Resilient Design for Process and Runtime Variations

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

der Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Farshad Firouzi

Tag der mündlichen Prüfung: 12. February 2015

Erster Gutachter: Prof. Dr. Mehdi B. Tahoori, Chair of Dependable Nano Computing, Department of Computer Science and Engineering, Karlsruhe Institute of Technology (KIT), Germany

Zweiter Gutachter: Prof. Dr. Krishnendu Chakrabarty, Department of Electrical and Computer Engineering, Duke University, USA

Farshad Firouzi
76131 Karlsruhe

Hiermit erkläre ich an Eides statt, dass ich die von mir vorgelegte Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen, Internet-Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen – die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

_____

Farshad Firouzi

ACKNOWLEDGMENTS

First of all, I would like to express my sincere gratitude to my advisers Prof. Mehdi Tahoori and Prof. Krish Chakrabarty for their support and guidance in every single step of my doctorate studies. Their insightful advice helped me to significantly improve my research philosophy and attitude.

I would also like to express my deepest appreciation to Dr. Sani Nassif for his multi-disciplinary feedback and his hospitality during my visit to IBM. I am always overwhelmed by his genius and morality. My special thanks to my colleagues who worked shoulder by shoulder with me. I owe many thanks to Fangming Ye for his collaboration.

A special word of thanks for the members of my defense exam committee (in alphabetical order): Prof. Wolfgang Karl, Prof. Jörn Müller-Quade, Prof. Hartmut Prautzsch, and Prof. Peter Sanders.

Every now and then I am also very grateful to my teacher Mr. Sohrabi. When I think about inspiration I think about one thing in my life, my spiritual master Prof. Sied Mehdi Fakhraie. Words can never express the feelings, nor my thanks that I owe him. Rest in peace Mehdi.

I dedicate this dissertation to my parents, my family, and my wife for their endless love, support, and encouragement to learn and explore. Thank you all and I love you so much.

Finally, thank you for picking up my dissertation.

To my family, for their love and support.

The most beautiful thing we can experience is the mysterious.
Albert Einstein

LIST OF OWN PUBLICATIONS INCLUDED IN THIS THESIS

[1] F. Firouzi, F. Ye, K. Chakrabarty, and M. Tahoori, "Aging- and variation-aware delay monitoring using representative critical path selection," in *ACM Transactions on Design Automation of Electronic Systems (Under Review)*, 2015.

[2] F. Ye, F. Firouzi, Y. Yang, K. Chakrabarty, and M. Tahoori, "On-chip voltage-droop prediction based on support-vector machines and feature selection," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (Under Review)*, 2015.

[3] F. Firouzi, F. Ye, S. Kiamehr, K. Chakrabarty, and M. Tahoori, "Adaptive mitigation of parameter variations," in *Asian Test Symposium (ATS)*, 2014.

[4] F. Firouzi, F. Ye, K. Chakrabarty, and M. Tahoori, "Chip health monitoring using machine learning," in *International Symposium on VLSI (ISVLSI)*, 2014.

[5] F. Firouzi, F. Ye, K. Chakrabarty, and M. Tahoori, "Representative critical-path selection for aging-induced delay monitoring," in *International Test Conference (ITC)*, 2013, pp. 1–10.

[6] F. Firouzi, S. Kiamehr, M. Tahoori, and S. Nassif, "Incorporating the impacts of workload-dependent runtime variations into timing analysis," in *Design, Automation, and Test in Europe (DATE)*, 2013, pp. 1022–1025.

[7] F. Firouzi, S. Kiamehr, and M. Tahoori, "Statistical analysis of BTI in the presence of process-induced voltage and temperature variations," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2013, pp. 594–600.

[8] F. Firouzi, F. Ye, K. Chakrabarty, and M. Tahoori, "Power-aware minimum NBTI vector selection using a linear programming approach," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 32, no. 1, pp. 100–110, 2013.

[9] F. Firouzi, F. Ye, K. Chakrabarty, and M. Tahoori, "NBTI mitigation by optimized NOP assignment and insertion," in *Design, Automation, and Test in Europe (DATE)*, 2012, pp. 218–223.

[10] F. Firouzi, S. Kiamehr, and M. B. Tahoori, "Modeling and estimation of power supply noise using linear programming," in *International Conference on Computer-Aided Design (ICCAD)*, 2011, pp. 537–542.

[11] F. Firouzi, F. Ye, K. Chakrabarty, and M. Tahoori, "A linear programming approach for minimum NBTI vector selection," in *Great Lakes Symposium on VLSI (GLSVLSI)*, 2011, pp. 253–258.

PUBLICATIONS

[1] F. Firouzi, F. Ye, A. Koneru, A. Vijayan, K. Chakrabarty, and M. Tahoori, "Re-using BIST for circuit aging monitoring," in *European Test Symposium (Under Review)*, 2015.

[2] F. Firouzi, F. Ye, K. Chakrabarty, and M. Tahoori, "Aging- and variation-aware delay monitoring using representative critical path selection," in *ACM Transactions on Design Automation of Electronic Systems (Under Review)*, 2015.

[3] F. Ye, F. Firouzi, Y. Yang, K. Chakrabarty, and M. Tahoori, "On-chip voltage-droop prediction based on support-vector machines and feature selection," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (Under Review)*, 2015.

[4] F. Firouzi, F. Ye, S. Kiamehr, K. Chakrabarty, and M. Tahoori, "Adaptive mitigation of parameter variations," in *Asian Test Symposium (ATS)*, 2014.

[5] F. Firouzi, F. Ye, K. Chakrabarty, and M. Tahoori, "Chip health monitoring using machine learning," in *International Symposium on VLSI (ISVLSI)*, 2014.

[6] S. Wang, F. Firouzi, F. Oboril, and M. B. Tahoori, "Stress-aware P/G TSV planning in 3D-ICs," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2014.

[7] F. Ye, F. Firouzi, Y. Yang, K. Chakrabarty, and M. Tahoori, "On-chip voltage-droop prediction using support-vector machines," in *VLSI Test Symposium (VTS)*, 2014, pp. 1–6.

[8] S. Kiamehr, F. Firouzi, M. Ebrahimi, and M. B. Tahoori, "Aging-aware standard cell library design," in *Design, Automation, and Test in Europe (DATE)*, 2014, pp. 261:1–261:4.

[9] S. Wang, F. Firouzi, F. Oboril, and M. B. Tahoori, "P/G TSV planning for IR-drop reduction in 3D-ICs," in *Design, Automation, and Test in Europe (DATE)*, 2014, pp. 44:1–44:6.

[10] F. Oboril, F. Firouzi, S. Kiamehr, and M. B. Tahoori, "Negative bias temperature instability-aware instruction scheduling: A cross-layer approach," *Journal of Low Power Electronics*, vol. 9, no. 4, pp. 389–402, 2013.

[11] F. Firouzi, F. Ye, K. Chakrabarty, and M. Tahoori, "Representative critical-path selection for aging-induced delay monitoring," in *International Test Conference (ITC)*, 2013, pp. 1–10.

[12] S. Kiamehr, F. Firouzi, and M. Tahoori, "A layout-aware x-filling approach for dynamic power supply noise reduction in at-speed scan testing," in *European Test Symposium (ETS)*, 2013, pp. 1–6.

[13] S. Kiamehr, M. Ebrahimi, F. Firouzi, and M. Tahoori, "Chip-level modeling and analysis of electrical masking of soft errors," in *VLSI Test Symposium (VTS)*, 2013, pp. 1–6.

[14] F. Firouzi, S. Kiamehr, M. Tahoori, and S. Nassif, "Incorporating the impacts of workload-dependent runtime variations into timing analysis," in *Design, Automation, and Test in Europe (DATE)*, 2013, pp. 1022–1025.

[15] Y. Hara-Azumi, F. Firouzi, S. Kiamehr, and M. Tahoori, "Instruction-set extension under process variation and aging effects," in *Design, Automation, and Test in Europe (DATE)*, 2013, pp. 182–187.

[16] S. Kiamehr, F. Firouzi, and M. Tahoori, "Aging-aware timing analysis considering combined effects of NBTI and PBTI," in *International Symposium on Quality Electronic Design (ISQED)*, 2013, pp. 53–59.

[17] F. Firouzi, S. Kiamehr, and M. Tahoori, "Statistical analysis of BTI in the presence of process-induced voltage and temperature variations," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2013, pp. 594–600.

[18] F. Firouzi, S. Kiamehr, and M. Tahoori, "Power-aware minimum NBTI vector selection using a linear programming approach," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 32, no. 1, pp. 100–110, 2013.

[19] F. Oboril, F. Firouzi, S. Kiamehr, and M. Tahoori, "Reducing NBTI-induced processor wearout by exploiting the timing slack of instructions," in *International Conference on Hardware/Software Codesign and System Synthesis (CODES)*, 2012, pp. 443–452.

[20] F. Firouzi, A. Azarpeyvand, M. E. Salehi, and S. M. Fakhraie, "Adaptive fault-tolerant DVFS with dynamic online avf prediction," *Microelectronics Reliability*, vol. 52, no. 6, pp. 1197–1208, 2012.

[21] S. Kiamehr, F. Firouzi, and M. B. Tahoori, "Input and transistor reordering for NBTI and HCI Reduction in Complex CMOS Gates," in *Great Lakes Symposium on VLSI (GLSVLSI)*, 2012, pp. 201–206.

[22] F. Firouzi, S. Kiamehr, and M. Tahoori, "NBTI mitigation by optimized NOP assignment and insertion," in *Design, Automation, and Test in Europe (DATE)*, 2012, pp. 218–223.

[23] F. Firouzi, S. Kiamehr, and M. B. Tahoori, "Modeling and estimation of power supply noise using linear programming," in *International Conference on Computer-Aided Design (ICCAD)*, 2011, pp. 537–542.

[24] F. Firouzi, A. Yazdanbakhsh, H. Dorosti, and S. Fakhraie, "Dynamic soft error hardening via joint body biasing and dynamic voltage scaling," in *Euromicro Conference on Digital System Design (DSD)*, 2011, pp. 385–392.

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

[26] F. Firouzi, M. E. Salehi, F. Wang, and S. M. Fakhraie, "An accurate model for soft error rate estimation considering dynamic voltage and frequency scaling effects," *Microelectronics Reliability*, vol. 51, no. 2, pp. 460–467, 2011.

[27] S. Kiamehr, F. Firouzi, and M. B. Tahoori, "Stacking-based input reordering for NBTI aging reduction," *ITG-Fachbericht-Zuverlässigkeit und Entwurf*, 2011.

[28] L. Chen, F. Firouzi, S. Kiamehr, and M. B. Tahoori, "Fast and accurate soft error rate estimation at rtl level," *ITG-Fachbericht-Zuverlässigkeit und Entwurf*, 2011.

[29] F. Firouzi, M. E. Salehi, F. Wang, S. M. Fakhraie, and S. Safari, "Reliability-aware dynamic voltage and frequency scaling," in *International Symposium on VLSI (ISVLSI)*, 2010, pp. 304–309.

[30] F. Firouzi, M. E. Salehi, F. Wang, S. M. Fakhraie, and S. Safari, "Reliability considerations in dynamic voltage and frequency scheduling schemes," in *International Conference on Design and Technology of Integrated Systems in nanoscale Era*, 2010.

[31] F. Firouzi, M. E. Salehi, A. Azarpeyvand, S. M. Fakhraie, and S. Safari, "Analysis of single-event effects in embedded processors for non-uniform fault tolerant design," in *International Conference on Innovations in Information Technology*, 2009, pp. 141–145.

[32] F. Firouzi, S. Kiamehr, P. Monshizadeh, M. Saremi, A. Afzali-Kusha, and S. Fakhraie, "A model for transient fault propagation considering glitch amplitude and rise-fall time mismatch," in *Asia Symposium on Quality Electronic Design (ASQED)*, 2010, pp. 89–92.

[33] A. Azarpeyvand, M. Salehi, F. Firouzi, A. Yazdanbakhsh, and S. M. Fakhraie, "Instruction reliability analysis for embedded processors," in *International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, 2010, pp. 20–23.

[34] M. Salehi, A. Azarpeyvand, F. Firouzi, and A. Yazdanbakhsh, "Reliability analysis of embedded applications in non-uniform fault tolerant processors," in *International Conference on Future Information Technology (FutureTech)*, 2010, pp. 1–5.

ABSTRACT

Computing systems, ranging from high performance computers such as servers to embedded systems such as smart phones and handheld medical devices, have been working their way into many aspects of daily life. As a result, semiconductor industry has encountered persistent pressure to improve chip performance and functionality while decreasing cost and time to market. Down-scaling of the transistor feature size, increasing operating frequencies, and increasing transistor count per chip are the most important drivers to allow for high performance. However, these drivers also pose a major challenge due to parameter variations at manufacturing and runtime, induced by imperfections in the manufacturing process and variations in workloads and environment. Parameter variations are considered as the dominant source of lifetime and performance limiters of circuits, increasing the chip failure rate. Nevertheless, the circuits have to be designed to maintain a specific level of performance and power consumption over a specified lifetime in the presence of parameter variations.

The main objective of this thesis is to tackle the impact of parameter variations in order to improve the chip performance and extend its lifetime. To achieve this goal, it is required to: 1) understand and analyze parameter variations and consider their interdependency, 2) develop fast and accurate reliability evaluation platforms to incorporate the combined effects of variations and transistor aging into *Very Large Scale Integration* (VLSI) design process, 3) develop techniques to track and monitor lifetime-delay-power changes of the chip during in-field operation, and 4) develop design-time and runtime adaptive techniques for alleviating the effects of variations to guarantee the resilience of the chip throughout its lifetime

This thesis presents a set of unified analysis and design techniques for resilient systems in the presence of combined effects of parameter variations. First, a holistic aging- and variation-aware timing analysis framework is de-

veloped and integrated into commercial *Electronic Design Automation* (EDA) tools to evaluate the impacts of aging, voltage, temperature, and process variations on circuit performance-power-lifetime. Using this framework, a novel delay monitoring technique is proposed which enables us to dynamically track the delay and the lifetime of the circuit in-field under the influence of parameter variations. Finally, based on the proposed timing analysis framework, and chip monitoring system, a set of static and adaptive mitigation techniques are designed to tackle the detrimental impacts of parameter variations on performance, power, and lifetime of the circuit.

ZUSAMMENFASSUNG

Computersysteme haben Einzug in viele Bereiche unseres täglichen Lebens gehalten. Diese Systeme reichen von leistungsfähigen Computern, wie Servern, bis hin zu eingebetteten Systemen, wie Smartphones und portablen medizintechnischen Geräten. Als Folge ist die Halbleiterindustrie einem stetigen Druck ausgesetzt, um die Chipleistung und Funktionalität zu verbessern und gleichzeitig die Kosten und die Entwicklungszeit zu senken. Um hochperformante Systeme weiter zu entwickeln, bedient man sich verschiedener Mechanismen. Dazu zählt die Verkleinerung der Transistorgröße, die Steigerung der Frequenz und eine Erhöhung der Anzahl an Transistoren auf einem Chip. Diese Einflussfaktoren stellen allerdings eine große Herausforderung im Bezug auf Prozessvariation bei der Herstellung und im Betrieb dar. Diese entstehen durch Schwankungen im Fertigungsprozess und durch unterschiedliche Arbeitslast und Umgebungen. Die Parametervariationen erhöhen die Chip Ausfallrate und werden als Hauptursache für die Lebenszeit- und Performanceeinschränkungen gesehen. Dennoch müssen die Schaltkreise dafür entworfen sein, ein gewisses Maß an Performance und Leistungs-stabilität über eine definierte Laufzeit in Anwesenheit von Parametervariationen gewährleisten zu können.

Das Hauptziel dieser Arbeit ist es den Einfluss der Parametervariationen zu bewältigen, um die Performance und Lebenszeit von Chips zu verlängern. Dazu müssen folgende Punkte erfüllt sein: 1) die Parametervariationen und deren gegenseitige Abhängigkeit müssen verstanden und analysiert werden, 2) schnelle und genaue Verlässlichkeitsevaluierungsplattformen zu entwickelen, um die Effekte der Variation und des Transistor Alterungeffektes mit dem *Very Large Scale Integration* (VLSI) Design Prozess verbinden zu können, 3) Techniken müssen entwickelt werden, um die Lifetime-Delay-Power-Veränderungen des Chips während des Betriebs zu messen und zu analysieren, und 4) adaptive Entwurfszeit- und Laufzeittechniken müssen entwickelt werden,

um Effekte von Variationen zu lindern, damit die Widerstandsfähigkeit eines Chips über seine Lebenszeit garantiert werden kann.

Diese Arbeit stellt eine vereinheitlichete Menge von Analysen und Entwurfstechniken für widerstandsfähige Systeme vor. Zuerst wird ein ganzheitliches Alterungs und variationsgewahres Zeitanalyseframework entworfen und in ein kommerzielles *Electronic Design Automation* (EDA) Werkzeug integriert. Damit ist es möglich die Einflüsse von Alterung, Spannung, Temperatur und Prozessvariationen auf die Performance/Leistungs-verbrauch/Lebenszeit zu evaluieren. Mit diesem Framework wird eine neuartige Delay-Monitor-Technik präsentiert, die es ermöglicht, die Verzögerung und die Lebenszeit der Schaltkreise im Betrieb unter dem Einfluss von Prozessvariationen, zu beobachten. Zuletzt werden mit Hilfe des beschriebenen Frameworks und des Delay-Monitor-Systemes eine Reihe von statischen und adaptiven Migrationstechniken entworfen, um die schädlichen Einflüsse der Parametervariation auf Performance, Energie und die Lebenszeit des Schaltkreises zu beheben.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Integrated Circuits (ICs) have become the principal elements of many aspects of our lives. They strongly impact the way we communicate, work, study, travel, sport, etc. [8, 9]. Over the years, in response to economics and market demands, the scaling of transistor feature size to atomic range has enabled the performance, and density of ICs to dramatically increase. In fact, semiconductor industry has successfully followed the Moors law [10]. According to this law, observed by Intel co-founder Gordon Moore in 1965, transistor count per chip is doubling every 15-18 months [11] (See Fig. 1.1). This trend allows us to implement more functionality per chip at a reduced device cost. Despite this promising degree of integration, technology scaling also poses many reliability issues. Reliability is defined as the ability of a digital chip to be operational with a given specifications for a specified period of time in the presence of stated conditions. Failing to effectively address this emerging criterion in critical applications such as medicine, automotive, space applications, etc can lead to unintended catastrophes. Unreliability can also cause other severe consequences such as product delays, and yield loss that ultimately lead to decrease in revenues and profits [12].

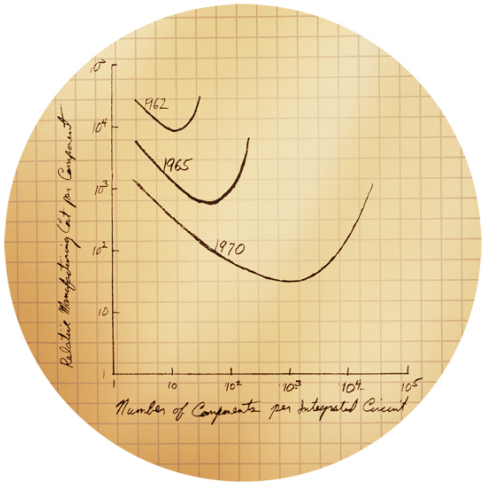


Figure 1.1: Original sketch of Moor's law. Decades later it remains true.

Reliability issues arise from different sources, however, *parameter variation* is recognized today as the major undesirable consequence of technology scaling and the dominant source of lifetime and frequency limiters [2, 13, 14, 15]. Due to parameter variations, physical and electrical characteristics of a fabricated chip vary from the design specifications over time. Parameter variation is an expensive issue, since even a small degree of parameter variations may

tions may translated to significant deviation in circuit frequency and lifetime. Parameter variations can be classified into two categories:

1. *Static variations* are mainly due to process variations caused by inaccuracy in the chip manufacturing process. These variations are fixed and do not change over chip lifetime.

2. *Runtime variations* denote the uncertainties in operating and environmental conditions (e.g., voltage and temperature fluctuations as well as transistor aging) over time when the circuit is operating in-field.

Due to increased level of imperfections in manufacturing process of ICs, process parameters of fabricated chips such as gate length and threshold voltage usually vary from the expected design value resulting in considerable timing mismatch between design time and runtime [16, 17, 18, 13, 19, 20, 21, 22].

In biology, aging is defined as "the collection of changes that render human beings progressively more likely to die" [23]. The same phenomenon happens for ICs. The main source of transistor aging is threshold voltage increase of transistors over time [2]. Consequently, the circuit delay is gradually degraded, and eventually, the circuit may exhibit timing violations if the delay in the critical paths exceeds the timing constraints for which it was designed [24, 25, 26, 27]. Ultimately the circuit fails (dies) due to timing violations.

In the presence of voltage variations, the actual supply voltage level seen by individual devices decreases [28]. Voltage variations are caused by the instantaneous switching current drawn from the power delivery network, and fluctuations in chip activity. This phenomenon causes the gate delay to increase and eventually reduces the system performance. In addition, voltage variations can also lead to intermittent logic and timing failures. Although technology scaling decrease the power dissipation of transistors, due to placement of more devices in a single chip the overall power consumption and power density is dramatically increased [2, 29]. The increase in power consumption and power density result in temperature variations. Temperature variation has detrimental side-effects on both performance and reliability of the chip [30, 31]. In addition, resistance of the power delivery network is impacted by temperature resulting in larger voltage variations.

Fig. 1.2 shows the failure rate of an IC over time in the presence of parameter variations. This curve, a.k.a. bathtub curve, consists of three parts [12, 15]:

1. *Infant mortality*: Due to manufacturing defects, the failure rate of this part is very high. However, the failure rate rapidly decreases as the faulty ICs are identified and discarded using burn-in process.

2. *Mid-life*: The next part represents the useful lifetime of the IC. During this part, the chip is susceptible to soft errors as well as timing violations imposed by runtime variations. The failure rate of this part is a function of operating and environmental conditions.

3. *Late life*: In the late life of the chip, again the chip experiences increasing failure rate due to aging-induced failures.

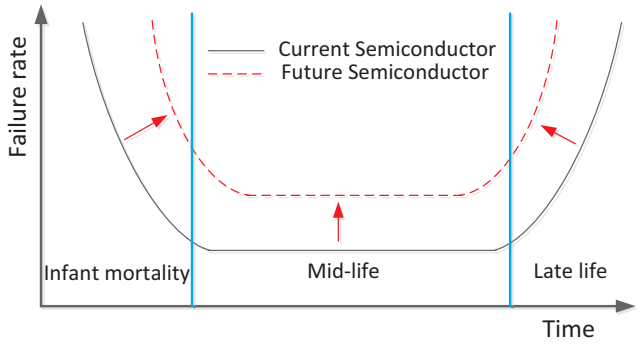


Figure 1.2: Failure rate of a chip over time [12, 15].

In Fig. 1.2, the dashed line demonstrates the impact of technology scaling on the failure rate. This figure shows that variations become more severe for future technology, since scaling leads to 1) higher degree of inaccuracy in chip fabrication process [3], 2) higher electrical field [2], and 3) higher power density [3]. Since transistor feature size is scaled more aggressively than operating voltage, as shown in Fig. 1.3, the electric field across the gate oxide is increased. As a result of the elevated temperature and/or higher electric field, the aging is expected to happen earlier and hence, lifetime is decreased.

Runtime temperature variation is expected to increase because of higher power consumption in smaller technology nodes (See Fig. 1.4). Higher clock frequencies and escalating transistor densities cause more devices to switch

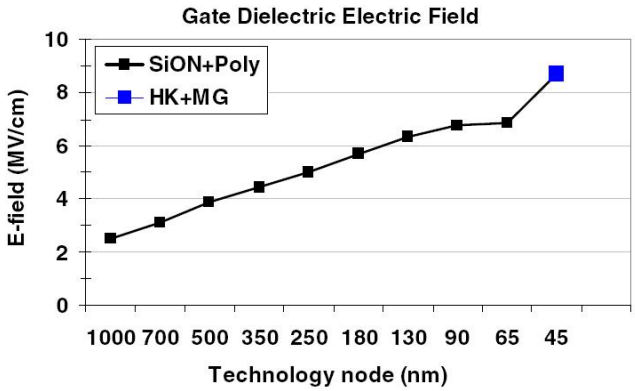


Figure 1.3: Electric field across gate oxide for different technologies [1].

simultaneously in atomic dimensions. This leads to elevated level and change rate of instantaneous current change and as a result the undesirable voltage variation is exacerbated.

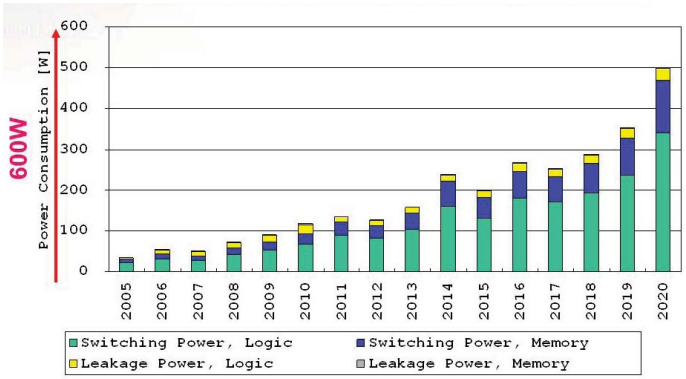


Figure 1.4: Power consumption trend [2].

1.1 Research Contributions

To enable semiconductor industry to continue to scale down the device dimensions, it is important to prevent the emerging problems of parameter variations by incorporating them into digital design methodologies and *Computer-Aided Design* (CAD) tools. To achieve this goal, we need to: 1) accurately model the combined effects of parameter variations and their detrimental effects on delay and lifetime, 2) precisely track the variations-induced delay increase of chips in-field, and 3) add appropriate countermeasures to the chip in order to compensate, and tackle the parameter variations in order to meet the specified design specifications. This thesis presents various novel

techniques to improve and extend the state-of-the-art methods in the way described below:

1.1.1 Modeling

Parameter variations are interdependent, although occurring at different time scale, which makes their timing impacts extremely complex to analyze. Figure 1.5 shows the interdependence of parameter variations. These strong correlations and the increasing sensitivities of nano-scaled transistors to variations make the state-of-the-art methodologies of analyzing the circuit delay to be significantly inefficient. To avoid under-design and/or over-deign of digital chips, an accurate variations-aware timing analysis technique is highly required. Facing this emerging problem motivated us to focus on addressing the challenging task of variations-aware timing analysis.

This thesis proposes a holistic and an accurate timing analysis framework which significantly improves the limitations of state-of-the-art methodologies by incorporating the combined effects of workload-dependent aging, process, and runtime variations, occurring at different time scales, into timing analysis flow. We discovered that neglecting the interaction among parameter variations in existing techniques results in considerable error in design margin and performance loss. The proposed framework is built on top of commercial *Electronic Design Automation* (EDA) tool chains and therefore it scales very well. Thanks to our novel approach, not only the pessimistic timing margin of circuits can be significantly reduced, but also other parts of digital design flow such as design-time and runtime mitigation techniques derive benefit from this technique.

1.1.2 Monitoring

The common practice to tackle the adverse impact of parameter variation on circuit delay is utilizing a safety timing margin , a.k.a. guardbands. Runtime variations tend to vary from workload to another, and between different time intervals during the execution. Since it is almost infeasible to accurately predict the operating conditions, this design-time approach may impose considerable performance loss. For example, as highlighted by IBM, a 20%

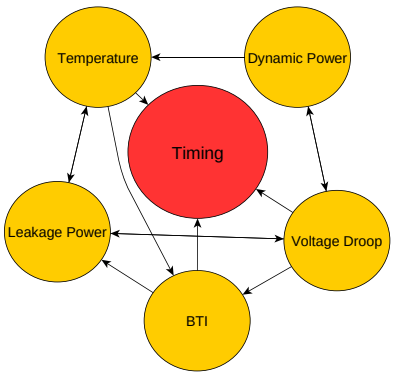


Figure 1.5: Interdependence of different sources of variation and their impact on circuit delay and lifetime.

timing margin is considered for voltage variations in POWER7 processors [32]. This loss tends to increase with technology scaling, causing an even larger performance gap between the nominal and the worst-case condition [14]. Runtime mitigation have been advocated as a promising alternatives that enables the dynamic adjustment of reliability knobs based on the actual variations seen during runtime. Evolving runtime techniques demands effective, accurate in-field variations-aware delay monitoring systems.

The focus of the most state-of-the-art techniques for delay/age monitoring is on sensor design. However, to apply these sensors, we need to tolerate a significant overhead unless sensors are carefully placed for very selective locations. Another challenge is to infer the information regarding the delay/age of every critical paths of the chip with limited information obtained by the monitoring sensors. In this thesis, we propose a new aging- and variations-aware methodology that utilizes different machine-learning techniques to monitor the delay of a small set of critical paths by leveraging already available sensors and dynamically assess the impact of variations for every reliability-critical path of the chip. In particular, this contribution provides answers to the questions: 1) how to identify the most-relevant silicon data and chip features, 2) how to select the most-appropriate machine-learning strategies for delay monitoring, and 3) how to realize the delay/age monitoring systems.

1.1.3 Mitigation

In general, to tackle parameter variations, two different categories of mitigation techniques, namely *design-time solutions* and *runtime solutions* are developed. Design-time techniques are based on aggregation of the strategies of model, predict, and margin, while run-time solutions are based on sense and adapt methodology. Note that these two categories can be regarded as being complementary to each other to tighten the overheads. This thesis presents how the significant overheads of exiting design-time guard-banding (i.e., adding timing margin) can be reduced with the help of the proposed variations-aware timing analysis flow. In addition, with the help of proposed delay/age monitoring system, a novel fine-grained reconfigurable active healing technique based on *Input Vector Control* (IVC) is presented to co-optimize lifetime and power consumption during inactive period considering the operating conditions of the circuit. Finally, a new *Dynamic Frequency Scaling* (DVF) that enables the dynamic adjustment of clock frequency based on the actual variations seen during runtime is proposed in order to significantly increase the system performance.

1.2 Organization of the Dissertation

The rest of this dissertation is organized as follows.

Chapter 2 discusses the physical mechanisms of parameter variations, modeling techniques, as well as the impact of parameter variations on reliability of the digital circuits. This chapter provides the fundamental background which is important for understanding the next chapters of this dissertation.

In Chapter 3, state-of-the-art timing analysis techniques are presented, advantage and disadvantage of each of them are discussed. This is followed by explaining the details our proposed variation-aware timing analysis. It also highlights the importance of considering the combined impacts of parameter variations during timing analysis.

Chapter 4 lists current status of the existing monitoring systems for tracking the status of circuits in-field. It then discusses the major limitations of state-of-the-art techniques and how they can be improved by our novel techniques. Our proposed variation-aware monitoring approach is presented in details and its accuracy is compared against previous work.

Chapter 5. describes design-time and runtime techniques previously proposed to tackle the adverse impacts of parameter variations. It also demonstrates how the proposed variations-aware timing analysis flow and the proposed monitoring system can be used to improve existing mitigation techniques to further increase frequency and extend the lifetime of the chip considering the power limit.

Finally, the thesis is concluded in Chapter 6 and points out the possible extension of this work as well as the applications of proposed techniques.

CHAPTER 2

BACKGROUND AND MODELING

Very-large-scale integration (VLSI) chips manufactured at nano-scale technology nodes face various reliability challenges [33, 14, 2]. Process variations as well as runtime variation including transistor aging together with workload-dependent voltage and temperature variations are considered as the major sources of unpredictability in VLSI designs. Process variations result in considerable timing mismatch between design specifications and the specifications of the manufactured (post-silicon) chips. In addition, timing specifications of the manufactured chip may also vary over the time due to workload-dependent runtime variations [34, 35]. This chapter introduces the physical mechanisms of these undesirable parameter variations, as well as their impacts on circuit frequency, power, and lifetime. This is followed by presenting methodologies and frameworks to model parameter variations.

2.1 Process Variations

Due to imprecision in the fabrication process, when a chip is fabricated, the obtained value of numerous chip parameters such as oxide thickness, gate length, and impurity density from are deviated from the intended design specifications. The uncertainties in physical parameters in turn lead to variations in electrical characteristics of transistors and interconnects, ultimately resulting in circuit delay variations and timing failures. Among different physical parameters, effective gate length and threshold voltage are mostly suffer from process variations [12, 36]. The main reason of deviations in gate length is that optical lithography cannot be scaled with the same pace of transistor scaling, and hence the feature size of transistors are far smaller than the available wavelength of lithography. The main source of threshold voltage variations lies in *Random Dopant Fluctuation* (RDF) due to the random nature of ion implantation. Since in the smaller technology nodes the total number of dopants is very few, RDF tends to significantly alter the threshold voltage. As shown in Fig. 2.1, variations in physical parameters can be categorized as follow [3]:

- Systematic variations: these deterministic variations are geometry-dependent or layout-dependent and can be modeled by practical equations or look-up tables.

- Non-systematic or random variations: as the name suggests, these variations are really uncertain and independent of design implementation. Therefore, only statistical random variables can be used for modeling of these parameters. Random variations themselves either can be further categories to die-to-die or within-die variations. Due to die-to-die variations, all the transistors placed on the same die are impacted in the same way. On the other hand, due to within-die variations, each transistors in the same die varies in a different way. Finally, within-die variations can either be totally independent or be spatially correlated. Lithographic-based variations such as gate length mostly show strong spatial correlations, on the other hand, non-lithographic variations such as RDF are almost random with negligible spatial correlation.

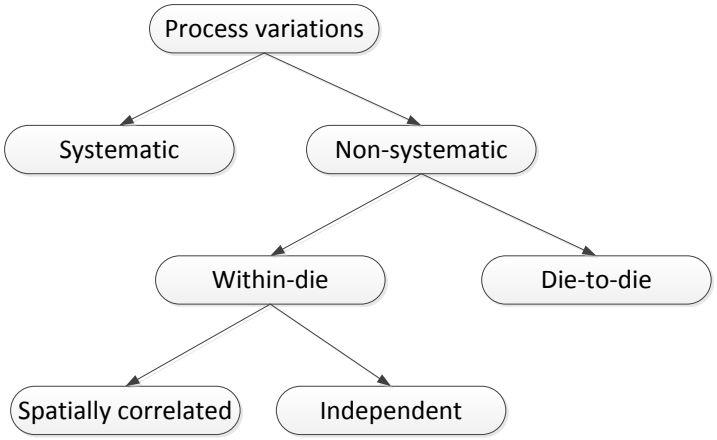


Figure 2.1: Categories of process variations [3].

According to [37], the variation of *Physical Parameter*s (e.g. effective gate length: ΔL) can be represented by the following equation:

$$\Delta PP_{total} = \Delta PP_{d2d} + \Delta PP_{wd,cor} + \Delta PP_{wd,rand}, \tag{2.1}$$

where ΔPP_{d2d} represents die-to-die variation. $\Delta PP_{wd,cor}$ represents the spatially correlated variation and $\Delta PP_{wd,rand}$ denotes the within-die independent random variation. In general, gates located in close proximity may exhibit similar parameter variations. The most common approach to exactly model the spatial correlation of within-die process variation, is based on a technique presented in [38]. As shown in Fig. 2.2, in this method, the die area is divided into several square tiles and correlation between two tiles is modeled

by a diminishing function of $exp(-\alpha \cdot d)$, where d is the distance of these two tiles and α is the diminishing factor.

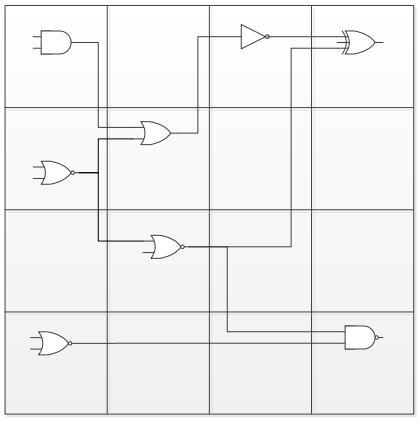


Figure 2.2: Spatial correlation modeling approach.

To keep track of the correlations among variations, *Principal Component Analysis* (PCA) can be used to map the correlated variables (e.g. ΔL) to a new set of parameters whose elements are mutually independent (orthogonal). As an example Equation 2.2 shows how ΔL_i can be represented by *Principal Components* (PC). Note, PCs have standard normal distribution and are same for all correlated variables:

$$\Delta L_i = \sum_{i}^{n} a_i \cdot PC_{\Delta L,i} + \mu_i, \tag{2.2}$$

where a_i is a coefficient which depends on the covariance matrix of the original correlated set of ΔL_i and μ_i is the mean value of ΔL_i.

2.2 Transistor Aging

There are different mechanisms such as *Bias Temperature Instability* (BTI), *Time-dependent Dielectric Breakdown* (TDDB), and *Hot Carrier Injection* (HCI) that cause transistor delay degrades over time resulting in timing violations and failure. Due to higher operating temperature and higher electric field in atomic-scaled semiconductor technology, the impacts of aging phenomena are escalated which leads to shorter circuit lifetime and higher timing failures. Therefore it is highly required to understand, model, track, and mitigate transistor aging to guarantee the specified reliability of the chip.

2.2.1 BTI

Negative BTI- (NBTI-) induced threshold voltage increase has grown in importance as technology scales down and among different aging mechanisms, it is considered as the dominant degradation phenomenon which should be appropriately addressed in nano-scaled technology [2]. The NBTI effect causes $|V_{th}|$ of PMOS transistor to increase. In general, physical mechanisms of NBTI are primarily associated with *reaction-diffusion* and *trapping-detrapping* phenomena.

According to reaction-diffusion (See Fig. 2.3), at the $Si - SiO_2$ interface of PMOS transistors, most of the Si atoms are connected to O atoms, while the rest of them are bonded to H atoms. Therefore, under high electric field ($Vgs = -Vdd$), some of the SiH bonds of PMOS transistor may be broken. The generated H and H_2 can diffuse towards the gate and the remaining dangling bonds (i.e., traps) increase the threshold voltage of the transistor. On the other hand, when stress is removed ($Vgs = 0$), the migrated H atoms return to the interface and compensate donging bonds resulting in threshold voltage decrease [24, 25, 39, 40, 33].

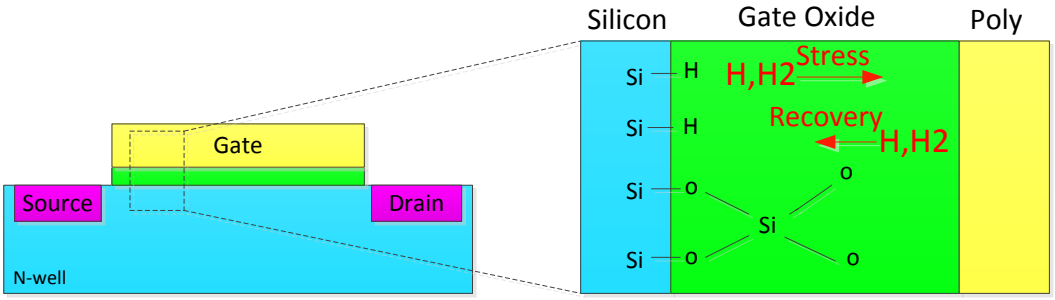


Figure 2.3: Reaction-diffusion BTI model.

In trapping-detrapping model (See Fig. 2.4), it is believed that some pre-existing traps are located in the dielectric of the PMOS transistors. Due to electric field, these traps can be filled with the holes migrating from the channel area contributing to threshold voltage increase. When electric field is removed, some of the filled traps are emptied and hence, threshold voltage decreases [41, 42, 43].

In both reaction-diffusion model and trapping-detrapping model, the first phase is refereed as *stress* phase, and the second phase is refereed as *recovery* phase. As shown in Fig. 2.5, in the stress phase, the threshold voltage of

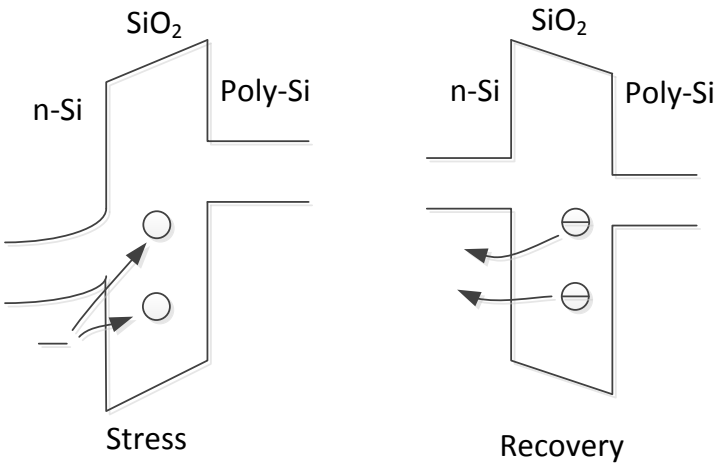


Figure 2.4: Trapping-detrapping BTI model.

PMOS transistor increases over the time, whereas in the recovery phase the threshold voltage decreases towards its initial value. It should be noted that the recovery phase cannot completely alleviate the effect of the stress and hence, the overall effect of NBTI is a positive shift in the threshold voltage of the PMOS transistors.

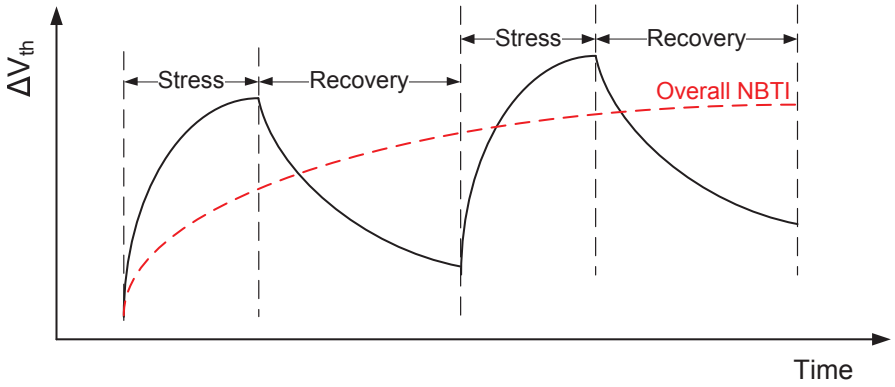
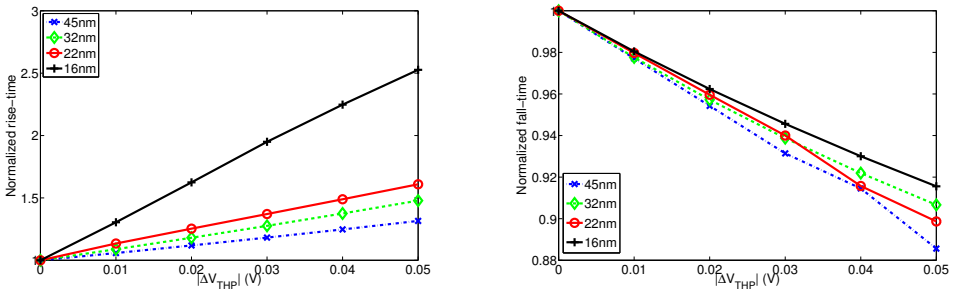


Figure 2.5: Stress and recovery mode.

As a result of NBTI, the rise time of a CMOS logic gate increases. Figure 2.6(a) shows the rise-time delay of a simple inverter versus $|\Delta V_{th}|$ of its PMOS transistor for 16 nm to 45 nm PTM technologies. As shown in this figure the rise-time delay sensitivity to $|\Delta V_{th}|$ of the PMOS transistor increases with technology scaling. However, the NBTI-induced $|\Delta V_{th}|$ of the PMOS transistor, leads to a decrease in fall-time delay of the CMOS logic gate as well. This is due to the fact that NBTI makes the PMOS transistor, in the pull-up network, weaker and as a result during the output fall time, the

PMOS transistor switches faster which eventually results in a lower fall-time delay. This effect is depicted in Figure 2.6(b).



(a) rise-time delay (b) fall-time delay

Figure 2.6: NBTI-induced delay degradation of a simple inverter for different $|\Delta V_{th}|$ of its PMOS transistor.

In the previous technology nodes, the *Positive BTI* (PBTI) effect on NMOS transistors was negligible in comparison to the effect of NBTI on PMOS transistors. However, by introduction of high-κ metal-gate technologies, PBTI has emerged as a major reliability concern for NMOS in and the effect becomes more significant with technology and voltage scaling [4, 44, 45] (see Fig. 2.7).

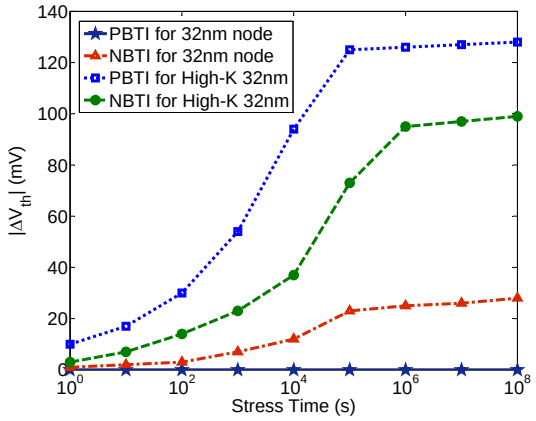


Figure 2.7: Vth shift induced by NBTI and PBTI [4].

BTI-induced timing degradation strongly depends on operating context parameters including supply voltage, temperature, and input patterns [46, 33] which are non-uniform and significantly vary from gate to gate and time to

time [40]. The overall BTI effect can be modeled by the following equation [39, 46]:

$$\Delta Vth_{NBTI}(t) = \left(\frac{\sqrt{{K_v}^2 \alpha T_{clk}}}{1 - \beta(t)^{1/2n}} \right)^{2n}, \tag{2.3}$$

where α is the duty cycle (i.e., the ratio between stress time to the total time), T_{clk} is the clock cycle. n is a fabrication process constant ($n = 1/4$ for hydrogen atoms and $n = 1/6$ for hydrogen molecules). The other parameters are described in [33].

BTI is impacted by an intrinsic variation factor [47, 48]. This fluctuation is rooted in intrinsic charge fluctuation of generated BTI-induced interface traps, similar to random dopant fluctuation. The effect of the intrinsic fluctuation on BTI-induced V_{th} shift can be modeled by [49]:

$$\sigma(\Delta V_{th-BTI}(t)) = \sqrt{\frac{K}{L.W} \mu(\Delta V_{th-BTI}(t))}. \tag{2.4}$$

2.2.2 HCI

HCI affects mainly NMOS transistors when the gate of NMOS is making a transition. Carriers in the channel are subjected to different electric fields when traveling between the source to the drain. If these hot carriers collide with the gate oxide interface, some electron-hole pairs are generated. Some of these generated electrons are energetic enough to accelerate and get trapped in the gate oxide. These interface traps are generated at the $Si - SiO_2$ interface near the drain causing the threshold voltage to increase [50] (Fig. 2.8). Since hot electrons are generated when the gate of the NMOS switches, the threshold voltage change due to HCI has a direct dependency with the operational frequency. The threshold voltage change can be estimated by Equation (2.6) [51].

$$\Delta V_{th} = A_{HCI} \times \alpha \times f \times e^{\frac{V_{DD} - V_{th}}{t_{ox} E_1}} \times t^{0.5}, \tag{2.5}$$

where A_{HCI} is a technology dependent constant, α is the activity factor, and f is the clock frequency. V_{th} and V_{DD} are the threshold voltage and supply voltage, respectively. t_{ox} is the oxide thickness, E_1 is a constant equal to $0.8V/nm$ [52] and t is the total time [53, 54].

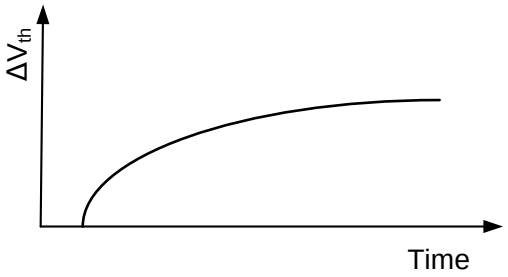


Figure 2.8: V_{th} change due to HCI.

2.2.3 TDDB

TDDB is an aging mechanism which is the result of formation of a conducting path between gate and substrate due to electric field (See Fig. 2.9). TDDB leads to an increasing leakage current and in turn resulting in reduced switching frequency and failure. The increase in gate oxide current can be expressed by the following equation:

$$\Delta I_{gate} = K(V_{gd})^p e^{t/\beta}, \tag{2.6}$$

where k, p, β are technology dependent fitting parameters. V_{gd} is the voltage between gate and drain and t is time.

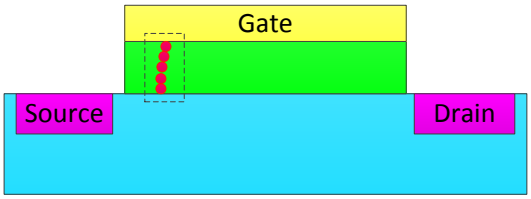


Figure 2.9: Physical mechanism of TDDB.

2.3 Power Model

Although with technology scaling the power consumption of each transistor is reduced, due to incredibly higher transistor count per area, the overall power density and as a result temperature and voltage variations are dramatically increased [2]. Therefore, chip designers should carefully address this emerging issue to avoid the ultimate undesirable consequences. Power dissipation in CMOS circuits is composed of two main components: dynamic and leakage power dissipation. Dynamic power occurs during switching of logics and

mainly is due to switching power and short circuit power. Switching power is the power required to charge/discharge the load capacitance. This power can be reduced by reducing the operating voltage, frequency, switching activity, and load capacitance. Short circuit power is due to direct current between supply rails (i.e., V_{DD} and GND) while PMOS and NMOS are both ON at the same time for a short period of time during switching because of finite rise and fall times. Static power is the power dissipated due to leakage currents drawn continuously from the power supply.

In general, the total power is estimated based on the workload-dependent operating conditions of the chip as follow:

- Powered off (Power Gating): The total power is zero.
- Powered on, clocks off (Clock Gating): The dynamic power is equal to zero and therefore the total power is equal to the leakage power.
- Powered on, clocks on, no input change: The chip has only dynamic power in the clock network and the total power is equal to the sum of the leakage power and the dynamic power of the clock network.
- Powered on, clocks on, with input change: The total power is the sum of the dynamic and the leakage powers.

2.3.1 Dynamic Power

Here, we explain the model for dynamic power of a gate with a simple buffer gate shown in Fig. 2.10. For simplicity only the capacitance between gate and source is depicted in this figure.

If the input of the first inverter goes from low to high, there is a fall transition in node B and a rise transition in the output of the second inverter (node C). When the voltage of node A increases, we have the following effects: 1) discharge of the capacitors C_{gs11} and C_{gs22} 2) charge of the capacitors C_{gs12}, C_{gs21}, and C_L 3) activation of a temporary path between supply and ground. The total current drawn from the supply voltage node contains three different components [55]:

- A gate capacitor differential current I_d, which charges the gate capacitor. The maximum peak value of this current coincide with the input transitions(I_{11} and I_{21}).

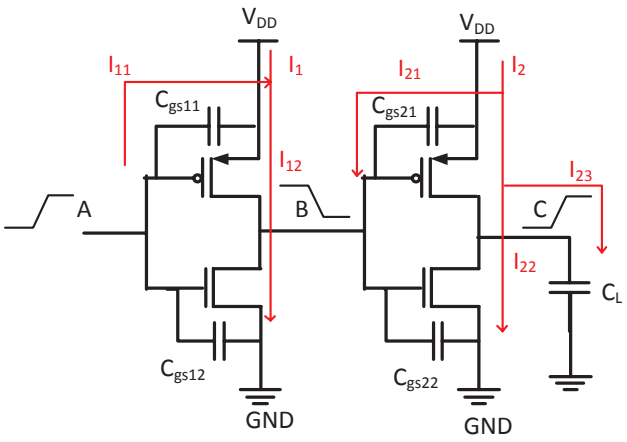


Figure 2.10: A two stage CMOS circuit with two inverter gates.

- A short circuit current I_s which occurs only when the output makes a transition (here I_{12} and I_{22}). This current is due to the fact that while the gate switches between ON and OFF states, for a short time period both pull-up and pull-down networks are ON. This current strongly depends on the slew rate (defined as the rate of change of voltage per time unit) of the input signal and increases with rise and fall time, since for a longer period of time there is a path directly from Vdd to ground.

- A charging current I_c, that increases the charge of the internal and load capacitors and only exists when the output goes from low to high (I_{23}).

In the above example, the total current drawn from supply is I_1 and I_2(see Fig. 2.10). The current I_1 shown in Fig. 2.11 consists of two components: 1) I_{11} is the gate capacitor differential current which is negative and occurs when node A makes a rising transition, 2) I_{12} is the short circuit current which is positive. Current I_2 illustrated in Fig. 2.12 consists of three components: 1) I_{21} is the gate capacitor differential current which is positive, 2) I_{22} is the positive short circuit current, and 3) the load charging current I_{23}.

2.3.2 Static Power

As shown in Fig. 2.13, different components contribute to static power [56]:

- I_{SUB}: sub-threshold leakage is due to the current between source and drain of a OFF transistors. This component of static power exponen-

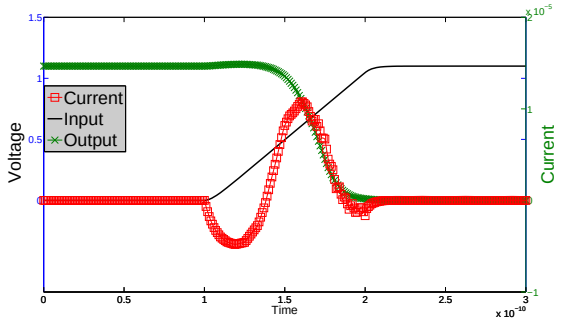


Figure 2.11: Switching current of an inverter during output fall time.

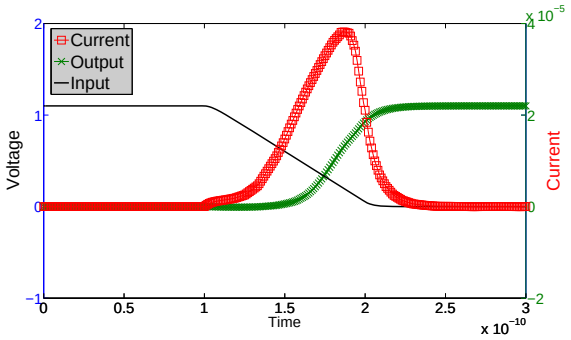


Figure 2.12: Switching current of an inverter during output rise time.

tially depends on temperature and actual power supply voltage. Moreover, the sub-threshold leakage current increases exponentially with the threshold voltage which varies over the time due to BTI effect. The following equation summarizes the dependence of the leakage power to temperature (T), supply voltage (V_{dd}^{cell}), and threshold voltage shift due to BTI (Vth) [57]:

$$P_{leakage} \propto exp^{(\alpha \cdot T + \beta \cdot V_{dd}^{cell} + \gamma \cdot Vth(T, V_{dd}^{cell}))}. \tag{2.7}$$

- I_{BTBT}: leakage through P-N junction between drain (source) and body.
- I_{GIDL}: gate-induced barrier lowering current through drain-body that depends on gate voltage.
- I_{GATE}: current that leaks via thin oxide layer between gate and body.
- I_{DG}: drain to gate oxide tunneling current.

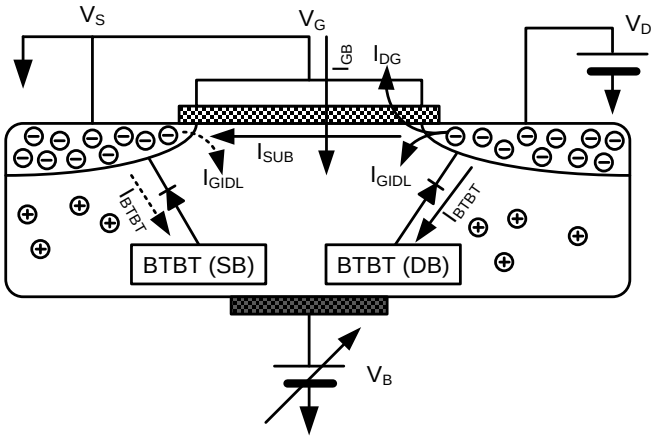


Figure 2.13: Leakage current components.

2.4 Voltage-droop

Voltage-droop is emerging as a challenging issue in nanometer digital designs. Voltage droop consists of two components: $IR - drop$ and inductive ΔI noise [58]. IR-drop is rooted in instantaneous current through the resistance of the power mesh network, power pads, and device package. Inductive ΔI noise is induced by rapid current change drawn from the inductance of the power mesh network, power pads, and device package which is proportional to Ldi/dt. Excessive voltage variations in the *Power Delivery Network* (PDN) decreases the switching speeds of transistors which may lead to timing failures. With technology scaling, due to higher power consumption of the chip, this phenomenon is even getting worse which highlights the importance of finding efficient modeling techniques as well as counter-measures to appropriately combat the detrimental impacts of voltage-droop on circuit reliability.

Fig. 2.14 illustrates schematic of a PDN [59]. According to this figure, total die area of a chip is categorized into two groups: core area and pad-frame. In the core area, the logic of the chip is placed, while the pad-frame is dedicated to all pads including I/O, and power pads. The current of power pads is supplied by package either using *Controlled Collapse Chip Connection* (C4) pads or using wire-bond pads. Two rings surround the core area. One of these rings delivers Vdd and the other one is connected to ground. Moreover, several horizontal and vertical power stripes create sort of a mesh network to evenly deliver the power from surrounding rings to the standard cells that

are placed in the core area.

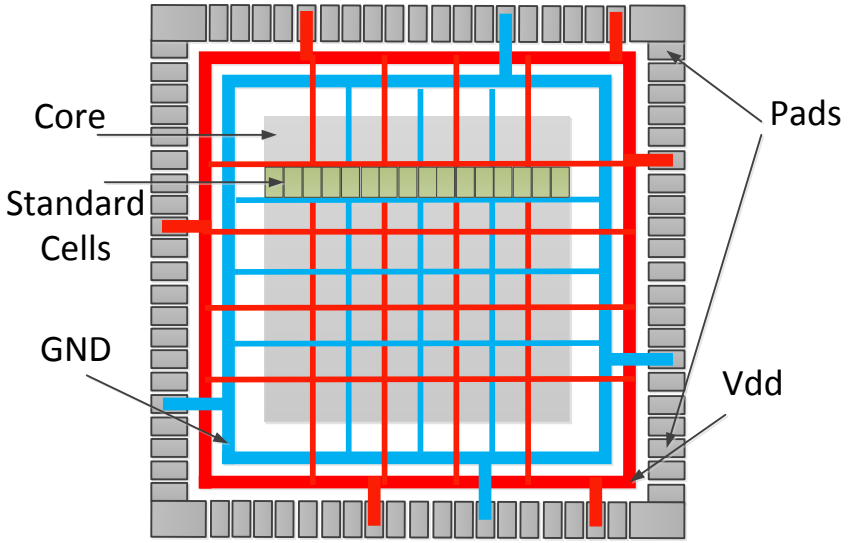


Figure 2.14: Equivalent circuit model of a power grid.

To compute Ldi/dt, the PDN can be modeled by a RLC mesh network as shown in Fig. 2.15. To find the voltage of each grid the following equation should be solved:

$$GV + CV' = u(t), \tag{2.8}$$

where V is a voltage vector, G is the conductance matrix, C includes the capacitance and inductance terms, and $u(t)$ is current [60].

In steady state power delivery analysis to calculate IR-drop, as shown in Fig. 2.16, PDN can be modeled by a resistance mesh network which is distributed over the core area [61]. Therefore, the voltage droop as a function of drawn current is written as follows [62]:

$$V = G^{-1}I, \tag{2.9}$$

where V is the vector of supply voltages of the grids. I is the vector of current drawn off the power grids and G is the conductance matrix. The current drawn from each grid can be calculated by adding the dynamic and leakage current of all the gates inside the grid. Resistance (R) of the power network is a function of the operating temperature (T) and it can be expressed by:

$$R = r_0(1 + cT), \tag{2.10}$$

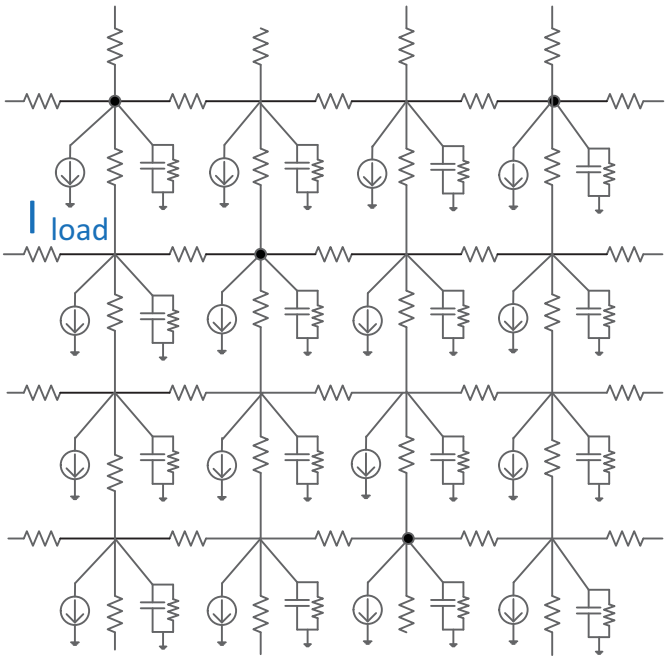


Figure 2.15: Equivalent RLC model of a power grid.

where r_0 is the resistivity at the nominal temperature and c is the temperature coefficient of metal used in the power grid [62].

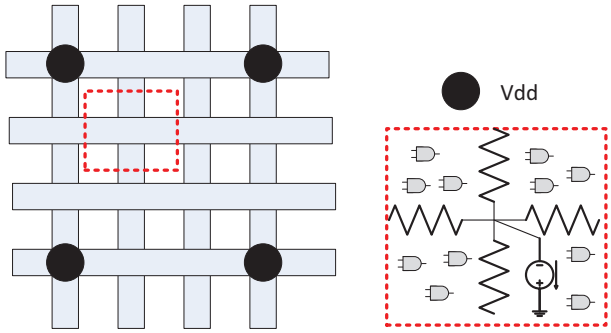


Figure 2.16: Equivalent circuit model of a power grid.

2.5 Temperature Model

Temperature variations is another major source of parameter variations that can result in dramatic fluctuations in circuit delay during runtime [5, 34, 63]. It also strongly escalates aging mechanisms and increases voltage-droop. Since in nano-scaled era power consumption and power density are increased,

variations in temperature get larger, resulting in more timing failures. Therefore, temperature modeling approaches and mitigation techniques should be evolved with the same pace in order to achieve a reliable design. In general, heat generation is a function of workload-dependent chip activity and leakage power. On the other hand, heat dissipation is related to the chip floorplan, thermal conductance of the chip, and the cooling system [5]. Fig. 2.17 shows a modern *Ceramic Ball Grid Array* (CBGA) package [64]. In general, there are two heat flow paths in the package. The first one starts from silicon bulk through the thermal interface material, heat spreader and heat sink, to the ambient air [5]. The second one, starts from silicon bulk through the interconnect layer, C4 pads, ceramic substrate, CBGA join to the printed circuit board.

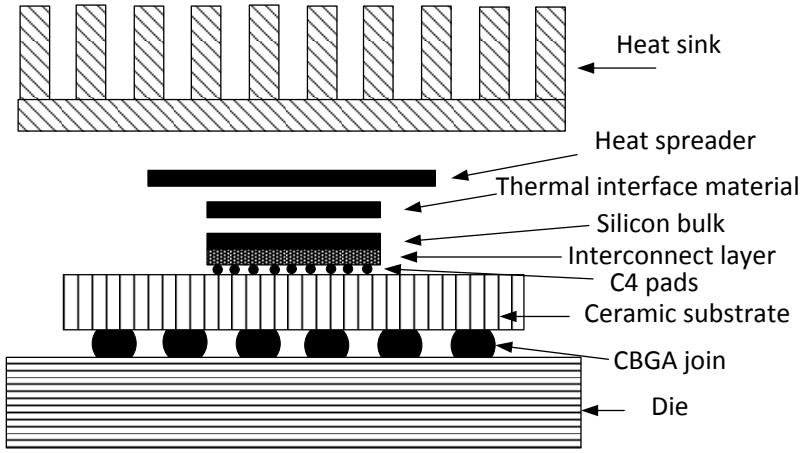


Figure 2.17: Stacked layers in a typical ceramic ball grid array (CBGA) package [5].

The typical scheme for extracting the thermal profile is by partitioning the chips into several cubes (i.e. vertically discretizing the chip into several layers and each layer is then laterally divided into rectangular grids). Therefore, each grid has one vertical thermal resistance to its neighbor located in the next layer and also has several lateral resistances to its neighbors in the same layer. The on-chip steady state temperature profile is governed by the following heat conduction equation subject to proper boundary conditions [65]:

$$\nabla.(k(\vec{r})\nabla T(\vec{r}) + P(\vec{r})) = 0, \tag{2.11}$$

where $\vec{r}$ represents the location in the 3D space, k is the thermal conductivity of the material, T is the temperature, and P denotes the power density of the heat source. The most common approach to solve the above equation is to make an analogy between thermal and electrical model. Table 2.1 shows the equivalent electrical parameters of thermal parameters. Finally, Kirchhoffs current-law is exploited to analyze the equivalent electrical model and the corresponding linear system of equations [66].

Table 2.1: Duality between thermal and electrical models [5].

Thermal quantity	Unit	Electrical quantity	Unit
Q, Heat transfer rate, power	W K	I , Current	A
T , Temperature difference	K/W	V , Voltage difference R	V
Rth , Thermal resistance	J/K	Electrical resistance C ,	Ω
Cth, Thermal capacitance		Electrical capacitance	F

2.6 Summary and Conclusions

As semiconductor technology scales to the deep nanoscale regime, parameter variations are posing a major challenge for integrated circuits. Variations are considered to be a dominant source of lifetime and frequency limiters and they have a significant impact on power consumption. Parameter variations are induced by process variations, as well as workload-dependent runtime variations such as voltage and temperature fluctuations and transistor aging. Process variations arise from imperfections in the manufacturing process. Voltage variations are caused by the instantaneous switching current drawn from the power-grid network. Temperature variations can be attributed to fluctuations in leakage/dynamice power across the chip. Finally, transistor aging, mostly due to BTI, is caused in large part by pre-existing traps in SiO2 and trap generation at the interface of Si/SiO2, resulting in a gradual increase in V_{th} and hence circuit delay over time. In this chapter, the physical mechanisms and the corresponding adverse impacts of parameter variations were discussed. In addition, it was shown that how each source of parameter variations can be modeled.

CHAPTER 3

AGING- AND VARIATIONS-AWARE TIMING ANALYSIS TECHNIQUES

This chapter overviews the state-of-the-art timing analysis techniques and their pros and cons. It is followed by explanation of the proposed variations-aware timing analysis technique. Finally, the accuracy of the proposed variations-aware timing analysis technique is compared against previous techniques. As will be discussed in the next chapters, the proposed timing analysis framework allows to perform circuit monitoring and different mitigation techniques to cope with reliability challenges.

3.1 State-of-the-arts

State-of-the-art aging-aware timing analysis tools can be classified into two main categories: transistor-level simulation and gate-level simulation. In the transistor-level method, first, the fresh circuit is simulated in order to determine the operating statistics of each transistor. Next, based on the collected workload information, the aging-induced threshold voltage shift is calculated for each transistor. Finally, the obtained aging-induced ΔV_{th} is applied to each transistor and the aged circuit delay is calculated [67]. Although the transistor-level method provides an accurate aging-aware timing information, it suffers from high simulation runtime which makes it infeasible for large circuits. Gate-level techniques either can be based on equations or based on Look-up Table (LUT). In the first approach, aged delay is estimated according to the ΔV_{th} using alpha-power-law model [68, 69, 70]. The shortcoming of the equation-based model is that it cannot capture the gate delay relation to runtime variation effects and the effect of the input slope. Moreover, the aged gate delay is obtained based on one equivalent ΔV_{th} for each gate. However, considering one equivalent ΔV_{th} instead of using different ΔV_{th} for all transistors inside the gates with multiple inputs, can result in a considerable inaccuracy. LUT-based gate delay model is adapted to take aging effects into account [71, 68]. This approach improves the accuracy in comparison with equation-based methods while it is compatible with commercial timing analysis.

Recently, a few studies tried to analyze the combined effect of process variations and NBTI in timing analysis. In [72], a new Vth model is proposed to capture the variation of the BTI effect considering process variations. [73] proposes a comprehensive reliability framework considering both process

variation and BTI. In [48], the effect of BTI and process variation is modeled under input pattern variation for a register file and a Kogge-Stone adder. In [49], the authors analyzed the effect of process variation on transistor aging using a Monte-Carlo based transistor-level simulation. While all the aforementioned techniques study the combined effect of NBTI and process variation, none of them considers temperature and voltage droop. There are a few techniques that consider temperature and voltage profiling during timing analysis [74, 35, 75]. In [35], temperature profile is extracted by considering the deterministic power sources which is later used for adjusting the gate delay. In [74], the profiling is improved by considering the dependence of the leakage power on temperature. However, the BTI effect and the effect of the voltage droop on leakage-dynamic power are not considered in prior techniques that results in significant timing error.

3.2 Variations-aware Timing Analysis

In this section, we present our proposed LUT-based technique for calculating the gate and ultimately circuit delay in the presence of aging. The overall flow of this method is depicted in Figure 3.1. According to this figure, in the first step, which has to be performed only once, either SPICE simulations or automatic library characterizer EDA tools such as Cadence Encounter Library Characterizer [59] are used to characterize each cell in the technology library in n+4 corners. The first n corners are dedicated to capture the effects of V_{th} shifts n transistors (n transistors inside the cell. The other four corners capture input slew, output load, temperature, and voltage of the cell. An important issue during LUT generation is accuracy, i.e. sampling frequency (table index) of each dimension. We observed that 10 °C, $0.05v$, $0.02v$ as the sampling intervals are reasonable choices for temperature, voltage, and threshold voltage, respectively, for a good trade-off between runtime and accuracy. For the other dimensions (i.e. input slope and output load) we use the default sampling rate as defined in the original technology library file.

In the next phase of our proposed flow, the circuit is synthesized and mapped to the characterized standard cell library. Then, workload and the extracted gate level netlist is fed to a fast logic simulator to extract the workload-dependent usage (logic level usage) details and signal probability

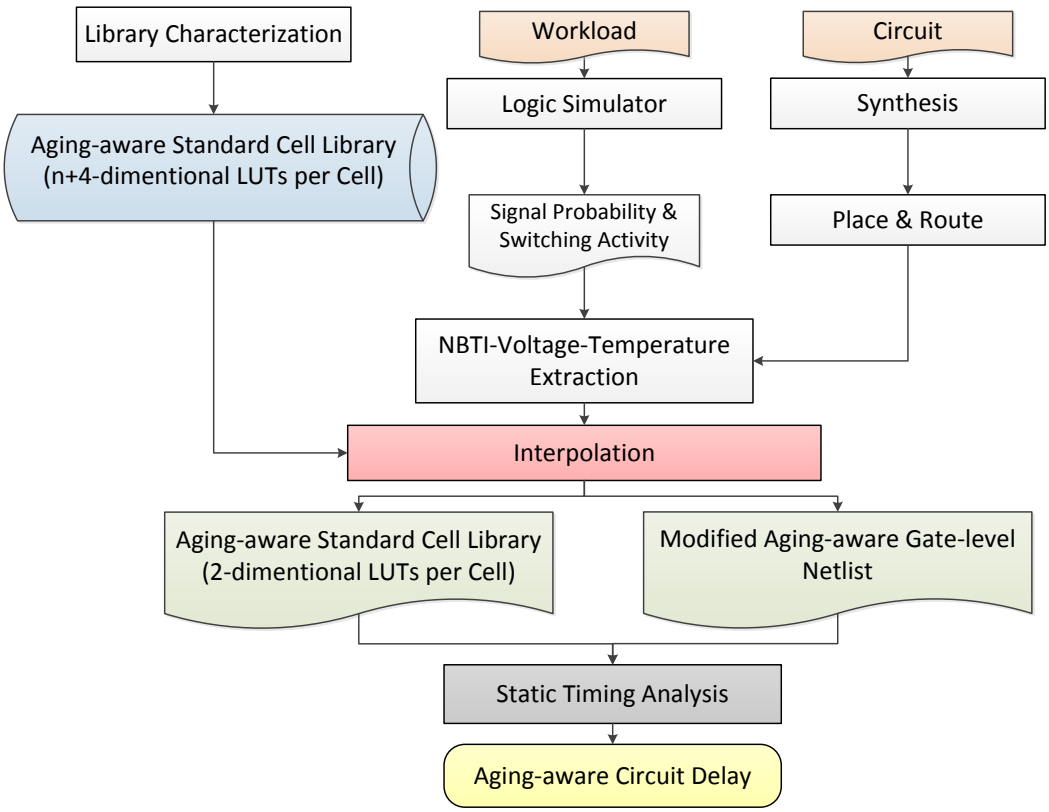


Figure 3.1: Overall flow of the proposed runtime variations aware static timing analysis.

and activity factor of each transistor inside the netlist. The effective duty cycle of each transistor in every gate is calculated by considering the stacking effect using the extracted signal probabilities. The extracted workload-dependent usage and duty cycle of each device is then translated to voltage, temperature and BTI-induced threshold voltage change. Considering the correlation and interdependence among different sources of runtime variations while accounting their different time scales is a major challenge. Voltage droop has a short term variation (ns) which is a result of different input vectors which are applied to the circuit. Temperature varies at higher time scale (ms) and as a result for thermal profiling, only considering the DC-behavior of the voltage droop would be sufficient [62]. On the other hand, BTI is a phenomenon which increases the circuit delay gradually (several weeks and months). Therefore, to estimate the BTI-induced delay degradation over time, it is sufficient to use average value of the temperature and the supply voltage at the time scale which BTI is considered.

The overall algorithm for obtaining the power, voltage droop, and temper-

ature profiles as well as BTI is depicted in Figure 3.2. In this flow, two loops are used to accurately model the interdependence among the voltage droop, temperature, and BTI. In the inner loop which is based on [34, 62, 75] power, temperature and voltage droop are obtained. First, power consumption of each grid is calculated by adding up the power consumption of each cells located inside the given grid. Once the power profile is obtained, it is converted into the temperature profile. Temperature profile can be extracted by the flow described in previous section or by using a sign-off thermal-profiling tool (e.g. HotSpot). Afterward, the resistance of the power mesh network is updated based on the temperature profile. Power mesh network information together with power profile and temperature profile are used to extract the voltage droop of each grid. Since, power consumption depends on temperature and voltage, the obtained temperature and voltage droop profiles are used to update the gate power and in turn power profile. This loop is iterated until convergence is reached. In the second loop, BTI-induced threshold-voltage change is estimated. The new threshold voltage is then used to update the power, temperature, and voltage profiles. In other words, the inner loop and BTI-estimation are parts of the outer loop. These two loops are iteratively executed until all the profiles reach a convergence. According to our observations, each loop, at worst case, only needs 10 iterations to converge.

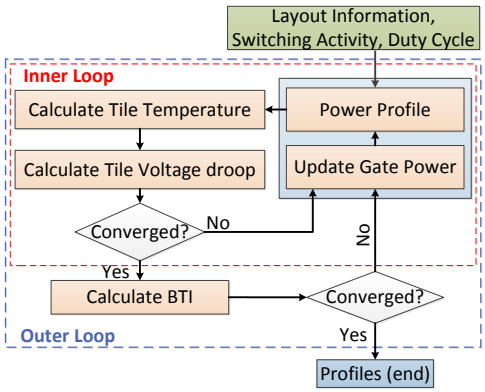


Figure 3.2: Overall flow of the proposed Power-Temperature-Voltage profiling and BTI estimation method.

In conventional static timing analysis tools (e.g. Synopsys PrimeTime), gate delay and gate output transition time are modeled as a function of only input transition time and output load capacitance (2-dimensional LUTs). Therefore, we need to reduce the dimensions of the $n + 4$-dimensional stan-

dard cell library. For this purpose, we use interpolation. Interpolation is a technique to construct a new data point within the range of an already known data points. After analyzing the netlist and dimension reduction, each gate in the netlist will be mapped to a newly generated library element which captures the post aging delay of that gate, based on the V_{th} characterization of the original library cells and netlist simulations for activity analysis (See Fig. 3.3). Such dimension reduction and representation of post-aging delay in library cell format make this flow compatible with standard timing analysis flow.

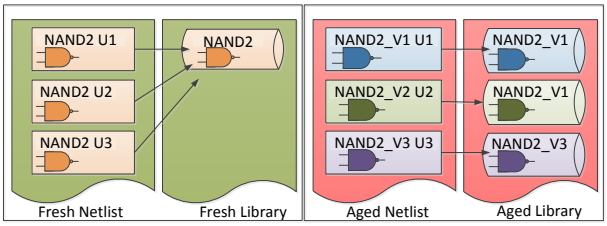


Figure 3.3: Frech STA versus aged STA.

Finally, the modified gate-level netlist and the generated runtime variations-aware technology library (one element per each cell in the netlist) are given to a static timing analysis tool to determine the circuit delay. Since $n + 4$-dimensional standard cell libraries are able to capture the effect of different parameters (such as temperature, voltage information and ΔV_{th} of different transistors within a cell) on gate delay, the estimated gate delay by this approach is very close to transistor level SPICE information. Another advantage of our method is that, it can be extended to handle other aspects of gate delay by augmentation of LUTs with other parameters such as process variation. Moreover, our LUT-based approach has the capability of a space/accuracy trade off and can be calibrated with post-silicon data as well.

3.3 Incorporating the Impact of Process Variations

In this section, we explain how our proposed methodology can be extended to consider the impact of process variations as well. This section is mainly based on [34, 62, 75]. In the presence of process variations, extracting the temperature, voltage droop and BTI becomes a statistical problem. Our proposed statistical flow for calculating thermal-voltage profile and BTI is

depicted in Figure 3.4. First, the die area is partitioned into virtual rectangular grids. Considering the spatial correlation among transistors on a die, process variation of each transistor is modeled by a normal distribution and represented by Equation (2.1). Next, PCA is performed to express the process variations in a canonical form as shown in Equation (2.2). The rest of the process is divided into three different steps: 1) Statistical Thermal Profile, 2) Statistical Voltage Droop Profile, and 3) Statistical BTI Analysis. In each of these steps, the first two moments (mean and sigma) of variable distributions are calculated. Here, based on prior studies [34, 62, 75], leakage and temperature are modeled with lognormal distributions. Since voltage and BTI are related to temperature and leakage by a set of Sum and Multiply operations, we model these variables (voltage and BTI) by lognormal distributions as well. These three steps are performed iteratively until sigma and mean value of distributions are converged (See Figure 3.4). Please note that during these steps based on [75], all of the equations and analysis are performed on a set of independent Principal Components (PC) derived during the PCA step. This enables us to fast and accurately capture the dependence among PVT and BTI. To the best of our knowledge, this is the first work that considers/models the combined effect of BTI and process variation in thermal-voltage profiling. Moreover, our proposed BTI analysis technique accurately captures the effect of process-induced voltage variations which is neglected in prior methods.

The leakage power of a gate depends on temperature and supply voltage with a quadratic function [76]. Moreover, the subthreshold leakage current increases exponentially with the threshold voltage. Since the threshold voltage is a function of the gate length, the leakage power is modeled as an exponential function of gate length [29]. Due to BTI effect, the threshold voltage of the gate increases over the time and hence exponentially affects the leakage power. However to be able to keep all of variables in normal and lognormal distribution, we model the dependency of the leakage to BTI-induced threshold voltage shift by a quadratic polynomial function. All of the aforementioned models are verified by accurate HSPICE simulation of a 7-stage ring oscillator in 45-nm technology. According to our results, the models match simulation data with R^2(R-squared) > 0.996. The following equation summarize the leakage power model as a function of temperature (T), supply voltage (V), threshold voltage shift due to BTI (V_{th-BTI}), and

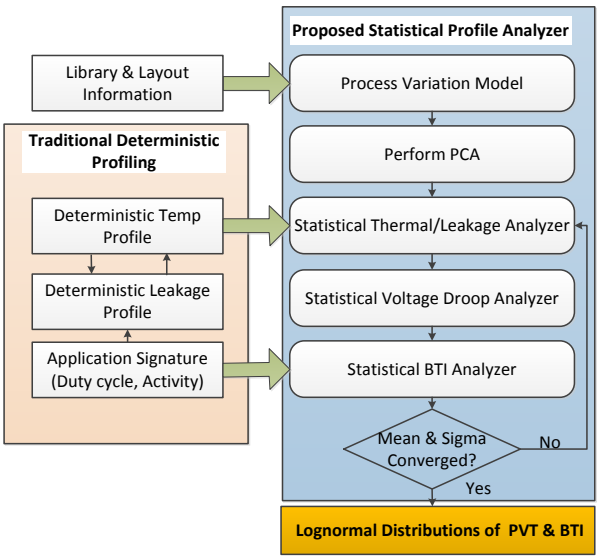


Figure 3.4: Flow of the proposed statistical leakage, temperature, voltage droop, and BTI profile analyzer.

gate length (ΔL):

$$P_{leakage} = P_{leakage}^{nominal} \cdot (1 + a_1 \cdot T + a_2 \cdot T^2).(1 + a_3 \cdot V + a_4 \cdot V^2). \\ (1 + a_5 \cdot Vth_{BTI} + a_6 \cdot Vth_{BTI}^2) \cdot exp(b \cdot \Delta L), \tag{3.1}$$

where $P_{leakage}$ stands for leakage power considering process variation. $P_{leakage}^{nominal}$ corresponds to the leakage power without any process variation at $T = 0°C$ and a_i , b are coefficients.

For computing the temperature profile from the power profile, initially a die is partitioned into n equal grids. The temperature of a grid T_i can be expressed based on the power-consumption of the grids in the die by a weighted sum [34] :

$$T_i = \sum_{j=1}^{n} a_{ij} \cdot P_j + a_{im} \cdot P_m, \tag{3.2}$$

where P_j represents the power of the grid j and a_{ij} is a coefficient reflects the sensitivity of a grid's temperature to the power change of the other grids on a die . P_m represents the chip to the ambient removing power and a_{im} captures the heat resistance from the heat sink to the air.

There is a positive feedback between leakage power and temperature of a chip. Moreover, process variation affects leakage power (by changing threshold voltage and effective gate length), which in turn results in a temperature variation [34]. The idea which is based on [34, 62, 75] is to update distribu-

tions of the temperature and power in an iterative way to reach a convergence in mean and sigma. Note that compared to [34, 62, 75], we only added the impact of BTI to analysis. Algorithm 1 which is adjusted from [34, 62, 75] shows the detail of the proposed statistical thermal profile analyzer. The proposed flow consists of two different phases: 1) Deterministic Leakage-Temperature Calculation 2) Statistical Leakage-Temperature Calculation [34, 62, 75]. In the first phase, nominal power of each grid is calculated by adding up the power of the gates located in the grid (without considering the effect of process variations). Next, a deterministic (nominal) thermal-leakage profile is obtained by considering the leakage-thermal loop effect (Performing Line 5-6 in the Algorithm iteratively). In the second phase of the algorithm, process variation is added to the leakage-temperature models. According to Equation (3.1), leakage power is exponentially related to process variation (e.g. gate length). Since process variation is represented in canonical form, leakage power is expressed by a lognormal canonical form as Equation (3.3) [34, 62, 75]:

$$L_A = exp(A), \quad A = \mu_A + \sum_{i}^{n} a_i \cdot X_i, \tag{3.3}$$

where A is a normal distribution (in the exponent of L_A), and X_i's are principal components. Moreover, temperature is a linear function of leakage power (see Equation (3.2)). Therefore, temperature can also be expressed by a lognormal canonical form. After generating the thermal-leakage distributions, Equations (3.1) and (3.2), by considering the leakage-thermal distributions, should be iteratively updated until a convergence (sigma and mean value of lognormal distribution) is reached. For this purpose, we need to be able to calculate lognomral sum and lognomral multiply operations. Suppose, L_A and L_B are two lognormal distributed variables expressed by Equation (3.3). Multiplication of two lognormal variables which are expressed by principal components is similar to the multiplication of two powers of the same base. Therefore, the lognormal-multiply is calculated by just adding the exponents (A *and* B). For estimating of lognormal sum ($L_C = L_A + L_B$), Wilkinson's method [77] is used. In this approach, mean and standard deviation of L_C

are calculated as following [75]:

$$\mu(L_C) = \mu(L_A) + \mu(L_B),$$
$$\sigma^2(L_C) = \sigma^2(L_A) + \sigma^2(L_A) + 2 \cdot cov(L_A, L_B), \tag{3.4}$$

Then by matching the first two moments, the parameters (mean(C), $\sigma(C)$) of the lognormal random variable (L_C) are extracted by [75]:

$$\mu_C = log(\mu^2(L_C)/\sqrt{\sigma^2(L_C) + \mu^2(L_C)}),$$
$$\sigma_C = \sqrt{log(\frac{\sigma^2(L_C)}{\mu^2(L_C)} + 1)}, \tag{3.5}$$

where C refers to the normal variable that is in the exponent of L_C. Finally we need to represent C in canonical form (C'). For this purpose we exploit a method that is proposed in [75]. Using this approach, c'_i is calculated by [75]:

$$c'_i = log(\frac{\mu_A \cdot exp(a_i) + \mu_B \cdot exp(b_i)}{\mu_A + \mu_B}). \tag{3.6}$$

Since all X_i (principal components) are independent, the variance of C' can be computed as: $\sum_i^n (c'_i)^2$. Obviously there is a difference between σ_C and the standard deviation of estimated canonical form ($\sigma_{C'}$) [75]. In order to diminish this error, the value of c'_i are normalized by $\frac{\sigma_C}{\sum_i^n (c'_i)^2}$ [38]. In addition, μ'_C is set to μ_C.

To assess the voltage droop, we augment the leakage-thermal profile analyzer [34, 75] with considering the voltage effect during analysis [62]. The new flow consists of two nested loops [62]: the inner loop belongs to the temperature-leakage loop; and the outer loop is used to statistically find the lognormal distribution of the voltage droop [62]. This process is iteratively performed until the mean and sigma value of the lognormal distributions (representing thermal, leakage, and voltage droop) converge (See Figure 3.4).

There is a feedback (loop) between BTI and thermal-voltage profiles. To assess the BTI and adjust the thermal-voltage profiles based on the BTI-induced V_{th} shift (which is our difference compared to [34, 62, 75]), we need to accurately consider this feedback. For this purpose, we model the BTI as a lognormal distribution. Next, Statistical Voltage Droop Profile (presented

Algorithm 1 Leakage-Thermal profile analyzer using PCA (based on [34, 62, 75]).

divide the die into n grids
P_i : *Power of each grid i*
//Deterministic Temp and Leakage Calculation
while *Temp and Leakage not converged* **do**
  $T = T_i \leftarrow \sum_{j=1}^{n} a_{ij} \cdot P_j + a_{im} \cdot P_m$
  $P_{leakage} \leftarrow P_{leakage}^{nominal} \cdot (1 + a_1 \cdot T + a_2 \cdot T^2)$
end while
//Statistical Temp and Leakage Calculation
Generate lognormal distributions of Temp and Leakage
while $Moments(\mu, \sigma)$ *of Leakage and T not converged* **do**
  Update $\mu_{Leakage}, \sigma_{leakage}$:
  Equation(3.1) *with nominal* $Voltage_{droop}$ *and* $BTI(V_{th})$
  Update μ_T, σ_T : $T_i \leftarrow \sum_{j=1}^{n} a_{ij} \cdot P_j + a_{im} \cdot P_m$
end while

in previous steps) is augmented by adding another loop for inserting the BTI effect to the analysis. Therefore, BTI profiling algorithm consists of three nested loops: Two inner nested loops for extracting leakage, thermal, and voltage droop profiles; The outer loop for obtaining the BTI profile. We iteratively execute these three statistical nested loops until mean and sigma of lognormal distributions of all profiles (leakage, thermal, voltage droop, BTI) converge (see Figure 3.4). Please note that all of the variables are expressed with a set of independent PCs, therefore the dependence among PVT and BTI is accurately considered.

3.4 Experimental Results

Several IWLS and ISPD benchmark circuits [78] are used to evaluate the efficiency and accuracy of the proposed methodology. Circuits are synthesized by Synopsys Design Compiler [79] using Nangate 45 nm library [80] and then the gate-level netlists are placed using Cadence SOC Encounter. Besides, each cell in the library is characterized by accurate HSPICE simulations. HotSpot [1] is used to obtain the thermal profile of the circuit. BTI-induced threshold voltage change is estimated by assuming a delay degradation of 10% in 5 years. To show how BTI, Voltage droop, and temperature affect the circuit delay, we consider six different scenarios listed in Table 3.1.

Table 3.1: Different scenarios to show the effects of runtime-variations on delay.

1	**-V-T-BTI**	No run-time variations
2	**+V-T-BTI**	Only voltage droop
3	**-V+T-BTI**	Only Temperature
4	**+V-T+BTI**	Only BTI
5	**Additive margin**	Independent summation of different margins from Scenarios 2,3,4
6	**+V+T+BTI (Proposed)**	Combined effect of all sources of runtime variation (V, T, BTI)

State-of-the-art statistical thermal profiling methods do not consider the aging and voltage droop effects on temperature. To show how these factors affect the accuracy of thermal profiling, we perform an experiment for a 7-stage inverter chain in 45 nm technology node with four different scenarios. According to the Table 3.2, neglecting the effects of aging and voltage droop results in up to 2.38% and 67% error in the estimation of the mean and standard deviation of the estimated temperature of the chip, respectively. Hence, this overestimation of the temperature profile leads to considerable error (8.54% in mean and 14.45% in standard deviation) in BTI wearout estimation.

Table 3.2: Error of incomplete consideration of the interdependence among PVT and BTI in Temperature and BTI (ΔVth) estimation compared to our proposed technique (+V+T+BTI)

	max_{Temp}	μ_{Temp}	σ_{Temp}	$\mu_{\Delta Vth}$	$\sigma_{\Delta Vth}$
-V+T-BTI	89.23%	2.38%	67.00%	8.54%	14.45%
+V+T-BTI	14.09%	0.50%	13.60%	0.00%	1.03%
-V+T+BTI	38.07%	1.61%	38.40%	8.54%	12.14%

Table 3.3 shows the circuit relative delay increase (w.r.t. *-V-T-BTI*) due to runtime variations with different schemes. Comparing the seventh (proposed method) and sixth (simple additive margin) columns of the table reveals that independent analysis of temperature, voltage, and BTI leads to 17% inaccuracy in circuit delay estimation in average. To verify the scalability of our method the runtime is calculated when all of the simulations are performed on a workstation with Intel Xeon E5540 2.53GHz (2 quad-core

processors), 16GB RAM. As shown in Table 3.3, even for very large circuits such as leon3mp processor, the runtime of our proposed method is less than an hour.

Table 3.3: Relative circuit delay increase (w.r.t. **-V-T-BTI**) due to runtime variations ($Error = (Proposed - additive\ margin)/Proposed$).

Circuit	# of cells	+V-T-BTI	-V+T-BTI	-V-T+BTI	additive margin	Proposed	Error	Time (s)
b17	27k	6%	6%	6%	17%	22%	25%	654
b18	88k	9%	6%	6%	21%	25%	16%	978
b19	165k	8%	7%	8%	22%	32%	29%	1071
b22	40k	9%	7%	6%	17%	23%	24%	658
dsp	42k	2%	6%	17%	25%	28%	13%	444
leon2	995k	3%	9%	11%	23%	29%	20%	3245
leon3mp	721k	3%	7%	15%	25%	30%	18%	2458
vga_lcd	114k	5%	16%	21%	41%	48%	14%	1059
risc	61k	10%	10%	13%	33%	39%	16%	754
des_perf	84k	2%	19%	19%	40%	44%	10%	1060
average							17%	

Next, we investigate the impact of temperature and voltage on BTI. In [40] the effect of temperature (and partially voltage variations) on BTI analysis is well studied. Unfortunately, their proposed timing analysis flow only considers some corner cases. According to Table 3.4, assuming a constant temperature ($T_{nom} = 25°C$) leads to 10% error in the estimated BTI-induced delay degradation (compared to $+V+T$). Considering a constant power supply voltage ($VDD_{nom} = 1V$) results in 12.8% inaccuracy in estimated BTI-induced delay increase.

Table 3.4: The effect of neglecting voltage and temperature variations on BTI-induced delay degradation (error are calculated w.r.t Scheme: +V+T).

Circuit	-V-T	+V-T	-V+T
b17	-5.0%	-10.0%	30.0%
b18	-15.1%	-15.9%	1.59%
b19	-11.9%	-13.9%	4.3%
b22	-2.0%	-4.0%	5.0%
dsp	-19.2%	-21.9%	20.6%
leon2	-12.4%	-14.7%	9.5%
leon3mp	-14.0%	-16.7%	16.2%
vga_lcd	-21.4%	-23.8%	9.5%
risc	-1.2%	-6.7%	3.4%
average	-10.2%	-12.8%	10.0%

The effect of the BTI on voltage and temperature profiles are investigated and shown in Figure 3.5. Neglecting the effect of BTI (which changes the power density and in turns voltage-temperature profiles) leads to 4.8% and 8.8% error in estimated temperature and voltage droop, respectively.

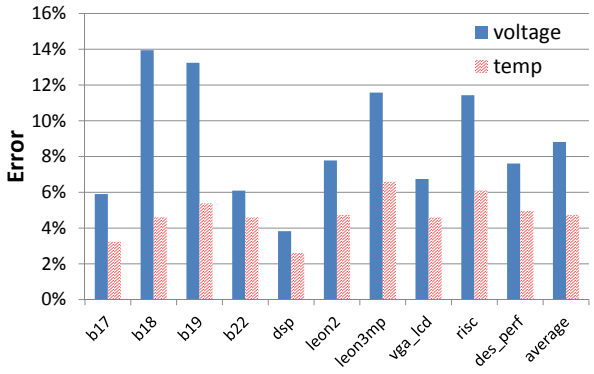


Figure 3.5: The error caused by neglecting BTI on the voltage and the temperature.

Input activity due to workload variation influences the voltage and temperature profiles and in turns affects the BTI. Figure 3.6 shows the circuit delays at different primary input activity factors (0.2,0.5,0.8). Higher inputs activity factors leads to larger circuit delay.

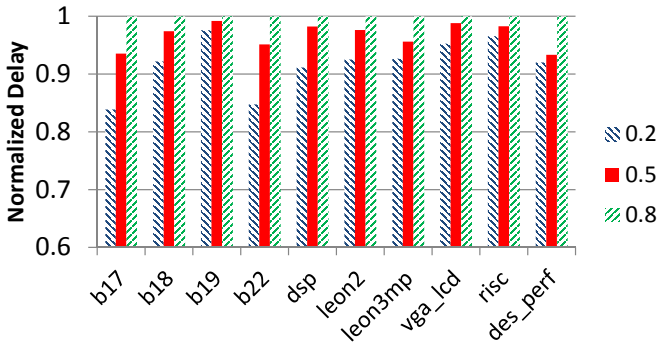


Figure 3.6: The effect of activity factor on the circuit delay.

3.5 Conclusions

In nano-scale regime, process variations as well runtime variations due to voltage, temperature, and transistor aging introduce remarkable uncertainty

and unpredictability to circuit delay and its lifetime. Consideration of short-term and long-term workload-dependent runtime variations at design time and the interdependence of various parameters are major challenges of timing analysis. However, a novel approach to tackle all these issues and their interdependence was missing. In this chapter, we presented a novel timing analysis framework to accurately capture the combined effects of various workload-dependent runtime variations happening at different time scales, by making the link between system-level runtime effects and circuit-level design. The proposed frameworks can be fully integrated with existing commercial EDA toolset, making it scalable for very large designs. Using the proposed timing analysis technique, we observed that treating each aspect independently and ignoring their intrinsic interactions can lead to inaccurate results. The proposed timing analysis technique can be used to accurately identify the timing margin in order to prevent over/under design.

CHAPTER 4

CHIP DELAY/AGE MONITORING USING MACHINE-LEARNING

Parameter variations degrade path delay over time and may eventually induce circuit failure due to timing variations. Therefore, in-field tracking of path delays and prediction of operational frequency and lifetime of chips are essential to realize runtime adaptive mitigation techniques in order to cope with the detrimental effects of variations. Several delay sensor designs have been proposed in the literature. However, due to the significant overhead of these sensors and the large number of critical paths in today's IC, it is infeasible to monitor the delay of every critical path in silicon. This chapter overviews state-of-the-art monitoring systems and then presents our novel aging- and variations-aware representative path-selection technique based on machine learning that allows us to measure the delay of a small set of paths and infer the delay of a larger pool of paths that are likely to fail due to delay variations.

4.1 State-of-the-arts

Variation-aware delay/age monitoring can be implemented in four different ways [81]:

- *On-line self-test:* this method is based on periodical delay-test using pre-stored test-patterns [82]. However, the normal operation of the circuit might be interrupted to be able to apply test-patterns [83].

- *Replica circuit:* in this technique a stand-alone (i.e., replica) circuit such as a set of ring-oscillators is inserted in various places in the circuit to mimic the delay degradation of the original circuit [84, 85]. However, replica-circuits might fail to entirely capture the effects of running workload and variations on the original circuit [81].

- *In-situ delay sensor:* in this case, dedicated sensors are inserted next to the flip-flops of functional critical paths to directly measure the corresponding delay during field-operation [86, 87]. However, since the number of critical paths can grow exponentially to the number of gates, it is infeasible to monitor each and every critical path [81].

- *Representative path monitoring:* in this technique, the delay of a small set of paths are monitored and based on that the delay of other critical

paths are inferred [88]. However, existing techniques only consider the effects of process variations, while the impacts of runtime variations including transistor aging are ignored.

4.2 Problem Statement and Overview of Proposed Method

In adaptive mitigation techniques, the circuit behavior is monitored at runtime, and a suitable knob is tuned based on the feedback. Such runtime adaptation schemes rely on a in-field chip delay/age monitoring infrastructure. One approach to monitor circuit delay in the presence of parameter variations is to target a large pool of target (long) paths that are more likely to have timing failures; such paths are referred to as Critical Paths (CPs). However, monitoring such a large number of CPs is not feasible due to the cost associated with the placement of too many sensors. A solution to this problem lies in the selection of only a small set of Representative Critical Paths (RCPs) from the large pool of target paths. The delays of the RCPs are accurately measured either by on-chip sensors or via delay testing, and the measured values are mapped to the delays of the other critical paths by exploiting the similarities in timing characteristics between CPs and RCPs. In other words, our objective is to select an optimal number of paths as RCPs to predict delays of a large pool of CPs while ensuring that the prediction error is minimized by accurately taking the effects of variations and transistor aging into account. The proposed flow consists of two different phases: 1) *feature extraction*, and 2) *identifications of RCPs*.

We utilize our proposed variations-aware timing analysis framework, presented in Chapter 3, to obtain the critical paths of the circuit. Next, all possible CPs within the circuit are enumerated and then are encoded into a vector. Afterwards, we use different learning-machine techniques to select the RCPs. Finally, to verify the accuracy and efficiency of the proposed RCP selection method, we compare the actual measured path delays (under different aging and variations) with the predicted delay values.

4.3 Feature Extraction

In general, there are a variety of topological and electrical similarities among CPs. For example, CPs that are in proximity in the layout of the chip tend to have a similar voltage droop and temperature. Moreover, CPs might have a large number of gates in common. We propose to use a machine-learning approach to capture the correlations among CPs, based on which we can select a small set of RCPs. Suppose each CP p_i can be encoded by a vector $p_i = [x_{i1}, x_{i2}, \ldots, x_{iM}]$ with M chip features. Each of these features captures the sensitivity of the path delay to one source of uncertainty (e.g., process variation, voltage, temperature, aging, etc.). The delay of path p_i, d_{p_i}, can be calculated by the following equation:

$$\begin{aligned} d_{p_i} &= p_i F, \\ F &= [F_{i1}, F_{i2}, \ldots, F_{iM}], \end{aligned} \tag{4.1}$$

where F is a vector that captures the value of features. We use a hypothetic circuit, shown in Fig. 4.1, as an example to illustrate the proposed path-encoding approach. This circuit consists of a CP, namely P_{CP}, which is located in three different grids of power delivery network, namely V_1, V_2, and V_3, respectively. P_{CP} can be described as follows:

$$\begin{aligned} P_{CP} &= [x_1, x_2, x_3], \\ d_{P_{CP}} &= P_{CP} F, \\ F &= [V_1, V_2, V_3], \end{aligned} \tag{4.2}$$

where x_1, x_2, x_3 are the sensitivities of the $d_{P_{CP}}$ to V_1, V_2, and V_3, respectively. V_1, V_2, and V_3 represent the actual values of voltage features.

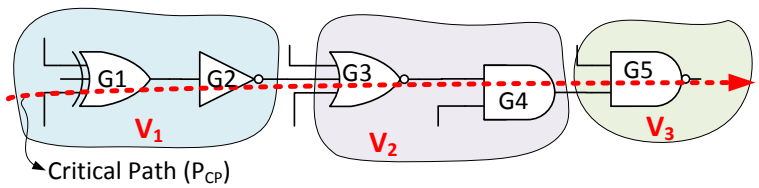


Figure 4.1: A hypothetic circuit for illustrating path-encoding algorithm.

Fig. 4.2 shows the overall flow of the proposed path-encoding approach, which can effectively explore the uncertainty space of variations. The path encoding features include topological feature, process-variation feature, BTI feature, temperature feature, and voltage feature, respectively, discussed below.

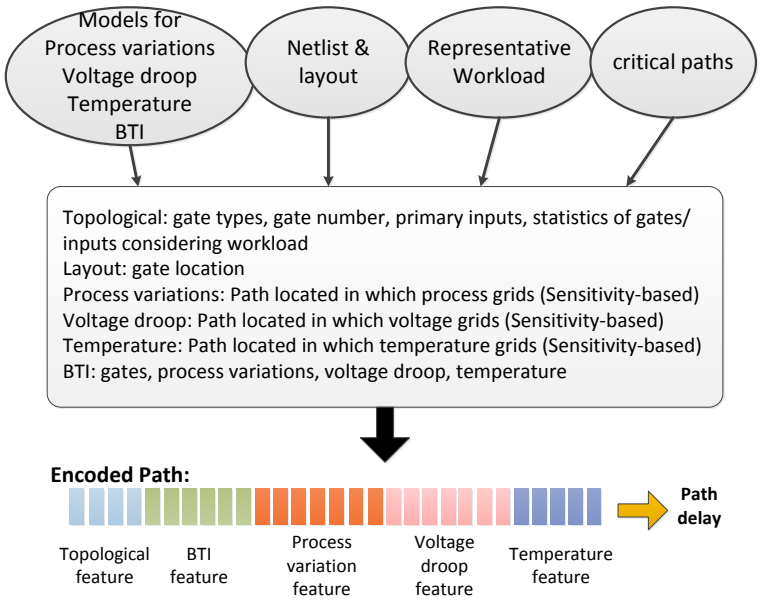


Figure 4.2: Path encoding flow.

1. Topological features: this feature models: (i) which gates are located in each CP; (ii) the gate types; (iii) location of the gates in the floorplan.

2. Process-variation features: this feature models the process variation of the circuit. In this work, without loss of generality, the total process variation is modeled by the summation of die-to-die, within-die partially-correlated, and independent random variations. To accurately capture the within-die spatial correlation, the chip layout is first divided into rectangular grids. Finally, a CP is encoded in a way that the corresponding vector reflects the grids to which the path is sensitive.

3. : Temperature feature: one of the major sources of runtime variations, which strongly influences circuit delay, is temperature. In order to encode a path with respect to temperature, the chip area is divided into several grids. The representative temperature feature of the path reflects the grids in which the path passes through.

4. Voltage-droop feature: voltage droop, which has been shown to vary significantly over time and from-gate-to-gate, significantly affects circuit delay. Since power delivery network can be modeled as a resistive network distributed over the die, a CP is encoded in such a way that the corresponding vector reflects the grids to which the path is sensitive.

4.4 Identification of RCPs

Recall that our goal is to select RCPs for monitoring in order to estimate the chip performance. Based on the measured delays in RCPs, we could accurately estimate the delay of other CPs. Given N CPs, we use an $M \times N$ matrix $P = [p_1, p_2, \ldots, p_N]^T$ to denote these paths. Note that each path p_i is encoded with M features as described in Section 4.3. The delay of N paths can be expressed as a vector $D = [d_1, d_2, \ldots, d_N]^T$. The selected RCPs can be represented as an $M \times R$ matrix $P_R = [p'_1, p'_2, \ldots, p'_r]^T$, where $R << N$. Similarly, the delay measurements of the RCPs are of the form of a vector $D_R = [d'_1, d'_2, \ldots, d'_R]^T$.

In order to identify the RCP set, we rely on unsupervised machine-learning techniques, such as the SVD-QRcp method and clustering, which are discussed below. The choice of unsupervised learning is motivated by the fact that we have no data available on the behavior of the chip for supervised learning. In this work, we propose to use an adaptive method in Section 4.4.3, which uses both SVD-QRcp method and C-means method. We will first introduce SVD-QRcp method and C-means clustering method in Section 4.4.1 and Section 4.4.2, respectively.

4.4.1 SVD-QRcp Method

Singular-value decomposition and QR decomposition with column pivoting (SVD-QRcp) is an orthogonal transformation technique that has been widely used for feature selection in many areas, such as signal processing, control theory, and network optimization [89, 90]. Using the SVD-QRcp method, an RCP set P_R can be selected from the complete CP set P. The delay for each CP can be estimated using a linear combination of measured delays in

RCPs. The estimated delay $\overline{D}$ can be expressed as follow:

$$\overline{D} = PP_R^T(P_R P_R^T)^{-1} D_R, \tag{4.3}$$

where $()^{-1}$ denotes the inverse matrix. The corresponding estimation error can be measured using relative root mean-squared error ($rRMSE$), defined as:

$$rRMSE = \frac{\sqrt{\sum(\overline{D} - D))^2}}{N \cdot range(D)} \times 100\%, \tag{4.4}$$

where $range(D)$ is the range of $D = max(D) - min(D)$. To accurately predict delays in critical paths using delays in representative paths, the selection of representative paths represents a tradeoff between number of RCP R and prediction error Err. To select RCPs, we rely on SVD factorization, which transforms the matrix P into a product of three matrices. The decomposition can be written as:

$$P = U\Sigma V^T, \tag{4.5}$$

where matrix $U \in R^{N \times N}$ and $V \in R^{M \times M}$ are orthogonal matrices, and $\Sigma = \text{diag}(\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_M \geq 0)$. The diagonal elements of Σ are called the singular values of P. An important property of SVD is that it reveals the rank of P. In Equation (4.5), $\text{rank}(P) = \text{rank}(\Sigma)$. Consequently, the number of non-zero singular values indicates the rank of the matrix P. However in our application, we can get an even smaller number $R < \text{rank}(\Sigma)$, since the existence of smaller singular values σ_i implies the presence of redundancy or less important rules among the rules that forms the complete set [91].

In order to determine appropriate R, we adopt the criterion p_{ex}, i.e., the percentage of "energy" explained by singular values [92]. It is defined as:

$$p_{ex} = \frac{\sum_{i=1}^{R} \sigma_i^2}{\sum_{i=1}^{N} \sigma_i^2} \times 100, \tag{4.6}$$

where R is the number of RCPs for which the energy explained by the corresponding R number of singular values is p_{ex} percentage of the total energy. In this work, we determine a minimum of R RCPs to meet $P_{ex} > P_{ex-th} = 99\%$, whereby these R RCPs can represent nearly the entire set of CPs.

Once we determine the optimum number of critical paths R, we then select the positions of these RCPs based on QR decomposition and column pivoting

Algorithm 1: SVD-QRcp method

Require: P, p_{ex-th}
Ensure: P_R, R
1: $N \leftarrow$ number of critical paths in P
2: Singular-value decomposition: $[U, \Sigma, V] = \text{SVD}(P)$;
3: $R \leftarrow 0$, $p_{ex} \leftarrow 0$, array $S \leftarrow$ diag (Σ);
4: **while** $p_{ex} < p_{ex-th}$ **do**
5: $\quad R \leftarrow R + 1$;
6: $\quad p_{ex} = \frac{\Sigma_1^R s_i}{\Sigma_1^N s_i}, \quad \forall s_i \in S$;
7: **end while**
8: Select first R columns in U, $U_R = U(:, 1:R)$;
9: QR-decomposition with column pivoting:
$[Q, R, \Pi] = \text{QR}\ (U_R^T)$;
10: $P_n = \Pi^T P$;
11: $P_R = P_n(:, 1:R)$;
12: **return** P_R and R

Figure 4.3: Procedure for the SVD-QRcp method.

(QRcp), using the following equation:

$$U_R^T = QR\Pi^T, \tag{4.7}$$

where the input to this procedure is U_R, a sub-matrix formed by the first R columns of U [88, 83]. Note that Q is a unitary matrix and R is an upper triangular matrix. The permutation matrix Π can transform P, reflected in U_R^T, so that the critical paths in $P_n = \Pi^T P$ appear in a decreasing order of corresponding importance. Then we take the sub-matrix P_R formed by the first R rows of P_n to be the RCP set. The complete algorithm is presented in Fig. 4.3. The computational complexity of SVD-QRcp method depends on the SVD algorithm, which has the computational complexity of O(min{M^2N, MN^2}), where min{$*$} is the operation of obtaining the smaller value, and M and N are the row length and column length, respectively, for matrix P.

Next, we present a small example to illustrate the selection of RCPs based on SVD-QRcp. Suppose we have a CP set P_0 and *delay contribution* vector

T_0 as follow:

$$P_0 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}, \quad T_0 = \begin{bmatrix} 2 \\ 1 \\ 2 \\ 2 \\ 3 \\ 1 \\ 2 \\ 1 \\ 2 \end{bmatrix}, \tag{4.8}$$

where P_0 consists of 8 CPs as row vectors, each of which has 9 features. An entry of 1 indicates that the CP corresponding to that row exhibits the feature for that column. Delay of CP set is $D = P_0 \cdot T_0 = \{d_1 = 11, d_2 = 8, d_3 = 9, d_4 = 4, d_5 = 8, d_6 = 9, d_7 = 5, d_8 = 13\}$. According to the SVD-QRcp algorithm shown in Fig. 4.3, if we set number of RCPs R to be 5, the selected RCP set is $P_R = \{p_2, p_3, p_5, p_6, p_8\}$, and $P_e x = 0.95$ according to Equation (4.6). By measuring the delay of RCPs in P_R and using Equation (4.3), we can predict the delay of remaining CPs $d'_1 = 10.8$, $d'_4 = 3.6$, and $d'_7 = 4.9$. The $rRMSE$ is calculated to be 1.2%. If we consider fewer RCPs, e.g. $R = 3$, the selected $P_R = \{p_2, p_5, p_8\}$. Then the predicted delay of the remaining CPs are $d'_1 = 10.9, d'_3 = 7.2, d'_4 = 3.2, d'_6 = 7.6$, and $d'_7 = 4.3$, and $rRMSE$ is 7.2%, which is still quite low.

4.4.2 C-means Clustering Method

C-means clustering incorporates fuzzy logic, whereby each critical path has a probability of belonging to each cluster [93], thus each critical path can probabilistically belong to two or more clusters rather than only one cluster. This fuzzy set membership can be interpreted as that any two paths may share partial features, thus any path is a combination of multiple features that can be regarded as clusters. The objective of C-means clustering is to maximize the inter-cluster variance and minimize the intra-cluster variance. The training of C-means clustering is based on minimization of the objective

function J_m as shown below:

$$J_m = \sum_{i}^{N} \sum_{j}^{C} u_{ij}^{m} ||p_i - c_j||, \tag{4.9}$$

where $m \geq 1$ is a weighting factor, and $|| * ||$ is the Euclidean norm. The parameter N is the number of CPs. Set C consists of k clusters, in which c_j is the centroid of each cluster, and u_{ij} is the probability that a path p_i belongs to c_j. The optimization approach follows two iterative steps involving centroid c_j and probability u_{ij}, such that:

$$c_j = \frac{\Sigma_i^N u_{ij}^m \cdot p_i}{\Sigma_i^N u_{ij}^m} \quad \forall j \in C, \tag{4.10}$$

$$u_{ij} = (\sum_{k}^{C} (\frac{||p_i - c_j||}{||p_i - c_k||}))^{-1} \quad \forall i \in N, j \in C. \tag{4.11}$$

Note that computation of the updated probability u_{ij} is necessary for the minimization of the objective function J_m [Bezdek 2984]. The complete algorithm is shown in Fig. 4.4. The algorithm will eventually converge to a minimum objective function, under the condition that the change in J_m is below some threshold ξ. The computational complexity of C-means clustering algorithm is O(ndc^{2}i), where n is the number of data point, d is the number of features, c is the number of clusters, and i is the number of calculation iterations [Cai 2007].

The effectiveness of the clustering method depends on the choice of the number of clusters. If we select too few clusters, we may not cover all the segments in the design. If we select too many clusters, we may exceed the upper limit on the number of CP monitors. We determine the number of clusters by the monitoring resources, e.g. number of sensors.

To illustrate the C-means clustering method, we again use the CP set P_0. If we use 5 clusters, reflected as 5 selected representative paths, the membership Matrix U as obtained using the algorithm of Fig. 4.4 is shown

Algorithm 2: C-means clustering method

Require: P, N, k, ξ
Ensure: U, C
1: Initialize $U = \{u_{ij}\}$ matrix;
2: Initialize $J_m \leftarrow$ very large value;
3: **repeat**
4: $J_{old} \leftarrow J_m$;
5: **for** each $1 < j < k$ **do**
6: update $C = \{c_j | c_j = \frac{\sum_i^N u_{ij}^m \cdot p_i}{\sum_i^N u_{ij}^m}\}$;
7: update $U = \{u_{ij} | u_{ij} = (\sum_k (\frac{||p_i - c_j||}{||p_i - c_k||}))^{-1}\}$;
8: **end for**
9: $J_m = \sum_i^N \sum_j^C u_{ij}^m ||p_i - c_j||$;
10: **until** $||J_m - J_{old}|| < \xi$
11: **return** U and C

Figure 4.4: Procedure for the C-means clustering method.

below:

$$U = \begin{bmatrix} 0.32 & 0.11 & 0.00 & 0.00 & 0.98 & 0.20 & 0.01 & 0.05 \\ 0.03 & 0.40 & 0.00 & 0.98 & 0.01 & 0.20 & 0.01 & 0.04 \\ 0.03 & 0.14 & 1.00 & 0.00 & 0.00 & 0.20 & 0.00 & 0.05 \\ 0.02 & 0.18 & 0.00 & 0.00 & 0.01 & 0.30 & 0.97 & 0.03 \\ 0.87 & 0.21 & 0.00 & 0.00 & 0.00 & 0.10 & 0.01 & 0.82 \end{bmatrix},$$

where each column corresponds to the membership of a CP to each cluster. Here, RCP set $P_R = \{p_1, p_3, p_4, p_5, p_7\}$ is selected, as these paths have the highest scores in each cluster. The delays of the remaining CPs are calculated using Equation (4.3) as $d_2 = 6.9, d_6 = 7.8$, and $d_8 = 11.8$. Thus the $rRMSE$ is 5%, which is low, but comparatively higher than the $rRMSE$ obtained using SVD-QRcp with the same number of RCPs.

4.4.3 Adaptive Method

Aging effects lead to increased delay on functional paths. However, the delay increase in one path differs from another path due to potentially different stress on each segment in these paths. To account for path-delay change over time, we propose an update mechanism to reduce the mismatch in

aging-induced delay. At design time, we leverage the SVD-QRcp method (Algorithm 4.3) to generate a base representative path set P_R. We place delay sensors on all paths in P_R. At run-time, in addition to P_R, we dynamically monitor a set of additional paths P_A using path delay testing. The selection of P_A depends on the resource budget available for monitoring, which also determines the number of clusters that can be utilized. The set P_A is determined using the C-means clustering algorithm, as shown in Fig. 4.4. The complete set of monitoring paths $P' = P_R \cup P_A$ can thus be generated, as shown in Fig. 4.5,

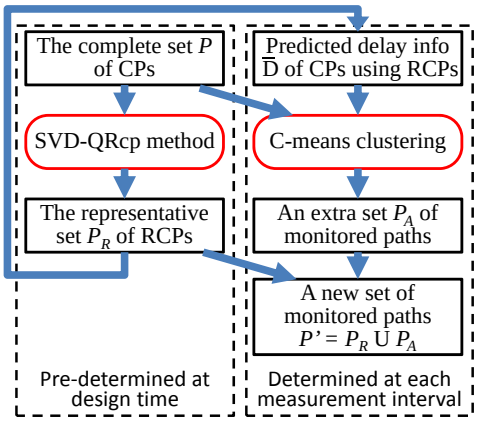


Figure 4.5: Algorithm for selecting RCPs using a combination of SVD-QRcp and C-means clustering for runtime monitoring.

When we determine the RCP set for monitoring chip performance, we use the delay prediction mechanism shown in Fig. 4.6. First, we measure the delays D_R in the RCP set P_R to get the base predicted delay set $\overline{D}$. Based on the predicted delay set $\overline{D}$ and chip topological features, we cluster P in order to select the extra path set P_A. The measured delays in the extra paths D_A are then compared to the predicted delay $\overline{D_A}$. We can thus obtain the offset $\Delta(D_A) = D_A - \overline{D_A}$ to estimate $\Delta(D)$ for all CPs in P. The eventual prediction model in each interval can thus be updated to account for the prediction errors.

To illustrate the effectiveness of the adaptive method, we revisit our previous example. Assume that T_0 is the delay contribution vector at $t = 0$. If we consider aging, the delay contribution vector changes at $t > 0$. Assume that T_1 is the delay contribution vector at some point $t = t_i$ during system

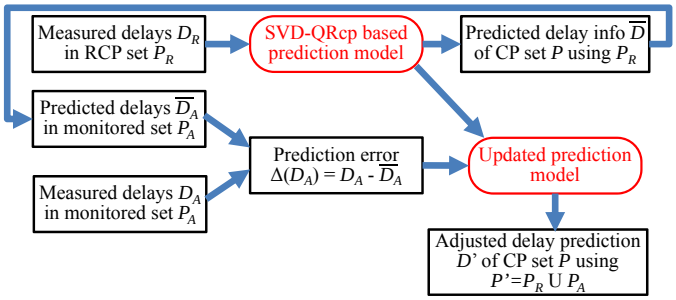


Figure 4.6: Adaptive delay prediction mechanism.

runtime, as shown below:

$$T_1 = \begin{bmatrix} 4 & 6 & 7 & 5 & 7 & 4 & 9 & 5 & 6 \end{bmatrix}^T. \tag{4.12}$$

When we use a set of 5 RCPs out of 8 CPs and the algorithm SVD-QRcp method alone, the $rRMSE$ is 1.2% at $t = 0$. When $t = t_i$, the delay set $D_1 = P_0 \cdot T_1$ is $\{d_1 = 33, d_2 = 28, d_3 = 29, d_4 = 15, d_5 = 29, d_6 = 26, d_7 = 21,$ and $d_8 = 38\}$. Based on the measured delays of RCP set $P_R = \{p_2, p_3, p_5, p_6, p_8\}$, the predicted delays of the remaining 3 CPs are $d'_1 = 29.1, d'_4 = 12.9,$ and $d'_7 = 19.2$. Therefore, the $rRMSE$ increases to 5.8%, which is much larger than the prediction accuracy obtained at $t = 0$.

The prediction errors can be mitigated using the adaptive method proposed in Fig. 4.6. We form a set of 5 RCPs, which consists of 3 fixed RCPs (using SVD-QRcp method) and 2 dynamic RCPs (using C-means clustering method). The RCP set P_R selected by SVD-QRcp is $P_R = \{p_2, p_5, p_8\}$. At $t = 0$, the other 2 dynamic RCPs are selected based on C-means method using a matrix P'_0, as follow:

$$P'_0 = [P_0|\overline{D}] = \left[\begin{array}{ccccccccc:c} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 10.9 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 8 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 7.2 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 3.2 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 8 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 7.6 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 4.3 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 13 \end{array}\right], \tag{4.13}$$

where $\overline{D_0}$ consists of the measured and the SVD-QRcp-based predicted delays. The additional RCP set P_A is selected to be $\{p_1, p_6\}$ and the entire

RCP set is $P' = P_R \cup P_A = \{p_1, p_2, p_5, p_6, p_8\}$ for $t = 0$. Comparing the measured delays of P_A and corresponding predicted delays obtained above, we can obtain the prediction errors $\Delta d'_1 = -0.1$ and $\Delta d'_6 = -1.4$. The delay compensation can thus be calculated for the remaining 3 CPs using Equation (4.3), such that $\Delta d_3 = -1.3, \Delta d_4 = -0.9$, and $\Delta' d_7 = -0.5$. Therefore the complete prediction delay set $\overline{\overline{D_0}} = \{d''_1 = 11, d''_2 = 8, d''_3 = 8.5, d''_4 = 4.1, d''_5 = 8, d''_6 = 9, d''_7 = 4.8, d''_8 = 13\}$, and $rRMSE$ is compensated to be 0.9%, which is less than using either SVD-QRcp or C-means clustering alone.

At $t = t_i$, similar compensation also applies to the delay prediction. We select the RCP set to be $P' = \{p_2, p_5, p_8\} \cup \{U_6, U_7\}$. Note that we select a different P_A because the predicted delay using SVD-QRcp is different at $t = t_i$. The predicted delays based on the adaptive method is then $\overline{\overline{D_1}} = \{d''_1 = 30.1, d''_2 = 28, d''_3 = 30.4, d''_4 = 13.9, d''_5 = 29, d''_6 = 26, d''_7 = 21$, and $d''_8 = 38\}$. The $rRMSE$ is 4.9%, which is better than the SVD-QRcp method only.

4.5 Experimental Results

4.5.1 Experimental Setup

Experiments are performed on several IWLS'05 and ITC'99 benchmark circuits [78, 94] to evaluate the efficiency and accuracy of the proposed methodology. Circuits are synthesized using Synopsys Design Compiler and mapped to the Nangate 45 nm library [80]. The extracted netlists are placed and routed using Cadence SOC Encounter. Learning algorithms are implemented using the Matlab 2011b statistics toolbox. Experiments are run on a 64 bit Linux systems with 12 GB of RAM and quad-core Intel i7 processors running at 2.67 GHz.

4.5.2 Effectiveness of RCPs

We first evaluate the effectiveness of delay prediction when aging-aware features are considered. For a system with a large number of paths (millions or more), we select only the top 5% of critical paths to form a targeted CP set based on corresponding timing slacks. We use the SVD-QRcp method

Table 4.1: Information about ITC'99 and IWLS'05 benchmark designs.

	b17	b18	b19	b22	RISC
# of gates	27k	88k	185k	40k	61k
# of gate-type features	54	54	54	54	54
# of temperature features	100	100	100	100	100
# of voltage features	400	400	400	400	400
# of process -variation features	400	400	400	400	400
# of critical paths	1021	604	524	722	1562
# of RCPs[1]	33	18	19	18	38

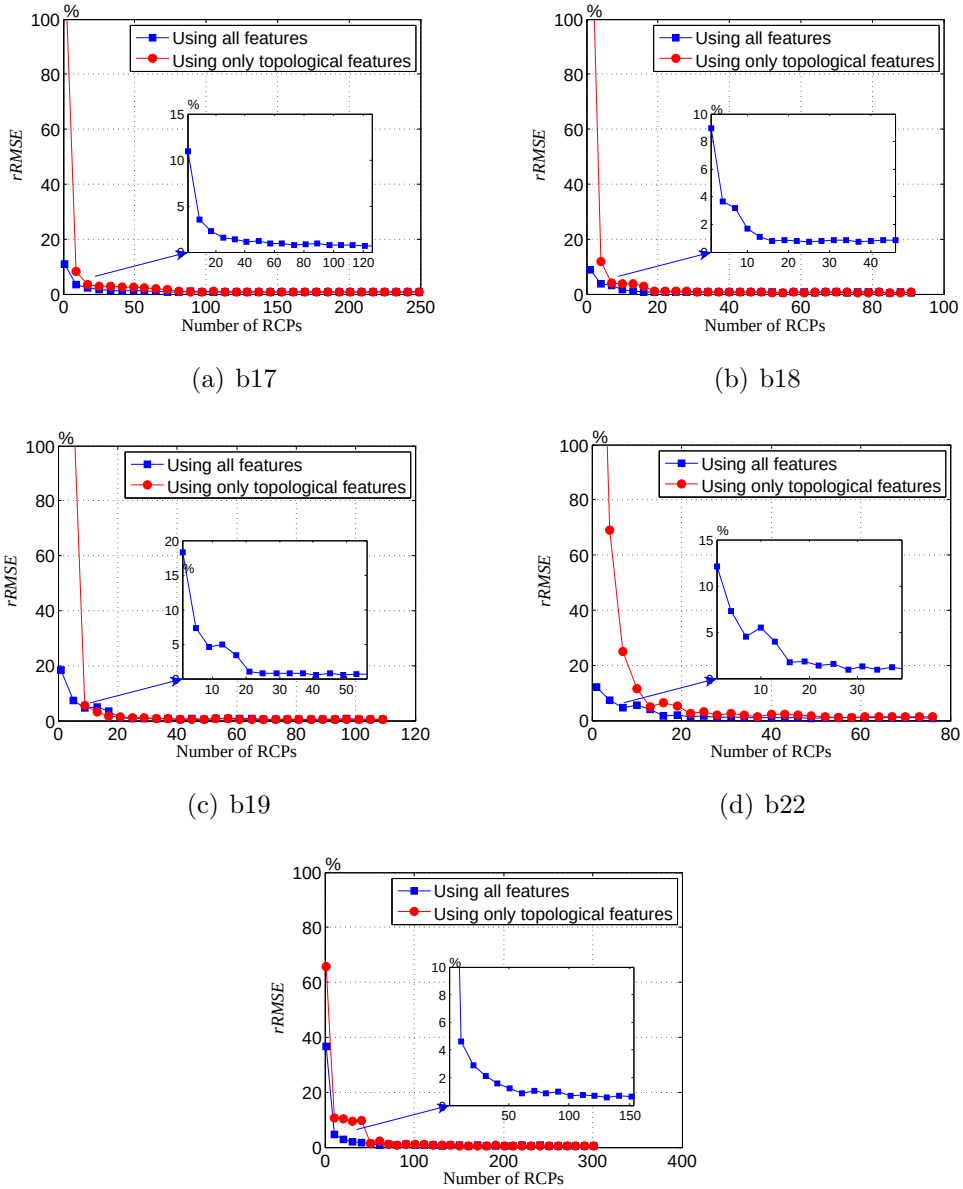
[1]The number of RCPs obtained when required timing accuracy is set to higher than 97%.

(Fig. 4.3) to select the RCP set from the entire CP set. Prediction accuracy is evaluated based on $rRMSE$ the metric, defined in Equation (4.4). Table 4.1 lists the total number of CPs, the optimal number of RCPs for different benchmark circuits when the required timing accuracy is set to be higher than 95%. Note that for different circuits, the numbers of selected RCPs are different based on the calculation of P_{ex} and the total number of CPs. According to this table, we observe that the number of RCPs is significanlty smaller than number of CPs. For example, we select 35 RCPs out of a total of 3021 CPs for b17, and 46 RCPs out of a total of 3662 CPs for RISC processor. These results show that with only a small number of RCPs, we can predict the delays of a large set of CPs with high accuracy.

The prediction accuracy for six benchmarks at measurement point t_{3y} (the third year in system runtime) is plotted in Fig. 4.7. First, we observe that $rRMSE$ drops fast when the number of RCP increases. For example in b22, if we take all features into account for delay prediction, the $rRMSE$ obtained by using only 5 RCPs is 10.2%, while the $rRMSE$ obtained for 18 RCPs is only 1.3%. Second, the $rRMSE$ is found to remain constant when the number of RCPs is larger than 25 for b22. The results show that there is a clear knee in the graphs for all circuits, which indicates that increasing the number of RCPs beyond a certain point does not have a significant impact on accuracy. Therefore, we are able to achieve high prediction accuracy for a large pool of target CP set by monitoring only a few paths and using Equation

(4.3). Note that we can exploit P_{ex} using Equation (4.6) to determine the knee point effectively. Third, we observe that $rRMSE$ drops faster when we use a more detailed model that considers all features than when we use a simple model that includes only topological features. These results highlight the effectiveness of the use of aging- and variation-aware features for delay prediction, as described in Section 4.3.

Note that our method is of more general use than [83], whereby we can predict delay of every critical path in the circuit, in contrast to the overall delay of the entire circuit. Moreover, we consider more complete features such as process variations, voltage droop, and temperature to achieve more accurate results. As depicted in Fig. 4.7, by considering all the features, the accuracy is improved by 17.6% on average if we use the same number of RCPs.



(a) b17

(b) b18

(c) b19

(d) b22

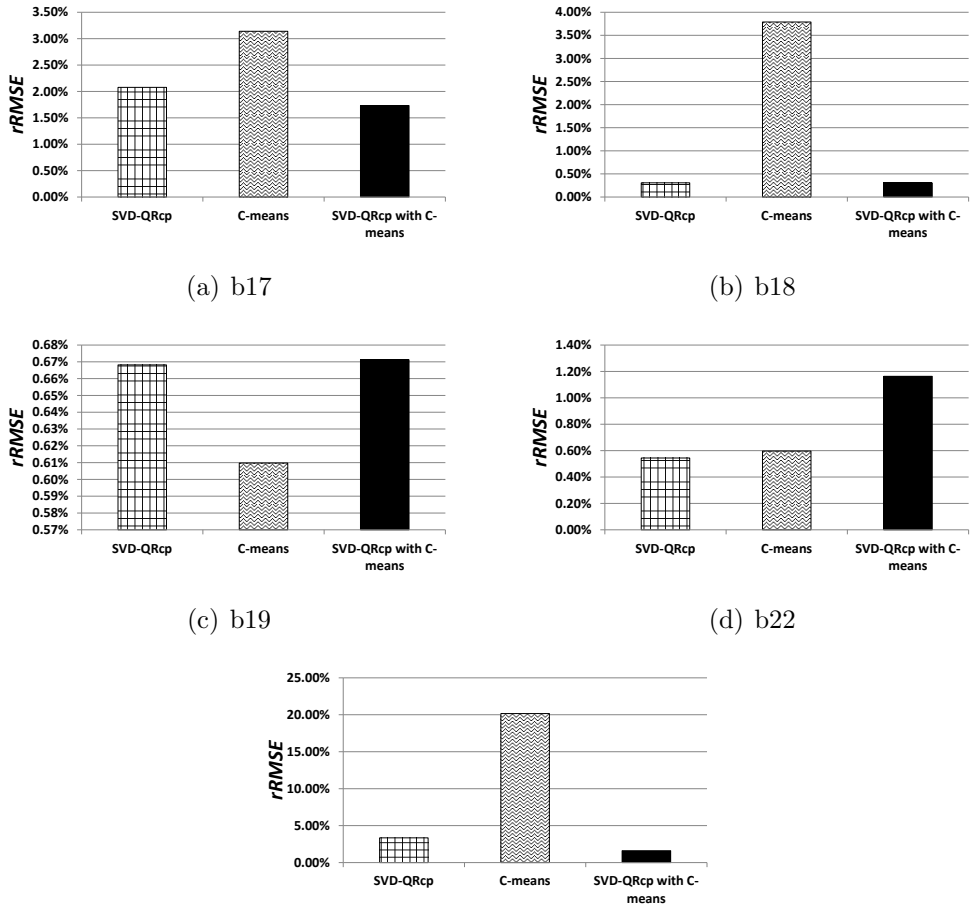
(e) RISC

Figure 4.7: Prediction accuracy obtained 1) using all features and 2) using only topological feature at t_{3y} for ITC'99 and ISWL'05 benchmark circuits.

Next, we illustrate the effectiveness of using adaptive prediction method (i.e., SVD-QRcp with Cmeans) for six benchmark circuits under different degree of variations. For this purpose we consider two scenarios: 1) RCPs are extracted based on only topological feature, 2) RCPs are extracted by considering both topological feature and additional variation features. The average prediction accuracies of first and second scenarios are shown in Fig.

4.8 and Fig. 4.9, respectively. The adaptive prediction method (i.e., SVD-QRcp with C-means) is a combination of SVD-QRcp and C-means clustering, as described in Fig. 4.5 and Fig. 4.6, respectively. We compare the prediction accuracy obtained using same number of RCPs selected by the adaptive method to two static path-selection methods, namely SVD-QRcp (Fig. 4.3) and C-means clustering (Fig. 4.4). The number of RCPs selected by the C-means clustering method equals the number of clusters. In addition, we have implemented an iterative clustering method based on [95], thereby selecting the clustering setting with high prediction accuracy. The prediction accuracy is the average of multiple measurement points during the time. In addition, for both scenarios, we run Monte-Carlo simulation and in each iteration, physical characteristics, voltage, and temperature of each gate are updated according to the corresponding variation model. Comparison of Fig. 4.8 and Fig. 4.9 indicates that when RCPs are extracted by considering all features, prediction error is significantly reduced under variations. Moreover, we observe higher prediction accuracy when we use adaptive prediction method, compared to the other two static methods. For example in b17 as shown in Fig. 4.9, $rRMSE$ is 1% if we use the adaptive prediction, while $rRMSE$ is 1.8% if we use SVD-QRcp and 1.6% if we use C-means clustering.

Finally, in Fig. 4.10, we present the prediction accuracy trends during runtime to analyze the effectiveness of RCPs under aging effects. In almost all cases, where each case corresponds to a circuit and a prediction point, the dynamic method (SVD-QRcp+C-means) offers higher accuracy in comparison to the two static methods.



(a) b17

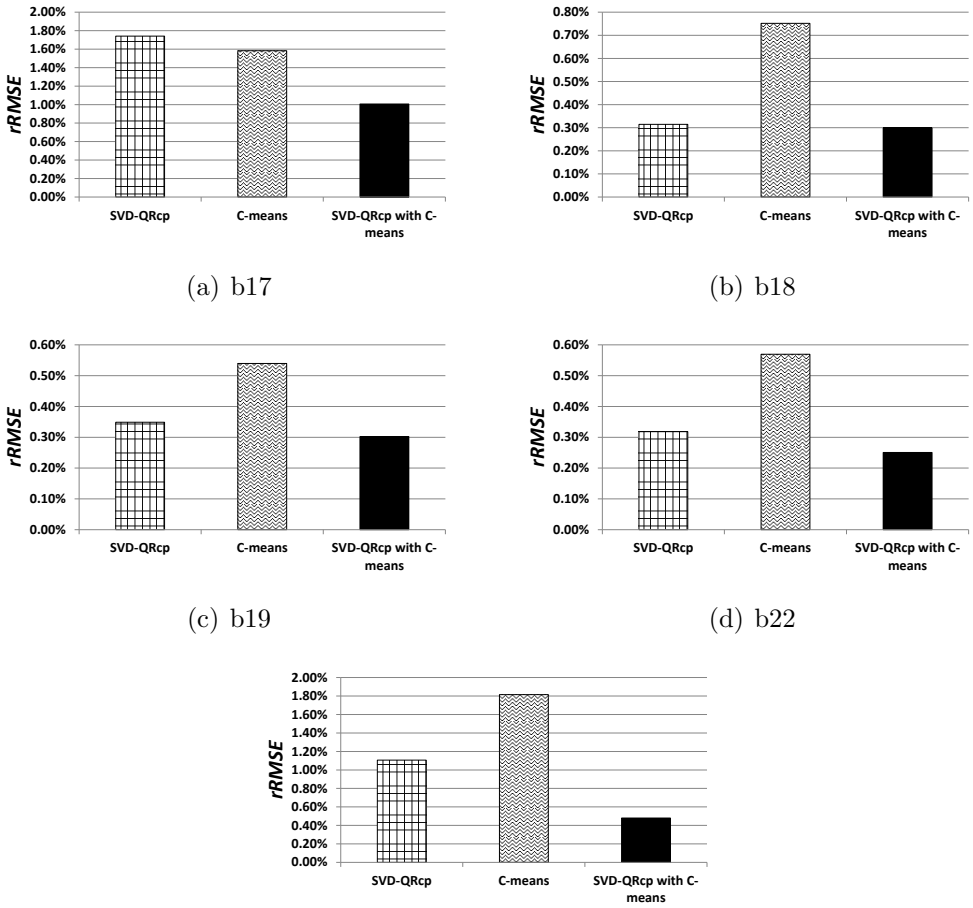
(b) b18

(c) b19

(d) b22

(e) RISC

Figure 4.8: Comparison of average prediction accuracy between different RCP selection methods (using only topological features) for ITC'99 and IWLS'05 benchmark circuits.



(a) b17

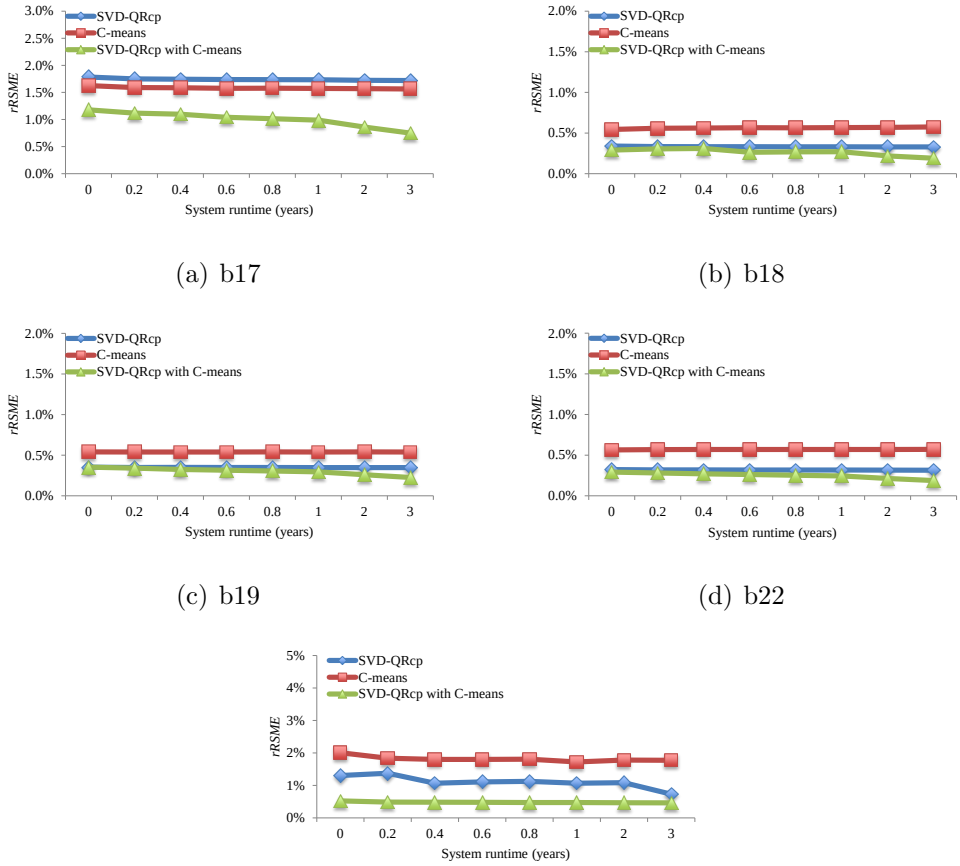
(b) b18

(c) b19

(d) b22

(e) RISC

Figure 4.9: Comparison of average prediction accuracy between different RCP selection methods (using all features) for ITC'99 and IWLS'05 benchmark circuits.



(a) b17

(b) b18

(c) b19

(d) b22

(e) RISC

Figure 4.10: Comparison of runtime prediction accuracy between different RCP selection methods for ITC'99 and IWLS'05 benchmark circuits.

4.5.3 Monitoring Sensors

For runtime adaptation, either delay testing or sensors must be implemented to collect data on delay changes due to parameter variations of the CPs. Various in-situ delay sensors have been proposed in the literature [[96, 6]; in this work, we use a sensor similar to the one presented in [6]. As shown in Fig. 4.11, this sensor consists of a latch, two inverters, and three NAND gates that are inserted at the end point of each CP. As discussed in [6], the area overhead of this sensor is only 22 transistors. This sensor detects late transitions on functional CPs and generates pulses. The widths of these pulses represent the timing margins of the CPs. Next, the measured timing margin is converted to a digital value using a *measurement unit* that consists

of two multiplexers, a NAND gate, a ring oscillator, an N-bit counter, and a LUT. The details of the sensor design are discussed in [6]. Note that only one measurement unit is used for all sensors. Table 4.2 shows the area and power overheads of the monitoring sensors for RCP monitoring. Note that we use the same number of RCPs in SVD-QRcp with Cmeans method, SVD-QRcp method, and the C-means clustering method, thereby the overheads of all these three methods are same. Results obtained using Synopsys Design Compiler shows that the overhead decrease when we increase the size of the circuit.

Table 4.2: Overhead due to the monitoring sensors.

Benchmark	No.of gates	Sensor Overhead	
		Area	Power
b17	27K	2%	0.9%
b18	88K	1.1%	0.3%
b19	185K	0.5%	0.1%
b22	40K	1.4%	0.6%
RISC	61K	1.0%	0.4%

4.5.4 Error in Delay Sensors

Next, we evaluate the robustness of the proposed adaptive method with the presence of inaccurate data provided by the delay sensors. We use Gaussian distribution to inject errors in the readouts from the delay sensors. For example, 1% error means that delay value read from the sensor has a mean value of the actual delay and a deviation of 1%. In Fig. 4.12, the readout errors are 0%, 3%, 6% and 10%; these values are delibrately set to be larger than what has typically been reported in the literature [6]. Mento Carlo (MC) simulation is used for evaluation. We use 40 trials in the MC method, since the delay-prediction error does not change much as we increase the number of trials beyond 40. We observe that the delay-prediction error increases gradually as the reading error increases in the sensor. The delay-prediction accuracy depends on the sensor reading accuracy. Nevertheless, the prediction error remains insignificant in most cases. There are also several existing calibration techniques to tackle the detrimental effects of variations on delay sensors [6], however this is out of the scope of this thesis. Moreover, delay testing can also be combined by in-situ based RCP monitoring to improve the accuracy.

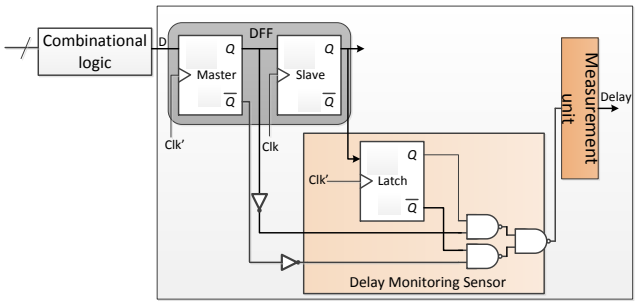
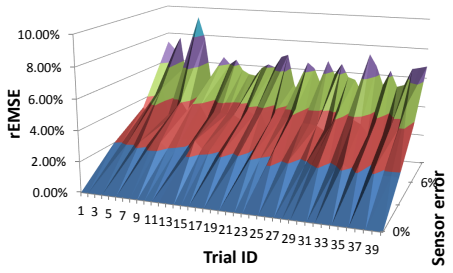
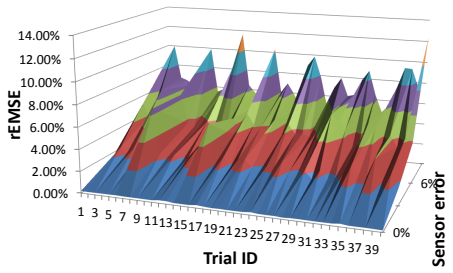


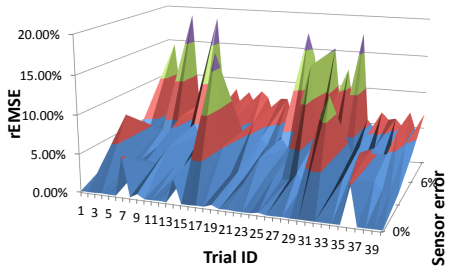
Figure 4.11: A design of one in-field variation-aware delay sensor. Adopted from [6].



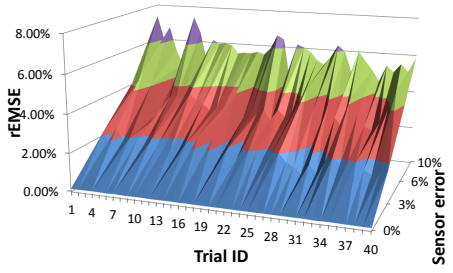
(a) b17



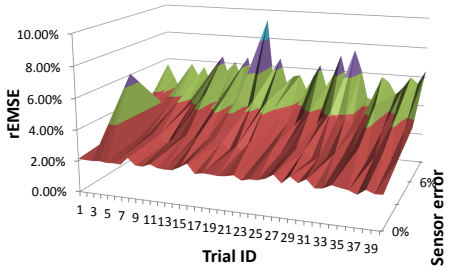
(b) b18



(c) b19



(d) b22



(e) RISC

Figure 4.12: Effects of inaccuracies in delay-sensor readouts on the accuracy of delay prediction.

4.6 Conclusions

The complexity associated with advanced technology nodes requires the monitoring of a large pool of critical paths to ensure desired performance of a chip over its lifetime. A small set of representative critical paths are usually adopted as a surrogate to estimate delays for the complete set of critical paths. However, uncertainties introduced by process and runtime variations and aging reduce prediction accuracy when a small of representative critical paths is used. As a result, system adaptation effectiveness and resilience are adversely affected. In this chapter, we have shown how reasoning methods based on machine-learning can be used to account for uncertainties in chip parameters. Simulation results for a range of benchmark circuits highlight the efficiency of the proposed techniques for predicting critical path delays in the presence of parameter variations.

CHAPTER 5

MITIGATION TECHNIQUES

In previous chapters, modeling of parameter variations as well as in-field delay/age monitoring systems are addressed. This chapter presents different but yet complementary static and adaptive techniques to mitigate the detrimental impact of parameter variations. These techniques that are developed on top of our proposed timing analysis framework and monitoring systems can significantly improve the state-of-the-art techniques by extending the lifetime of the chips and by increasing the circuit frequency while satisfying the power constraints. Our static methods such as guard-banding are based on model, predict, and margin. On the other hand, adaptive methods can consider workload-dependent time-varying characteristics of the circuit based on a sense and adapt strategy.

5.1 State-of-the-arts

There has been considerable research work for alleviating the variations effect on digital chips at different design levels [97]. The common approach to combat parameter variations is adding timing margin during design time. Although the complexity of this method is very low, it significantly suffers from performance loss. Various gate and transistor sizing are proposed in literature to compensate the impact of variations [98, 99, 100]. The overhead of these design-time techniques are medium, but their efficiency can be very low depending on the working conditions. Recently, high level synthesis techniques are also adjusted to consider parameter variations [101]. Input vector control and internal node control are another design-time approaches to tackle transistor aging. The basic idea is to apply a specific input vector during idle-time in order to maximize recovery time of most critical gates [102, 7, 103, 104, 105, 106]. Adaptive voltage and threshold voltage scaling can be used during runtime to compensate variations [107, 108, 109, 110, 111, 112]. Power gating of circuits during their idle-time is an efficient way to reduce transistor aging and leakage power [113, 114, 115, 116]. Degradation rate balancing try to balance the workload and idle time over all logic blocks (e.g., ALU) in a system. Adaptive re-indexing of cache modules, instruction scheduling, and task scheduling fall into this category [117, 118, 119].

5.2 Static Input Vector Control (Static-IVC)

Input vector affects both NBTI and leakage power, but not in the same direction [120]. In other words, the best input vector resulting in minimum aging might not lead to minimum leakage power. It implies that, a set of Pareto points has to be extracted and afterwards during the standby mode at runtime, based on the system conditions and requirements, a suitable input vector can be selected and applied to the circuit. In this section we describe our proposed *Linear Programming* (LP) based method for co-optimization of NBTI and leakage power by obtaining an optimal input vector to apply to the combinational circuit during the standby mode.

5.2.1 NBTI Minimization

Here we explain the logic network and also the NBTI-induced gate delay increase relations by LP constraints. Linear programming is an efficient mathematical optimization approach consisting of an objective which needs to be optimized, and a set of linear constraints in a specific format as follows:

$$Minimize\ C^T x, \quad subject\ to\ Ax \leq b, \tag{5.1}$$

where x represents a vector of optimization (controlling) variables, C and b are vectors of coefficients, and A is a matrix of coefficients. In this work, our objective is to minimize the overall circuit delay increase due to NBTI by considering (i.e. taking a maximum over) post-aging delay in all critical and near-critical paths. For each path, the path delay increase is the sum of gate delay increase for all the gates along that path. The result of this LP minimization gives us the minimal post-aging circuit delay as well as the input vector corresponding to the minimal circuit delay increase. This input vector can be used during standby mode.

In fact, each input combination for a given gate in the library leads to a different NBTI-induced delay increase in the standby mode. We exploit a pseudo-Boolean function to formulate such NBTI effect for different gates in the LP compatible format. For instance, considering a NAND gate, we can

write the function corresponding to NBTI-induced delay increase as:

$$\begin{aligned} object\ function &= \Delta delay \\ &= D_{00}\bar{a}\bar{b} + D_{01}\bar{a}b + D_{10}a\bar{b} + D_{11}ab \\ &= D_{00}(1-a)(1-b) + D_{01}(1-a)b \\ &+ D_{10}a(1-b) + D_{11}ab, \end{aligned} \tag{5.2}$$

where a and b are the inputs of the gate, $a, b \in 0, 1$, and D_{ab} indicates delay change due to the NBTI effect corresponding to gate inputs ab. D_{ab} can be extracted from NBTI model using precise HSPICE simulations. By applying the Boole-Shannon expansion we reach:

$$\begin{aligned} \Delta delay &= (D_{00} - D_{01} - D_{10} + D_{11})ab \\ &+ (D_{10} - D_{00})a + (D_{01} - D_{00})b + D_{00}. \end{aligned} \tag{5.3}$$

In order to express the object function in LP format, it has to be linearized. Since in a NAND gate, output(c)$= 1 - (ab)$, the above equation can be rewritten as

$$\begin{aligned} \Delta delay &= (D_{00} - D_{01} - D_{10} + D_{11})(1-c) \\ &+ (D_{10} - D_{00})a + (D_{01} - D_{00})b + D_{00}. \end{aligned} \tag{5.4}$$

With the same approach the object function of the NOR and NOT gates can be extracted, as shown in Table5.1.

Table 5.1: LP object functions for gate Δdelays.

Function	Logic operation	Object function
INV	$b = NOT(a)$	$D_0 b + D_1 a$
NAND	$c = NAND(a, b)$	$(D_{10} - D_{00})a +$ $(D_{01} - D_{00})b +$ $(D_{10} + D_{01} - D_{00} - D_{11})c$ $+(2D_{00} + D_{11} - D_{01} - D_{10})$
NOR	$c = NOR(a, b)$	$(D_{11} - D_{01})a +$ $(D_{11} - D_{10})b +$ $(D_{11} + D_{00} - D_{10} - D_{01})c$ $+(D_{10} + D_{01} - D_{11})$

Next, a set of linear constraints are required to represent the functionality

of the logic gates. This would be the LP representation of the logic network (gate-level netlist). There exist two sets of such constraints to represent the functionality of logic gate [121, 122]. Table 5.2 illustrates the set of constraints based on these two models for basic logic gates.

Table 5.2: LP constraints for basic logic operations.

Function	Logic operation	Constraints Form I [121]	Constraints Form II [122]
INV	$b = NOT(a)$	$b + a = 1$	$b + a = 1$
NAND	$c = NAND(a, b)$	$c \leq 2 - (a + b + 1)/2$ $c \geq 1 - (a + b)/2$	$c \leq 2 - a - b$ $c \geq 1 - a$ $c \geq 1 - b$
NOR	$c = NOR(a, b)$	$c \leq 1 - (a + b)/2$ $c \geq 1 - (a + b)$	$c \geq 1 - (a + b)$ $c \leq 1 - b$ $c \leq 1 - a$

As mentioned before, the optimization objective is to minimize the overall circuit delay increase due to NBTI. To accurately take this into account, post-aging delays of all critical and near-critical paths have to be considered. It should be noted that the NBTI-induced delay increase of a near-critical path p_i could be more than that of a critical path p_j ($D(p_i) < D(p_j)$) such that $D(p_i) + \Delta D(p_i) > D(p_j) + \Delta D(p_j)$. The list of all critical and near-critical paths (i.e. the *vulnerable* paths) can be extracted from static timing analysis of the circuit by setting a threshold for the slack of such paths. The goal is to optimize (minimize) the post-aging delays of the vulnerable paths of the circuit. The NBTI-aware Object Function (NOF) can be expressed by Equation 5.5:

$$minimize : NOF = \max_{j=0}^{N} \sum_{\forall\ g_{ji}\ in\ VP_j} \Big(D(g_{ji}) + \Delta D(g_{ji}) \Big), \tag{5.5}$$

where j is a vulnerable path, N is number of vulnerable path, g_{ji} is gate i in the vulnerable path VP_j.

To linearize the "max" operation, we replace Equation 5.5 by a set of constraints as follows.

$$\forall j: \; x \geq D(VP_j) = \sum_{\forall \; g_{ji} \; in \; VP_j} \Big(D(g_{ji}) + \Delta D(g_{ji}) \Big) \tag{5.6}$$
$$minimize : x$$

PBTI is becoming an important transistor aging factor by the introduction of high-κ/metal gate transistors. This phenomenon affects NMOS transistors in the similar way which NBTI affects PMOS transistors. As a result the same equation as NBTI can be used to estimate the threshold shift due to PBTI. Therefore, similar LP formulations can be exploited as described above to find best input vector in terms of PBTI.

5.2.2 Leakage Power Minimization

Since leakage power of each gate in the cell library is strongly dependent of its input vectors, the total leakage power of the circuit is a function of its primary input vector. Here, similar to [121][122], we describe a linear programming based method to find the best input vector which results in minimum leakage power of the circuit. The overall methodology is similar to that presented in Sec. IV.A. Here, a pseudo-Boolean function is also used to formulate leakage power for different gates in the LP compatible format. The object function for different gates is similar to the table 5.1. However, D_{ab} should be replaced with P_{ab}. , where P_{ab} indicates the leakage power corresponding to gate inputs ab which can be extracted form a look-up table. The total leakage power can be obtained by simple adding the leakage power of all individual gates in the circuits. As a result, the Power-aware Object function (POF) can be written by the following equation:

$$minimize : POF = \sum_{i=0}^{N} P(g_i), \tag{5.7}$$

where N is the number of gates of the circuit.

5.2.3 NBTI and Leakage Co-optimization

Finally, for co-optimization of NBTI and power, three different objectives can be defined.

- In Aging-constraint leakage minimization (e.g. in high performance or high reliability applications), the post-aging delay has to be less than a certain threshold and power should be minimized.

$$\begin{aligned} minimize : POF_{total} \\ NOF_{total} \leq D_{target} \end{aligned} \tag{5.8}$$

- In Leakage-constraint aging minimization (e.g. for low power applications), the power has to be less than a certain value and post aging delay should be minimized.

$$\begin{aligned} minimize : NOF_{total} \\ POF_{total} \leq P_{target} \end{aligned} \tag{5.9}$$

- Another approach is to minimize the combination object function.

$$\begin{aligned} minimize : \alpha.POF_{total} + \beta.NOF_{total} \\ \alpha + \beta = 1 \end{aligned} \tag{5.10}$$

  where α and β are pre-determined constants based on the importance of the power and/or aging for a given application.

The proposed methodology for co-optimization of each mentioned objective is summarized in Figure 5.1. Our proposed timing analysis tool is used to extract the vulnerable paths of the circuit. Next, the generated gate level netlist of the circuit is given to a logic simulator to calculate signal probabilities of all internal nodes. This signal probability or duty cycle is used as an input in the gate-level NBTI models to estimate delay changes due to different input vectors for each of the gates of the circuit. Besides, each cell in the technology library is characterized regarding the leakage power. This information is stored in a look-up table to be used in the next step for generating the LP formulation. Afterwords, NBTI-Power-aware LP

generator takes the calculated NBTI coefficient of each gate, obtained from the NBTI model and the extracted look-up table for leakage power, as well as the netlist of the circuit and generates an LP model. Finally, an LP-solver is exploited to solve the generated LP constraints and find the optimal input vector which co-optimize the NBTI and leakage power. A Monte-Carlo simulation is used for evaluating the proposed methodology in terms of accuracy and runtime.

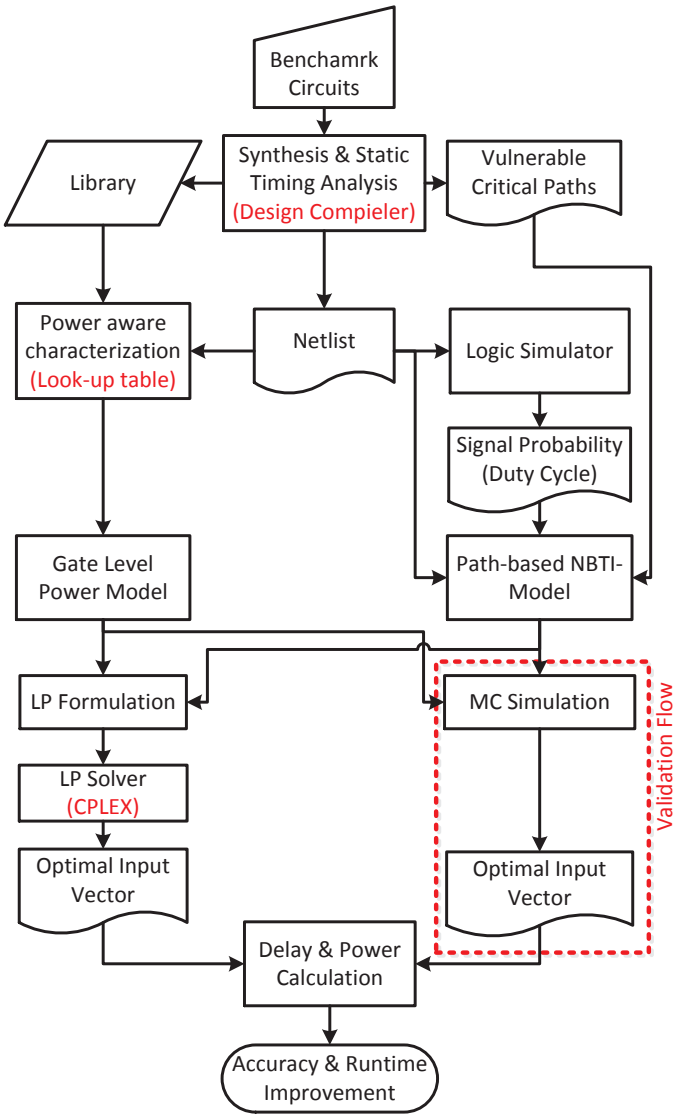


Figure 5.1: Flowchart of the proposed power-aware minimum NBTI input vector selection.

5.2.4 Running Example

To illustrate the presented flow, let us consider the circuit depicted in Figure 5.2.

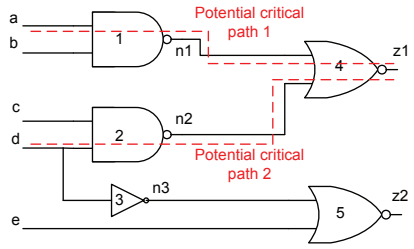


Figure 5.2: An example circuit for LP formulation.

For finding the NBTI-aware object function, we assume the circuit has two potential critical (vulnerable) paths, Path1 and Path2. Path1 consists of gates g_1 and g_4. The NBTI-aware object function (NOF) of the Path1 is written as:

$$
\begin{aligned}
&NOF_{Path1} = NOF_{g_1} + NOF_{g_4} = \\
&\Big(a(D_{10} - D_{00}) + b(D_{01} - D_{00}) + n1(D_{10} + D_{01} - D_{11} - D_{00}) \\
&+2D_{00} + D_{11} - D_{10} - D_{01}\Big) + \Big(n1(D_{11} - D_{01}) + \\
&n2(D_{11} - D_{10}) + z1(D_{11} + D_{00} - D_{10} - D_{01}) + \\
&D_{10} + D_{01} - D_{11}\Big)
\end{aligned}
$$

Path2 consists of gates g_2 and g_4. The object function of the Path2 is calculated as follows:

$$
\begin{aligned}
&NOF_{Path2} = NOF_{g_2} + NOF_{g_4} = \\
&\Big(c(D_{10} - D_{00}) + d(D_{01} - D_{00}) + n2(D_{10} + D_{01} - D_{11} - D_{00}) \\
&+2D_{00} + D_{11} - D_{10} - D_{01}\Big) + \Big(n1(D_{11} - D_{01}) + \\
&n2(D_{11} - D_{10}) + z1(D_{11} + D_{00} - D_{10} - D_{01}) + \\
&D_{10} + D_{01} - D_{11}\Big)
\end{aligned}
$$

The final NBTI-aware object function for minimizing the NBTI effect of

the circuit can be expressed as:

$$NOF_{total} = \max(NOF_{Path1}, NOF_{Path2})$$

To represent the above constraint in a linear format, it can be replaced by the following constraints.

$$\begin{aligned} &minimize: \ x \\ &subject\ to: \ x \geq NOF_{Path1},\ x \geq NOF_{Path2} \end{aligned}$$

To represent the logic network functionality, the logic constraints of all the gates in forms of Table 5.2 need to be added to the set of constraints.

If our objective is minimizing the power, then the total power of the circuit should be considered. In this example, the Power-aware object function (POF) can be obtained as:

$$\begin{aligned} &POF_{total} = POF_{g_1} + POF_{g_2} + POF_{g_3} + POF_{g_4} + POF_{g_5} = \\ &\Big(a(P_{10} - P_{00}) + b(P_{01} - P_{00}) + n1(P_{10} + P_{01} - P_{11} - P_{00}) \\ &+2P_{00} + P_{11} - P_{10} - P_{01}\Big) + \Big(c(P_{10} - P_{00}) + d(P_{01} - P_{00}) \\ &+n2(P_{10} + P_{01} - P_{11} - P_{00}) + 2P_{00} + P_{11} - P_{10} - P_{01}\Big) + \\ &\Big(d(P_1) + n1(P_0)\Big) + \Big(n1(P_{11} - P_{01}) + n2(P_{11} - P_{10}) + \\ &z1(P_{11} + P_{00} - P_{10} - P_{01}) + P_{10} - P_{01} - P_{11}\Big) + \Big(n3(P_{11} - P_{01}) \\ &+e(P_{11} - P_{10}) + z2(P_{11} + P_{00} - P_{10} - P_{01}) + P_{10} - P_{01} - P_{11}\Big) \end{aligned}$$

5.2.5 Solving LP Constraints

Binary Integer LP (BILP)

In this form of the linear programming, all the controlling variables (here the input vectors and all the internal nodes) can only take values of 0 or 1 [122]. This is consistent with the actual situation for logic circuits where all the nodes are 0 or 1. Solution to the BILP formulations is *exact* (optimum solution). Since the number of binary variables of the BILP model is equal

to the number of circuit nodes, the runtime of BILP solver increases with the number of circuit nodes. The choice of the constraint forms used for logic network representation (Table 5.2) has a strong effect on the runtime of BILP solver. Representing the circuit with constraints of form I (2 constraints for 2-input gates) results in unacceptable runtime. For instance, it takes about 2 hours for C499 circuit from ISCAS85 benchmark. Based on this observation, constraints of form I are infeasible to be applied to large circuits. On the other hand, constraints of form II (3 constrains for each 2-input gate) leads to considerable runtime reduction for BILP solver. For example, it solves the formulation for C499 in just 0.27 seconds. Therefore, we use the logic representation of form II for BILP solution.

Relaxed LP (RLP)

In complete relaxed LP solution, all variables can get real values anywhere in the range between 0 to 1 [121]. This is in contrast with the BILP solution. However, it needs a subsequent step to convert final results to binary values. A commonly used method to assign a binary value to each input is based on random rounding [121]. Based on this method an input with the real value of $0 \leq p \leq 1$ is converted to 0 by the probability of $(1 - p)$ and converted to 1 by the probability of p. Using constraints of form I and II in the relaxed mode improves the LP solver runtime significantly, however, due to this random conversion from real to binary, the optimality of the solution is greatly reduced. In other words, the solutions obtained by RLP are extremely sub-optimal. In our experiments, unlike the BILP mode, we found out that logic representation of form I is more suitable (better accuracy) for the RLP option.

Mixed-Integer LP (MILP)

As described above, BILP provides optimal solutions but its runtime increases as the number of gates and vulnerable paths increases. On the other hand, RLP has a more reasonable runtime at the expense of providing inaccurate results. To have the best of two worlds, Mixed Integer LP (MILP) can be used. The key idea is to force some selective variables to take only binary values and let the other variables to be real (relaxed). By this approach, we

take advantage of the accuracy of BILP and fast runtime of RLP. For this purpose we exploit constraints in form II (Table 5.2) because it is the only set of constraints which guarantees that if the gate inputs are binary then the output is binary too. According to this property, if only the primary inputs of the circuit have binary values then all the intermediate nodes in the circuit will have binary values even if they are relaxed (i.e. allowed to take real values). Hence, the accuracy of this method would be the same as pure BILP solution. Another major advantage of using this method is that the number of explicit binary variables is reduced to the number of primary inputs. Thus, the LP solver runtime decreases and becomes comparable to RLP runtime. For example for C6288 circuit from ISCAS85 benchmark, the MILP method gives the optimal solution while it is 17X faster in comparison with BILP method.

5.2.6 Experimental Results

We have evaluated the efficiency of the proposed methods using experiments on selected ISCAS'85, ISCAS'89, and MCNC benchmark circuits. We have implemented a Monte Carlo (MC) simulation to obtain the optimal input vector from the random simulation flow. Since, the simulation-based method is very time-consuming, we consider a bound. In other words, if it cannot find the optimal solution and does not improve the accuracy of the obtained solution more than 0.1% after 100,000 iterations, the simulation is terminated and the last obtained result is reported. In addition, we have also implemented the Probability-Based (PB) method proposed in [120]. In each iteration, the random input vectors are generated based on the probability of 0/1 obtained from the best solution of last iteration. For the proposed LP-based technique, we use the flow shown in Figure 5.1. All LP instances are solved using CPLEX, a mixed integer linear programming solver [123]. In this experiments, Ratio of Active to Standby time (RAS) is set to be 3:7. The effectiveness of IVC to mitigate NBTI effect has an opposite relation with RAS. In other words, the higher RAS is, the lower effectiveness of IVC to mitigate NBTI effect is. The gate delays are also extracted from the standard cell library. All vulnerable (critical and near-critical) paths with maximum 5% slack are extracted using PrimeTime static timing analyzer.

Consideration of such paths as the vulnerable paths is consistent with the previous work suggesting near-critical paths are possible to become critical due to NBTI [124].

To compare the results obtained from various LP solutions and MC simulations, we define error factor as $(LP\_opt - MC\_opt)/(MC\_opt)$, where LP_opt and MC_opt are the minimum circuit delay degradation due to NBTI obtained by LP and MC methods, respectively. Table 5.3 compares the error factor and runtime of different LP options (BILP, RLP, MILP), PB method, and the MC simulation. The experiments are conducted on a workstation with Intel Xeon E5540 2.53GHz (2 quad-core processors), 16GB RAM, and the operating system of Windows 7 Enterprise 64bit.

The number of gates and the number of extracted vulnerable paths for each circuit are shown in table 5.3. The fourth column (Δd_{min}) corresponds to the minimum circuit delay increase obtained by MC method. The choice of input vector during the standby mode significantly impacts the delay degradation due to NBTI. The range of delay degradation ($\Delta delayRange = (\Delta d_{max} - \Delta d_{min})/\Delta d_{min}$), obtained by the MC simulation, is shown in the fifth column. According to these results, in average, the delay degradation can vary 50% for a circuit based on the input vector (the range from the best case to the worst case). The next three columns compare the optimization results obtained by various LP options and PB [120] with MC simulations. BILP and MILP have the same accuracy and can improve the optimization by 12% compared to the MC simulation. However, the optimality of the results obtained by RLP is 2% worse compared to the MC simulation. Moreover, PB provides more accurate results than MC simulation, but the results obtained by MILP and BILP are more optimized. As shown in the table, the proposed BILP and MILP methods always have an error less than or equal to zero. This implies that BILP/MILP methods always find better solution than MC simulation because MC simulation cannot necessarily find the best solution due to limited exploration of the search space. On the other hand, since RLP method is not exact, the solution of this method is not optimal and in some cases the solution is even worse than the solution found by MC simulation which leads to positive errors in the table.

In terms of runtime, Monte-Carlo simulation has the highest runtime, followed by PB, BILP, MILP, and RLP. The runtime of PB method is better than MC simulation however it has higher runetime compared to all types

of our proposed LP approaches in average. As shown in this table, runtime has a direct relationship with the number of gates as well as the number of vulnerable paths (VP). All LP-based methods are 4-5 orders of magnitude faster than MC simulation. MC simulation is infeasible for medium to large circuits. MILP provides a runtime balance between RLP and BILP. RLP, despite having the fastest runtime, provides sub-optimal results (about 2% error). While MILP and BILP have the same best accuracy, MILP can further reduce the runtime by 7%, compared to BILP. The runtime of proposed method can be improved by using a technique proposed in [125, 126]. In this technique, the large circuits can be transformed into some trees to improve the runtime of LP approach. This can be done in three steps. First, the circuits is divided into some cicuit trees by a link-deletion algorithm. In this algorithm, the connections between gates are deleted in a way that each gate fans out to at most one other gate. Performing this algorithm results in some dangling inputs which has no fanin gate. Afterwards, The linear programming approach can be performed on each tree to find the best input vector. Finally, new values are assigned to the dangling nodes and this algorithm is performed iteratively until it converges.

Table 5.4 illustrates the minimum and maximum NBTI-induced delay degradation and their corresponding leakage power. It also shows the minimum and maximum leakage power and their corresponding NBTI-induced delay degradation. The purpose of this table is to show how the two dimensional search spaces with respect to the minimum and maximum NBTI as well as leakage is distributed, and how the optimization of one parameter affects the other one. This table reveals that the leakage power corresponding to the input vector with the best NBTI-induced delay degradation is not equal to the minimum leakage power or even worse leakage power. This is also true for the NBTI-induced delay degradation corresponding to input vector with minimum leakage power. This is due to the fact that an input vector does not affect the NBTI and leakage power in the same direction.

Table 5.3: Comparison of proposed linear programming models with accurate Monte-Carlo simulations in terms of NBTI.

Benchmark Circuit	# of gates	# of VPs	Δd_{min} (ps) by MC	$\Delta delay$ Range	Optimization error			Runtime (sec)				
					PB [120]	$B/MILP$	RLP	MC	PB [120]	$BILP$	$MILP$	RLP
C432	176	22,350	31.91	58%	-1%	-5%	-4%	78,925	3,480	3.20	2.15	0.62
C499	533	2,238	29.46	36%	+1%	-7%	13%	6,942	260	0.27	0.33	0.09
C880	415	47	28.18	61%	-4%	-9%	-2%	285	182	0.03	0.06	0.02
C1355	539	1,457	26.57	37%	-1%	-1%	14%	3,944	266	0.17	0.55	0.09
C2670	696	454	33.45	72%	-6%	-19%	6%	1,805	1,825	0.11	0.06	0.03
C3540	868	10,000	43.39	48%	0%	-7%	16%	44,279	2,432	3.92	19.45	0.42
C5315	1,644	402	8.15	157%	+2%	-13%	2%	788	679	1.11	0.16	0.06
C6288	2,770	30,000	120.46	157%	0%	-5%	3%	43,916	900	9,986.73	590.03	11.17
i2	217	429	16.15	28%	0%	-10%	10%	575	21	0.01	0.03	0.02
i3	132	265	8.15	41%	0%	-18%	16%	210	10	0.01	0.01	0.01
i4	220	385	17.47	62%	-1%	-30%	-30%	500	514	0.01	0.01	0.02
i5	198	1,345	12.76	119%	0%	-10%	0%	1,032	607	0.01	0.01	0.01
i6	491	1,493	5.99	111%	0%	0%	19%	895	862	0.01	0.01	0.01
i7	649	1,867	9.26	88%	0%	0%	14%	1,526	61	0.02	0.01	0.01
S09234	958	2,001	23.47	52%	-26%	-32%	-22%	4,239	4,241	0.38	0.28	0.05
S13207	2,370	703	27.56	56%	-11%	-16%	2%	2,361	2,373	0.33	0.44	0.09
S15850	3,199	101	37.51	36%	-4%	-16%	-3%	1,243	892	0.38	0.58	0.11
S35932	8,614	250	11.65	157%	-10%	-22%	-22%	2,520	2,303	1.09	0.95	0.90
Average				50%	-3%	-12%	2%		11.89X	28,059X	29,912X	52,889X

Table 5.4: IVC results for NBTI-induced circuit degradation and leakage power.

Benchmark Circuit	Worst NBTI		Best NBTI		Worst Leakage		Best Leakage	
	ΔD (ps)	P_{leak} (nW)	ΔD (ps)	P_{leak}(nW)	P_{leak}(nW)	ΔD (ps)	P_{leak}	ΔD (ps)
C432	48.0	491.6	30.3	509.0	550.5	39.4	415.6	37.2
C499	37.3	1,320.0	27.3	1,360.0	1,486.4	32.3	1,247.5	34.8
C880	41.2	1,085.4	25.6	1,112.5	1,313.9	35.2	977.1	38.7
C1355	36.1	1,317.8	26.4	1,297.7	1,421.0	32.9	1,216.3	30.0
C2670	46.8	1,779.1	27.2	1,826.0	2,040.3	42.9	1,635.4	45.5
C3540	59.5	2,365.8	40.1	2,355.4	2,531.8	52.8	2,122.4	45.0
C5315	18.1	3,996.6	7.0	4,472,9	4,845.7	12.6	3,996.6	18.1
C6288	146.5	7,418.7	114.1	7,366.2	8,070.4	136.4	6,734.6	138.4
i2	20.5	485.4	14.5	492.1	951.5	15.7	248.5	17.1
i3	10.9	390.1	6.7	397.7	516.7	9.0	323.9	6.7
i4	26.7	505.4	12.2	479.9	646.9	19.5	359.2	22.9
i5	24.2	468.5	11.5	479.7	542.6	11.5	367.8	24.2
i6	11.3	1,446.9	6.0	1,245.0	1,639.6	8.7	1,178.4	8.5
i7	14.1	1,729.2	9.3	1,712.4	1,990.1	11.2	1,403.8	11.2
S09234	24.91	2,419.0	16.0	2,486.5	2,930.3	23.0	2,156.7	24.4
S13207	31.34	5,990.64	23.2	6,065.5	7,284.3	30.7	5,139.5	30.3
S15850	47.94	8,039.9	31.6	8,557.5	9,609.8	40.6	6,945.4	40.4
S35932	12.54	23,478.1	9.1	23,990.8	29,888.8	10.4	20,117.0	11.6
Average	36.5	3,596.0	24.3	2,678.2	4,347.8	31.4	3,143.6	32.5

Table 5.5 shows how the input vectors affect different primitive gates in terms of NBTI-induced aging and leakage. The values are shown as the percentage of worst aging and leakage, respectively. As shown in this table, for NAND gate the input pattern (00) resulting in the minimum leakage leads to the maximum NBTI degradation. On the other hand, in NOR and inverter gates the best input vector for leakage power is the best choice for NBTI.

Table 5.5: The impact of input vector on leakage and NBTI (Normalized to max).

	NAND		NOR			Inverter	
	Leakage	Aging	Leakage	Aging		Leakage	Aging
00	17%	100%	100%	100%	0	100%	100%
01	100%	50%	88%	50%			
10	45%	50%	8%	0%	1	48%	0%
11	49%	0%	12%	0%			

This phenomenon implies that, IVC is a co-optimization problem in terms of NBTI and leakage. As a result, we need to obtain a leakage-NBTI pareto-curve (or a set of Pareto points) for each circuit [120]. As an example, Figure 5.3 shows the distribution of normalized delay degradation and leakage (to the worst case) of various input vectors for circuit C880. It also shows the pareto-curve. Each point in the pareto-curve corresponds to an input vector which results in minimum NBTI-induced delay degradation with regards to a special leakage power. The set of Pareto points for each circuit can be obtained based on the approaches described in Section 5.2.3. Using such Pareto points, when an idle time is detected, a suitable input vector is selected according to leakage/NBTI requirement of the system and applied to the circuit.

Table 5.6 shows the minimum NBTI $\Delta delay$ of the selected benchmark circuits with different power constraints. This table can be viewed as the digitalized version of the leakage-NBTI pareto-curve in with 5% steps of the leakage power compared to the minimum value. The results show that for almost all of the circuits, the best input vector leading to minimum NBTI degradation can be obtained by only 20% relaxation of leakage power constraints. For instance, in circuit C6288 by only 5% more leakage power compared to the minimum value offered by IVC, NBTI minimization approach

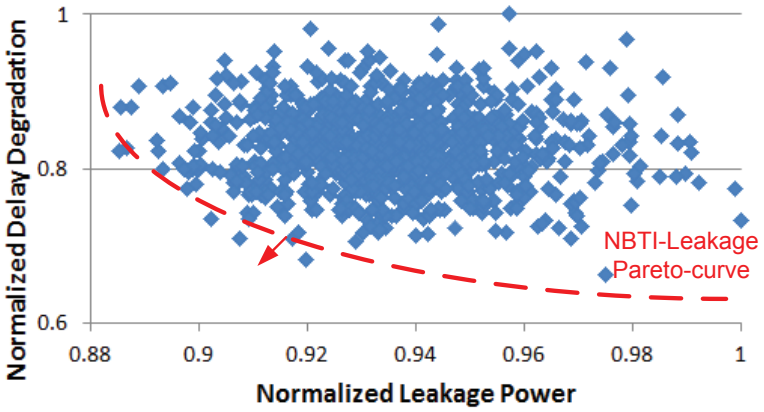


Figure 5.3: Co-optimization of input vector in terms of NBTI and Leakage-power for C880 benchmark circuit.

is saturated. Basically, with modest relaxation in power constraint, an IVC offering the minimum NBTI degradation can be achieved.

As shown in Table 5.3, the runtime of the LP method not only depends on the number of gates, but also strongly depends on the number of critical paths. For example, although S13207 has more gates than C6288, its LP runtime is extremely smaller. This is due to the fact that S13207 has lower amount of vulnerable paths in comparison with C6288 benchmark circuit. However, based on a technique proposed in [124], the number of aging vulnerable paths can be reduced by a factor of 50, in average. The benefit of using such approach is that it can significantly reduce the number of vulnerable paths to be considered while the accuracy is not affected. We can exploit such technique to start with fewer vulnerable paths, which can further improve the scalability of our method.

Table 5.6: Co-Optimization results of NBTI and leakage power with different power constraints.

Benchmark Circuit	Minimum NBTI ΔD (ps) with Different Power Constraints (Normalized to Minimum Leakage Power)					
	0%	5%	10%	15%	20%	25%
C432	37.18	33.19	31.58	30.30	30.30	30.30
C499	34.85	30.35	28.02	28.02	27.46	27.30
C880	38.73	36.02	32.19	29.47	26.20	25.65
C1355	29.97	28.69	27.97	26.36	26.36	26.36
C2670	45.55	34.18	28.74	27.24	27.24	27.24
C3540	45.05	43.40	41.61	40.17	40.17	40.17
C5315	18.09	10.88	9.44	8.16	7.05	7.05
C6288	138.48	114.12	114.12	114.12	114.12	114.12
i2	17.09	10.88	10.55	10.55	10.55	10.55
i3	6.71	6.71	6.71	6.71	6.71	6.71
i4	22.92	20.37	17.81	14.93	13.48	12.21
i5	24.25	17.87	12.76	11.49	11.49	11.49
i6	8.54	5.99	5.99	5.99	5.99	5.99
i7	11.21	10.32	10.32	10.32	9.26	9.26
S09234	22.03	16.92	15.98	15.98	15.98	15.98
S13207	27.06	23.23	23.23	23.23	23.23	23.23
S15850	40.45	31.57	31.57	31.57	31.57	31.57
S35932	11.65	9.10	9.10	9.10	9.10	9.10

5.3 NoP Assignment

Due to data and control hazards and memory stalls, pipelined processors need to execute instructions that have no effect on the state of the processor [127]. These special instructions are referred as NOP and their effects are actually to occupy the hardware resources for a certain instruction slots with no effect on program execution. It should be noted that there are multiple cases of instruction which can act as NOP (e.g $SLL\ R0, R0, 0$ or $ADD\ R0, R0, 0$). Since NOPs do not change the state of the executed application, the time spent for executing NOP instructions in a processor can be viewed as a pseudo-idle time. Based on out observation, a considerable fraction of total executed instructions of SPEC2000 benchmark programs are NOP instructions. This implies that, there are plenty of opportunities for alleviating NBTI effect. Indeed, NBTI effect strongly depends on the input vector. Therefore, the impact of the NBTI can be reduced by executing a suitable instruction as a NOP. *The key idea in this technique is finding a*

new instruction with no effect on the program state to replace the processor's default NOP instruction in order to minimize the NBTI effect.

A key requirement to successfully exploit a NOP instruction for aging reduction is understanding the effect of different instructions on aging. For this purpose, we investigate the impact of all possible instruction opcodes and instruction source operands on the delay-degradation imposed by NBTI. Our observations show that the NBTI degradation effects of the instruction opcodes that can be used as NOP are almost the same and minimal. On the other hand, source operands have a significant influence on the amount of NBTI degradation to the processor. Based on this observation, we use a linear programming approach for finding the best *Maximum Aging Reduction* (MAR) NOP (opcode and source operand values) which leads to minimum NBTI-induced delay degradation while has no effect on the state of the executed program and acts like a normal NOP. Finally, two different techniques (software-based and hardware-based) are proposed to show how the extracted MAR NOP can be applied to the processor. We evaluate our proposed approach on a MIPS processor with various SPEC2000 benchmark applications in terms of lifetime improvement, power and area overheads. We show that the lifetime of the processor can be extended by 37% in average while the observed area and power overheads are less than 1%.

5.3.1 MAR NOP Selection

NBTI Effect of Possible NOPs

NOP is an instruction with no effect on the program execution and since it has a neutral effect, it can be inserted at any location in the program execution. For example in a MIPS processor the default NOP instruction is:

$$SLL\ R_0,\ R_0,\ 0$$

This instruction denotes, the content of R_0 is shifted left zero times. Since the R_0 is hardwired to 0, this instruction has no effect on the status of the program. Many instructions such as ADD, OR, SUB with R_0 or immediate operands can be used alternatively as a NOP. It should be noted that, using any other register rather than R_0 even as a source operand, may cause a data

hazard. For example, the following instruction is an alternative for default NOP.

$$ADDI\ R_0,\ R_0,\ 8$$

Both introduced NOPs (default NOP and the ADDI example) can be used as NOP with no effect on program execution. However, since they have different opcode and source operands, they may cause different amount of NBTI-induced delay degradation on the processor. We investigate the effect of applying different NOP candidates on NBTI-induced delay degradation of the processor, based on the flowchart depicted in Figure 5.4(a).

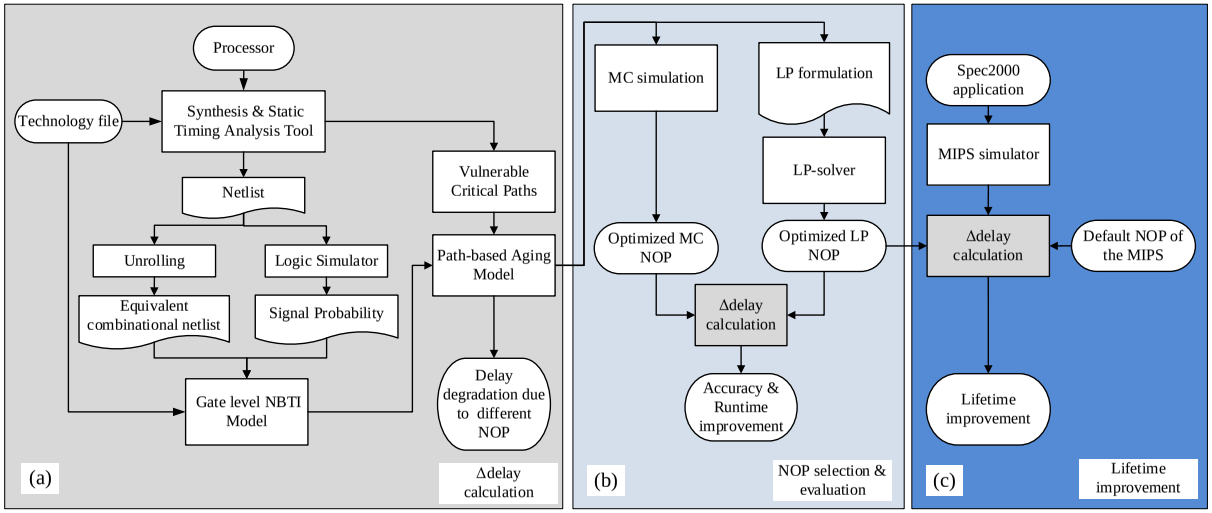


Figure 5.4: Overall flow of the proposed NBTI-aware NOP selection and evaluation.

We investigate the effect of different instruction opcodes and operands on the NBTI-induced delay degradation of the processor using the proposed variations-aware timing analysis technique. For each instruction opcode, the delay degradation imposed by NBTI is calculated for 100,000 randomly generated source operands (e.g. the immediate values and the data stored in source registers). According to the results illustrated in Figure 5.5, the average of the delay degradations for all instruction opcodes are almost the same. On the other hand, the variation of the NBTI-induced delay degradation is very sensitive to the values of source operands (the ranges shown in Figure 5.5). Therefore, it can be concluded that NBTI-induced delay degradation of a processor is mostly affected by the source operands rather than the instruction opcode. This phenomenon is mainly due to the following reasons:

- According to the analysis of critical paths, most of the aging vulnerable paths which can change the post-aging delay of the processor, are located in the EX-stage. Since the instruction opcode mostly affects the decoder rather than the EX-stage (specially ALU), the effect of the instruction opcode on the NBTI-induced delay degradation of the processor is negligible.

- Since the width of the opcode is considerably smaller than the operand width, the number of gates affected by the opcode is less than those affected by source operands.

Based on the above observations, to exploit NOP as a mechanism to efficiently minimize NBTI effect, one need to choose both opcode instruction and the values of source operands of NOP precisely. As a result two concerns should be addressed. First, optimized source operand values should be obtained for each opcode instruction in terms of NBTI. For this purpose, a Linear Programming approach is presented. Second, a mechanism should be devised to replace the default NOP and apply the opcode and its corresponding optimized source operands as a MAR NOP.

Linear Programming Approach

The straightforward solution for finding MAR NOP is to exhaustively apply and analyze all possible NOPs (and operand values) which is infeasible.

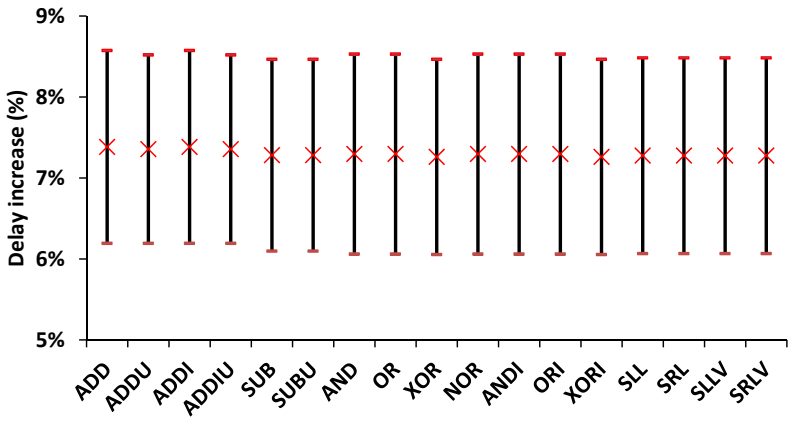


Figure 5.5: The effect of different NOPs (opcode and operand values) on NBTI-induced delay degradation (the range shows the impact of operand values).

Therefore, we use our proposed LP-based technique which was discussed in Section 5.2 to obtain MAR NOP resulting in minimum NBTI-induced delay degradation. The main characteristic of NOPs is that they do not change the state of the program. Therefore, only a subset of the instruction set can be employed as NOP. Moreover, as discussed, the NBTI effect mostly depends on the source operand rather than the instruction opcode. As a result, we need to modify NOP in a way that, it consists of not only a suitable instruction opcode, but also the corresponding NBTI-optimized operands value. To find a MAR NOP, first, all possible instruction opcodes that can be act as a NOP are considered. The first column of Table 5.7 shows all possible instruction opcodes that can be used as NOP instruction opcode. Next, for each opcode, the best operand values which minimize the NBTI effect on the processor is extracted. The proposed LP approach is used to find the optimized operand value for each opcode. Finally, the pair of opcode and corresponding operand values resulting in the minimum delay degradation is selected as MAR NOP.

For each opcode, the objective is to minimize the overall post aging delay of the processor imposed by NBTI considering all critical paths. The result of this optimization is a source operand leading to minimum NBTI of a processor for each instruction opcode. This objective can be represented by

the following equation:

$$minimize : x \mid \forall j : \; x \geq \tau(CP_i) = \sum_{g_i \; in \; CP_j}^{i} \Big(\tau(g_i) + \Delta\tau(g_i) \Big) \tag{5.11}$$

where CP is a critical path and g_i is the gate i in the critical path. For each critical path, the post-aging delay is the sum of the post-aging delay of all the gates along that path. The post-aging delay of each gate is equal to summation of pre-aging delay of the gate, $\tau(g_i)$, and the NBTI-induced delay increase, $\Delta\tau(g_i)$. As mentioned before, the NBTI-induced delay increase of each gate depends on the state of its inputs. To represent $\Delta\tau(g_i)$ of each primitive gate in a LP compatible format, we use the technique introduced in Section 5.2.

5.3.2 Applying MAR NOPs

In this section, we present two different methods for applying the instruction opcodes and their corresponding optimized source operand values as an MAR NOP.

Software-based Implementation

In order to apply the operand values of MAR NOP without affecting the program execution, we need to reserve some registers. These registers are dedicated only to save the corresponding operands of the MAR NOP (not available for application anymore) and are loaded right before the application is executed. Table 5.7 shows all possible instructions of MIPS which can be used as a NOP instruction. For example, $ADD\ R0, R_i, R_j$ can act as a NOP instruction only if the registers R_i and R_j are reserved. Otherwise, since the application might use these registers, the output of the program could be affected.

The last column of Table 5.7 shows the number of registers needed to be reserved for the corresponding NOP instructions. It should be noted that, these instructions are selected in a way that, the data stored in the reserved registers does not change during the NOP execution. In other words, the reserved registers keep the NBTI-optimized source operand during NOP

Table 5.7: NOP candidates of MIPS processor in the software-based implementation.

Operation (OP)	Operand	# of reserved registers
ADD, ADDU, SUB SUBU, XOR	$R_0 \leftarrow R_i\ OP\ R_j$	2
	$R_i \leftarrow R_i\ OP\ R_0$	1
OR	$R_0 \leftarrow R_i\ OP\ R_j$	2
	$R_i \leftarrow R_i\ OP\ R_0$	1
	$R_i \leftarrow R_i\ OP\ R_i$	1
ADDI, ADDIU ORI, XORI	$R_0 \leftarrow R_i\ OP\ Imm$	1
	$R_i \leftarrow R_i\ OP\ 0$	1
AND	$R_0 \leftarrow R_i\ OP\ R_j$	2
	$R_i \leftarrow R_i\ OP\ R_i$	1
ANDI	$R_0 \leftarrow R_i\ OP\ Imm$	1
	$R_i \leftarrow R_i\ OP\ 1$	1
NOR	$R_0 \leftarrow R_i\ OP\ R_j$	2
SRA, SLL, SRL	$R_0 \leftarrow R_i\ OP\ SA$	1
	$R_i \leftarrow R_i\ OP\ 0$	1
SRLV, ROTRV SLLV,SRAV	$R_0 \leftarrow R_i\ OP\ R_j$	2
	$R_i \leftarrow R_i\ OP\ R_0$	1
ROTR	$R_0 \leftarrow R_i\ OP\ SA$	1
	$R_i \leftarrow R_i\ OP\ 0$	1
	$R_i \leftarrow R_i\ OP\ 32$	1
Default NOP of MIPS	$R_0 \leftarrow R_0\ SLL\ 0$	0

execution. In conclusion, to apply a MAR NOP in software-based approach the following steps are performed:

1. Modify the compiler directives to generate the binary/assembly code while reserving the required (one or two) registers.

2. Replace the default NOPs in the code with MAR NOP.

3. Add necessary instructions to the beginning of the code to assign those reserved registers to the optimal values of MAR operands.

Another alternative of software realization of MAR NOP is round-robin allocation of the registers to the operands of the MAR NOP, however, it requires further modifications to the compiler.

Hardware-based Implementation

Here we present a hardware-based method for replacing the default NOP with the MAR NOP applying them during program execution. In this approach, we modify the input multiplexers of the ALU in a way that the NBTI-optimized source operand for the NOP-instruction is directly provided in the EX-unit (see Figure 5.6). For this purpose, an extra input is added to each of the input multiplexer of the ALU. These inputs provide the NBTI-optimized data for MAR NOP. In addition, decoder should be slightly changed accordingly to support the modification of the input multiplexer of the ALU. Since the operand values of the NOP are available in the EX-stage, the hardware-based NOP implementation can handle all the situations stem in hazards (e.g when a NOP is needed to be inserted from the EX-stage into the processor). Moreover, to insert a NOP instruction from IF stage, due to branch hazard, the Hazard Detection Unit should be accordingly changed to reset the instruction field of the IF stage to the MAR NOP (In base MIPS processor, this field is set to the default NOP).

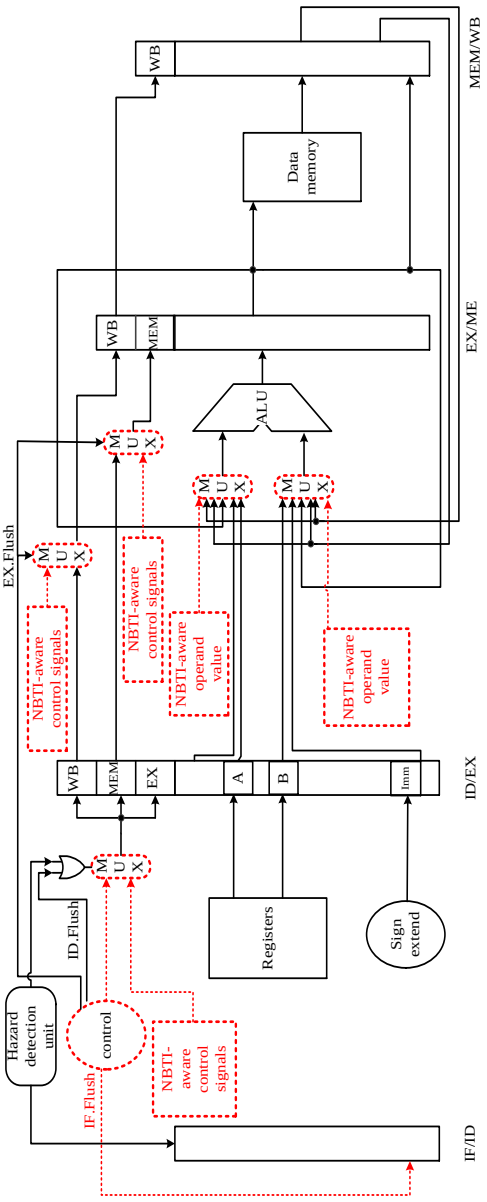


Figure 5.6: Hardware-based implementation of NOP in MIPS architecture.

Comparing Hardware versus Software Implementations

As mentioned, the software-based NOP implementation needs at least one reserved register. This implies that, the number of available registers for compiling a program is reduced. As a result, the performance might be decreased. Another limitation of the software-based approach is that, it cannot be used for all types of hazard which have been handled by the traditional NOP. There are some situations which might occur when the forwarding unit cannot avoid data hazards. As an example, consider an instruction which needs data that is provided by a preceding load instruction. In this case, the Hazard Detection Unit in the ID-stage identifies this situation in advance and inserts a stall between these dependent instructions. As a result, we should force the EX stage to perform a special instruction that does not change the state of the processor. Since here NOP is inserted from EX stage, software-based method cannot handle this situation. This is due to the fact that the source registers are read in the ID-stage and since this type of NOP is inserted after the ID-stage, the registers which contain the NBTI-optimized operands cannot be read. This phenomenon might also occur in control hazards. The most well-known method for resolving control hazards and reducing the branch penalty is branch prediction method. In case of misprediction, all of the instructions fetched according to the prediction, are flushed from the pipeline. To flush an instruction, the instruction field of the register is set to a NOP instruction. Depending on the branch execution unit, NOPs might be inserted from the EX-stage. Similar to the previous situation in case of data hazards, software-based MAR NOP implementation cannot be used here as well. In order to overcome these cases, compiler can be configured to take care of all possible hazards and stalls and hence bypass the hardware supports by applying the necessary NOPs statically at the compile time. However, this may result in some performance degradation, particularly for branches.

Despite the discussed limitations of software-based implementation, this approach is flexible and does not need any special hardware support or modification. In addition, since the critical paths of the circuit might change due to the different data patterns of the applications during the circuit lifetime, the optimized operands of the MAR NOP might change as well. Changing the appropriate operands of the MAR NOP in a software-based implemen-

tation is very straightforward and only needs to load new operands into the corresponding reserved registers before the program execution.

The main advantage of the hardware-based implementation of MAR NOP is that it can be used for all situations when a NOP must be inserted to the processor even from the EX-stage. However, since the optimized operands of the MAR NOP are provided by a hardwired method, it is not as flexible as the software-based implementation. To remove this drawback it is possible to use the already available scan-chain registers for modifying the operands according to the current state of the processor in terms of timing properties of the critical paths, with some additional design changes and overhead.

In in-order processors, when an instruction cannot be executed due to hazards and stalls, all the following instructions should be stalled and NOP instructions are inserted until the new instruction can be executed. On the other hand, in out-of-order superscalar processors, Functional Units (FUs), such as ALU, reservation stations, reorder buffers and physical registers, are isolated from each other with some sort of buffers. Therefore, when an instruction faces a hazard or stall, the following instructions can be executed. Typically, the utilization of FUs is far less than 100% and clock-gating is used during idle cycles to reduce power consumption. Therefore, the idle time of each FU can be exploited to apply MAR NOP. For each FU, based on its functionality and gate-level implementation, a suitable MAR NOP can be extracted and can be applied to the corresponding FU for any cycle that the FU is idle. In this case, the hardware-based implementation of MAR NOP is more favorable because software-based implementation might lead to some performance overhead.

5.3.3 Experimental Results

To evaluate the efficiency of the proposed method a five-stage MIPS processor is used. It should be noted that our methodology is generic and can be applied to other processors. The processor is synthesized by Synopsys Design Compiler and is mapped to TSMC 65nm standard cell library. By assuming a delay degradation of 10% in 3 years, all critical paths with 10% positive slack are extracted using PrimeTime static timing analyzer. Then, the unrolling method is applied to remove the logic feedbacks and convert

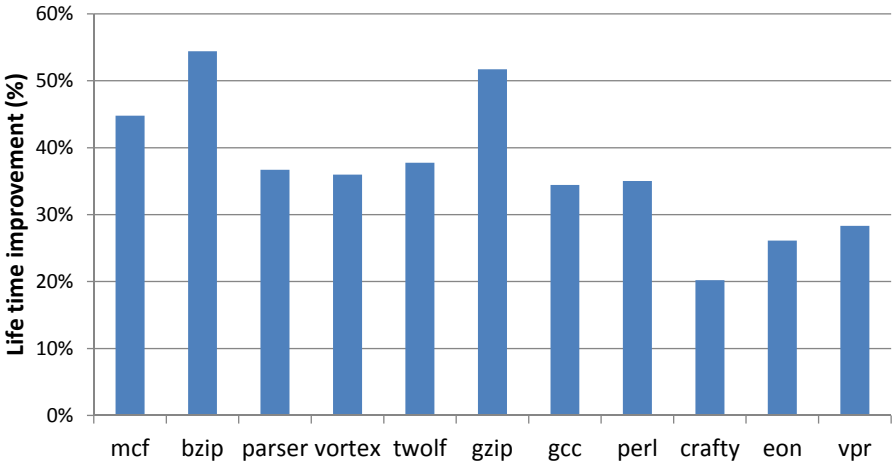


Figure 5.7: Lifetime improvement for selected spec2000 application using NBTI-aware NOP assignment.

the sequential structure of the processor to a combinational one. The signal probability (duty cycle) of each transistor is calculated by a logic simulator as well (see Figure 5.4(a)). By the method presented in 5.3.1 the problem of finding the best source operand for each opcode instruction is obtained using CPLEX (LP solver) and then the best pair (opcode and corresponding operand values) is selected as MAR NOP. We have also implemented a Monte Carlo (MC) simulation to obtain the optimal source operand for each opcode instruction to validate the LP approach. According to the results, due to large input set of the processor, the MAR NOP obtained by LP is better optimized than MC (5%) while reduces the runtime by 150x in average.

To analyze the efficiency of the extracted MAR NOP on the processor lifetime, we choose SPEC2000 benchmarks. We do a profiling on these applications with the M5 simulator [128]. Based on the output of the profiling, extracted MAR NOP, and default MIPS NOP the lifetime improvement is calculated by using the flow depicted in Figure 5.4(c). Figure 5.7 shows the lifetime improvements for the SPEC2000 applications, when the default NOPs are replaced with MAR NOPs. According to the results our proposed approach can extend the lifetime of the processor by 37% in average. It should be noted that the actual results strongly depend on the technology node, circuit design, and architecture which varies from one processor to another.

The software-based implementation needs to reserve one or two registers

for storing the optimized source operands of the MAR NOP. To investigate the effect of register reservation on the performance of the processor, we apply it to several selected SPEC2000 benchmarks. Each of the application is compiled with gcc-3.4.3 with -O1. According to the results illustrated in Table 5.8, reserving one register registers reduce the Instructions Per Clock (IPC) by only 0.1%. Moreover, IPC decreased around 0.5% due to reservation of two registers.

Table 5.8: Register reservation overhead on IPC.

Application	One register	Two registers
mcf	0.0%	0.2%
bzip2	0.0%	0.0%
parser	0.1%	0.5%
vortex	0.0%	-0.4%
twolf	0.0%	0.4%
gzip	0.0%	2.1%
gcc	0.6%	1.1%
perl	0.0%	0.0%
Average	0.1%	0.5%

To analyze the hardware-based approach, the modifications, have been applied to the RTL description of the MIPS processor and the modified version is synthesized with Synopsys Design Compiler. The results, as shown in Table 5.10, confirm that the overhead of this approach is quite negligible.

5.4 Adaptive Mitigation Techniques

Although static techniques are effective ways for mitigating parameter variations, they are not sufficient and need to be complemented by adaptive techniques. In this section, we present two different adaptive techniques, namely *Adaptive IVC* and *Adaptive Guard-banding* to further improve re-

Table 5.9: Normalized overhead of Hardware-based implementation of NOP to original MIPS.

	Original	Modified	Overhead
$Power(mW)$	1.897	1.919	1.1%
$Area(\mu m^2)$	35591	35717	0.3%
$Delay(ns)$	4.38	4.38	0.0%

siliency of the chip against parameter variations. The idea is to track the status of the chip with respect to the variations-indued delay degradation using the proposed age/delay monitoring system (See Section 4) and adjust the corresponding reliability knobs (such as timing margin or input vector) during runtime.

5.4.1 Adaptive Input Vector Control (A-IVC)

Although the static-IVC technique (introduced in Section 5.2) attempts to co-optimize BTI and leakage power, it has two shortcomings which can be improved by an adaptive approach:

1) Critical paths may change over time due to parameter variations. Therefore, the selected IVs might not be efficient during runtime, since they target paths that are no longer critical.

2) The leakage-BTI pareto-curve is affected considerably by the PVT corner. For example, Fig. 5.8 shows the pareto-curve of the b19 benchmark circuit at two different temperatures. First, when the circuit temperature is 25°C, we select IV_i to minimize the delay while meeting the leakage limit. Next, when the circuit temperature is reduced to 20°C, the leakage-BTI pareto curve also shifts consequently. In the static-IVC method, the same IV (IV_i) is selected to meet the leakage constraint. However, more delay reduction can be achieved if we select another IV (IVj) adaptively, while satisfying the requirement of limiting leakage.

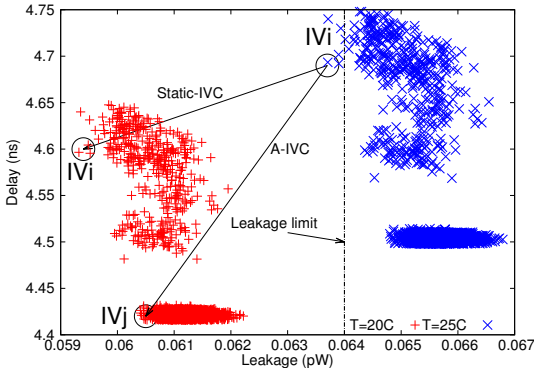


Figure 5.8: Comparison of the proposed A-IVC against static-IVC.

The concept of fine-grained monitoring and fine-grained adaptation enables us to address the above two shortcomings of static-IVC. Fig. 5.9 illustrates the overall flow of the proposed technique that allows us to continuously

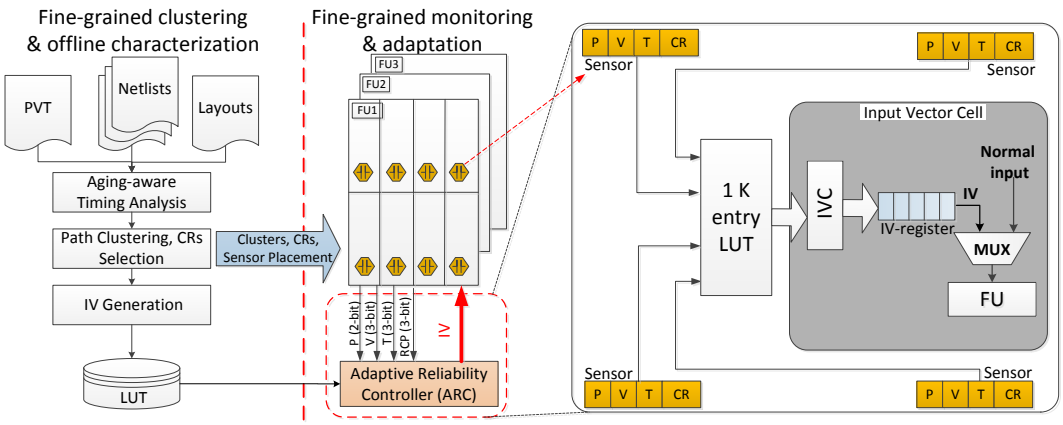


Figure 5.9: Fine-grained clustering, monitoring, and runtime adaptation.

re-evaluate chip conditions and apply countermeasures to efficiently tackle parameter variations by jointly co-optimizing frequency and leakage. As shown in the figure, the proposed technique consists of two different phases: 1) fine-grained clustering and offline Characterization, and 2) Fine-grained monitoring and adaptation.

Fine-grained clustering and offline characterization: In circuits with millions of *Critical Paths* (CPs), monitoring and adjusting the configuration of each CP individually is infeasible. To address this problem, we use machine learning to identify and exploit topological and electrical similarities among critical paths. We propose to group the critical paths into several clusters, such that the variation-induced delay increase of the paths in the same cluster are highly correlated. In other words, critical paths that belong to the same cluster tend to follow similar trends in terms of delay variation. This implies that if we monitor the delay of one path for each cluster, namely the CR, the status of the whole cluster in term of parameter variations can be determined with high accuracy. Note that to monitor the delay of each CR, we can place any available in-situ delay sensor such as described in [85, 129] at the downstream flip-flop of the path. For example, Fig. 5.10 illustrates how critical paths are grouped into four different clusters (groups). In this example, by only measuring the delay of four CR (represented by black circles), we can accurately estimate delays of other 14 CPs (represented by white circles) and hence the status of all four clusters.

In the offline phase, we also characterize each cluster at different parameter variations to find the corresponding near-optimal configuration (e.g., in A-

IVC, this configuration is IV) for each cluster. For this purpose, the BTI-leakage pareto-curve is obtained at different PVT corners for each generated path cluster. Pareto-curves of a cluster minimize the BTI-induced delay degradation of the cluster at a certain PVT corner under the constraint of leakage power for the entire circuit. The Pareto-curves obtained in this way are sampled and the corresponding IVs are stored a LUT of the ARC module to be used during runtime adaptation.

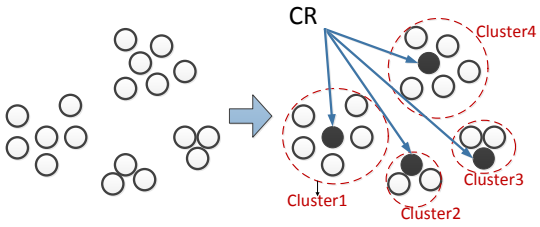


Figure 5.10: Illustration of path clustering and CR selection.

Fine-grained monitoring and adaptation: During runtime, an adaptive controller, namely *Adaptive Reliability Controller* (ARC) obtains the status of the PVT variations of the path clusters using available process, voltage droop, and thermal sensors such as in [130, 131]. Moreover, the proposed age/delay monitoring systems, the ARC module tracks the delay and BTI-induced delay degradation of path clusters. The ARC relies on a LUT, which is indexed based on readouts of the sensors, to adaptively adjust the configuration (e.g., suitable IVs) of each cluster. Note that the LUT is loaded with offline characterization data.

Implementation Issues of A-IVC

To pre-characterize the LUT (i.e., find appropriate IVs for different conditions), we adopt the linear programming (LP) approach presented in [7] to consider both path clustering and PVT corners. In addition, we rely on existing clock-gating units to realize the hardware implementation of IVC. A two-input multiplexer is added in front of the functional unit. One input is connected to the ARC module, which provides the input vectors in the IVC technique. The other input is the normal input of the logic core. A clock-gating unit is used to select the input to the multiplexer. As an example, Fig. 5.11 conceptually illustrates how aging-leakage aware IVs are applied

to the ALU of the LEON processor during idle times. Note that the IV is selected based on the LUT indices, which are the outputs of temperature, voltage, and delay sensors.

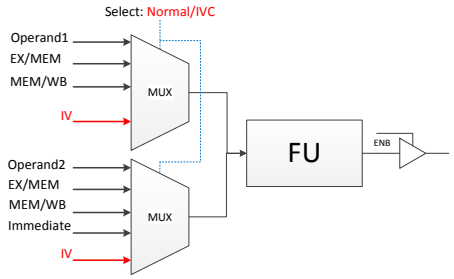


Figure 5.11: Hardware realization of A-IVC for the functional unit of the LEON processor.

The size of the LUT for the ARC module is a major practical concern in the proposed techniques. The LUT size depends on the number of sampling points of PVT corners and the number of sampling points of the delay sensors that are dedicated to monitor the delay of each CR/cluster. The sampling points represent a tradeoff between accuracy and overhead (i.e., the size of the LUT). To make a tradeoff, we use a non-uniform sampling approach such that more samples are used for the critical range of each parameter. To accomplish this goal, we perform two kinds of analysis. 1) Sensitivity analysis: We vary each parameter (e.g., channel length, threshold voltage, voltage, and temperature.) from its nominal value and observe the sensitivity of the corresponding path delay as well as the leakage power. 2) Frequency analysis: We determine what range of parameter variation is more likely to be experienced by the processor. Based on above analysis, we use 7, 7, 5, 4 sampling points for voltage, temperature, delay sensors, and process variation, respectively. Less than 1 KB of memory is required to store the LUT. Since process variation does not change over time, we can reduce the LUT size to less than 256 B by incorporating a software approach. In this method, the total size of the LUT (i.e., 1 KB) is divided into 4 pages with the size of 256 B corresponding to the 4 different process variation points. After fabrication, when sensors determine the actual process corner, the corresponding page is copied to the dedicated physical LUT in the ARC module.

Experimental Results

To evaluate the efficiency and accuracy of the A-IVC, experiments are performed on various ISCAS'89, IWLS'05, and ITC'99 benchmark circuits [78, 94]. Synopsys Design Compiler and Cadence SOC Encounter are used for synthesis and place-route, respectively, with the Nangate 45nm target library [80]. An iterative profiling process similar to [132] is conducted to obtain the temperature, voltage, and BTI of each individual gate within the circuit. The BTI-induced threshold voltage change is estimated by assuming a delay degradation of 10% in 3 years [7]. As in recent related work, circuit calibration and cell characterization are based on 120°C and 1 V as the operating temperature and the voltage, respectively [133]. Moreover, based on ITRS, we assume 10% variations in voltage droop and relevant fabrication process parameters [2]. The power profile and the corresponding temperature profile are calculated by running different randomly generated input vectors with a switching activity factor of 0.2. Synopsys Primetime is used to extract critical and near-critical paths of the circuit in the presence of aging and variations.

Fig. 5.12 compares the lifetime improvement obtained by the proposed adaptive-IVC to static-IVC presented in [7]. Static-IVC can extend the lifetime by 32% on average, while the proposed A-IVC extends the lifetime by 60% on average. Moroever, we observe that in the presence of PVT variations, leakage power improvement provided by A-IVC is 64% higher on average over all benchmark circuits compared to static-IVC. The reason for this considerable lifetime-leakage improvement is that the proposed technique is capable of tracking the usage and operating conditions of the chip. Therefore, it adaptively adjusts the IVs during runtime, which results in more power saving and longer lifetime.

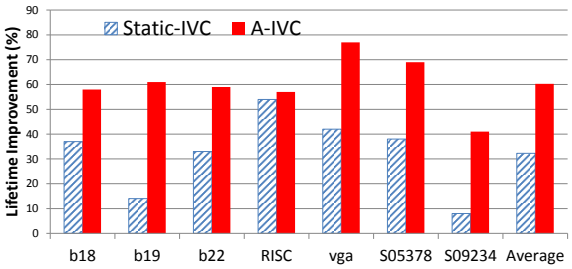


Figure 5.12: Lifetime improvement using A-IVC compared to static-IVC [7].

To estimate the overhead of the proposed technique, all the modification

depicted in Fig. 5.11 are implemented for ALU of LEON2 processor. The results obtained by Synopsys Design Compiler show that the overhead is negligible (See Table 5.10).

Table 5.10: Overhead of the of A-IVC for the LEON processor.

	Original	Modified	Percentage Overhead
$Power[mW]$	5.5976	5.5256	1.3%
$Area[\mu m^2]$	41305	41676	0.9%
$Delay[ns]$	1.58	1.59	0.6%

5.4.2 Adaptive Guard-banding

Static guard-banding is the most common design-time technique to tackle aging and variations. In this technique, at design time a conservative timing margin based on the worst-case variability corners combined with maximum target lifetime is considered for the circuit in order to avoid timing failures. However, aging is a gradual process and hence the entire margin is not required earlier in the lifetime. In addition, not all the fabricated chip suffers from wors-case process variations. Therefore, static guard-banding may lead to considerable performance loss. To minimize the performance overhead, the impact of transistor aging and variation on the overall circuit delay can be accurately obtained by the proposed age/delay monitoring (See Section 4). Based on the exact amount of delay, the actual value of delay degradation has to be set as timing margin to gain performance. Fig. 5.13 shows the performance gain for circuit when b17, when static guard-banding is replaced by adaptive guard-banding. As illustrated in this figure, in the adaptive method, the timing margin is gradually increased over time but in the static method timing margin is fixed. Note that performance gain is eroded over time. When the circuit is fresh ($t = 0$), performance gain is very high. However, when the circuit is already aged (e.g., after 10 years) the timing margins of adaptive and static approaches become same and hence we do not gain any performance. Note that our approach also enables us to reduce the timing margin that is allocated for tackling process variations. Table 5.11 shows the performance improvement of the adaptive guard-banding to static guard-banding for different circuits at $t = 0$ year and $t = 5$ years.

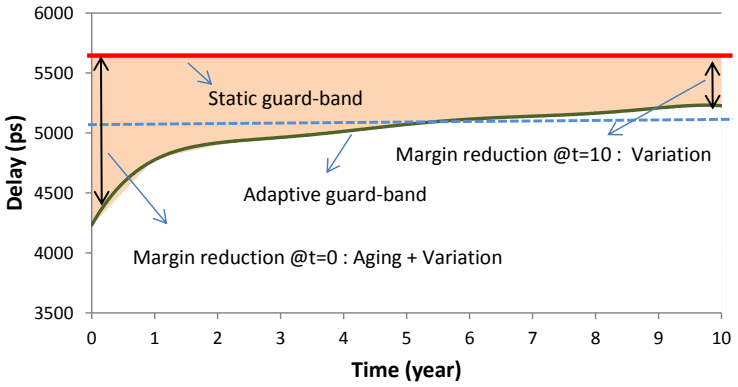


Figure 5.13: Performance gain obtained by adaptive guard-banding based on RCPs compared to static guard-banding for circuit b17.

Table 5.11: Performance improvements of adaptive guard-banding to static guard-banding.

Benchmark	No.of gates	Performance improvements (%)	
		$@t = 0year$	$@t = 5years$
b17	27K	30%	10%
b18	88K	25%	10%
b19	185K	25%	10%
b22	40K	29%	10%
RISC	61K	22%	10%

5.5 Conclusions

In this section, we introduced a set of complementary static and adaptive mitigation techniques to tackle the adverse impact of parameter variations. We proposed IVC method that can be used to co-optimize power consumption and aging-degradation during idle cycles by applying a suitable input vector. We also illustrated that how IVC method can be used for NOP assignment to maximum BTI relaxation on the processor. Two methods, software-based and hardware-based, are proposed to replace the original NOP with this maximum aging reduction NOP. Finally, we presented two adaptive techniques to adjust the timing margin and the IVC on the basis of dynamic workload-dependent variations using the proposed RCP-based delay/age monitoring system.

CHAPTER 6

SUMMARY AND CONCLUSION

Technology scaling is the key to continue the success of semiconductor industry by integrating faster while low power transistors in smaller area. As semiconductor integrated circuits become denser and more complex due to technology scaling, chip designers face several reliability challenges. Parameter variation is considered as a major reliability concern in the nanoscale regime for integrated circuits. Parameter variations can arise either from process variations or workload-dependent runtime variations such as temperature variations, voltage droop and transistor aging. Undesirable variations in chip parameters can result in mismatch between electrical design specifications and runtime characteristics. This inconsistency is then translated to to loss of performance, timing violations, unexpected failures, and reduced lifetime.

Process variations such as random dopant fluctuations are the results of the imperfection in the chip manufacturing process. Temperature variations is related to workload-dependent power consumption and power density of the chip. Voltage drop occurs when a large number of logic gates in the circuit draw high switching current from the on-chip power supply network. Transistor aging mainly due to BTI and HCI increases the threshold voltage of transistors over time. As a consequence of parameter variations, there is an increase in gate delays, resulting in higher path delays, and the eventual occurrence of intermittent and transient faults during chip operation.

To fully address the problem of parameter variations, it is important to 1) model and analyze the chip delay in the presence of parameter variations, 2) be able to monitor and track the adverse impacts of them on circuit delay and lifetime, and 3) compensate and mitigate their undesirable effects on the chip. Due to importance of parameter variations, over the past few years several attempts have been done to address each of these three aspects. The main shortcoming of state-of-the-art timing analysis techniques is that the interaction among parameter variations is neglected. However, all these phenomena are tightly coupled and hence the combined effect of all these variations on the circuit timing has to be considered. Existing monitoring techniques also suffer from huge area/performance overheads. In addition, these techniques do not provide fine-grained information about status of each individual critical path in the circuit. Finally, the available mitigation techniques cannot tackle the detrimental impacts of parameter variations in an efficient way.

The objective of this thesis was to appropriately address the shortcomings of prior techniques by different novel means. In Chapter 3, the interdependence between parameter variations were studied and modeled which enabled us to improve the accuracy of timing analysis flow. In Chapter 4, we proposed a learning machine techniques for age/delay monitoring of the chip in-filed. This technique allows to analyze the status of each critical paths in the circuit under the influence of parameter variations with negligible overheads during runtime. On top of the proposed timing analysis framework and monitoring system, a set of complementary static and adaptive techniques were also proposed in Chapter 5, which can significantly improve the lifetime and frequency of the chip.

Our novel techniques to model, track, and mitigate parameter variations can be exploited in a variety of applications. For example, critical applications such as automotive, medical, and space applications can benefit from them to improve the lifetime and resiliency against undesirable unexpected failures. By introduction of new reliability issues such as unintentional design-time attacks, and intentional hardware bugs which are inserted for malicious purpose, one extension of this work could be capturing these anomalies as well. Another promising extension of this thesis might be abstracting some of the proposed models to be able to consider them in other domains such as cloud computing.

REFERENCES

[1] J. Hicks, D. Bergstrom, M. Hattendorf, J. Jopling, J. Maiz, S. Pae, C. Prasad, and J. Wiedemer, "45nm transistor reliability." *Intel Technology Journal*, vol. 12, no. 2, 2008.

[2] International Technology Roadmap for Semiconductors (ITRS), available at http://www.itrs.net/.

[3] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer, "Statistical timing analysis: From basic principles to state of the art," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 4, pp. 589–607, 2008.

[4] 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 sio2/hfo2 stacks with fusi, tin, re gates," in *Symposium on VLSI Technology*, 2006, pp. 23–25.

[5] W. Huang et al., "Hotspot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, vol. 14, no. 5, pp. 501–513, 2006.

[6] S. Wang, M. Tehranipoor, and L. Winemberg, "In-field aging measurement and calibration for power-performance optimization," in *DAC*, 2011, pp. 706–711.

[7] F. Firouzi, S. Kiamehr, and M. B. Tahoori, "Power-aware minimum nbti vector selection using a linear programming approach," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 100–110, 2013.

[8] http://www.nobelprize.org.

[9] http://www.TI.com.

[10] J. L. Hennessy and D. A. Patterson, *Computer architecture: a quantitative approach*. Elsevier, 2012.

[11] G. E. Moore et al., "Cramming more components onto integrated circuits," 1965.

[12] T. McConaghy and J. Hogan, *Variation-Aware Design of Custom Integrated Circuits: A Hands-on Field Guide*. Springer, 2013.

[13] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *Design Automation Conference*, 2003, pp. 338–342.

[14] M. S. Gupta, J. A. Rivers, P. Bose, G.-Y. Wei, and D. Brooks, "Tribeca: design for pvt variations with local recovery and fine-grained adaptation," in *International Symposium on Microarchitecture*. ACM, 2009, pp. 435–446.

[15] M. Dietrich and J. Haase, *Process Variations and Probabilistic Integrated Circuit Design*. Springer, 2012.

[16] A. Agarwal, D. Blaauw, and V. Zolotov, "Statistical timing analysis for intra-die process variations with spatial correlations," in *Computer-aided design*, 2003, p. 900.

[17] H. Chang and S. S. Sapatnekar, "Full-chip analysis of leakage power under process variations, including spatial correlations," in *Design Automation Conference*, 2005, pp. 523–528.

[18] D. Boning and S. Nassif, "Models of process variations in device and interconnect," *Design of high performance microprocessor circuits*, p. 6, 2000.

[19] S. Bhunia, S. Mukhopadhyay, and K. Roy, "Process variations and process-tolerant design," in *International Conference on VLSI Design*, 2007, pp. 699–704.

[20] E. Humenay, D. Tarjan, and K. Skadron, "Impact of process variations on multicore performance symmetry," in *Design, automation and test in Europe*, 2007, pp. 1653–1658.

[21] M. Alioto, G. Palumbo, and M. Pennisi, "Understanding the effect of process variations on the delay of static and domino logic," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 5, pp. 697–710, 2010.

[22] X. Liang and D. Brooks, "Mitigating the impact of process variations on processor register files and execution units," in *International Symposium on Microarchitecture*, 2006, pp. 504–514.

[23] P. B. Medawar, *An Unsolved Problem of Biology*. H. K. Lewis, 1952.

[24] M. A. Alam and S. Mahapatra, "A comprehensive model of pmos nbti degradation," *Microelectronics Reliability*, vol. 45, no. 1, pp. 71–81, 2005.

[25] R. Vattikonda, W. Wang, and Y. Cao, "Modeling and minimization of pmos nbti effect for robust nanometer design," in *Design Automation Conference*, 2006, pp. 1047–1052.

[26] M. A. Alam, "A critical examination of the mechanics of dynamic nbti for pmosfets," in *International Electron Devices Meeting*, 2003, pp. 14–4.

[27] V. Huard, M. Denais, and C. Parthasarathy, "Nbti degradation: From physical mechanisms to modelling," *Microelectronics Reliability*, vol. 46, no. 1, pp. 1–23, 2006.

[28] H. H. Chen and D. D. Ling, "Power supply noise analysis methodology for deep-submicron vlsi chip design," in *Design Automation Conference*, 1997, pp. 638–643.

[29] H. Su, F. Liu, A. Devgan, E. Acar, and S. Nassif, "Full chip leakage estimation considering power supply and temperature variations," in *International symposium on Low power electronics and design*, 2003, pp. 78–83.

[30] A. K. Coskun, T. S. Rosing, K. A. Whisnant, and K. C. Gross, "Static and dynamic temperature-aware scheduling for multiprocessor socs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 9, pp. 1127–1140, 2008.

[31] Y. Zhan and S. S. Sapatnekar, "Fast computation of the temperature distribution in vlsi chips using the discrete cosine transform and table look-up," in *Asia and South Pacific Design Automation Conference*, 2005, pp. 87–92.

[32] N. James, P. Restle, J. Friedrich, B. Huott, and B. McCredie, "Comparison of split-versus connected-core supplies in the POWER6 microprocessor," in *IEEE International Solid-State Circuits Conference*, 2007, pp. 298–604.

[33] W. Wang et al., "The impact of NBTI effect on combinational circuit: modeling, simulation, and analysis," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, vol. 18, no. 2, pp. 173–183, 2010.

[34] J. Jaffari and M. Anis, "Statistical thermal profile considering process variations: Analysis and applications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 6, pp. 1027–1040, 2008.

[35] B. Lasbouygues, R. Wilson, N. Azemard, and P. Maurine, "Temperature-and voltage-aware timing analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,*, vol. 26, no. 4, pp. 801–815, 2007.

[36] R. Shen, S. X.-D. Tan, and H. Yu, *Statistical performance analysis and modeling techniques for nanometer VLSI designs*. Springer, 2012.

[37] J. Xiong, V. Zolotov, and L. He, "Robust extraction of spatial correlation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 4, pp. 619–631, 2007.

[38] H. Chang and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single pert-like traversal," in *International Conference on Computer-aided design*, 2003, p. 621.

[39] S. Bhardwaj et al., "Predictive modeling of the NBTI effect for reliable design," in *Proceedings IEEE Custom Integrated Circuits Conference*, 2006, pp. 189–192.

[40] W. Wang et al., "The impact of NBTI on the performance of combinational and sequential circuits," in *ACM/IEEE Proceedings Design Automation Conference (DAC)*, 2007, pp. 364–369.

[41] B. Kaczer, S. Mahato, V. V. de Almeida Camargo, M. Toledano-Luque, P. J. Roussel, T. Grasser, F. Catthoor, P. Dobrovolny, P. Zuber, G. Wirth et al., "Atomistic approach to variability of bias-temperature instability in circuit simulations," in *International Reliability Physics Symposium (IRPS)*, 2011, pp. XT–3.

[42] T. Grasser, B. Kaczer, W. Goes, H. Reisinger, T. Aichinger, P. Hehenberger, P.-J. Wagner, F. Schanovsky, J. Franco, M. T. Luque et al., "The paradigm shift in understanding the bias temperature instability: from reaction–diffusion to switching oxide traps," *IEEE Transactions on Electron Devices*, vol. 58, no. 11, pp. 3652–3666, 2011.

[43] J. Franco, B. Kaczer, M. Toledano-Luque, P. J. Roussel, J. Mitard, L.-A. Ragnarsson, L. Witters, T. Chiarella, M. Togo, N. Horiguchi et al., "Impact of single charged gate oxide defects on the performance and scaling of nanoscaled fets," in *IEEE International Reliability Physics Symposium (IRPS)*, 2012, pp. 5A–4.

[44] J. Kim, R. Rao, S. Mukhopadhyay, and C. Chuang, "Ring oscillator circuit structures for measurement of isolated nbti/pbti effects," in *Integrated Circuit Design and Technology and Tutorial, International Conference on*, 2008, pp. 163–166.

[45] J. Stathis, M. Wang, and K. Zhao, "Reliability of advanced high-k/metal-gate n-fet devices," *Microelectronics Reliability*, vol. 50, no. 9-11, pp. 1199–1202, 2010.

[46] S. Krishnappa, H. Singh, and H. Mahmoodi, "Incorporating effects of process, voltage, and temperature variation in bti model for circuit design," in *Latin American Symp. on Circuits and Systems*, 2010, pp. 236–239.

[47] R. Kanj, R. Joshi, C. Adams, J. Warnock, and S. Nassif, "An elegant hardware-corroborated statistical repair and test methodology for conquering aging effects," in *International Conference on Computer-Aided Design*, 2009, pp. 497–504.

[48] T. Siddiqua, S. Gurumurthi, and M. Stan, "Modeling and analyzing nbti in the presence of process variation," in *International Symposium on Quality Electronic Design*, 2011, pp. 1–8.

[49] S. Han, J. Choung, B. Kim, B. Lee, H. Choi, and J. Kim, "Statistical aging analysis with process variation consideration," in *International Conference on Computer-Aided Design*, 2010, pp. 412–419.

[50] W. Wang, V. Reddy, A. Krishnan, R. Vattikonda, S. Krishnan, and Y. Cao, "Compact modeling and simulation of circuit reliability for 65-nm CMOS technology," *IEEE Transactions on Device and Materials Reliability*, vol. 7, no. 4, pp. 509–517, 2007.

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

[52] http://ptm.asu.edu/reliability/.

[53] B. Linder and J. Stathis, "Statistics of progressive breakdown in ultra-thin oxides," *Microelectronic Engineering*, vol. 72, no. 1, pp. 24–28, 2004.

[54] R. Rodriguez, J. Stathis, and B. Linder, "Modeling and experimental verification of the effect of gate oxide breakdown on cmos inverters," in *International Reliability Physics Symposium Proceedings, 2003.*, 2003, pp. 11–16.

[55] J. Wang, "Current waveform simulation for CMOS VLSI circuits considering event-overlapping," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, pp. 128–138, 2000.

[56] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer cmos circuits," *Proceedings of the IEEE*, vol. 91, no. 2, pp. 305–327, 2003.

[57] A. Abdollahi, F. Fallah, and M. Pedram, "Leakage current reduction in cmos vlsi circuits by input vector control," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 12, no. 2, pp. 140 –154, feb. 2004.

[58] K. Arabi, R. Saleh, and X. Meng, "Power supply noise in SoCs: Metrics, management, and measurement," *Design & Test of Computers*, vol. 24, no. 3, pp. 236–244, 2007.

[59] http://www.Cadence.com.

[60] N. Sani, "Large signal steady state analysis of ic power grids," 2014.

[61] S. Nassif, "Power grid analysis benchmarks," in *Asia and South Pacific Design Automation Conference*, 2008, pp. 376–381.

[62] K. Haghdad and M. Anis, "Power yield analysis under process and temperature variations," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, no. 99, pp. 1–10, 2011.

[63] A. K. Coskun, T. S. Rosing, and K. C. Gross, "Utilizing predictors for efficient thermal management in multiprocessor socs," *TCAD*, vol. 28, no. 10, pp. 1503–1516, 2009.

[64] J. Parry, H. Rosten, and G. B. Kromann, "The development of component-level thermal compact models of a c4/cbga interconnect technology: The motorola powerpc 603 and powerpc 604 risc microprocessors," *IEEE Transactions on Components, Packaging, and Manufacturing Technology*, vol. 21, no. 1, pp. 104–112, 1998.

[65] Y. Cheng, *Electrothermal analysis of VLSI systems*. Springer, 2000.

[66] C. Tsai and S. Kang, "Cell-level placement for improving substrate thermal distribution," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 2, pp. 253–266, 2000.

[67] R. Tu, E. Rosenbaum, W. Chan, C. Li, E. Minami, K. Quader, P. Ko, and C. Hu, "Berkeley reliability tools-bert," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 10, pp. 1524–1534, 1993.

[68] J. Velamala, V. Ravi, and Y. Cao, "Failure diagnosis of asymmetric aging under nbti," in *International Conference on Computer-Aided Design*, 2011, pp. 428–433.

[69] K. Bowman, B. Austin, J. Eble, X. Tang, and J. Meindl, "A physical alpha-power law MOSFET model," in *International Symposium on Low power electronics and design*, 1999, pp. 218–222.

[70] H. Luo, Y. Wang, K. He, R. Luo, H. Yang, and Y. Xie, "A novel gate-level NBTI delay degradation model with stacking effect," *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*, pp. 160–170, 2007.

[71] D. Lorenz, G. Georgakos, and U. Schlichtmann, "Aging analysis of circuit timing considering nbti and hci," in *International On-Line Testing Symposium*, 2009, pp. 3–8.

[72] W. Wang, V. Reddy, B. Yang, V. Balakrishnan, S. Krishnan, and Y. Cao, "Statistical prediction of circuit aging under process variations," in *Custom Integrated Circuits Conference*, 2008, pp. 13–16.

[73] Y. Lu et al., "Statistical reliability analysis under process variation and aging effects," in *Design Automation Conference*, 2009, pp. 514–519.

[74] P. Li, "Critical path analysis considering temperature, power supply variations and temperature induced leakage," in *International Symposium on Quality Electronic Design*, 2006, pp. 6–pp.

[75] D. C. Artem Rogachev, Lu Wan, "Temperature aware statistical static timing analysis," in *ICCAD*. IEEE Computer Society, 2011, p. 900.

[76] A. Devgan and C. Kashyap, "Block-based static timing analysis with uncertainty," in *Computer-aided Design*, 2003, p. 607.

[77] S. Schwartz and Y. Yeh, "On the distribution function and moments of power sums with log-normal components," *Bell Syst. Tech. J*, vol. 61, no. 7, pp. 1441–1462, 1982.

[78] International Workshop on Logic and Synthesis 2005 Benchmark (IWLS'05), available at http://iwls.org/.

[79] http://www.Synopsys.com.

[80] NANGATE,available at http://www.nangate.com.

[81] F. Firouzi, F. Ye, K. Chakrabarty, and M. B. Tahoori, "Representative critical-path selection for aging-induced delay monitoring," in *International Test Conference*, 2013, pp. 1–10.

[82] A. H. Baba and S. Mitra, "Testing for transistor aging," in *VTS*, 2009, pp. 215–220.

[83] S. Wang, J. Chen, and M. Tehranipoor, "Representative critical reliability paths for low-cost and accurate on-chip aging evaluation," in *International Conference on Computer-Aided Design*, 2012, pp. 736–741.

[84] M. Agarwal et al., "Optimized circuit failure prediction for aging: Practicality and promise," in *International Test Conference*, 2008.

[85] E. Karl et al., "Compact in-situ sensors for monitoring negative-bias-temperature-instability effect and oxide degradation," in *International Solid-State Circuits Conference*, 2008, pp. 410–623.

[86] M. Agarwal et al., "Circuit failure prediction and its application to transistor aging," in *VLSI Test Symposium*, 2007, pp. 277–286.

[87] S. Wang, M. Tehranipoor, and L. Winemberg, "In-field aging measurement and calibration for power-performance optimization," in *Design Automation Conference*, 2011, pp. 706–711.

[88] L. Xie and A. Davoodi, "Representative path selection for post-silicon timing prediction under variability," in *Design Automation Conference*, 2010, pp. 386–391.

[89] J. Yen and L. Wang, "Simplifying fuzzy rule-based models using orthogonal transformation methods," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 29, no. 1, pp. 13–24, 1999.

[90] S. Haykin, *Adaptive filter theory (ISE).* Prentice-Hall, Englewood-Cliffs, NJ, 2003.

[91] G. Mouzouris and J. Mendel, "Designing fuzzy logic systems for uncertain environments using a singular-value-QR decomposition method," in *International Conference on Fuzzy Systems*, vol. 1, 1996, pp. 295–301.

[92] S. Chakroborty and G. Saha, "Feature selection using singular value decomposition and QR factorization with column pivoting for text-independent speaker identification," *Elsevier Speech Communication*, vol. 52, no. 9, pp. 693–709, 2010.

[93] N. Callegari, P. Bastani, L. Wang, S. Chakravarty, and A. Tetelbaum, "Path selection for monitoring unexpected systematic timing effects," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2009, pp. 781–786.

[94] International Test Conference 1999 Benchmark (ITC'99), available at http://www.cad.polito.it/downloads/tools/itc99.html.

[95] U. Fayyad, C. Reina, and P. S. Bradley, "Initialization of iterative refinement clustering algorithms," in *International Conference on Knowledge Discovery and Data Mining*, 1998, pp. 194–198.

[96] L. Lai, V. Chandra, R. Aitken, and P. Gupta, "Slackprobe: A low overhead in situ on-line timing slack monitoring methodology," in *Design, Automation and Test in Europe*, 2013, pp. 282–287.

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

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

[99] K. Kang, H. Kufluoglu, M. Alain, and K. Roy, "Efficient transistor-level sizing technique under temporal performance degradation due to nbti," in *International Conference on Computer Design*, 2007, pp. 216–221.

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

[101] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "Nbti-aware synthesis of digital circuits," in *Design Automation Conference*, 2007, pp. 370–375.

[102] Y. Wang, H. Luo, K. He, R. Luo, H. Yang, and Y. Xie, "Temperature-aware nbti modeling and the impact of standby leakage reduction techniques on circuit performance degradation," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 5, pp. 756–769, 2011.

[103] J. Abella, X. Vera, and A. Gonzalez, "Penelope: The nbti-aware processor," in *International Symposium on Microarchitecture*, 2007, pp. 85–96.

[104] Y. Wang, X. Chen, W. Wang, Y. Cao, Y. Xie, and H. Yang, "Leakage power and circuit aging cooptimization by gate replacement techniques," *IEEE Transactions on Very Large Scale Integration Systems*, no. 99, pp. 1–14, 2011.

[105] D. R. Bild, R. P. Dick, and G. E. Bok, "Static nbti reduction using internal node control," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 17, no. 4, p. 45, 2012.

[106] C. Lin, C.-H. Lin, and K.-H. Li, "Leakage and aging optimization using transmission gate-based technique," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 87–99, 2013.

[107] S. Gupta and S. S. Sapatnekar, "Gnomo: Greater-than-nominal vdd operation for bti mitigation," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2012, pp. 271–276.

[108] X. Chen, Y. Wang, Y. Cao, Y. Ma, and H. Yang, "Variation-aware supply voltage assignment for simultaneous power and aging optimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 11, pp. 2143–2147, 2012.

[109] L. Zhang and R. Dick, "Scheduled voltage scaling for increasing lifetime in the presence of NBTI," in *ASPDAC*, 2009, pp. 492–497.

[110] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "Adaptive techniques for overcoming performance degradation due to aging in cmos circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 4, pp. 603–614, 2011.

[111] E. Mintarno, J. Skaf, R. Zheng, J. B. Velamala, Y. Cao, S. Boyd, R. W. Dutton, and S. Mitra, "Self-tuning for maximized lifetime energy-efficiency in the presence of circuit aging," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 5, pp. 760–773, 2011.

[112] G. Karakonstantis, C. Augustine, and K. Roy, "A self-consistent model to estimate nbti degradation and a comprehensive on-line system lifetime enhancement technique," in *International On-Line Testing Symposium (IOLTS)*, 2010, pp. 3–8.

[113] A. Calimera, E. Macii, and M. Poncino, "NBTI-Aware power gating for concurrent leakage and aging optimization," in *International Symposium on Low power electronics and design*, 2009, pp. 127–132.

[114] A. Calimera, E. Macii, and M. Poncino, "Nbti-aware clustered power gating," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 16, no. 1, p. 3, 2010.

[115] A. Calimera, M. Loghi, E. Macii, and M. Poncino, "Partitioned cache architectures for reduced nbti-induced aging," in *Design, Automation & Test in Europe*, 2011, pp. 1–6.

[116] A. Sinkar and N. S. Kim, "Analyzing and minimizing effects of temperature variation and nbti on active leakage power of power-gated circuits," in *International Symposium on Quality Electronic Design*, 2010, pp. 791–796.

[117] A. Calimera, M. Loghi, E. Macii, and M. Poncino, "Dynamic indexing: concurrent leakage and aging optimization for caches," in *International Symposium on Low Power Electronics and Design*, 2010, pp. 343–348.

[118] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "Impact of nbti on sram read stability and design for reliability," in *International Symposium on Quality Electronic Design*, 2006, pp. 6–pp.

[119] F. Oboril, F. Firouzi, S. Kiamehr, and M. Tahoori, "Reducing NBTI-induced processor wearout by exploiting the timing slack of instructions," in *International conference on Hardware/software codesign and system synthesis*, 2012, pp. 443–452.

[120] Y. Wang, X. Chen, W. Wang, V. Balakrishnan, Y. Cao, Y. Xie, and H. Yang, "On the efficacy of input Vector Control to mitigate NBTI effects and leakage power," in *ISQED*, 2009, pp. 19–26.

[121] S. Naidu and E. Jacobs, "Minimizing stand-by leakage power in static CMOS circuits," in *Design, Automation and Test in Europe*, 2001, pp. 370–376.

[122] F. Gao and J. Hayes, "Exact and heuristic approaches to input vector control for leakage power reduction," *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 11, pp. 2564–2571, 2006.

[123] I. Cplex, "10.0," *Users Manual*, 2006.

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

[125] Y. Wang, X. Chen, W. Wang, Y. Cao, Y. Xie, and H. Yang, "Leakage power and circuit aging cooptimization by gate replacement techniques," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, no. 99, pp. 1–14, 2010.

[126] L. Cheng, L. Deng, D. Chen, and M. Wong, "A fast simultaneous input vector generation and gate replacement algorithm for leakage power reduction," in *Design Automation Conference*, 2006, pp. 117–120.

[127] J. Hennessy, D. Patterson, and D. Goldberg, *Computer architecture: a quantitative approach*. Morgan Kaufmann, 2003.

[128] N. Binkert, R. Dreslinski, L. Hsu, K. Lim, A. Saidi, and S. Reinhardt, "The M5 simulator: Modeling networked systems," *Micro, IEEE*, vol. 26, no. 4, pp. 52–60, 2006.

[129] A. Cabe et al., "Small embeddable NBTI sensors (SENS) for tracking on-chip performance decay," in *International Symposium on Quality of Electronic Design (ISQED)*, 2009.

[130] J. Tschanz et al., "Adaptive frequency and biasing techniques for tolerance to dynamic temperature-voltage variations and aging," in *International Solid-State Circuits Conference*, 2007, pp. 292–604.

[131] S. Pant and D. Blaauw, "Circuit techniques for suppression and measurement of on-chip inductive supply noise," in *European Solid-State Circuits Conference*, 2008, pp. 134–137.

[132] F. Firouzi, S. Kiamehr, M. Tahoori, and S. Nassif, "Incorporating the impacts of workload-dependent runtime variations into timing analysis," in *Design, Automation and Test in Europe*, 2013, pp. 1022–1025.

[133] E. Mintarno et al., "Workload dependent NBTI and PBTI analysis for a sub-45nm commercial microprocessor," in *International Reliability Physics Symposium (IRPS)*, 2013, pp. 3A.1.1–3A.1.6.