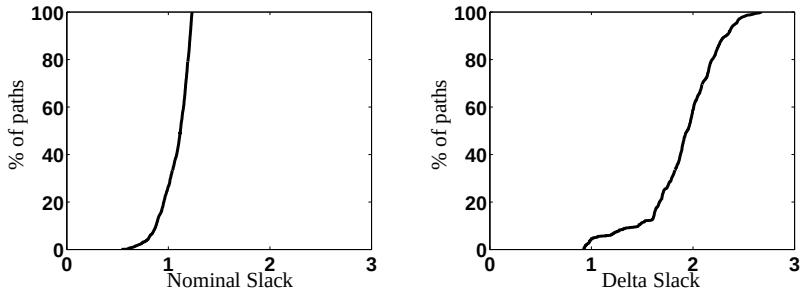
The effects of temperature on gate delay variations are becoming especially acute in nanoscale technologies. Elevated temperatures reduce both the threshold voltage and the carrier mobility; depending on which of these wins out, gate delays may increase, decrease, or vary nonmonotonically with temperature. Consequently, the worst-case delays for different gates, or different blocks, may correspond to different temperatures. Further, these worst-case conditions may be in the interior of the allowable range, rather than at a "corner". The latter part of Chapter 4 details a procedure for finding the worst-case delay of a circuit under thermal variations, first by finding the corresponding value for a block (Section 4.7.2), and then by formulating an optimization problem that finds the most critical path (Section 4.9) across the circuit, at the full-chip level. Our results in Section 4.9.2 indicate that the method can reduce the pessimism associated with a worst-case approach in determining the delay of a circuit.

4.1 A Framework for Timing Sensitivity Analysis

Microprocessors are designed under nominal or typical conditions where process parameters are assumed to be at their nominal values. Unlike ASICs, which are designed under worst-case assumptions, microprocessor designers have relied upon at-speed testing of manufactured parts to grade parts by frequency, with higher frequency parts selling at higher prices. However, due to increasing levels of parameter variations, as well as aggressive design styles that strive for the best performance at the lowest power, designing at the nominal point causes surprises in silicon [87]. Often paths with large positive slacks turn out to be speed limiting in silicon. Further, from a design perspective, ordering paths by nominal slack, as is customary, does not provide a complete prioritization of paths to work on. For example, the slack distribution[1] of paths in a modern microprocessor is as seen in Fig. 4.1.

Due to power performance trade-offs, a steep timing "wall" is created, where a large number of paths have the same slack. When the first silicon arrives, it may so happen that the drive strength of certain kinds of devices - for example, low power devices - turns out to be lower than what was assumed during design. As a result, paths that are more susceptible to variations in this device type are likely to show up as speed limiting in silicon, necessitating costly design re-spins. In the above

[1] Slack values are normalized with respect to the FO4 delay of an inverter, throughout the paper.



(a) Nominal Slack. (b) Slack Sensitivity.

Figure 4.1: CDF (cumulative density function) of the slack of the top 1000 paths on a design block: (a) shows the nominal slack while (b) plots the sensitivity in slack, i.e., Δ slack when the drain current I_d of low power devices is 20% weaker.

example (Fig. 4.1), however, if in addition to nominal slacks the slack sensitivity of paths to the drive strength of the low power devices were available, it would have been possible to fix paths that are very sensitive to the drive current variations of this particular device before tape-out. Here, by slack sensitivity we mean the change in slack for a given change in a parameter. For example, the slack sensitivity distribution of the same set of paths, when the drive current of all low-power devices is weaker by 20% is shown in Fig. 4.1(b). Note that the steep timing wall now has a finite slope when viewed from a sensitivity perspective - paths that appeared equivalent in terms of slack appear different in terms of sensitivities. Thus, by taking into account the nominal slack and the sensitivity of the slack to parameters in conjunction with the amount of variations in the parameters - often specified as a range, rather than a statistical distribution - a more effective prioritization of paths to work on can be provided to the designer.

4.2 Problem Statement

In this work, we propose a block-based algorithmic framework for solving the following problem: Given a set of parameters and their ranges (bounds), compute accurate arrival times (slacks) for all settings of the parameters within the specified ranges in a single timing run. This allows us to compute the arrival time (slack) sensitivity at any given point - within the range of variations - by

simply querying for timing information in the neighborhood of the point. Even though the nominal values are specified as numbers they are not exact and are best treated as ranges. From Fig. 4.1(b), it can be seen that this framework allows us to compute the change in slack - slack sensitivity - for any variation in drive current within the 20% bound.

It is generally accepted that block-based techniques have certain advantages over path-based methods, fast run times, incrementality and timing aware optimization, etc., and is the method of choice in industrial timing analyzers. We distinguish our approach from block-based SSTA [47, 48] which assumes that the distributions and their correlations are known *a priori*. This is usually not the case, but the bounds or ranges of parameters are easier to obtain. Further, certain parameters such as V_{dd} and Miller Coupling Factors (MCF) are not statistical in nature, and are therefore naturally described in terms of ranges. Recently, a block-based static timing algorithm that also works with parameter ranges was presented in [88]. The primary goal of that work was to preserve the accuracy at only the worst-case corner. To achieve this goal, the arrival time sensitivities to the parameters were adjusted during the output arrival time computation. As a result, the arrival times at non-worst-case settings were not accurate (we show this later on in Table 4.5.3). Our goal in this work is different: we wish to compute accurate timing at all points, not just the worst-case point. This enables us to compute the sensitivity of a timing parameter by evaluating the timing at two different points in the parameter space and computing the change in the timing quantity.

4.3 Previous Work and Limitations

In this section, we briefly describe a previous approach [88] that provides an upper bound on the worst-case arrival times over all settings of the parameters within a specified range. Suppose the delay of a gate depends on n *independent* parameters, denoted $p_1, \ldots, p_n$. Assuming a first order variation model, the delay d can be written as:

$$d = \overline{d} + \sum_{i=1}^{n} a_i X_i \tag{4.1}$$

where a_i is the delay sensitivity to some parameter p_i, and X_i is the normalized variable of p_i s.t. $-1 \leq X_i \leq 1$ for all $i = 1, \ldots, n$. We refer to (4.1) as the delay hyperplane. We assume that

the physical parameters such as channel length L, V_{dd}, etc. have been transformed into the abstract parameters X_i by means of the affine transformation as described in [88, 89]. We use $\mathbf{X}$ to denote the set of points in the hypercube defined by $-1 \leq X_i \leq 1$. Since the arrival time at the output node of a timing path is simply the summation of delays along the path, we express the arrival time A at the output node, as the arrival time hyperplane:

$$A = \overline{A} + \sum_{i=1}^{n} b_i X_i \tag{4.2}$$

where $\overline{A}$ is the nominal arrival time. Thus, the arrival time at the output node of a path has a simple representation that faithfully captures the sensitivity of the path to the parameters, given by the b_i terms in (4.2).

However, such a simple linear representation of arrival times is not helpful for computing sensitivities when many paths converge on a node. Consider the scenario depicted in Fig. 4.2 which shows four different paths with different arrival times (denoted as A_1, A_2, A_3, and A_4 respectively), and different sensitivities to some parameter X_i. If these four paths converge at the same node, the arrival time at that node is given by the max of the arrival time of the four paths, and unlike (4.2), the arrival time is a nonlinear function of the parameters.

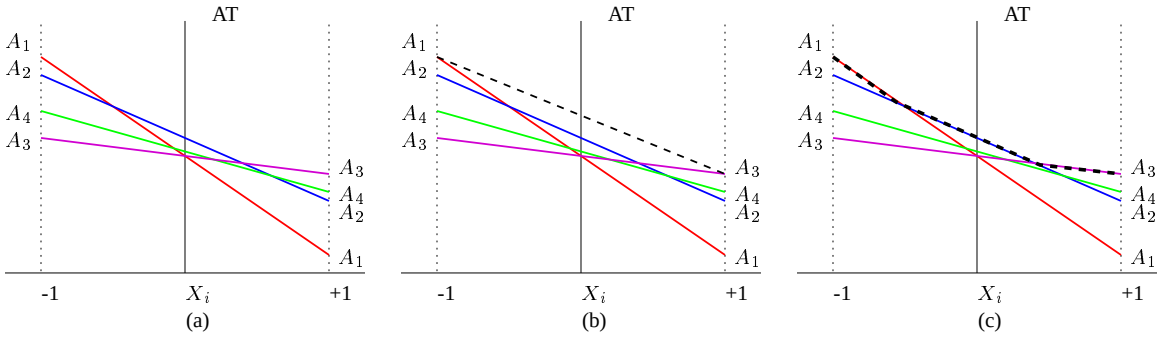


Figure 4.2: MAX arrival time (AT) of four paths shown in (a) using: (b) a bounding hyperplane based on [88] and (c) an exact piece-wise planar representation.

The maximum arrival time (AT) at the nominal value of X_i is given by path 2 (hyperplane A_2). However, if the parameter changes by -0.5, path 1 is the dominant path whereas if the parameter changes by +0.7, path 3 is the dominant one. Therefore, we require a representation of arrival times

that is faithful to the fact that different paths dominate for different settings of the parameters, and as a result gives the correct arrival time for any setting of the parameters.

One representation of the arrival times in the presence of a max function is to use a bounding hyperplane as shown by the dotted line in Fig. 4.2(b) (In one dimension it is a line.) [88]. The authors of that work propose an algorithm that provides a tight upper bound on the worst-case arrival time value at the node. The advantage of this method is that the representation is still linear and canonical. This also ensures that the worst-case delay of the circuit is an upper bound on the true delay of the circuit. However, as mentioned in the introduction, in microprocessor design we are interested in the sensitivities around a design point rather than the worst-case delay. As Fig. 4.2 shows, the bounding hyperplane is significantly inaccurate at non-worst-case settings of the parameters, particularly around the nominal value, since it is designed to be tight only at the worst-case corner. Further, no information as to which path is dominant and under what conditions is provided. However, if we relax the requirement that we propagate only a single arrival time hyperplane, then a piecewise-planar representation of the MAX function is possible (as shown in Fig. 4.2(c)). We allow a set of hyperplanes such that each one of them is the maximum hyperplane for some setting of the parameters within the allowed ranges. Intuitively, each hyperplane represents a path up to that node in the timing graph.

Thus, we seek a means of determining a subset of paths converging at every node, such that the arrival time hyperplane corresponding to every path in the subset contributes to the MAX function. In this context, for a set of m arrival time hyperplanes given by:

$$A_j = \overline{A_j} + \sum_{i=1}^{n} b_{ji} X_i, j = 1, \ldots, m \tag{4.3}$$

we say that a hyperplane A_j is prunable if and only if:

$$\max(A_1, \ldots, A_j, \ldots, A_n) = \max(A_1, \ldots, A_{j-1}, A_{j+1}, \ldots, A_n) \tag{4.4}$$

For example, in Fig. 4.2(c), A_4 can be pruned because it does not contribute to the MAX function (shown by the dotted line). A set of hyperplanes is said to be irreducible if no hyperplane in the set is prunable. Applying this definition to the four paths in Fig. 4.2(a), we see that the three hyper-

planes A_1, A_2, A_3 form an irreducible set and the MAX is shown by the dotted line in Fig. 4.2(c) which forms a piece-wise linear representation (piece-wise planar in higher dimensions). While this raises the possibility of exponential blow up in the number of hyperplanes under the worst-case, since every path (there are exponential number of paths in the worst-case) could become critical at some setting of the parameters, we show in Section 4.5 that this is not the case on practical industrial designs.

We mention in passing that recently in [89], a branch and bound method was proposed to compute the exact worst-case path delay using the method of [88] to provide the bounds required to prune the search space. While the method could be adapted to compute the circuit delay at any settings of the parameters, it involves searching through the path space any time the parameter setting changes. Further, the run time depends on the quality of the upper bounds. As shown in Fig. 4.2(b), the upper bound can be very loose at non-worst-case settings of the parameters, particularly at the nominal corner. Finally, this method does not explicitly provide us with a technique for determining under what conditions a particular path could become the most critical, something that is useful for the designers to know.

4.4 Propagation of Arrival Times

The basic operations of static timing analysis are SUM and MAX operations. Given a set of hyperplanes $\mathbf{A_1}$ at the input of a gate, and a delay hyperplane d_{12} from the input node to the output node, the output hyperplane A_2 is given by:

$$\mathbf{A_2} = A_j + d_{12} | A_j \subset \mathbf{A_1} \tag{4.5}$$

Thus, the SUM operation is canonical and the cardinality of the output hyperplane is equal to that of the input hyperplane, i.e., $|\mathbf{A_2}| = |\mathbf{A_1}|$.

However, for a MAX[2] operation, given a set of arrival times at the inputs of a gate, the arrival time at the output of the gate is the union of the sets of arrival times at the inputs. We refer to the set

[2]The algorithms described in the paper can be adapted for the MIN operation in a straightforward manner.

of hyperplanes at the output node as $\mathbf{U} = A_1, \ldots, A_m$. We must perform a pruning operation on $\mathbf{U}$ to determine a set $\mathbf{A} \subset \mathbf{U}$ (ideally $\mathbf{A}$ would be irreducible). Such a pruning operation is necessary in order to ensure that the number of hyperplanes on every node does not increase exponentially as we perform a forward propagation of arrival times along the timing graph.

Two different ideas are explored in this regard, leading to five different algorithms. The two pruning strategies are explained below.

4.4.1 Pairwise Pruning

Given two hyperplanes A_1 and A_2, we write $A_2 \prec A_1$ if $A_1 - A_2 \geq 0$ for all values of X_i in $\mathbf{X}$:

$$\overline{A_1} - \overline{A_2} + \left(\sum_{i=1}^{n} (b_{1i} - b_{2i}) X_i \right) \geq 0 \tag{4.6}$$

which is always true if:

$$\overline{A_1} - \overline{A_2} \geq \left(\sum_{i=1}^{n} (|b_{1i} - b_{2i}|) \right) \tag{4.7}$$

Given two hyperplanes A_1 and A_2, we have one of:

1. $A_1 \prec A_2$, in which case we prune A_1
2. $A_2 \prec A_1$, in which case we prune A_2
3. neither of the above is true, in which case we retain both A_1 and A_2

4.4.2 Feasibility Check Based Pruning

The pairwise pruning strategy described above does not guarantee that the set $\mathbf{A}$ is irreducible. This can be illustrated by Fig. 4.2 where A_4 will not be marked as nonprunable (using an algorithm based on Section 4.4.1), since (4.6) does not hold for any hyperplane that is compared with A_4. In order to determine if a hyperplane in $\mathbf{U}$ is prunable or not, it must be simultaneously compared with all other nonprunable hyperplanes in $\mathbf{U}$. Thus, to determine if a hyperplane A_j in $\mathbf{U}$ can be pruned,

the following condition must be satisfied:

$$\begin{aligned} A_j \geq A_k, \forall k = 1, \ldots, m, k \neq j \\ -1 \geq X_i, i = 1, \ldots, n \\ \text{has no feasible solution} \end{aligned} \tag{4.8}$$

4.4.3 Exact Algorithms for Pruning Events During MAX Computations

Based on the above two pruning strategies, we present three *exact* algorithms to prune events. The word *exact* is used to distinguish these algorithms, since the arrival times computed at any node using these algorithms is accurate.

1. PAIRWISE: The pairwise pruning strategy (Section 4.4.1) is used to determine prunable events. Since, the condition to prune events in (4.6) is sufficient but not necessary, the algorithm does not produce an irreducible set, resulting in some redundant events being carried forward. However, it is fast, and the pairwise checking of m events can be performed in $O(m^2)$ time. The algorithm for PAIRWISE pruning is outlined in Algorithm 6. The PAIRWISE pruning algorithm begins with the set **U**, and initially assumes that all the hyperplanes in **U** are nonprunable. A hyperplane in **U** is compared with all other nonprunable hyperplanes in **U** and if it is not prunable, it is added to the set **A**.

Algorithm 6 PAIRWISE pruning.

```
1: {U = A_1, ..., A_m}
2: A = {}
3: Mark all hyperplanes non-prunable
4: for i = 1 : m do
5:    If A_i is marked pruned continue
6:    for j = 1 : m do
7:       If A_j is marked pruned continue
8:       If A_j ≺ A_i mark A_j pruned
9:    end for
10: end for
11: Add all non-prunable hyperplanes to A
```

2. FEASCHK: The feasibility check based pruning strategy (Section 4.4.2) can be applied on

events to produce the true list of nonprunable or dominant events. Algorithm 2 outlines the FEASCHK pruning strategy. Since feasibility checking is supported by all LP (linear programming) solvers, we use the commercial optimization package CPLEX [90] which performs feasibility checking efficiently. We note that the algorithm is inherently parallelizable since the feasibility check for each hyperplane can be performed in parallel, if a multiprocessor machine were available. Also, unlike the method in [89], FEASCHK can easily be adapted to find a point in $\mathbf{X}$ where a given hyperplane (path) is nonprunable (critical).

However, this process potentially involves solving an expensive linear program on every event at every node, and hence may lead to larger runtimes[3] in comparison with PAIRWISE.

Algorithm 7 FEASCHK pruning.

```
1: {U = A_1, ..., A_m}
2: A = {}
3: Mark all hyperplanes non-prunable
4: for i = 1 : m do
5:    If A_i is marked pruned continue
6:    Formulate (4.8) and check for feasibility
7:    {Only include A_k not marked}
8:    if Solution to (4.8) is feasible then
9:       A = A ∪ A_j
10:   else
11:      Mark A_j as pruned
12:   end if
13: end for
14: Add all non-prunable hyperplanes to A
```

3. PAIRWISE_FEASCHK_THRESH: In order to optimize the runtime spent in determining the set of hyperplanes to propagate, FEASCHK algorithm can be applied selectively. If the number of hyperplanes on the node exceeds a certain user-specified threshold N, we apply FEASCHK; else the PAIRWISE algorithm is used to prune the hyperplanes. This implies that some redundant hyperplanes that can be pruned are carried forward, until the threshold is reached.

[3] Although the complexity of FEASCHK is also of the order of $O(m^2)$ (since the complexity of linear programming is $O(mn)$ for m equations in n unknowns), i.e., the same as that of PAIRWISE, the constants are significantly higher.

4.4.4 Exploring Run-Time Accuracy Trade-offs

While the algorithms described in the previous subsections are exact, the runtime depends on the nature of the logic cone and the number of parameters considered. In the worst-case, the runtimes could be exponential since the number of hyperplanes carried can increase exponentially. We now describe two methods which trade-off accuracy for runtime.

A Soft-Pruning Strategy

The PAIRWISE pruning strategy can often be inefficient, leading to an exponential blow-up in the number of nonprunable hyperplanes, if most of them cannot be pruned using (4.7). Instead, if we relax (4.6) as follows, then additional pruning can be achieved:

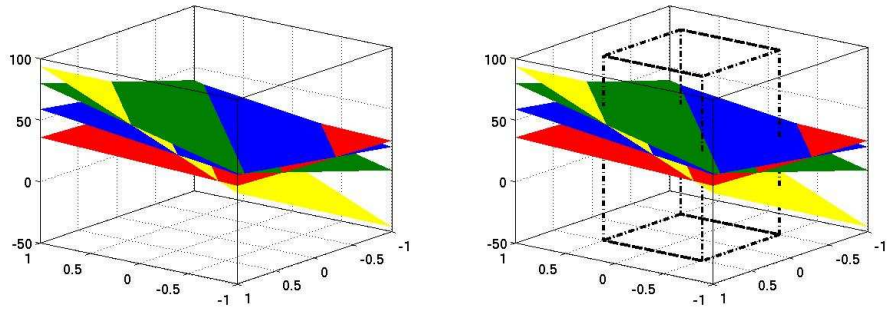
$$\overline{A_1} - \overline{A_2} + \left(\sum_{i=1}^{n} (b_{1i} - b_{2i}) X_i \right) \geq \epsilon, \epsilon < 0 \tag{4.9}$$

Intuitively, this implies that we mark A_2 as prunable even if it can exceed A_1 by a small amount, for some setting of the parameters. In order to account for the fact that pruning A_2 may lead to an inaccurate timing estimate at some setting of the parameters, we raise hyperplane A_1 by increasing its nominal arrival time , by the minimum amount required for (4.6) to be satisfied. This not only allows us to prune A_2 but may also allow several other planes to be pruned by the raised hyperplane of A_1, thereby considerably decreasing the number of hyperplanes that must be propagated.

There is a trade-off between the number of hyperplanes pruned and the pessimism in the actual arrival time numbers due to raising some of the hyperplanes, based on the value of ϵ. However, in practice, a small value of ϵ, (-0.5% of $\overline{A_1}$, based on our experiments) provides a considerable speedup without significantly overestimating the arrival times. In order to ensure that the increase in the nominal arrival times (due to the raising a hyperplane) does not get cascaded during forward propagation, an upper bound may be placed on the maximum amount by which a hyperplane can be raised. In our implementation, we use this idea of a soft threshold for pruning our hyperplanes as a preprocessing stage to reduce the cardinality of **U**, before applying FEASCHK. Each hyperplane is allowed to be raised at most once during PAIRWISE pruning if (4.9) is true but (4.6) is still false. This algorithm is referred to as PAIRWISE_SOFT_PRUNE_FEASCHK.

Shrinking Hypercube Method

We now describe a method that allows accuracy to be traded-off for runtime by limiting the number of hyperplanes carried. This idea is explained in Fig. 4.3 where four hyperplanes in an irreducible set $\mathbf{A}$ are shown. X_i is in [-1, 1] as before. However, if X_i is restricted to lie in [-0.5, 0.5], A_1 and A_4 are prunable as shown in Fig. 4.3(b), while at the nominal value of X_i (interval size is zero) A_1, A_3, and A_4 are all prunable. Thus, by reducing the size of the range of parameters, fewer hyperplanes required to be propagated. Hyperplanes that are not prunable outside the range are replaced with a bounding hyperplane using the method of [88]. While this method is pessimistic outside the reduced range of X_i, it is faster since fewer hyperplanes are propagated. Further, by preserving accuracy within the reduced range which is centered on the nominal point, the arrival times around the nominal are still calculated accurately.



(a) Four nonprunable hyperplanes (A_1, A_2, A_3, A_4).

(b) Shrunk hypercube for the four planes in (a), given by -0.5 $\leq X_i \leq$ 0.5, such that only A_2 and A_3 are nonprunable.

Figure 4.3: Shrinking hypercube method.

More formally, for every node we have a triple consisting of the nonprunable hyperplanes, the hypercube where the hyperplanes are nonprunable, and a bounding hyperplane which is an upper bound of all the hyperplanes at that node. Consider an m input gate, s.t. at each input we have the triple: $< \mathbf{A}, \mathbf{X_a}, B >$ where $\mathbf{A}$ is an irreducible set, $\mathbf{X_a}$ is the set of points in the reduced hypercube given by $-1 \geq X_i \geq 1, i = 1, \ldots, n$ and B is the bounding hyperplane. To compute the

triple at the output, we start with the initial triple $< \mathbf{U}, \mathbf{X_0}, \mathbf{U_b} >$ where $\mathbf{U} = \cup \mathbf{A_j}, j = 1, \ldots, m$, $\mathbf{X_0}$ is the smallest hypercube from the inputs, and is given by $\mathbf{X_0} = \cap \mathbf{X_{aj}}, j = 1, \ldots, m$, and $\mathbf{U_b} = B_1, \ldots, B_m$. We prune $\mathbf{U}$ such that the number of nonprunable hyperplanes is less than the user specified threshold. We do this by iteratively shrinking $\mathbf{X_0}$ if necessary (by some delta), and the algorithm, denoted as SHRINK_HYPERCUBE is detailed in Algorithm 4.4.4. A bounding hyperplane B on $\mathbf{U_b}$ is also computed using [88] or other such methods.

Algorithm 8 SHRINK_HYPERCUBE pruning.

1: $\{$(Inputs: $\mathbf{U}, \mathbf{X_0}, \mathbf{U_b}, N$)$\}$
2: $\{$(Outputs:$\mathbf{A}, \mathbf{X}, B$)$\}$
3: $\{\mathbf{U} = A_1, \ldots, A_m\}$
4: $\{\mathbf{U_b} = B_1, \ldots, B_m\}$
5: $\{N =$ maximum number of nonprunable hyperplanes allowed$\}$
6: $\{\mathbf{X_0}$ = Initial hypercube$\}$
7: Apply FEASCHK algorithm with bounds on $\mathbf{X}$ from $\mathbf{X_0}$, to obtain the irreducible set of $\mathbf{A}$
8: **while** size($\mathbf{A} > N$) **do**
9: Shrink hypercube $\mathbf{X_0}$ by Δ
10: Apply FEASCHK with new bounds on $\mathbf{X}$ to obtain the new irreducible set of $\mathbf{A}$
11: **end while**
12: Compute bounding hyperplane B on $\mathbf{U_b}$

The SHRINK_HYPERCUBE method is equivalent to FEASCHK when $\mathbf{X}$ is in [-1, 1], and reduces to the method in [88] if $N = 1$. This algorithm can be extended to shrink each dimension by different amounts and to also use some form of binary search in the *while* loop in Algorithm 4.4.4.

4.5 Simulation Results on Microprocessor Blocks

In this section, we present simulation results obtained on a 45nm based commercial microprocessor design. Global variations in four different parameters types, namely supply voltage (V_{dd}), Miller Coupling Factor (MCF), channel length of NMOS transistors (L_n), and channel length of PMOS transistors (L_p), are considered. L_n is divided into two different types, based on whether the device is nominal or low power, and further into three types based on layout dependent information. These six types are denoted as L_{n1}, L_{n2}, L_{n3}, L_{nlp1}, L_{nlp2}, and L_{nlp3} accordingly (lp indicates that the device is of low-power type). Similarly, L_p is further divided into six different types L_{p1}, L_{p2}, L_{p3}, L_{plp1}, L_{plp2}, L_{plp2}, and L_{plp3}. Each of the individual L parameters is now assumed to vary

independently of each other, thereby resulting in 14 different parameters (12 L types, MCF, and V_{dd}). The ranges of these parameters are shown in Table 4.5.

Table 4.1: Range of variations for parameters.

Parameter (total of 14)	Range of variations
V_{dd}	0 to -18%
L_n and L_p (12 different types)	$\pm$10%
MCF	$\pm$33%

The bounds on these parameters are provided as an input to the timing engine. We found that the delay is linear in the parameter variations within these ranges. The library characterization flow has been enhanced to compute the delay sensitivities on all timing arcs with respect to each of the above parameters as a function of input slopes and output loads. The pruning algorithms described in Sections 4.4.3 and 4.4.4 are applied on four different design blocks. Table 4.5 presents information about the benchmark circuits. The timing engine is implemented in C++, with an interface to CPLEX [90], to perform FEASCHK pruning. The arrival times are computed for RISE and FALL transitions at the MAX and MIN modes as is typical in a static timing tool.

Table 4.2: Benchmark information for microprocessor design blocks.

	Block 1	Block 2	Block 3	Block 4
No. of Registers	623	1086	2510	1021
No. of Nodes	21425	22384	40972	50599
No. of timing arcs	14143	10044	16879	46647

4.5.1 Run-time Comparisons

The runtimes for performing a forward propagation on the timing graph computing the set of irreducible arrival time hyperplanes on every node are shown in Table 4.5.1. The runtime numbers are relative to nominal timing, where the hyperplane with the largest (smallest) arrival time at the nominal point for MAX (MIN) analysis is propagated.

Table 4.3: Run-times (relative to nominal) with 14 parameters.

Method	Block 1	Block 2	Block 3	Block 4
PAIRWISE	-[4]	1.20X	1.15X	1.69X
FEASCHK	14.11X	1.67X	1.40X	1.74X
PAIRWISE_FEASCHK_THRESH[5]	14.44X	1.20X	1.16X	1.76X

The results indicate a significant difference between the runtimes on Block 1 versus the other blocks. As shown in Fig. 4.4, this is because there are a large number of reconvergent paths in Block 1 and consequently a larger fraction of nodes that contain 100 or more hyperplanes. Evidently, since Blocks 2 - 4 consist of fewer critical paths that remain critical over all settings of the parameters, the runtime increase is extremely small, particularly in comparison with a multicorner timing analysis method, where the runtime is linear in terms of the number of corners chosen, and can potentially be exponential in terms of the number of parameters.

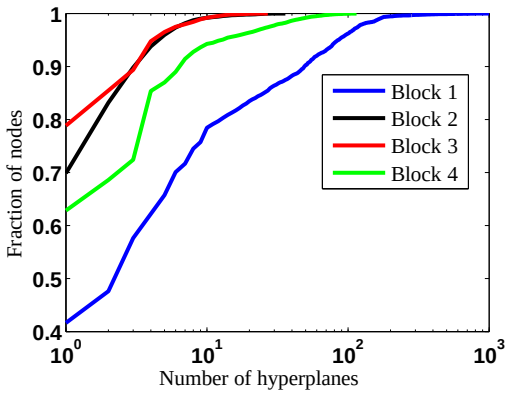


Figure 4.4: CDF of the number of hyperplanes on the four blocks for MAX operations, performed using FEASCHK.

It is also interesting to see that, for Block 1, PAIRWISE takes an order of magnitude more runtime than FEASCHK although the complexity of PAIRWISE is less than that of FEASCHK,

[4]This run did not finish due to insufficient memory on a machine with a 3GB RAM, with 10-20 nodes yet to be pruned, and propagated. The total runtime until that run however was already over 50X. Expectedly, the largest fraction of the runtime is spent on these final few nodes, where the logic cone converges at the register, and these nodes had around $O(1000)$ hyperplanes each.

[5]Using a threshold of 50, i.e., use FEASCHK to prune if number of hyperplanes exceeds 50, and PAIRWISE otherwise.

which can be explained as follows. Since PAIRWISE is a sufficient but not a necessary condition for pruning, it carries forward a significant number of prunable hyperplanes, which has a cascading effect as the hyperplanes are propagated through the circuit. FEASCHK on the other hand does more work to find truly prunable hyperplanes at every node and the number of hyperplanes it carries forward is therefore significantly reduced. For example, there were 93 nodes that had more than 1000 hyperplanes with PAIRWISE whereas there were only 15 such nodes with FEASCHK.

The runtime scaling with number of parameters is compared by plotting the relative runtimes (with respect to nominal) for cases of 4 parameters (obtained by lumping the six L_n types and six L_p types into L_n and L_p respectively), 8 parameters (nominal and low power types are lumped into a single type), and 14 parameters. The results are shown for Block 1 in Fig. 4.5. The results indicate that both FEASCHK and PAIRWISE_FEASCHK_THRESH show better scaling of runtimes with respect to the number of parameters, as compared to PAIRWISE.

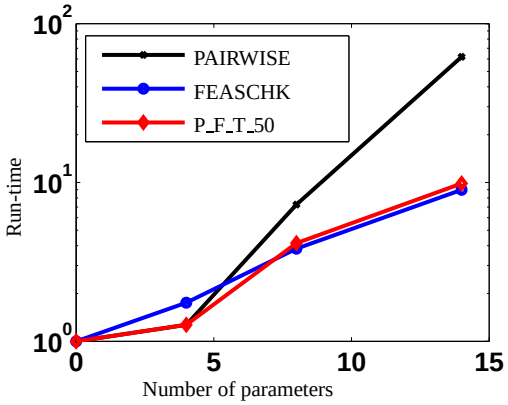


Figure 4.5: Run-times are shown for a test-run where some of the primary input arrival times were adjusted differently from that used in Table 4.5.1 and other experiments, so as to let the PAIRWISE run complete.

To summarize, our experiments on the four circuit blocks indicate that PAIRWISE_FEASCHK_THRESH provides significantly better runtime performance over the PAIRWISE method for circuits with large number of hyperplanes (as seen in Block 1). At the same time it performs better than FEASCHK on Blocks 2-4. Thus, it is the method of choice for performing exact pruning of arrival time hyperplanes. Since Block 1 has large number of equally critical paths, we focus on that block in the rest of the section.

4.5.2 Approximate Methods

In order to explore runtime accuracy trade-offs, the SHRINK_HYPERCUBE method, described in Section 4.4.4 is applied on Block 1 for different values of N, where N denotes the maximum number of hyperplanes that can be propagated on every node. The runtimes and the smallest size of the hypercube, computed across the inputs of the 623 sequential gates in the design, for the case of 14 parameters, are shown in Table 4.4. A step-size of 0.25 is used to shrink the hypercube in each iteration. The runtimes are compared with respect to the FEASCHK runtime in Table 4.5.1. The results indicate a good trade-off between the runtimes, the threshold N, and the size of the hypercube (denoted in Section 4.4.4 by a, where $-a \leq X_i, \leq a, i = 1, \ldots, n$).

Table 4.4: SHRINK_HYPERCUBE method on Block 1 with 14 parameters.

No. of hyperplanes allowed	Speedup over FEASCHK	Size of hypercube
50	0.69X	0.25
100	0.74X	0.25
200	0.83X	0.50
400	0.93X	0.75
800	0.99X	0.75

A cdf of the size of the hypercube for each of the timing cones[6] in Block 1 is shown in Table 4.5, for the case where N was set to 100, in the SHRINK_HYPERCUBE algorithm. The results indicate that more than 95% of the timing cones have a hypercube of size 1, implying less than 100 hyperplanes on them, and hence the arrival times computed on all these cones are exact, for any setting.

Table 4.5: Distribution of the hypercube size on Block 1.

Size of hypercube	No. of cones	Cumulative %
0.25	2	1.19%
0.50	2	2.38%
0.75	51	3.28%
1.00	1619	100.00%

To further explore runtime accuracy trade-offs, the PAIRWISE_SOFT_PRUNE_FEASCHK al-

[6] A cone is a set of combinational gates in the transitive fanin of a sequential element.

gorithm, explained in Section 4.4.4 was applied on Block 1. The results are compared with FEASCHK algorithm in Table 4.6. The table shows a reduction in the maximum number of hyperplanes on a node by a factor of three, when compared with FEASCHK. Accordingly, a 33% speedup over FEASCHK is obtained at the expense of a small overestimation (maximum of 1.6%) in the nominal arrival times.

Table 4.6: Results for PAIRWISE_SOFT_PRUNE_FEASCHK on Block 1 with 14 parameters.

	FEASCHK	PAIRWISE_SOFT_PRUNE_FEASCHK
Run-time	14.11X	9.58X
No. of hyperplanes on the largest cone	948	382

4.5.3 Slack Computation

We briefly explain how we compute the slacks at the inputs of the sampling registers on a cone in a block. In our framework, the arrival times at the data and clock inputs of the sampling register are irreducible sets of hyperplanes denoted as $\mathbf{A_d}$ and $\mathbf{A_c}$, respectively. The required arrival time at the data input of the sampling register is given by:

$$\mathbf{R_d} = A_{j_c} + T - S | A_{j_c} \in \mathbf{A_c} \tag{4.10}$$

where T is the cycle time and S is the setup time. We do not consider the setup time variations in this work. On all our benchmarks, the cardinality of $\mathbf{A_c}$ was one since there was no fanin in the clock network. The margin at the data input of the register is given by:

$$\mathbf{M_d} = R_{i_d} - A_{j_d} | A_{j_d} \in \mathbf{A_d}, R_{i_d} \in \mathbf{R_d} \tag{4.11}$$

Thus, (4.11) can be computed in $O(|\mathbf{A_d}||\mathbf{R_d}|)$ time. $\mathbf{M_d}$ may be further pruned using the techniques in Section 4.4.3.

In order to evaluate the sensitivity of the slack of the various paths to parameter variations, we first compute the set of irreducible slack hyperplanes at the data input of each of the registers on Block 1, for the case of 14 parameters. We now consider the cone with the highest number of irre-

ducible slack hyperplanes on Block 1 (948 hyperplanes). Table 4.5.3 shows the slacks (computed as a minimum of the margins of the 948 irreducible hyperplanes) at different settings of the parameters, (none of which are worst-case), the settings being:

1. nominal values of all parameters (Nominal)
2. all devices 5% faster (Fast L)
3. low V_{dd}, high MCF (Low V_{dd}, High MCF)
4. certain layout type devices being 5% slower due to lithography effects during frbrication. (Slow Layout)
5. low power devices being 5% slower (Slow Low Power).
6. all parameters at their worst-case (Worst-Case).

The slack at each of these settings is significantly different from the nominal slack, demonstrating that paths have different sensitivities and the ability of our method to predict that.

Table 4.7: Slacks at different settings of the parameters.

Setting	Normalized Slack	Delta Slack w.r.t. Nominal	AT Overestimation using [88]
Nominal	-0.16	-	24.42%
Fast L	+0.64	+0.80	33.21%
Low V_{dd}, High MCF	-1.13	-0.98	20.23%
Slow Layout	-0.74	-0.58	17.47%
Slow Low Power	-1.45	-1.29	17.57%
Worst-Case	-5.28	-5.12	0.00%

We also compute the upper bound on the arrival times (AT) at each of these settings using the upper bounding hyperplane method in [88] in order to determine the extent of pessimism induced by using such an upper bounding method, and the results are shown in the last column in Table 4.5.3. Expectedly, at the worst-case corner setting, the arrival time computed using [88] is exact, and there is no overestimation, whereas at other settings of the parameters, particularly at the nominal, the arrival times computed using [88] are higher by as much as 20-30%.

Fig. 4.6 shows the plot of the nominal slacks versus the new slack computed at some point in X for the 948 hyperplanes, determined such that the path marked with a P in the figure, which has a large nominal slack, becomes the most timing critical at that setting. The setting corresponded to a nominal V_{dd}, close to worst-case MCF, certain layout transistor types being fast, others being slow, and some of which were not at their extreme values in the range. The set of paths that are encircled are the most sensitive when the parameters change from nominal to this particular setting. Note that this information is not obtained in current timing flows based on nominal slacks. In this case, the path marked with a P would not have been considered critical. However, in our flow we can compute the slack at any setting of the parameters, thus enabling a what-if analysis.

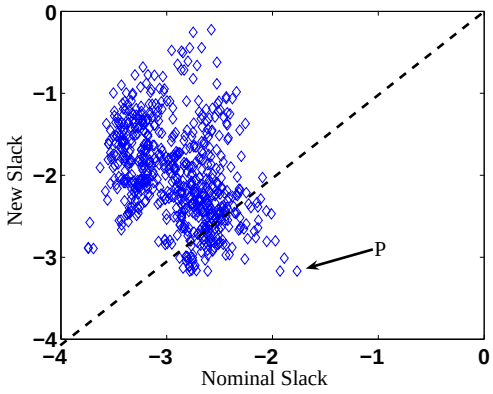


Figure 4.6: Slacks at a different setting of **X** such that the path (marked P) in the figure, with a relatively large nominal slack becomes the most timing critical.

4.6 Timing Analysis under Mixed Temperature Dependence

The traditional assumption that has guided timing analysis is that the delays of library cells increase monotonically with temperature, due to a decrease in the mobility of transistors, and therefore, the drain current. This phenomenon is commonly referred to as negative temperature dependence (NTD). However, with transistor scaling and the use of lower supply voltages, the effect of thermal variations on the transistor threshold voltage (V_t) has also begun to significantly affect delays. An increase in temperature causes V_t to decrease, and the drain current to increase. In other words, elevated temperatures affect the drain current in opposite ways.

More concretely, the drain current, I_d, of a transistor can be expressed in the form:

$$I_d = \mu(T) C_{ox} \frac{W}{L} \left(V_{gs} - V_t(T)\right)^{\alpha}, \tag{4.12}$$

where the terms have their usual meanings [91]. The delay of a gate varies inversely with I_d, i.e., higher values of I_d correspond to lower gate delays, and vice versa. Thermal variations can affect I_d in two ways:

- The mobility, μ, of charge carriers in a transistor reduces with increasing temperature, T, according to:

  $$\mu(T) = \mu(T_0) \left(\frac{T}{T_0}\right)^{-m} \tag{4.13}$$

  where T_0 is the room temperature (typically, 300K), and $m > 0$ is the mobility temperature exponent, with a typical value of 1.7 in highly doped silicon, and 1.4 in nanometer-thin silicon layers, where boundary scattering becomes important [92]. This reduction in μ lowers I_d.

- The threshold voltage of a transistor, V_t, decreases with increasing temperature along the trajectory

  $$V_t(T) = V_t(T_0) - \kappa \left(T - T_0\right) \tag{4.14}$$

  where $\kappa > 0$ is the threshold voltage temperature coefficient with a typical value of 2.5mV/K [93]. This trend makes it easier for a transistor to switch on as temperatures rise, and implies a tendency for increased values of I_d.

It is clear that the two phenomena above have opposite effects on I_d, and therefore, the gate delays,

and the trend of delay with temperature depends on which of the two is more dominant. For certain operating conditions, such as in low V_{dd} circuits, or circuits that use low-V_t cells, the increase in drain current due to a reduction in V_t can outweigh the reduction in mobility, creating a scenario where the delay may decrease with temperature (an effect known as positive temperature dependence (PTD) or inverted temperature dependence (ITD)), or may change nonmonotonically with temperature (i.e., mixed temperature dependence (MTD)) [49–51, 93]. Even the same gate type may show a mix of various types of temperature dependence, depending on the capacitive load and the input slew rate. The net result is that it is no longer possible to assume that the maximum circuit delay occurs at the highest temperature, for a given process and voltage corner, as documented in Section 4.7.2.

In previous work, it has been suggested that to reduce computational runtimes associated with multicorner timing analysis, the circuit may be timed at a fixed setting, and the timing at each other settings is computed using derating factors [94], based on the sensitivities of the circuit delays to the varying parameters. Other approaches such as [41, 88, 95] assume that the delay can be expressed as a linear function of the varying parameters, and the first order delay-sensitivities computed about the nominal setting, and perform timing analysis to determine the delays at other settings. However, we show results indicating that under mixed temperature dependence, the variation in the delay of the circuit with temperature is not only nonlinear and nonmonotonic, but also distinct for different settings of process, and supply voltage. This renders such corner-based and sensitivity-based techniques ineffective.

The work in [49] handles ITD by determining the maximum delay of each gate, based on its loading conditions, over the full range of operating temperatures. These delays are added for all gates along a path to obtain the worst-case delay of the path, assuming the temperature of each gate in the design to be independent of the others. As a result, the computed delays can correspond to significantly different temperatures, even for neighboring gates. However, it is well-known, and documented in [96, 97], that over a circuit block, the spatial variation of temperature is gradual. The method of [49] is oblivious to this fact, and may well assign worst-case delays to adjacent gates, even though these delays are achieved at vastly different temperatures. This implies that while such an approach provides an upper bound on the delay, this bound may be pessimistic.

An alternative, as proposed in this work, factors in the spatial relationships associated with an on-chip temperature profile. Unlike [49], we assume the temperatures of all gates within a circuit block to be the same, while allowing the temperature of each individual block is expected to vary with time, considering changes in the power density under differing workload conditions, environmental effects, etc, subject to thermal and power constraints. This approach reduces the pessimism in [49] by imposing additional realistic constraints that capture the spatial variations in temperature across blocks, as documented in Section 4.8.

Noting that the maximal delay of the circuit can occur at any intermediate temperature, we enumerate the delays at multiple temperatures during static timing analysis (STA), and thereby determine the maximal delay, and the corresponding operating temperature. However, since such a naive enumerative approach has a large runtime and capacity overhead, due to storing and propagating the delays at multiple temperature points for each timing arc, we use a quadratic representation of the delay with respect to temperature, and perform STA using this model. The results indicate that using a temperature-aware STA, as opposed to merely considering the worst-case delays, based on [49], leads to a lower number for the maximal delay of the circuit. Similarly, considering the thermal relationship across different blocks on a chip, due to spatial, power, and thermal correlations, results in a lower number for the maximal delay of the critical paths, that span across multiple blocks in a chip.

For the problem of computing the worst-case delay under thermal variations, Section 4.7 demonstrates the limitations of a corner-based approach or a method based on [49]. Next, Section 4.8 describes how the delay within a block can be computed over a temperature range. Section 4.9 then demonstrates how the worst-case temperature over an entire circuit, consisting of multiple blocks, is computed by solving a small optimization problem.

4.7 Temperature Dependence Trends

4.7.1 Temperature Dependence of Library Cell Delays

In order to investigate the temperature dependence of library cells, we perform simulations using two different PTM [75] 45nm technology model files, each having distinct V_t values. The nominal

value of V_{dd} is chosen as 0.9V, while $T_{\min}$ = 0°C and $T_{\max}$ = 130°C are the minimum and maximum temperature corner setting, respectively, and a step size of 10°C is used to characterize the cell delays. The cell library consists of NOT, NAND2, NAND3, and NOR2 gates, of different sizes, with two versions of each type, i.e., consisting of low V_t and high V_t devices, respectively. The use of such gates with differing V_t values is quite common in most standard cell libraries, in order to reduce the leakage power of the circuit. The delays of the timing-arcs for each of the gates are characterized at several C_L-τ pairs, and a look-up table is constructed, with interpolation for intermediate values.

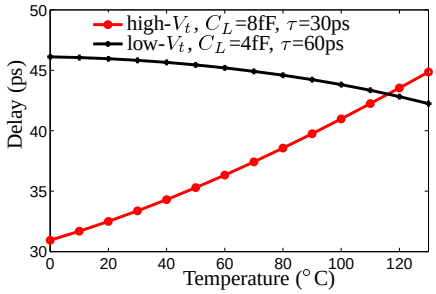


Figure 4.7: Delay of two-input NAND gates showing instances of positive and negative temperature dependence, based on the input slope (τ), and capacitive load (C_L), and whether the transistors are high-V_t or low-V_t.

Two distinct cases of temperature dependence for the delay of a two-input NAND gate are shown in Fig. 4.7, corresponding to different values of capacitive loads (C_L), input slopes (τ), and V_t values. For example, the delay of a two-input NAND gate with high-V_t transistors for a load of 8fF and an input slope of 30ps shows NTD. On the other hand, a two-input NAND gate, with low-V_t transistors for a load of 4fF and an input slope of 60ps, shows PTD. While Fig. 4.7 shows the delay-temperature curve for two instances of a NAND gate, for a sample C_L-τ combination, the library delays across different C_L-τ values show distinctly varying slopes, and temperature dependences. In general, gates with high-V_t transistors tend to show a greater degree of PTD, as compared with an identical gate that has low-V_t transistors, due to the sensitivity of the delay to temperature in (4.12). For low V_t devices, since $(V_{gs} - V_t(T_0))$ = $(V_{dd} - V_t(T_0))$ is larger to begin with, as V_t decreases with an increase in temperature the change in $(V_{gs} - V_t)^\alpha$ dominates the mobility term in (4.12), causing the drain current to increase with temperature. However, the exact nature of

temperature dependence (PTD, NTD, or MTD) of the gates, and the spread in the delay over the range of temperatures, are strong functions of C_L, τ, transistor type (low-V_t or high-V_t), as well as the structure of the gate itself.

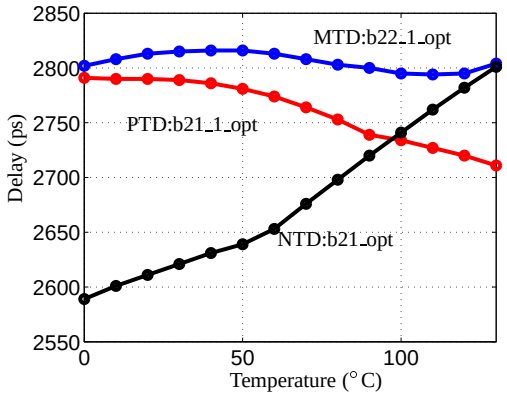


Figure 4.8: Delay of three ITC99 benchmarks showing the three different cases of temperature dependence: (a) positive temperature dependence (PTD) for b21_1_opt, (b) negative (NTD) for b21_opt, and (c) mixed (MTD) for b22_1_opt.

4.7.2 Temperature Dependence in Circuits

Typical circuits designed for optimal performance and power consumption include gates that show positive, negative, or mixed temperature dependence. Therefore, at the circuit level, one may expect the thermal trends to be a result of this mix. Fig. 4.8 demonstrates the temperature dependence of the delays of three ITC99 benchmarks, synthesized using the 45nm based library described above. The circuit b21_1_opt, which uses a large number of low-V_t cells, shows PTD, while the circuit b21_opt, which uses high-V_t cells, shows NTD. On the other hand, circuit b22_1_opt shows MTD within the temperature range, with the maximum occurring at 50°C, and the minimum at 110°C. Expectedly, the spread in the delay with temperature is small for the MTD case. Intuitively, this corresponds to the positive delay trend shown by PTD cells being countered by an opposite negative trend in the NTD cells.

4.8 Block Level Timing Analysis

Different blocks in a chip can operate at different temperatures, based on factors such as the spatial and temporal distribution of power, signal activities, workload conditions, and environmental variations. As described earlier, these blocks consist of gates whose temperature dependence can vary based on the transistor threshold voltages, loading capacitance, input slope, and the supply voltage. A key step in full-chip timing analysis is in the analysis of individual circuit blocks.

The approach in [49] provides such a technique, but assumes the worst case delay for each gate within the operating temperature range, $[T_{\min}, T_{\max}]$. However, these worst-case delays could correspond to drastic on-chip temperature variations between a gate and its neighbor, which is an unrealistic scenario. As shown in thermal images published in [96,97], on-chip temperatures exhibit a vast degree of spatial and temporal correlation. This implies that gates inside a circuit block can be reasonably assumed to operate at the same temperature, at any given instant of time[7].

It is easily seen that the approach in [49] provides an upper bound on the maximal delay of the full-chip path, but this bound may be pessimistic. In this section, we develop a procedure for a more realistic analysis, and we quantify the level of pessimism that it removes in the analysis of a single block.

4.8.1 An Enumerative Approach

Unlike conventional static timing analysis (STA) which uses a worst-case temperature corner, typically at the highest temperature, we can see that a mix of PTD, NTD, and MTD cells imply that the delay of a circuit may be maximized at any intermediate temperature. Our first approach performs a simple enumeration of the delay analysis within the temperature range. The library is characterized at multiple temperatures within the range and the characterized values are stored in a lookup table: in particular, in our experiments, we use the temperature range between $T_{\min}$ = 0°C and $T_{\max}$ = 130°C, with a of 10°C, thus characterizing the library at 14 temperature points. We perform separate forward propagation of the arrival times at each of these 14 temperature points, and

[7] The time constant associated with the rate of change of temperature of a block is several magnitudes larger than the delay of the circuit block itself. Hence, it can be safely assumed that during the course of data propagation from the inputs to the outputs, the temperature of the block is invariant.

determine the maximum delay of the circuit over all temperatures.

Table 4.8: Block level static timing analysis under thermal variations using enumeration.

Benchmark	Width	Height	Minimum		Maximum		Spread
	(mm)	(mm)	T (°C)	Delay (ps)	T (°C)	Delay (ps)	%
b14_1	0.96	0.35	0	2369	130	2480	5%
b14_1_opt	1.20	0.37	0	2076	130	2214	7%
b15_1	0.72	0.76	130	2102	30	2195	4%
b15_opt	0.60	0.91	130	2466	20	2593	5%
b17_1_opt	1.39	0.93	0	2254	80	2270	1%
b17_opt	0.95	1.64	130	2932	30	3057	4%
b20_1_opt	0.90	0.72	0	2572	130	2708	5%
b20_opt	0.84	0.93	0	2228	130	2823	27%
b21_1_opt	0.61	0.97	130	2711	0	2791	3%
b21_opt	1.08	0.68	0	2589	130	2801	8%
b22_1_opt	0.60	1.40	110	2794	50	2816	1%
b22_opt	2.14	0.89	0	2736	130	2845	4%
C6288	0.47	0.46	130	5113	0	5478	7%
dalu	0.43	0.13	130	2264	40	2345	4%
des	0.17	0.93	130	1409	30	1478	5%
i8	0.33	0.73	80	998	30	1009	1%
i10	0.43	0.33	130	2006	20	2031	3%
t481	0.78	0.38	130	1125	0	1181	5%

Table 4.8 shows the results of applying this analysis on a variety of large circuits from the ISCAS85, LGSYNTH93, and ITC99 suites. These circuits are mapped to a 45nm [75] based library consisting of low and high V_t NOT, NAND2, NAND3, and NOR2 gates of different sizes. The maximum and minimum delays of the most critical path within [0°C, 130°C] are computed, and the difference between these delays is the spread in timing variation of the circuit due to thermal effects. Here, the minimum delay denotes the smallest delay of the longest critical path in the temperature range, [0°C, 130°C], and should not be confused with the minimum path delay of the circuit that is used for hold-time calculations.

For each benchmark, successive columns of the table show the dimensions (width and height) of the circuit block, the minimum longest-path delay for the circuit over the temperature range and the temperature at which it is achieved, and the corresponding numbers for the maximum delay.

The last column shows the spread of delays over the temperature range.

Of these circuits, we notice that benchmark b20_opt, in particular, shows a large spread in the delay. This can be attributed to the fact that its implementation consists primarily of high-V_t devices, and its delay significantly increases with temperature, as compared with the remaining circuits. On the other hand, the circuits with a low delay spread tend to have a mix of low-V_t and high-V_t cells on the critical path: the former tend to show PTD while the latter often show NTD, as outlined in Section 4.7.

4.8.2 Quadratic Delay Model for STA

Clearly, the enumerative approach in the previous section is not scalable and requires a high characterization overhead; however, it uses exact timing models and is accurate within the limitations of the granularity of the enumeration. For the results shown, we have verified that the step size of 10°C is sufficient to accurately capture the shape of the delay versus temperature curve. Therefore, we propose a simpler analytic approach for delay analysis under thermal variations.

Linear delay models have been used extensively in timing analysis under process perturbations, for example, in SSTA methods such as [47, 48] and in corner-based methods such as [41, 88, 95]. Since MTD causes the gate delays to be nonlinear and nonmonotonic, such a model is incapable of capturing these thermally-driven variations. Therefore, we propose to use a quadratic model of the gate delays with respect to temperature. We experimentally verify that the fit obtained through this model, with respect to the standard-cell library delay-data is extremely accurate, and the average error is less than 0.5% across all the characterization points. Further, this model can be characterized using only three points per timing arc, and hence provides savings in memory to store the library delays, as compared with the 14 points used in the enumerated delay model in Section 4.8.1.

It has widely been observed that STA requires two computations: a sum and a max operation. Therefore, under the delay model proposed above, STA requires accurate closed-form computations for the sum and $\max$ of two quadratic functions. Of these, the sum operation is easily computed: given two quadratic delays D_1, and D_2, the sum $D_3 = D_1 + D_2$ can be computed in $O(1)$ time by simply adding up the corresponding coefficients for each term of the polynomial.

However, the $\max$ computation is more involved, since the maximum of two quadratic functions

may not be a quadratic. Previous curve-fitting based techniques to compute an efficient canonical expression for $\max$, in [95] do not guarantee the $\max$ to be an upper bound on the arrival times, a crucial requirement for safe delay estimation. Hence, we present a method to compute an upper-bounded quadratic function that is the maximum of two given quadratic functions, in $O(1)$ time, thereby ensuring canonicity in representation, safety in estimation, and efficiency in computation. This method considers the various cases arising during the $\max$ computation, based on whether the delay-temperature curves are a constant, linear, monotonically increasing or decreasing quadratics, or are either convex (concave) with a minimum (maximum) in $[T_{\min}, T_{\max}]$. The precise computations require case enumerations, and all details are described in the Appendix.

Table 4.9 tabulates the maximum delays in the temperature range, [0°C, 130°C], computed for the eighteen benchmarks considered in Table 4.8, using the three methods: enumeration, quadratic delay model, and the worst-case method from [49]. Columns 2 and 3 repeat the enumeration results from Table 4.8: these are considered to be the accurate values. The corresponding results using the quadratic model are shown in Columns 4 and 5. Column 8 shows the results obtained from worst-case model from [49]. It can be seen that the worst-case method may overestimate the delay of the circuit by up to 4.5%, while the quadratic model reduces these errors to a maximum of 2%. Like the worst-case method, the quadratic model maintains pessimism; however, the corresponding errors are, on average, 2% lower than those of the worst-case method.

The sources of error in the quadratic delay model come from two sources: first, from errors in curve-fitting the library delays to the quadratic model, and second, from errors in the $\max$ computation. The accuracy of the quadratic approach is dependent on the nature of temperature dependence of the benchmark circuit. For instance, Fig. 4.8 demonstrates that the benchmark b22_1_opt shows mixed temperature dependence, where the shape of the curve is neither convex nor concave in [0°C, 130°C]. Expectedly, a quadratic representation of the delay for such cases may lead to an overestimation at some temperature points. Nevertheless, the results indicate that the **amount of overestimation** in the delays (as compared with the enumerated delay model) is **less** than that using the worst-case method.

The column denoted as "Runtime" in Table 4.9 shows the relative STA runtime using the quadratic delay modeling approach, as compared with the enumerated delay model. Expectedly,

Table 4.9: Comparison of delays using enumerated, quadratic, and worst-case delay models.

Benchmark	Enumerated		Quadratic				Worst-case	
	Maximum Delay		Maximum Delay			Fractional CPU time	Delay	Increase
	T (°C)	Delay (ps)	T (°C)	Delay (ps)	Error %		(ps)	%
b14_1	130	2480	130	2480	0.00%	0.79	2577	3.9%
b14_1_opt	130	2214	130	2216	0.09%	0.83	2284	3.2%
b15_1	30	2195	0	2204	0.41%	0.88	2207	0.5%
b15_opt	20	2593	20	2593	0.00%	0.86	2657	2.5%
b17_1_opt	80	2270	110	2305	1.54%	0.99	2373	4.5%
b17_opt	30	3057	30	3058	0.03%	0.94	3068	0.4%
b20_1_opt	130	2708	130	2706	-0.07%	0.90	2799	3.4%
b20_opt	130	2823	130	2825	0.07%	0.92	2884	2.2%
b21_1_opt	0	2791	0	2779	-0.18%	0.90	2840	1.8%
b21_opt	130	2801	130	2799	-0.07%	0.91	2914	4.0%
b22_1_opt	50	2816	70	2865	1.74%	0.87	2926	3.9%
b22_opt	130	2845	70	2866	1.44%	0.88	2961	4.1%
C6288	0	5478	0	5478	0.00%	0.52	5485	0.1%
dalu	40	2345	40	2345	0.00%	0.49	2370	1.1%
des	30	1478	30	1478	0.00%	0.75	1479	0.1%
i8	30	1009	90	1016	0.69%	0.64	1048	4.2%
i10	20	2031	20	2072	2.01%	0.69	2075	2.5%
t481	0	1181	0	1181	0.00%	0.70	1181	0.0%

the runtime of the quadratic approach is lower than the enumerated delay model, due to the reduced number of library delays that are required to be parsed, and stored during timing analysis. The runtime savings varies with the size of the benchmarks (smaller benchmarks show more savings due to the larger fraction of time spent in loading and storing the library delays), and the nature of the delay-temperature curve, which determines the complexity of the $\max$ computation. While for the enumerated delay model, the runtime for computing the $\max$ of the arrival times at the output of a gate is fixed, for the quadratic delay model, the number of computations involved in determining the quadratic $\max$ of two quadratic delay functions strongly varies with the shape, monotonicity, and convexity/concavity of the delay curves, as can be seen from the Appendix. Further, the characterization time for the quadratic model is significantly lower than that for the enumerated delay model, since only three temperature points are needed for the former, as opposed to 14 for the latter, for each library timing-arc. Thus, the quadratic approach is faster than the enumeration-based approach, and also has a significant reduction in storage requirements, as compared with the table lookup based enumerated delay model. To summarize, the quadratic delay model provides a reasonable trade-off between accuracy in estimation (amount of overestimation in the maximal delays) with the characterization and STA runtime, and the memory usage.

We now explore the impact of MTD on the shape of the delay-versus temperature curve, under process and voltage variations, since it is vital to determine the delay of the circuit accurately, at all operating conditions.

4.8.3 Impact of Process and Voltage Variations

Previous works such as [94] have proposed the use of derating factors to allow designers to predict the delay D at a given process corner as a linear function of temperature, thus easing the task of timing analysis under thermal variations. However, the use of such scaling factors is valid only if the variation of delay with temperature is monotonic, and the maximum occurs at an extreme point. However, our simulation results, shown in Fig. 4.8 and Table 4.8, illustrate that the delay versus temperature curve can be nonlinear as well as nonmonotonic, implying that such derating factors cannot be used. Instead, the circuit delay must be computed at each desired corner setting separately.

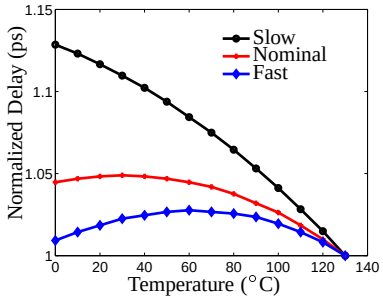


Figure 4.9: Delay variation with temperature of benchmark des at different PV corner settings (The delays are normalized with respect to the delay at each corner at T = 130°C, so as to enable better comparison of the shape of the curves, and the temperature dependence in each case.).

In fact, even for the same circuit, the nature of delay variation under thermal changes could be different at various process corners. Fig. 4.9 plots the delay of an LGSYNTH93 benchmark des at the nominal, slow, and fast corners, normalized with respect to the delay at T = 130°C at the corresponding corner. It can be seen that the three curves show different monotonicity characteristics, and the delays are maximized at different temperatures. The temperature that maximizes the delay increases, as we move from the slow corner toward the fast corner, due to the sensitivity of the delays to V_t, μ (both of which depend on T), and V_{dd}. The curves for the remaining benchmarks show similar characteristics, with the maximal delays occurring at different temperatures, across various PV corners. This example illustrates that the performance of a circuit under PVT variations cannot be accurately predicted using derating factors or sensitivities around a worst-case (or any other) point. Instead, the gate delays and signal arrival times at different temperatures must be estimated separately at each corner.

4.9 Full Chip Timing

While Section 4.8 presents a method for determining the maximal temperature of a block, occurring at any temperature within $[T_{\min}, T_{\max}]$, different blocks in a chip can operate at different temperatures, based on their localized workloads, activity factors, leakage and active power distributions, and physical locations. In this section, we present a framework for estimating the worst-case delay of a circuit at a given PV corner.

4.9.1 Delay Maximization of Critical Paths

The operating temperatures of different blocks in a chip at any instant of time, are a strong function of their spatial locations (which determine the lateral conductance of heat across blocks, and dissipation to the heat-sink), and the corresponding power distribution. To capture this information, we use the thermal-electrical duality based on the finite difference discretization of the heat equation. Thermal resistances between adjacent blocks are determined, based on their spatial separation, and the material properties. Additional nodes are added for the interface, spreader, and heat-sink for each of these blocks, as in [98]. Given a power-map $\mathbf{P}$ of the blocks, the temperatures of the blocks, $\mathbf{T}$, are determined as:

$$G\mathbf{T} = \mathbf{P} \tag{4.15}$$

where G is the thermal conductance matrix, provided by applying nodal analysis on the network of thermal resistances.

The power of the blocks can vary with factors such as the application that is begin run on the system, the workload conditions, and the ambient temperature. The temperature on the chip however cannot vary arbitrarily. Its spatial relationships are captured by (4.15): if upper and lower bounds on $\mathbf{P}$ are available, upper and lower bounds on $\mathbf{T}$ can be computed as linear functions of these values.

The relationship is based on the well-known property of any conductance matrix, that its inverse has no negative entries. Intuitively, this can be seen from the fact that the $(i,j)^{\text{th}}$ element of G^{-1} is the effective resistance between nodes i and j, which must be nonnegative in a physical system. This property is particularly useful in translating if upper and lower bounds on $\mathbf{P}$ to upper and lower bounds on $\mathbf{T}$. In general, given $A\mathbf{x} = \mathbf{b}$, one cannot guarantee that $\mathbf{x}_{\min} = A^{-1}\mathbf{b}_{\min}$, where $\mathbf{x}_{\min}$ and $\mathbf{b}_{\min}$ are the minimum values of $\mathbf{x}$ and $\mathbf{b}$, respectively. However, for the relationship $G\mathbf{T} = \mathbf{P}$, since all elements of G^{-1} are nonnegative, it is true that

$$T_{\min} = G^{-1}\mathbf{P}_{\min} \tag{4.16}$$

$$T_{\max} = G^{-1}\mathbf{P}_{\max} \tag{4.17}$$

Therefore, since the entries of G depend only on the spatial locations of the blocks, which are fixed,

the maximum (minimum) temperature is a linear function of the maximum (minimum) power.

Different blocks in a chip may exhibit widely varying temperature dependences, implying that the temperatures at which their delays are maximized can be drastically different from one another, as seen from Table 4.8. While different blocks in a chip can operate locally at different temperatures, these temperatures cannot vary arbitrarily. Hence a setting where two adjacent blocks, one showing NTD, and the other PTD operate at temperatures $T_{\max}$, and $T_{\min}$, respectively, such that their delays are locally maximized, is quite unlikely. Given bounds on the active powers of the blocks, the maximum variation in temperatures across blocks is constrained by (4.16). Accordingly, the problem of optimizing the delay of a circuit under temperature variations is now set up as that of finding the largest allowable temperature that is consistent with the relationship in (4.15). The objective function is chosen to be the maximization of the sum of all block delays. Bounds on the power dissipation of the various blocks are given as an input to the system, and we find for each block, the temperature within bounds determined by (4.16), that results in the longest delays of the critical paths across the chip.

The delay of a circuit varies nonlinearly with temperature, as seen from the results in Fig. 4.9. The delay for each circuit block in the chip is determined as a function of temperature, as detailed in Section 4.8 using the table lookup based quadratic delay model, and is denoted as $D(T)$. The delay of the most-critical path, D_{max}, is assumed to be the sum of the delays of the individual critical paths in each of the m different blocks in the core, for illustrative proposes. Accordingly, we cast a nonlinear optimization problem, to determine a feasible assignment to the temperature of the blocks, and their corresponding power densities, such that the delay D_{max}, of the critical path, is maximized.

The nonlinear optimization problem for a chip, such as that shown in Fig. 4.10, whose floorplan has n different blocks, and an illustrative critical path lies on $m \leq n$ of these blocks, is as follows:

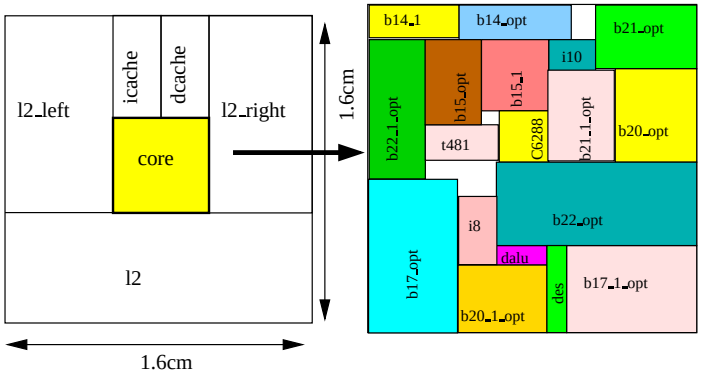


Figure 4.10: Floorplan based on [6], showing different benchmarks in the core.

$$
\begin{array}{rrr}
\text{Maximize} & D_{\max} = & \Sigma_{i=1}^{m} D_i(T_i) \\
\text{s.t.} & G\mathbf{T} = & \mathbf{P}(\mathbf{T}) \\
 & P_i = & A_i + L_i(T)\ \forall i = 1, \ldots, n \\
A_{\min_i} \leq & A_i \leq & A_{\max_i}\ \forall i = 1, \ldots, n \\
L_{\min_i}(T) \leq & L_i(T) \leq & L_{\max_i}(T)\ \forall i = 1, \ldots, n \\
 & & T_{\min} \leq T \leq T_{\max} \\
 & T_i > & T_{\text{interface}_i}\ \forall i = 1, \ldots, n \\
 & T_{\text{interface}_i} > & T_{\text{spreader}_i}\ \forall i = 1, \ldots, n \\
 & T_{\text{spreader}_i} > & T_{\text{sink}_i}\ \forall i = 1, \ldots, n \\
 & T_{\text{sink}_i} > & T_{\text{ambient}}\ \forall i = 1, \ldots, n \\
T_{\min} < & T_i < & T_{\max}\ \forall i = 1, \ldots, n \\
T_l < & T_{\text{ambient}} < & T_h
\end{array}
\tag{4.18}
$$

The first constraint in (4.18) is given by (4.15), except that the power and temperature of the blocks can both vary with time subject to other constraints. Bounds on the active and leakage powers of the blocks in the core, and the thermal constraints from (4.15) place restrictions on the relative

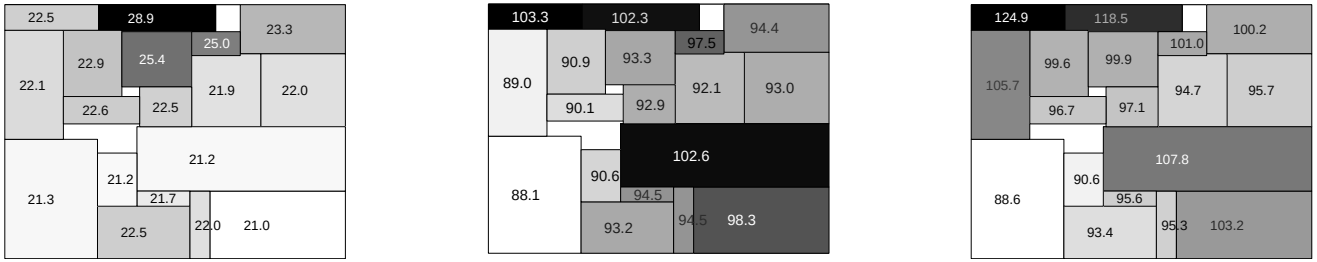
temperature of the blocks, thereby enabling us to find a feasible solution over the set of values of $\mathbf{T}$, such that D_{max} is maximized. If these constraints are not considered, the maximal critical path delays can be computed by simply assigning the temperature that maximizes the delay of each individual block, thereby leading to unrealistic cases where adjacent blocks may have their delays corresponding to the two corner temperatures, T_{min} and T_{max}, respectively.

In order to determine bounds on the power dissipation, we consider variations in both the active and leakage powers, with signal probabilities, and activity factors, as well as due to PVT variations. Accordingly, the P_i term in (4.18) denotes the sum of the active (A_i) and leakage (L_i) powers for the n different blocks in the chip, while the power of the remaining nodes are determined, based on [98]. Since leakage power $L_i(T)$, is a strong function of the operating temperature, and its value depends on the application run, signal probabilities of the nodes, as well as the ambient temperature, we characterize the leakage of our library cells at different temperatures, for varying signal probabilities. Leakage simulations are performed over several randomly chosen input signal probabilities for each circuit block, to determine the minimal and maximal leakage numbers at each temperature, and polynomial best-fit functions with respect to temperature, of the lower bound on the leakage $L_{\min}$, and the upper bound $L_{\max}$ are determined. Further, lower bounds on the leakage may be modified under additional standby mode optimization schemes, such as sleep-transistor settings, input nodal control, input vector control, reverse body biasing (RBB), etc. The lower and upper bounds are computed separately at each PV corner setting since leakage depends on both V_{dd} and V_t.

The active power, denoted as A_i for each of the n blocks in (4.18), is assumed to be temperature invariant, and hence lies within a lower bound A_{min} and an upper bound A_{max}, determined through simulations, based on the activity factors of these blocks, which can vary with different applications. If certain blocks have the ability to be power-gated or clock-gated, the lower bounds may be modified accordingly. Further, these bounds are computed at each voltage corner since active power is a quadratic function of V_{dd}.

Since the chip may be operated under different conditions, the ambient temperature T_{ambient}, is also assumed to vary over a certain range $[T_l, T_h]$, while the temperature of the blocks are assumed to vary between $T_{\min}$ and $T_{\max}$. Additional constraints are added since the temperature of the block

must be greater than the interface, which must be greater than the heat spreader, which in turn must be greater than that of the heat sink. The number of variables in (4.18) for a floorplan consisting of n blocks, is equal to $6n + 1$, $4n$ for the temperatures of the blocks, and their spreader, interface, and sink nodes, n each for the active and leakage powers, and one for the ambient temperature. It must be noted that the size of the nonlinear programming problem is linear in the number of blocks in the floorplan, and is therefore expected to be tractable. The size of the problem considered in this framework, and the runtimes are discussed in the next section.



(a) Slow corner (b) Nominal corner (c) Fast corner

Figure 4.11: Temperatures of the blocks at which the delay of the full chip path is maximized, at different corner settings: dark-colored blocks and relatively hotter than the light-colored ones.

While the above formulation does not consider the spatial correlations in power amongst different blocks, along with the spatial correlation in temperature, this information may also be suitably factored in the form of additional constraints in (4.18) based on microarchitectural simulations. The work in [99] provides examples of factoring in additional microarchitectural constraints such as certain functionally related blocks being concurrently on, maximal power constraint implying that at most l out of n blocks are active at a given time, etc. Such techniques can further enable us to reduce the pessimism associated with estimating the maximal delay of the critical paths, by placing additional restrictions on the relative power numbers of the blocks.

For simplicity in analysis, timing variations in the combinational data path alone are considered in this framework. The nonlinear programming formulation in (4.18) can however easily be modified to consider both the data and the clock networks by determining the lowest delay along the capturing path and the path with the highest delay along the generating logic cone, to check that the setup-time requirements are satisfied.

4.9.2 Results of Full-Chip Timing Analysis

Eighteen of the largest circuits from the ISCAS85, ITC99, and LGSYNTH93 combinational benchmark suites that were analyzed in Section 4.8 are assumed to constitute the core of a chip, whose floorplan is modeled upon the sample floorplan used in [6], and is shown in Fig. 4.10. The cells in these blocks are placed using Capo [100]. The delay of each of these blocks as a function of temperature is computed by using the quadratic delay model, as described in Section 4.8. Bounds on active and leakage power are also determined, by running multiple simulations using randomly chosen activity factors and signal probabilities for the primary input signals. The ambient temperature is assumed to vary between [0°C, 50°C], while the temperature of the chip itself is assumed to vary over [0°C, 130°C].

The nonlinear optimization problem is then solved using fmincon function of Matlab [101] to determine the temperatures of the circuit blocks, their active and leakage powers, and the maximum delay $D_{\max}$, of the full chip path, in (4.18). The problem formulation and optimization is repeated at each PV corner, since the delay, and the power numbers can change with PV variations. The complexity of the optimization problem depends on the number of blocks considered in the floor-

plan. The matrix G in (4.18) is a system of size $4n + 12$ for n blocks in the core, based on the modeling in [98], and n=23 for the chip shown in Fig. 4.10. Accordingly, the runtime for solving this nonlinear programming problem, consisting of around hundred variables, and a few hundred equality and inequality constraints is of the order of less than a minute, at each PV corner.

Table 4.10: Delay and temperatures for different corners for full-chip timing.

V_{dd}	Process	T_{avg} (°C)	T_{ambient} (°C)	D_{max} (ns)	D_{wc} (ns)	$\left(1 - \frac{D_{\text{max}}}{D_{wc}}\right)$ %
V_{min}	slow	22.77	12.37	68.1	72.1	5.5%
V_{nom}	slow	51.47	28.65	60.5	63.3	4.4%
V_{max}	slow	58.56	11.27	55.9	58.0	3.5%
V_{min}	nominal	20.38	0.17	48.3	50.7	4.7%
V_{nom}	nominal	94.47	35.12	44.6	46.1	3.2%
V_{max}	nominal	100.60	17.32	42.5	43.3	1.9%
V_{min}	fast	20.32	6.44	32.9	34.4	4.2%
V_{nom}	fast	70.52	27.85	31.2	32.2	2.9%
V_{max}	fast	96.70	27.74	30.3	30.9	2.0%

The results are tabulated in Table 4.10 for the nominal, slow, and fast process corners, for different settings of V_{dd}. The average temperatures (T_{avg}) of the 18 benchmarks, along with the ambient temperature (T_{ambient}) that results in the maximization of the objective function are tabulated. The results indicate that with increase in supply voltage, the temperatures of the blocks that maximize the delay $D_{\max}$, shift toward higher temperatures. This is consistent with Fig. 4.9, where the delays for the ITC99 benchmark des show lower PTD with increasing V_{dd}. These delays are also compared with the numbers computed using the worst-case method from [49] for each of the blocks (as shown in Table 4.8 of Section 4.8), and by simply summing these delays for the critical path (denoted as D_{wc}). Expectedly, our optimization approach reduces this upper-bound by up to 5.5% across different PV corner settings, as indicated by the last column of Table 4.10.

Fig. 4.11 shows the solution to the optimization problem in (4.18), i.e., the temperature of the various blocks that maximizes the delay, for the nominal, fast, and slow corners, whose settings are described in Section 4.8.3. Dark-colored blocks are relatively hotter than the light-colored ones. At the slow corner, V_{dd} is at its lowest setting, and the impact of V_t reduction with temperature is dominant leading to PTD, hence the delay is maximized at temperatures closer to $T_{\min}$ = 0°C,

as seen in Fig. 4.11(a). Similarly, at the fast corner, where V_{dd} is at its highest, the reduction in mobility dominates, and the delays largely exhibit a negative temperature dependence. Hence, in Fig. 4.11(c), the temperatures of the blocks are closer to $T_{\max}$ = 130°C. Further, the results indicate that the temperatures across the entire core at different settings vary by a maximum of around 40°C, while the temperatures across adjacent blocks vary by at most 20°C. This further justifies our approach of using a single localized temperature variable for all gates inside a circuit block, and also considering spatial correlations and power constraints, as opposed to merely using a worst-case method, based on [49].

Chapter 5

Conclusion

In this thesis, we presented several algorithms for reliability and variability-aware analysis and optimization of digital circuits.

We begin our thesis with an overview of Negative Bias Temperature Instability (NTBI), a PMOS transistor aging phenomenon. NBTI has become an important reliability concern in present day circuit design. The dynamics of interface trap generation and annealing (that govern the mechanism of NBTI) depend on a large number of complex factors, which can be analytically captured using the framework of Reaction-Diffusion (R-D) model. Our simple analytical model for NBTI, presented in the initial parts of Chapter 2 can be used to quantify the impact of transistor aging on its V_{th} degradation, and thereby the shift in the delay (frequency) of the circuit over its lifetime. Existing NBTI models fail to account for the effect of finite oxide thickness, and the role of the reaction phase during recovery, thereby leading to poor scalability, or an inaccurate fit with experimental data. A more detailed accurate model, that does not use the assumption that the oxide thickness is infinite is presented later on in Chapter 2. The framework for using this model in a multi-cycle gigahertz operation can be used to estimate the temporal delay degradation of digital circuits.

In the next chapter, we evaluate the impact of NBTI on the temporal degradation of a digital circuit. Further, with the use of high-k Hf-based dielectrics, the impact of PBTI on NMOS transistors has also become significant. Accordingly, in the beginning of Chapter 3, we present a framework to determine the V_{th} degradation of a transistor, using the reaction-diffusion (R-D) model, and integrate this into a static timing analyzer to compute the temporal delay degradation of digital circuits. A worst-case method to estimate the maximal impact of BTI in a pessimistic manner is detailed. The amount of overestimation in the delay of the circuit using this method is investigated. Our results indicate that the worst-case approach does not lead to a large amount of overestimation and hence can be used as a measure of the impact of BTI on digital circuit lifetime degradation. Besides, we also demonstrate the potential for an input vector control (IVC) approach that can mitigate the temporal degradation of circuits by arranging for the signal probabilities to be such that gates along the critical paths undergo minimal stress.

The latter part of Chapter 3 presents three circuit optimization techniques to robustly design cir-

cuits, accounting for BTI-induced temporal degradation. Gate sizing has previously been suggested to redesign the circuit, targeting it to tighter timing constraints, such that even after maximal degradation, the circuit still meets the original desired frequency of operation over its lifetime. However, we argue in Chapter 3 that incorporating the effects of BTI during logic synthesis better optimizes the circuit, thereby minimizing the area and power overhead. Accordingly, our work proposes a method to perform technology mapping, by taking into account the exact NBTI effect, and its dependency on the amount of time for which the gate has been stressed and relaxed. Circuits are synthesized to ensure optimal performance during the entire lifetime of around 10 years, despite NBTI induced temporal degradation. The results of this SP based NBTI-aware synthesis scheme are compared with a worst case NBTI library based synthesis, and the area-power savings that can be achieved are reported. Our experimental results indicate that an average of 10% savings in area and around 12% savings in power can be achieved using this method.

In the latter part of Chapter 3, we present a dynamic control technique to optimally design a circuit that can meet the original target frequency over its lifetime. Previous BTI-aware robust circuit design solutions in the presilicon design stage, aimed at guaranteeing reliable circuit performance, can lead to large area and associated power overheads. We propose an adaptive approach that determines the temporal degradation of the circuit, and compensates for it, through adaptive body biasing (ABB) and adaptive supply voltage (ASV). The results indicate that by combining the adaptive and synthesis approaches, circuits can be efficiently guardbanded over their lifetime, with a minimal overhead in area, and a small increase in power, as compared with a circuit designed only to meet the nominal specifications. Further, techniques such as those in [83] may be used to apply ABB/ASV to simultaneously counter the impact of aging, as well as process and temperature variations.

In the final part of Chapter 3, we analyze the impact of NBTI on the degradation of SRAM logic cells. We show that in particular, NBTI can worsen the static noise margin (SNM) of SRAM cells, and this can cause read stability issues. A novel technique of cell flipping has been proposed which can recover up to 30% of the noise margin degradation caused due to NBTI. Software and hardware approaches for implementing this technique in data caches are also discussed.

In the next chapter of this thesis we solve two problems involving variability-aware timing

analysis techniques. In the first part of Chapter 4, we present a framework to predict the timing of circuits when no knowledge of the exact distribution of the varying parameters is available, and only information about the parameters, and their ranges are known. We show how the delay of a gate can be modeled in this scenario as a normalized parameterized hyperplane in terms of the various parameters. We then present a block-based static timing analysis framework to determine the arrival times and margins at all timing-endpoints in a circuit, at any setting of the parameters within the hyperspace. We describe various pruning techniques during $\max$ propagation, in this paradigm. Our results indicate that this technique can efficiently determine the most timing critical paths in the circuit accurately, at different settings of the varying parameters, (min, max, worst-case, best-case, nominal, low-V_{dd}, high-coupling, etc), using a single sweep of forward propagation on the timing graph. Results are shown on 45nm based commercial microprocessor circuits, and this framework has been used to identify potential noncritical paths during design, that may end up as speed-limiting in silicon, due to their large sensitivities to variations.

The last part of this thesis deals with timing analysis under inverted temperature dependence (ITD). Existing solutions to handle ITD lead to a large pessimism in estimation, due to the underlying assumption that the temperature of each gate in a circuit block, can vary independently of the others. Accordingly, in the latter part of Chapter 4, we formulate an optimization problem to determine the temperatures of different blocks on a chip, subject to thermal and power constraints, such that the delay of the critical path is maximized. The problem is solved by first determining a means of estimating the maximal delay of a circuit block, and using the results of block-based STA in a nonlinear optimization framework. Modifications to the current corner based STA techniques are outlined. Our results show that using a thermal aware full-chip timing analysis formulation, the pessimism in estimating the maximal delays of the full-chip paths is reduced by an average of around 5%.

BIBLIOGRAPHY

[1] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "NBTI-Aware Synthesis of Digital Circuits." Presented at the Design Automation Conference June 2007.

[2] S. Chakravarthi, A. T. Krishnan, V. Reddy, C. Machala, and S. Krishnan, "A Comprehensive Framework for Predictive Modeling of Negative Bias Temperature Instability," in *Proceedings of the IEEE International Reliability Physics Symposium*, pp. 273–282, April 2004.

[3] A. T. Krishnan, C. Chancelor, S. Chakravarthi, P. E. Nicollian, V. Reddy, and A. Varghese, "Material Dependence of Hydrogen Diffusion: Implication for NBTI Degradation," in *IEEE International Electronic Devices Meeting*, pp. 688–691, December 2005.

[4] A. T. Krishnan. Private Communication.

[5] C. Shen, M. F. Li, C. E. Foo, T. Yang, D. M. Huang, A. Yap, G. S. Samudra, and Y.-C. Yeo, "Characterization and Physical Origin of Fast Vth Transient in NBTI of pMOSFETs with SiON Dielectric," in *IEEE International Electronic Devices Meeting*, pp. 333–336, November 2006.

[6] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-Aware Microarchitecture," in *Proceedings of the ACM International Symposium on Computer Architecture*, pp. 94–125, August 2004.

[7] D. K. Schroder and J. F. Babcock, "Negative Bias Temperature Instability: Road to Cross in Deep Sub-Micron Silicon Semiconductor Manufacturing," *Journal of Applied Physics*, vol. 94, pp. 1–18, January 2003.

[8] A. T. Krishnan, V. Reddy, S. Chakravarthi, J. Rodriguez, S. John, and S. Krishnan, "NBTI Impact on Transistor and Circuit: Models, Mechanisms and Scaling Effects," in *IEEE International Electronic Devices Meeting*, pp. 14.5.1–14.5.4, December 2003.

[9] V. Reddy, A. T. Krishnan, A. Marshall, J. Rodriguez, S. Natarajan, T. Rost, and S. Krishnan, "Impact of Negative Bias Temperature Instability on Digital Circuit Reliability," in *Proceedings of the IEEE International Reliability Physics Symposium*, pp. 248–253, April 2002.

[10] F. Crupi, C. Pace, G. Cocorullo, G. Groeseneken, M. Aoulaiche, and M. Houssa, "Positive Bias Temperature Instability in nMOSFETs with Ultra-Thin Hf-Silicate Gate Dielectrics," *Journal of Microelectronic Engineering*, vol. 80, pp. 130–133, June 2005.

[11] J. F. Zhang and W. Eccleston, "Positive Bias Temperature Instability in MOSFETs," *IEEE Transactions on Electron Devices*, vol. 45, pp. 116–124, January 1998.

[12] S. S. Sapatnekar, *Timing*, ch. Static Timing Analysis. Springer-Verlag New York, Inc, 2004.

[13] M. A. Alam, "A Critical Examination of the Mechanics of Dynamic NBTI for pMOSFETs," in *IEEE International Electronic Devices Meeting*, pp. 14.4.1–14.4.4, December 2003.

[14] 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, pp. 2004–2014, May 1977.

[15] S. Ogawa and N. Shiono, "Generalized Diffusion-Reaction Model for the Low-Field Charge-Buildup Instability at the Si-SiO2 interface," *Journal of Applied Physics*, vol. 51, pp. 4128–4230, February 1995.

[16] M. A. Alam and S. Mahapatra, "A Comprehensive Model of PMOS NBTI Degradation," *Journal of Microelectronics Reliability*, vol. 45, pp. 71–81, August 2004.

[17] S. Mahapatra, P. B. Kumar, and M. A. 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, pp. 1371–1379, September 2004.

[18] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "An Analytical Model for Negative Bias Temperature Instability (NBTI)," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 493–496, November 2006.

[19] R. Vattikonda, W. Wang, and Y. Cao, "Modeling and Minimization of PMOS NBTI Effect for Robust Nanometer Design," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 1047–1052, July 2006.

[20] S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, and S. Vrudhula, "Predictive Modeling of the NBTI Effect for Reliable Design," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 189–192, September 2006.

[21] T. Grasser, W. Gos, V. Sverdlov, and B. Kaczer, "The Universality of NBTI Relaxation and its Implications for Modeling and Characterization," in *Proceedings of the IEEE International Reliability Physics Symposium*, pp. 268–280, April 2007.

[22] H. Reisenger, O. Blank, W. Heinrigs, A. Muhlhoff, W. Gustin, and C. Schlunder, "Analysis of NBTI Degradation and Recovery Behavior Based on Ultra Fast Vt Measurements," in *Proceedings of the IEEE International Reliability Physics Symposium*, pp. 448–453, April 2006.

[23] H. Reisenger, O. Blank, W. Heinrigs, W. Gustin, and C. Schlunder, "A Comparison of Very Fast to Very Slow Components in Degradation and Recovery due to NBTI and Bulk Hole Trapping to Existing Physical Models," *IEEE Transactions on Devices and Materials Reliability*, vol. 7, pp. 119–129, March 2007.

[24] J. H. Stathis and S. Zafar, "The Negative Bias Temperature Instability in MOS Devices: A Review," *Journal of Microelectronics Reliability*, vol. 46, pp. 270–286, February 2006.

[25] C. R. Parthasarathy, M. Denais, V. Huard, G. Ribes, E. Vincent, and A. Bravaix, "New Insights into Recovery Characteristics Post NBTI Stress," in *Proceedings of the IEEE International Reliability Physics Symposium*, pp. 471–477, April 2006.

[26] S. Zafar, B. H. Lee, J. Stathis, A. Callegari, and T. Ning, "A Model for Negative Bias Temperature Instability (NBTI) in Oxide and High k pFETs," in *Digest of Symposium on VLSI Technology*, pp. 208–209, June 2004.

[27] B. Kaczer, V. Arkhipov, R. Degraeve, N. Collaert, G. Groeseneken, and M. Goodwin, "Disorder-Controlled Kinetics Model for NBTI and its Experimental Verification," in *Proceedings of the IEEE International Reliability Physics Symposium*, pp. 381–387, April 2005.

[28] V. Huard, C. R. Parthasarathy, C. Guerin, and M. Denais, "Physical Modeling of Negative Bias Temperature Instabilities for Predictive Exploration," in *Proceedings of the IEEE International Reliability Physics Symposium*, pp. 733–734, April 2006.

[29] V. Huard, M. Denais, and C. Parthasarathy, "NBTI Degradation: From Physical Mechanisms to Modeling," *Journal of Microelectronics Reliability*, vol. 46, pp. 1–23, January 2006.

[30] S. Mahapatra, K. Ahmed, S. Varghese, A. E. Islam, G. Gupta, L. Madhav, D. Saha, and M. A. Alam, "On the Physical Mechanism of NBTI in Silicon Oxynitride p-MOSFETs: Can Differences in Insulator Processing Conditions Resolve the Interface Trap Generation Versus Hole Trapping Controversy?," in *Proceedings of the IEEE International Reliability Physics Symposium*, pp. 1–9, April 2007.

[31] A. E. Islam, H. Kufluoglu, D. Varghese, and M. A. Alam, "Critical Analysis of Short-Term Negative Bias Temperature Instability Measurements: Explaining the Effect of Time-Zero Delay for on-the-fly Measurements," *Applied Physics Letters*, vol. 90, pp. 3505–3508, February 2007.

[32] A. E. Islam, H. Kufluoglu, D. Varghese, S. Mahapatra, and M. A. Alam, "Recent Issues in Negative Bias Temperature Instability: Initial Degradation, Field Dependence of Interface Trap Generation, Hole Trapping Effects, and Relaxation," *IEEE Transactions on Electron Devices*, vol. 54, pp. 2143–2154, September 2007.

[33] J. H. Lee, W. H. Wu, A. E. Islam, M. A. Alam, and A. Oates, "Separation Method of Hole Trapping and Interface Trap Generation and Their Roles in NBTI Reaction-Diffusion Model," in *Proceedings of the IEEE International Reliability Physics Symposium*, pp. 745–746, April 2008.

[34] J. Keane *et al.*, "An On-Chip NBTI Sensor for Measuring PMOS Threshold Voltage Degradation," in *Proceedings of the ACM International Symposium on Low Power Electronics and Design*, pp. 189–194, August 2007.

[35] T.-H. Kim, J. Liu, R. Persaud, and C. H. Kim, "Silicon Odometer: An On-Chip Reliability Monitor for Measuring Frequency Degradation of Digital Circuits," *IEEE Journal of Solid-State Circuits*, vol. 43, pp. 874–880, April 2008.

[36] B. C. Paul, K. Kang, H. Kufluoglu, M. A. Alam, and K. Roy, "Temporal Performance Degradation under NBTI: Estimation and Design for Improved Reliability of Nanoscale Circuits," in *Proceedings of the Design, Automation & Test in Europe*, pp. 1–6, March 2006.

[37] B. C. Paul, K. Kang, H. Kufluoglu, M. A. Alam, and K. Roy, "Impact of NBTI on the Temporal Performance Degradation of Digital Circuits," *IEEE Electron Device Letters*, vol. 26, pp. 560–562, August 2003.

[38] K. Kang, H. Kufluoglu, M. A. Alam, and K. Roy, "Efficient Transistor-Level Sizing Technique under Temporal Performance Degradation due to NBTI," in *Proceedings of the IEEE International Conference on Computer Design*, pp. 216–221, OCtober 2006.

[39] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "Impact of NBTI on SRAM Read Stability and Design for Reliability," in *Proceedings of the IEEE International Symposium on Quality Electronic Design*, pp. 205–210, March 2006.

[40] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "NBTI-Aware Synthesis of Digital Circuits," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 370–375, June 2007.

[41] S. V. Kumar, C. V. Kashyap, and S. S. Sapatnekar, "A Framework for Block Based Timing Sensitivity Analysis," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 688–693, July 2008.

[42] Y. Zhan, S. V. Kumar, and S. S. Sapatnekar, "Thermally Aware Design," *Foundations and Trends in Electronic Design Automation*, vol. 2, pp. 255–370.

[43] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "A Finite Oxide Thickness Based Model for Negative Bias Temperature Instability," *accepted for publication in IEEE Transactions on Devices and Materials Reliability*, 2009.

[44] S. V. Kumar and S. S. Sapatnekar, “Timing Analysis and Leakage Optimization Under Mixed Temperature Dependence,” *to be submitted to ACM Transactions on Design Automation of Electronic Systems*.

[45] S. V. Kumar and S. S. Sapatnekar, “Estimation of the Delay Degradation due to Aging (Bias Temperature Instability) in CMOS Circuits,” *Submitted to IEEE Transactions on VLSI Systems*.

[46] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, “Negative Bias Temperature Instability,” in *Proceedings of SRC TECHCON*, September 2008.

[47] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan, “First Order Incremental Block Based Statistical Timing Analysis,” in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 331–336, June 2004.

[48] H. Chang and S. S. Sapatnekar, “Statistical Timing Analysis under Spatial Correlations,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, pp. 1467–1482, September 2005.

[49] A. Dasdan and I. Hom, “Handling Inverted Temperature Dependence in Static Timing Analysis,” *Proceedings of the ACM Transactions on Design Automation of Electronic Systems*, vol. 11, pp. 306–324, April 2006.

[50] H. Ananthan, C. H. Kim, and K. Roy, “Larger than Vdd Forward Body Bias in sub-0.5V Nanoscale CMOS,” in *Proceedings of the IEEE International Symposium of Low Power Electronic Design*, pp. 8–13, 2004.

[51] V. Gerousis, “Design and Modeling Challenges for 90nm and 50nm,” in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 353–360, September 2003.

[52] M. A. Alam, “On the Reliability of Micro-electronic Devices: An Introductory Lecture on Negative Bias Temperature Instability,” in *Nanotechnology 501 Lecture Series*, September 2005. Available at `http://www.nanohub.org/resources/?id=193`.

[53] D. K. Schroder, "Negative Bias Temperature Instability: Physics, Materials, Process, and Circuit Issues," in *IEEE Solid-State Circuit Society*, August 2005. Available at `http://www.ewh.ieee.org/r5/denver/sscs/ Presentations/2005.08.Schroder.pdf`.

[54] B. Zhu, J. S. Suehle, Y. Chen, and J. B. Bernstein, "Negative Bias Temperature Instability of Deep Sub-micron p-MOSFETs Under Pulsed Bias Stress," in *IEEE International Integrated Reliability Workshop Final Report*, pp. 125–129, October 2002.

[55] G. Chen, M. F. Li, C. H. Ang, J. Z. Zheng, and D. L. Kwong, "Dynamic NBTI of p-MOS Transistors and its Impact on MOSFET Scaling," in *Proceedings of the IEEE International Reliability Physics Symposium*, pp. 196–202, April 2003.

[56] H. Kufluoglu and M. A. 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. 5, pp. 1101–1107, May 2007.

[57] S. Mahapatra, P. B. Kumar, T. R. Dalei, D. Saha, and M. A. Alam, "Mechanism of Negative Bias Temperature Instability in CMOS Devices: Degradation, Recovery and Impact of Nitrogen," in *IEEE International Electronic Devices Meeting*, pp. 105–108, December 2004.

[58] M. Ershov, R. Lindley, S. Saxena, A. Shibkov, S. Minehane, J. Babcock, S. Winters, H. Karbasi, T. Yamashita, P. Clifton, and M. Redford, "Transient Effects and Characterization Methodology of Negative Bias Temperature Instability in pMOS Transistors," in *Proceedings of the IEEE International Reliability Physics Symposium*, pp. 606–607, April 2003.

[59] G. Chen, K. Y. Chuah, M. F. Li, D. S. H. Chan, C. H. Ang, J. Z. Cheng, Y. Jin, and D. L. Kwong, "Dynamic NBTI of PMOS Transistors and its Impact on Device Lifetime," in *Proceedings of the IEEE International Reliability Physics Symposium*, pp. 196–200, April 2003.

[60] R. Fernandez, B. Kaczer, A. Nackaerts, S. Demuynck, R. Rodriguez, M. Nafria, and G. Groeseneken, "AC NBTI Studied in the 1 Hz – 2 GHz Range on Dedicated on-chip CMOS circuits," in *IEEE International Electronic Devices Meeting*, pp. 337–340, December 2006.

[61] W. Wang, V. Reddy, A. T. Krishnan, R. Vattikonda, S. Krishnan, and Y. Cao, "Compact Modeling and Simulation of Circuit Reliability for 65nm-CMOS Technology," *IEEE Transactions on Devices and Materials Reliability*, vol. 7, pp. 509–517, December 2007.

[62] S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, and S. Vrudhula, "Scalable Model for Predicting the Effect of Negative Bias Temperature Instability for Reliable Design," *IET Transactions on Circuits, Devices and Systems*, vol. 2, pp. 361–371, August 2008.

[63] S. Rangan, N. Mielke, and E. C. C. Yeh, "Universal Recovery Behavior of Negative Bias Temperature Instability in pMOSFETs," in *IEEE International Electronic Devices Meeting*, pp. 14.3.1–14.3.4, December 2003.

[64] W. Wand, S. Yang, S. Bhardwaj, S. Vrudhula, F. Liu, and Y. Cao, "The Impact of NBTI Effect on Combinational Circuit: Modeling, Simulation, and Analysis," *IEEE Transactions on VLSI Systems*, to appear 2009.

[65] M. Denais, V. Huard, C. Parthasarathy, G. Ribes, F. Perrier, N. Revil, and A. Bravaix, "Oxide Field Dependence of Interface Trap Generation During Negative Bias Temperature Instability in PMOS," in *IEEE International Integrated Reliability Workshop Final Report*, pp. 109–112, October 2004.

[66] A. E. Islam, E. N. Kumar, H. Das, S. Purawat, V. Maheta, H. Aono, E. Murakami, S. Mahapatra, and M. A. Alam, "Theory and Practice of On-The-Fly and Ultra-Fast-Vt Measurements for NBTI Degradation: Challenges and Opportunities," in *IEEE International Electronic Devices Meeting*, pp. 805–808, December 2007.

[67] N. K. Jha and V. R. Rao, "A New Oxide-Trap Assisted NBTI Model," *IEEE Electron Device Letters*, vol. 26, pp. 687–689, September 2005.

[68] B. Zhang. Private Communication.

[69] M. F. Li, G. Chen, C. Shen, X. P. Wang, H. Y. Yu, Y.-C. Yeo, and D. L. Kwong, "Dynamic Bias Temperature Instability in Ulthathin SiO_2 and HfO_2 Metal Oxide Semiconductor Field Effect Transistors and its Impact on Device Lifetime," *Japanese Journal of Applied Physics*, vol. 43, pp. 7807–7814, November 2004.

[70] K. Kang, H. Kufluoglu, K. Roy, and M. A. Alam, "Impact of Negative Bias Temperature Instability in Nano-Scale SRAM Array: Modeling and Analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, pp. 1770–1781, October 2007.

[71] H.-S. Jung, S. K. Han, M. J. Kim, J. P. Kim, Y.-S. Kim, H. J. Lim, S. J. Doh, J. H. Lee, M. Y. Yu, J.-H. Lee, N.-I. Lee, H.-K. Kang, S. G. Park, and S. B. Kang, "PBTI and HCI Characteristics for High-k Gate Dielectrics with Poly-Si and MIPS Gates," in *Proceedings of the IEEE International Reliability Physics Symposium*, pp. 50–53, April 2005.

[72] S. Zafar, Y. Kim, V. Narayanan, C. Cabral, V. Paruchuri, B. Doris, J. Stathis, A. Callegari, and M. Chudzik, "A Comparative Study of NBTI and PBTI Charge Trapping in SiO_2-HfO_2 Stacks with FUSI, TiN, Re Gates," in *Proc. Symposium on VLSI Technology*, pp. 23–25, October 2006.

[73] R. Burch, F. N. Najm, P. Yang, and T. N. Trick, "A Monte Carlo Approach for Power Estimation," *IEEE Transactions on VLSI Systems*, vol. 1, pp. 63–71, March 1993.

[74] D. Lee and D. Blaauw, "Static Leakage Reduction through Simultaneous Threshold Voltage and State Assignment," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 191–196, June 2003.

[75] "Predictive Technology Model." Available at `http://www.eas.asu.edu/~ptm`.

[76] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis," Tech. Rep. UCB/ERL M92/41, University of California, Berkeley, 1992. Available at `http://www-cad.eecs.berkeley.edu/research/sis`.

[77] K. Kang, S. Gangwal, and K. Roy, "NBTI Induced Performance Degradation in Logic and Memory Circuits: How Effectively can we Approach a Reliability Solution," in *Proceedings of the Asia-South Pacific Design Automation Conference*, pp. 726–731, January 2008.

[78] J. W. Tschanz, J. T. Kao, S. G. Narendra, R. Nair, D. A. Antoniadis, A. P. Chandrakasan, and V. De, "Adaptive Body Bias for Reducing Impacts of Die-to-Die and Within-Die Parameter

Variations on Microprocessor Frequency and Leakage," *IEEE Journal of Solid-State Circuits*, vol. 37, pp. 1396–1402, November 2002.

[79] A. Keshavarazi, S. Narendra, B. Bloechel, S. Borkar, and V. De, "Forward Body Bias for Microprocessors in 130-nm Technology Generation and Beyond," *IEEE Journal of Solid-State Circuits*, vol. 38, pp. 696–701, May 2003.

[80] J. W. Tschanz, N. S. Kim, S. Dighe, J. Howard, G. Ruhl, S. Vanga, S. Narendra, Y. Hoskote, H. Wilson, C. Lam, M. Shuman, C. Tokunga, S. Somasekhar, S. Tang, D. Finan, T. K. an N. Borkar, N. Kurd, and V. De, "Adaptive Frequency and Biasing Techniques for Tolerance to Dynamic Temperature-Voltage Variations and Aging," in *Proceedings of the IEEE International Solid-State Circuits Conference*, pp. 292–294, February 2007.

[81] M. Terauchi, "Impact of Forward Substrate Bias on Threshold Voltage Fluctuation in Metal-Oxide-Semiconductor Field-Effect Transistors," *Japanese Journal of Applied Physics*, vol. 46, pp. 4105–4107, July 2007.

[82] J. W. Tschanz, S. G. Narendra, R. Nair, and V. De, "Effectiveness of Adaptive Supply Voltage and Body Bias for Reducing Impact of Parameter Variations in Low Power and High Performance Microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 38, pp. 826–829, May 2003.

[83] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "Body Bias Voltage Computations for Process and Temperature Compensation," *IEEE Transactions on VLSI Systems*, vol. 16, pp. 249–262, March 2008.

[84] V. Reddy, J. Carulli, A. Krishnan, W. Bosch, and B. Burgess, "Impact of Negative Bias Temperature Instability on Product Parametric Drift," in *Proceedings of the IEEE International Test Conference*, pp. 148–155, October 2004.

[85] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu, "New Paradigm of Predictive MOSFET and Interconnect Modeling for Early Circuit Design," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 201–204, June 2000.

[86] C. Visweswariah, "Death, Taxes, and Failing Chips," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 343–347, June 2003.

[87] K. Killpack, C. V. Kashyap, and E. Chiprout, "Silicon Speedpath Measurement and Feedback into EDA Flows," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 390–395, June 2007.

[88] S. Onaissi and F. N. Najm, "A Linear Time Approach for Static Timing Analysis Covering All Process Corners," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 217–224, November 2006.

[89] L. G. Silva, M. Silveira, and J. R. Phillips, "Efficient Computation of the Worst-Delay Corner," in *Proceedings of the Design, Automation & Test in Europe*, pp. 1–6, April 2007.

[90] "ILOG CPLEX Version 7.1." available at http://www.ilog/com/products/cplex/index.cfm.

[91] Y. Taur and T. H. Ning, *Fundamentals of Modern VLSI Devices*. New York, USA: Cambridge University Press, 1999.

[92] E. Pop, S. Sinha, and K. E. Goodson, "Heat Generation and Transport in Nanometer-Scale Transistors," *Proceedings of the IEEE*, vol. 94, pp. 1587–1601, August 2006.

[93] K. Kanda, K. Nose, H. Kawaguchi, and T. Sakurai, "Design Impact of Positive Temperature Dependence on Drain Current in sub-1V CMOS VLSIs," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 1559–1564, May 1999.

[94] B. Lasbouygues, R. Wilson, N.Azemard, and P. Maurine, "Timing Analysis in Presence of Supply Voltage and Temperature Variations," in *Proceedings of the International Symposium on Physical Design*, pp. 10–16, April 2006.

[95] K. R. Heloue and F. N. Najm, "Efficient Block-Based Parameterized Timing Analysis Covering All Potentially Critical Paths," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 173–180, November 2008.

[96] H. Hamann, A. Weger, J. A. Lacey, Z. Hu, P. Bose, E. Cohen, and J. Wakil, "Hotspot-Limited Microprocessors: Direct Temperature and Power Distribution Measurements," *IEEE Journal of Solid-State Circuits*, vol. 42, pp. 56–65, January 2007.

[97] H. Hamann, J. Lacey, A. Weger, and J. Wakil, "Spatially Resolved Imaging of Microprocessor Power (SIMP): Hotspots in Microprocessors," in *Proceedings of the Thermal and Thermomechanical Phenomena in Electronics Systems*, pp. 5–9, June 2006.

[98] "Hotspot - Version 4.1." available at http://lava.cs.virginia.edu/HotSpot/documentation.htm.

[99] H. Qian, S. Nassif, and S. S. Sapatnekar, "Early-Stage Power Grid Analysis for Uncertain Working Modes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, pp. 676–682, May 2005.

[100] "Capo: A Large-scale Fixed-die Floorplacer." available at http://vlsicad.eecs.umich.edu/BK/PDtools/Capo.

[101] "Matlab Optimization Toolbox: fmincon." available at http://www.mathworks.com/access/helpdesk/help/toolbox/optim/index.html.

[102] M. Sipser, *Introduction to the Theory of Computation*. PWS Publishing Company, 2005.

Appendix A

Generalized Calculations for Determining the Number of Interface Traps at the End of any Stress/Relaxation Phase

A.1 Stress Phase

In this subsection, we derive the trap generation calculations for the $(k+1)^{th}$ stress phase that runs from time $2kt_0$ to $(2k+1)t_0$. The effective time t_{eff}, at the beginning of this period satisfies

$$N_{IT}(2kt_0) = s_{2k}k_{IT}(D_H t_0)^{\frac{1}{6}} = k_{IT}(D_H t_{\text{eff}})^{\frac{1}{6}} \tag{A.1}$$

This implies that $t_{\text{eff}} = s_{2k}^6 t_0$. Therefore, the effective diffusion front at the beginning of this period is at $x_d(2kt_0) = s_{2k}^3 (2Dt_0)^{\frac{1}{6}}$, and it expands during this period as:

$$x_d(2kt_0 + t) = \sqrt{2D(t + s_{2k}^6 t_0)} \tag{A.2}$$

Therefore,

$$\begin{aligned} N_{IT}(2kt_0 + t) &= k_H (N_H^0)^2 x_d(2kt_0 + t) \\ &= k_H (N_H^0)^2 \sqrt{D_H(t + s_{2k}^6 t_0)} \end{aligned} \tag{A.3}$$

Using techniques similar to those employed in the derivation for the second stress phase (Sec 2.1.6), we substitute this in (2.8) to obtain

$$N_{IT}(2kt_0 + t) = \left(\frac{t}{t_0} + s_{2k}^6\right)^{\frac{1}{6}} N_{IT}(t_0) \tag{A.4}$$

At $t = t_0$, we have

$$N_{IT}((2k+1)t_0) = (1 + s_{2k}^6)^{\frac{1}{6}} N_{IT}(t_0)$$

and therefore

$$s_{(2k+1)} = (1 + s_{2k}^6)^{\frac{1}{6}}.$$

A.2 Relaxation Phase

In this subsection, we derive the trap generation calculations for any k^{th} relaxation phase that runs from time $(2k-1)t_0$ to $2kt_0$. The hydrogen front during the k^{th} phase continues to diffuse in an identical manner as that in the first relaxation phase (Fig. 2.3(f)-(g) and Fig. 2.4(f)-(g)). However, since the number of interface traps available for back-diffusion increases in dependent on the diffusion front in the previous cycles, we account for this by introducing a new term x_{ratio}, such that:

$$x_{\text{ratio}}((2k-1)t_0+t) = \frac{x(t+(2k-1)t_0) - x(t+2(k-1)t_0)}{x(t+t_0)} \tag{A.7}$$

The key idea is that the recovery in any given cycle is dependent only on the interface traps generated in the immediately preceding stress cycle. Accordingly. the number of annealed traps, based on the calculations in Section 2.1.5, is given by:

$$N^*_{IT}((2k-1)t_0+t) = \frac{1}{2}x_{\text{ratio}}((2k-1)t_0+t)N^{\Delta}_{H_2}((2k-1)t_0+t)\sqrt{\xi 2Dt}$$

It can be seen that if we substitute $k=1$, we get $x_{\text{ratio}}(t_0+t)=1$ and the above equation reduces to (2.23), which is the backward diffusion equation for the first recovery case. The number of interface traps can be expressed as the sum of the interface trap density at the end of the previous relaxation phase and the new traps created during the current stress/relaxation phases. Accordingly, $N_{IT}((2k-1)t_0+t) \approx N_{IT}((2k-2)t_0)+\frac{1}{2}N^{\Delta}_{H_2}((2k-1)t_0+t)(x((2k-1)t_0+t)-x(2(k-1)t_0+t))$. Simplifying, we have

$$N^*_{IT}((2k-1)t_0+t) = (N_{IT}((2k-1)t_0+t) - N_{IT}((2k-2)t_0))\sqrt{\frac{\xi t}{t_0+t}} \tag{A.9}$$

Therefore, substituting this in

$$N_{IT}((2k-1)t_0+t) = N_{IT}((2k-1)t_0) - N^*_{IT}((2k-1)t_0+t),$$

we have

$$N_{IT}((2k-1)t_0+t) = \frac{N_{IT}((2k-1)t_0) + N_{IT}((2k-2)t_0)\sqrt{\frac{\xi t}{t_0+t}}}{1+\sqrt{\frac{\xi t}{t_0+t}}} \tag{A.11}$$

At $t = t_0$, we have

$$N_{IT}(2kt_0) = \frac{2}{3}N_{IT}((2k-1)t_0) + \frac{1}{3}N_{IT}(2(k-1)t_0)$$

and therefore

$$s_{2k} = \frac{2}{3}s_{2k-1} + \frac{1}{3}s_{2k-2}$$

Appendix B

NBTI Model Assuming Atomic Hydrogen Diffusion

Although it is widely speculated that the diffusing species inside the oxide due to NBTI is molecular hydrogen, some researchers believe that the diffusion phase of NBTI action involves atomic hydrogen species. While the analytical expressions and the methodology to compute the number of interface traps generated due to NBTI have been outlined in detail for the H_2 diffusion case, a similar analysis can also be performed for the H diffusion case. Solving the R-D model equations for the continuous stress case gives us the familiar result [13, 16, 17, 37]:

$$N_{IT}(t) = \sqrt{\frac{k_f N_0}{k_r}}(Dt)^{\frac{1}{4}} \tag{B.1}$$

The first relaxation phase and the subsequent stress and relaxation phases can also be solved in the same way as in Sections 2.1.4-2.1.7. Expressing the final results using the "s_k notation" outlined in Section 2.3.2, and computing $N_{IT}(t_0)$ using (B.1), we can write:

$$s_{nk+i} = \left\{ \begin{array}{ll} (i + s_{kn}^4)^{\frac{1}{4}} & nk < i \leq nk + m \\ \frac{s_{nk+m} + s_{nk}\left(\frac{i}{2(m+i)}\right)^{\frac{1}{2}}}{1+\left(\frac{i}{2(m+i)}\right)^{\frac{1}{2}}} & nk + m < i \leq (n+1)k \end{array} \right\} \tag{B.2}$$

Appendix C

Proof of (2.42)

We provide a proof for the approximation made in (2.42). From (2.38), we have:

$$s_k^6 = s_{k-1}^6 + 1 \tag{C.1}$$

$$s_{k-1} = \frac{2}{3}s_{k-2} + \frac{1}{3}s_{k_1-3} \tag{C.2}$$

$$s_{k-2}^6 = s_{k-3}^6 + 1 \tag{C.3}$$

Substituting for s_{k-3} in (C.2) using (C.3), and using the value of s_{k-1} from (C.2) in (C.1), we have

$$s_k^6 = \left(\frac{2}{3}s_{k-2} + \frac{1}{3}\left(s_{k_2}^6 - 1\right)^{\frac{1}{6}}\right)^6 + 1 \tag{C.4}$$

The above equation can be re-written as:

$$s_k^6 - 1 = y = (mx + (1-m)(x^6 - 1)^{\frac{1}{6}})^6$$

where $m = \frac{2}{3}$ and $x = s_{k-2}$. The right hand side can be simplified as follows:

$$y = x^6\left(m + (1-m)\left(1 - \frac{1}{x^6}\right)^{\frac{1}{6}}\right)^6$$

Using $(1-a)^{\frac{1}{6}} \approx 1 - \frac{a}{6}$ for small a,

$$\begin{aligned} y &\approx x^6\left(m + (1-m)\left(1 - \frac{1}{6x^6}\right)\right)^6 \\ &= x^6\left(1 - \frac{(1-m)}{6x^6}\right)^6 \end{aligned}$$

Using $(1-a)^6 = 1 - 6a$ for small a,

$$\begin{aligned} y &\approx x^6\left(1 - 6\frac{(1-m)}{6x^6}\right)^6 \\ &= x^6 - (1-m) \end{aligned}$$

Re-substituting for y, m and x, we have

$$s_k^6 = s_{k-2}^6 - \frac{2}{3}$$

which is used in (2.42).

Appendix D

Proof for NP Completeness of NBTI-Satisfiability Problem

In this section, we show that the problem of determining whether a satisfying assignment to the primary inputs exists such that the "max delay" is equal to the "worst-case" delay is NP complete. The problem can be stated as follows:

NBTI-SAT = $\{< \text{C} > \mid$ There exists some feasible signal probability assignment for the primary inputs of circuit C, such that its "max delay" is equal to the "worst-case" delay.$\}$

Theorem 1:

NBTI-SAT is NP-complete.

Proof:

We first show that the problem is NP-hard by providing a verifier, as well as a nondeterministic polynomial time Turing machine for NBTI-SAT.

We show a verifier V for NBTI-SAT, given some certificate, which in this case is some assignment $\mathbf{P} = p_1, \ldots, p_n$ to the primary inputs as:

V = "On input $<< \text{C} >, \mathbf{P} >$:

1. Compute the "worst-case" delay of the circuit.
2. Determine the critical path of the circuit C, and compute whether the SP of each node along the critical path must be a 0 or a 1, or can be a don't care.
3. Use the SP values from $\mathbf{P}$ for the primary inputs, to determine the signal probability of every other node in the circuit.
4. Compare these SP values with the SP values computed in Line 2.
5. If they are the same, accept; otherwise reject."

Thus, NBTI-SAT is in NP.

The following nondeterministic polynomial time Turning machine N can be used to decide NBTI-SAT:

N = "On input $< \text{C} >$:

1. Assume that the SP of all internal and input nodes of C are 1, and compute the "worst-case" delay of the circuit.

2. Determine the critical path of the circuit C, and compute whether the SP of each node along the critical path must be a 0 or a 1, or can be a don't care.

3. Nondeterministically assign p_i = 0 or 1, for $i = 1, \ldots, n$.

4. Determine the signal probabilities of every other node in the circuit.

5. Compare these SP values with the SP values returned from Line 2.

6. If they are the same, accept; otherwise reject."

Hence, NBTI-SAT is in NP.

We now show that NBTI-SAT is reducible to SAT in polynomial time, and use the fact that SAT is NP-complete [102] to show that NBTI-SAT is also NP-complete:

We first compute the "worst-case" delay of the circuit C, as mentioned above, and identify the critical path. We then determine the signal probability at the primary output of this critical path. Since the optimal SP values for the "worst-case" delay are guaranteed to be either 0 or 1, the SP of the primary output is also either a 0, or a 1, implying that the output is either logic 1 or 0. The problem is now reducible to determining whether:

1. A satisfying boolean assignment at the primary inputs exists such that the logic of this primary output node is a 0 or a 1, as required by the "worst-case" NBTI conditions.

2. This assignment also satisfies the SP requirements of every other node on the critical path.

Clearly, this problem is equivalent to that of a boolean satisfiability, along all nodes lying on the critical path.

Thus, NBTI-SAT is reducible to SAT in polynomial time, and is therefore, also NP-complete.

Appendix E

Max of Two Quadratic Functions

In this section, we show how the maximum of two quadratic functions can be upper bounded by a quadratic function in $O(1)$ time. Given two quadratic arrival times D_1 and D_2, as a function of the normalized temperature variable x:

$$\begin{aligned} D_1(x) &= a_1x^2 + b_1x + c_1 \\ D_2(x) &= a_2x^2 + b_2x + c_2 \end{aligned} \tag{E.1}$$

where x varies in [0,1] (for convenience), we seek to compute a quadratic upper bound on:

$$\begin{aligned} D_3(x) &= \max(D_2(x), D_1(x)) \\ &= \max(D_2(x) - D_1(x), 0) + D_1(x) \end{aligned} \tag{E.2}$$

Equivalently, we seek to compute a quadratic upper bound on $\max(D_4(x), 0)$, where $D_4 = a_4x^2 + b_4x + c_4 = (D_2 - D_1)$. Let this quadratic upper bound be represented as $D_5 = a_5x^2 + b_5x + c_5$. Further, it is desired that D_5 be computed in $O(1)$ time, and the overestimation be minimized.

Three different cases arise during the computation of $D_5 = \max(D_4, 0)$ as follows, based on the monotonicity and convexity of D_4, and the number of intersections with the x-axis,in [0,1]:

- **Trivial Case - Zero intersections with x-axis**

  If D_4 is always positive, then $\max(D_4, 0) = D_4$. Similarly, if D_4 is always negative, then the $\max$ is simply equal to 0.

- **D_4 varies monotonically, and intersects the x axis once.**

  Two subcases arise based on the shape of D_4:

  1. If D_4 **increases** monotonically, we have:

$$\begin{aligned} D_4(0) = c &< 0 \\ D_4(1) = a + b + c &> 0 \end{aligned} \tag{E.3}$$

A quadratic upper bound on the $\max$ may be obtained as:

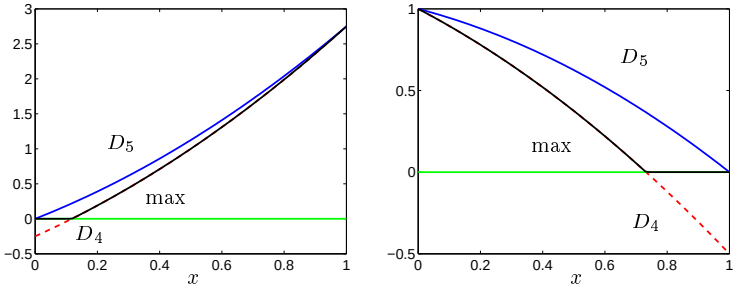
$$D_5 = ax^2 + bx + cx \tag{E.4}$$

since $ax^2 + bx + cx \geq ax^2 + bx + c$, for all x in [0,1] when $c < 0$. The minimum value of D_5 in [0,1] occurs at $x = 0$, and is equal to 0, while the maximum value (at $x = 1$) is equal to the maximum value of D_4 and is given by $a+b+c$, thereby satisfying $D_5 = \max(D_4, 0)$. An instance of this case in shown in Fig. E.1(a).

If D_4 is a linear curve that intersects the origin, D_5 reduces to

$$D_5(x) = (b + c)x \tag{E.5}$$

which is an increasing linear function, such that its value is 0 at $x = 0$, and $b + c$ at $x = 1$.



(a) D_4 increases monotonically.

(b) D_4 decreases monotonically.

Figure E.1: Max for a quadratic function.

2. Similarly, if D_4 **decreases** monotonically, we have:

$$\begin{aligned} D_4(0) &= c & &> 0 \\ D_4(1) &= a + b + c & &< 0 \end{aligned} \tag{E.6}$$

Accordingly, D_5 is given by:

$$D_5 = ax^2 - (a+c)x + c \tag{E.7}$$

since $ax^2-(a+c)x+c \geq ax^2+bx+c$, when $a+b+c<0$, for all $x \geq 0$. The minimum value of D_5 in [0,1] occurs at $x = 1$, and is equal to 0, while the maximum value (at $x = 0$) is equal to the maximum value of D_4 and is given by c, thereby satisfying $D_5 = \max(D_4, 0)$. This case is shown in Fig. E.1(b).

It can be seen that if D_4 is linear, D_5 in this case reduces to:

$$D_5(x) = c(1-x) \tag{E.8}$$

whose value is c at $x = 0$, and 0 at $x = 1$.

While the $\max$ of a linear function and 0 can be upper bounded using a linear expression as shown in (E.5), and (E.8), a tighter quadratic bound on the $\max$ can be obtained for the two cases as follows:

1. If D_4 is a linear increasing function such that $c < 0$ and $b + c > 0$, then $\max(bx + c, 0)$ is given by:

$$D_5^* = -cx^2 + (b+2c)x \tag{E.9}$$

The minimum value of D_5^*, which is a monotonically increasing quadratic function in [0,1], is equal to 0 at $x = 0$, while the maximum value in [0,1] is given by $b + c$. To assert that D_5^* is always greater than $D_4 = bx + c$ and is less than (E.5), we have:

$$\begin{aligned} D_5^* - D_4 &= [-cx^2 + (b+2c)x] - [bx + c] \\ &= -cx^2 + 2cx - c \\ &= -c(x-1)^2 \end{aligned} \tag{E.10}$$

which is > 0 for all x in [0,1], since c is negative, while:

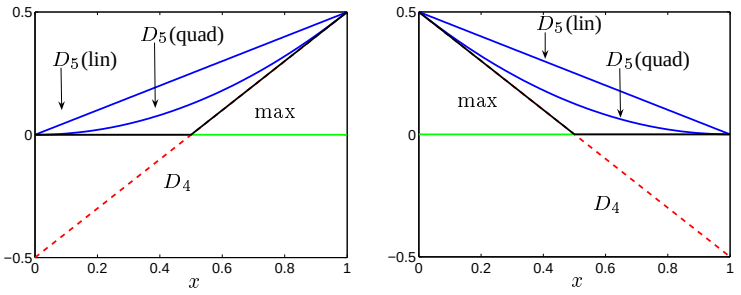
$$\begin{aligned} D_5^* - (b+c)x &= [-cx^2 + (b+2c)x] - [(b+c)x] \\ &= -cx^2 + cx \\ &= cx(1-x) \end{aligned} \tag{E.11}$$

which is < 0 for all x in [0,1], since c is negative. Thus, D_5^* from (E.9) provides a tighter upper bound on the $\max$, than D_5 from (E.5).

2. Similarly, for the case where D_4 is a monotonically decreasing linear function, such that $c > 0$, and $(b+c) < 0$, we have:

$$D_5^* = -(b+c)x^2 + bx + c \tag{E.12}$$

Fig. E.2 shows the linear and the quadratic $\max$ for the two cases, denoted as D_5(lin) and D_5(quad), respectively. The quadratic $\max$ is in effect tangential to the x-axis and to the linear curve $D_4 = bx + c$, at the two end points, respectively.



(a) D_4 increases monotonically.

(b) D_4 decreases monotonically.

Figure E.2: Max for a linear function.

- D_4 **is quadratic, and is nonmonotonic**

Two subcases arise when D_4 is **convex**, depending on the shape of the curve:

1. If the minimum value of D_4 is less than 0, this case is equivalent to the case where D_4 varies monotonically, since the minimum value, and the nonmonotone of D_4 below the x-axis has no effect on the shape of the $\max$, as compared with a case where D_4 increases monotonically.

2. Fig. E.3 shows the case where D_4 is a convex function that intersects the x-axis twice in [0,1], its values at $x = 0$ and $x = 1$ are both positive, while its minimum at some x^* in [0,1] is negative. An upper bounded convex quadratic on this function may be obtained by using the following conditions:

$$\begin{aligned} D_5(0) &= D_4(0) \\ D_5(1) &= D_4(1) \\ D_5(x^{\#}) &= 0 \\ \frac{dD_5}{dx} &= 0 \text{ at } x = x^{\#} \end{aligned} \tag{E.13}$$

   to solve for $x^{\#}$ (in [0,1] which is the point at which D_5 is minimized), a_5, b_5, and c_5. The resulting curve is shown in Fig. E.3.

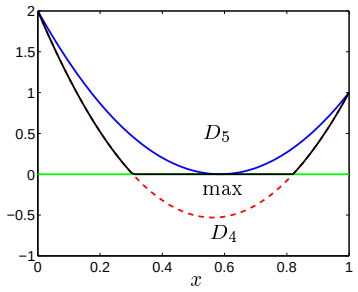
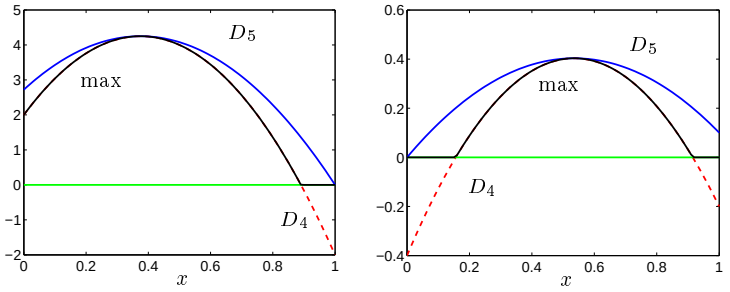


Figure E.3: Case where D_4 is convex, and intersects the x-axis twice in [0,1].

Similarly, two subcases arise when D_4 is **concave** with its maximal value occurring at some x^* in [0,1]:



(a) D_4 intersects x-axis once in [0,1]

(b) D_4 intersects x-axis twice in [0,1]

Figure E.4: Case where D_4 is concave.

1. If D_4 intersects the axis once in [0,1], as shown in Fig. E.4(a), and is nonmonotonic, the $\max$ can be determined in $O(1)$ time, by using the following conditions:

$$\begin{aligned} D_5(x^*) &= D_4(x^*) \\ \frac{dD_5}{dx} &= 0 \text{ at } x = x^* \end{aligned} \tag{E.14}$$

   and that the minimal value of D_5 is 0, at either $x = 0$ or $x = 1$, where D_4 is negative. D_5 is a raised version of D_4 such that its lowest value is 0, and its highest value is equal to that of D_4.

2. If D_4 intersects the axis twice in [0,1], as shown in Fig. E.4(b), and is nonmonotonic, the $\max$ can be determined in $O(1)$ time, by using the following conditions:

$$\begin{aligned} D_5(x^*) &= D_4(x^*) \\ \frac{dD_5}{dx} &= 0 \text{ at } x = x^* \end{aligned} \tag{E.15}$$

   along with $D_5(0) = 0$ if $x^* \geq 0.5$, or $D_5(1) = 0$ otherwise, thereby making D_5 a

positive raised version of D_4.

Thus, the $\max$ function of two quadratic delay arrival times can be computed efficiently in $O(1)$ time. Our simulation results show that given the nature of the variation of the arrival times along different gates in the circuit, the maximum overestimation in the delays is less than 2%, as seen from the results in Table 4.9.