Design and Optimization for Resilient Energy Efficient Computing

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

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

genehmigte

Dissertation

von

Mohammad Saber Golanbari

aus dem Iran

Tag der mündlichen Prüfung: 05.02.2019

Erster Gutachter: Prof. Dr. Mehdi B. Tahoori, KIT
Zweiter Gutachter: Prof. Dr. Ulf Schlichtmann, TUM

Acknowledgement

I would like to use this opportunity to express my gratitude to everyone who supported me throughout my Ph.D. study.

I want to sincerely thank my advisor *Prof. Dr. Mehdi Baradaran Tahoori* for his continuous support of my study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all aspects of research and writing of this thesis. Furthermore, I want to thank *Prof. Dr.-Ing. Ulf Schlichtmann* for taking the Korreferat and his useful and helpful comments.

Special thanks to my family. Words cannot express how grateful I am to *my parents* for all of their support throughout my entire life. I want to express my deepest gratitude to *my wife Mahdieh* for her love and encouragement during my Ph.D.; without her support and sacrifice, it was not possible to complete this journey.

Furthermore, I would like to thank my colleagues and friends in the Chair of Dependable Nano Computing at Karlsruhe Institute of Technology for their support and encouragement. In particular, I would like to express my gratitude to *Dr.-Ing. Saman Kiamehr* and *Dr.-Ing. Mojtaba Ebrahimi*, for all their help, support, constructive criticism, and for all the great personal and professional experiences we had together.

I also want to especially thank *Dr. Sani R. Nassif, Dr.-Ing. Fabian Oboril, Dr.-Ing. Rajendra Bishnoi, Dr.-Ing. Farzad Samie, Anteneh Gebregiorgis, Arunkumar Vijayan*, and *Nour Sayed* for their contributions to parts of my research and their stimulating discussions and friendly advices.

Finally, thank you for reading my dissertation.

Mohammad Saber Golanbari
Weiherstr. 18
76227 Karlsruhe

Hiermit erkläre ich an Eides statt, dass ich die von mir vorgelegte Arbeit selbstständig verfasst habe, dass ich die verwendeten Quellen, Internet-Quellen und Hilfsmittel vollständig angegeben haben 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.

Karlsruhe, Dezember 2018
Mohammad Saber Golanbari

Abstract

Modern electronic devices are an indispensable part of our everyday life. A major enabler for such integration is the exponential increase of the computation capabilities as well as the drastic improvement in the energy efficiency over the last 50 years, commonly known as Moore's law. In this regard, the demand for energy-efficient digital circuit, especially for application domains such as the Internet of Things (IoT), has faced an enormous growth. Since the power consumption of a circuit highly depends on the supply voltage, aggressive supply voltage scaling to the near-threshold voltage region, also known as Near-Threshold Computing (NTC), is an effective way of increasing the energy efficiency of a circuit by an order of magnitude.

Along with the attractive energy benefits, the NTC comes with a variety of design challenges. Reducing the supply voltage to the near-threshold voltage region also increases the sensitivity of the circuits to different variabilities, such as process variation, voltage fluctuation, and temperature variation, by more than one order of magnitude. Such large variation impacts hinder the reliability of the NTC circuits and are the main challenges towards widespread usage of NTC circuits. The traditional method of dealing with variabilities and the conventional designs and methods which are typically used in the nominal voltage range are inefficient for NTC circuit design, because they cannot deal with large performance variation and sensitivity of circuits in the near-threshold voltage region. Therefore, dealing with the NTC challenges requires a new design paradigm as well as design automation flow for the NTC.

The objective of this thesis is to provide a holistic approach for NTC design by tackling the major challenges, in the form of comprehensive design and design automation flow for NTC, mostly at circuit and logic levels. This also includes an in-depth analysis of the reliability issues for the NTC, as well as proposing optimization techniques for better reliability, performance, and energy efficiency. The contributions of this thesis are:

Variation-aware logic synthesis and timing closure: Satisfying timing and reliability requirements for NTC circuits is a challenging task. The impact of variabilities might be seen as large performance variation, which mandates expensive timing margins, or as high functional failure rates, for example, due to hold-time violations. The traditional approach to deal with circuit performance variation is to increase timing margins. However, this is inefficient for the NTC due to the extent of variations and the increased contribution of leakage energy.

This thesis proposes a *variation-aware synthesis and timing closure*, which reduces the sensitivity to variation and improves energy-efficiency, performance, and reliability while reducing the overheads of timing closure [1, 2]. Simulation results show that our proposed flow reduces the delay variation by 87% and improves the performance and energy efficiency by 25% and 7.4%, respectively, at the expense of only 4.8% area overhead.

Cross-layer reliability, energy efficiency, and performance optimization of data paths: Cross-layer analysis of processor data paths, from compiler all the way to circuit design, could reveal potential optimization approaches. A data path is a combination of several functional units with the ability to execute various instructions. Our analysis shows that the required time for executing an instruction at low supply voltage could be widely different which allows categorizing the instructions into fast and slow instructions. Moreover, the functional unit instructions can be categorized into frequently used and rarely used instructions.

This thesis proposes an *instruction multi-cycling* method to improve energy efficiency and resiliency of functional units [3]. In addition, we propose a *functional unit partitioning* method, in which a functional unit is partitioned into several smaller units to support fine-grained power-gating of the units which implement the rarely used instructions [4]. The proposed methods significantly improve the circuit timing, and at the same time considerably limit leakage energy, by employing a combination of circuit redesign and code replacement techniques. Simulation results show that the proposed methods improve performance and energy efficiency of an Arithmetic Logic Unit by 19% and 43%, respectively. Furthermore, the improved performance of the optimized circuits can be traded to improving the reliability [5, 6].

Post-fabrication calibration and runtime tuning: Process and runtime variations greatly affect the Minimum Energy Point (MEP) of NTC circuits, i.e., the supply voltage which leads to the best energy efficiency. Therefore, it is necessary to calibrate each NTC circuit to the correct operating condition (its MEP) based on variations to improve the energy efficiency.

This thesis proposes a *post-fabrication and runtime tuning* method which calibrates each digital circuit to its best MEP based on the post-fabrication measurements, such as speed and power consumption. The proposed method obtains the MEP in a per-chip basis to account for the impacts of process variation, and dynamically adapts supply voltage and frequency to account for time-dependent variations, such as workload and temperature. For this, the firmware of a chip is updated with a regression model which can determine the best MEP based on the data collected during the runtime about workload and temperature. The regression model is unique for each chip and is created based on the post-fabrication measurements. Simulation results show that the proposed method has high prediction accuracy and energy efficiency similar to hardware-implemented methods, but without imposing hardware overhead [7, 8].

Selective flip-flop optimization Ultra-low-voltage circuits may need to operate at nominal supply voltage mode to satisfy timing constraints required by running applications. In this case, the circuit is exposed to aging impact which degrades the transistors by increasing their threshold voltages. Our analysis shows that some flip-flops store a constant value, either '0' or '1', for a long time leading to a severe impact of aging. Compared to other components, the flip-flops are more sensitive to the aging impact and may fail to store a value correctly within the timing constraints. Furthermore, a slight voltage-drop could lead to timing violations if the aging-affected flip-flops are parts of the critical paths.

For this, this thesis proposes a selective flip-flop optimization methodology, in which the circuit is optimized for resiliency against the impacts of static aging and voltage drop. The proposed method first generates optimized variation-resilient flip-flops and adds them to the standard cell library. Then, timing-critical flip-flops that experience severe aging or voltage-drop impacts are replaced by the variation-resilient versions to improve the timing and reliability [9, 10]. Simulation results show that the proposed method prolongs the lifetime of a processor by 37% while imposing less than 0.1% leakage overhead.

While the NTC potentially offers large energy efficiency, the aforementioned challenges of the NTC design including high sensitivity to variations leading to reliability issues prevent the widespread application of the NTC to the emerging applications such as the IoT. In this dissertation, we proposed methods to address such challenges by improving the reliability and efficiency of NTC designs. Therefore, the contributions in this thesis, and other published work not included in this thesis [11–17], are key elements in making the NTC designs widely acceptable in the mainstream.

Zusammenfassung

Heutzutage sind moderne elektronische Systeme ein integraler Bestandteil unseres Alltags. Dies wurde unter anderem durch das exponentielle Wachstum der Integrationsdichte von integrierten Schaltkreisen ermöglicht zusammen mit einer Verbesserung der Energieeffizienz, welche in den letzten 50 Jahren stattfand, auch bekannt als Moore's Gesetz. In diesem Zusammenhang ist die Nachfrage von energieeffizienten digitalen Schaltkreisen enorm angestiegen, besonders in Anwendungsfeldern wie dem Internet of Things (IoT). Da der Leistungsverbrauch von Schaltkreisen stark mit der Versorgungsspannung verknüpft ist, wurden effiziente Verfahren entwickelt, welche die Versorgungsspannung in den nahen Schwellenspannung-Bereich skalieren, zusammengefasst unter dem Begriff Near-Threshold-Computing (NTC). Mithilfe dieser Verfahren kann eine Erhöhung der Energieeffizienz von Schaltungen um eine ganze Größenordnung ermöglicht werden.

Neben der verbesserten Energiebilanz ergeben sich jedoch zahlreiche Herausforderungen was den Schaltungsentwurf angeht. Zum Beispiel führt das Reduzieren der Versorgungsspannung in den nahen Schwellenspannungsbereich zu einer verzehnfachten Erhöhung der Sensibilität der Schaltkreise gegenüber Prozessvariation, Spannungsfluktuationen und Temperaturveränderungen. Die Einflüsse dieser Variationen reduzieren die Zuverlässigkeit von NTC Schaltkreisen und sind ihr größtes Hindernis bezüglich einer umfassenden Nutzung. Traditionelle Ansätze und Methoden aus dem nominalen Spannungsbereich zur Kompensation von Variabilität können nicht effizient angewandt werden, da die starken Performance-Variationen und Sensitivitäten im nahen Schwellenspannungsbereich dessen Kapazitäten übersteigen. Aus diesem Grund sind neue Entwurfsparadigmen und Entwurfsautomatisierungskonzepte für die Anwendung von NTC erforderlich.

Das Ziel dieser Arbeit ist die zuvor erwähnten Probleme durch die Bereitstellung von ganzheitlichen Methoden zum Design von NTC Schaltkreisen sowie dessen Entwurfsautomatisierung anzugehen, welche insbesondere auf der Schaltungs- sowie Logik-Ebene angewandt werden. Dabei werden tiefgehende Analysen der Zuverlässigkeit von NTC Systemen miteinbezogen und Optimierungsmethoden werden vorgeschlagen welche die Zuverlässigkeit, Performance und Energieeffizienz verbessern. Die Beiträge dieser Arbeit sind wie folgt:

Schaltungssynthese und Timing Closure unter Einbezug von Variationen: Das Einhalten von Anforderungen an das zeitliche Verhalten und Zuverlässigkeit von NTC ist eine anspruchsvolle Aufgabe. Die Auswirkungen von Variabilität kommen bei starken Performance-Schwankungen, welche zu teuren zeitlichen Sicherheitsmargen führen, oder sich in Hold-Time Verstößen ausdrücken, verursacht durch funktionale Störungen, zum Vorschein. Die konventionellen Ansätze beschränken sich dabei alleine auf die Erhöhung von zeitlichen Sicherheitsmargen. Dies ist jedoch sehr ineffizient für NTC, wegen dem starken Ausmaß an Variationen und den erhöhten Leckströmen.

In dieser Arbeit wird ein Konzept zur Synthese und Timing Closure von Schaltkreisen unter Variationen vorgestellt, welches sowohl die Sensitivität gegenüber Variationen reduziert als auch die Energieeffizienz, Performance und Zuverlässigkeit verbessert und zugleich den Mehraufwand von Timing Closures [1, 2] verringert. Simulationsergebnisse belegen, dass unser vorgeschlagener Ansatz die Verzögerungszeit um 87% reduziert und die Performance und En-

ergieeffizienz um 25% beziehungsweise 7.4% verbessert, zu Kosten eines erhöhten Flächenbedarfs von 4.8%.

Schichtübergreifende Zuverlässigkeits-, Energieeffizienz- und Performance-Optimierung von Datenpfaden: Schichtübergreifende Analyse von Prozessor-Datenpfaden, welche den ganzen Weg spannen vom Kompilierer zum Schaltungsentwurf, kann potenzielle Optimierungsansätze aufzeigen. Ein Datenpfad ist eine Kombination von mehreren funktionalen Einheiten, welche diverse Instruktionen verarbeiten können. Unsere Analyse zeigt, dass die Ausführungszeiten von Instruktionen bei niedrigen Versorgungsspannungen stark variieren, weshalb eine Klassifikation in schnelle und langsame Instruktionen vorgenommen werden kann. Des Weiteren können funktionale Instruktionen als häufig und selten genutzte Instruktionen kategorisiert werden.

Diese Arbeit stellt eine Multi-Zyklen-Instruktionen-Methode vor, welche die Energieeffizienz und Belastbarkeit von funktionalen Einheiten erhöhen kann [3]. Zusätzlich stellen wir einen Partitionsalgorithmus vor, welcher ein fein-granulares Power-gating von selten genutzten Einheiten ermöglicht [4] durch Partition von einzelnen funktionalen Einheiten in mehrere kleinere Einheiten. Die vorgeschlagenen Methoden verbessern das zeitliche Schaltungsverhalten signifikant, und begrenzen zugleich die Leckströme beträchtlich, durch Einsatz einer Kombination von Schaltungs-Redesign- und Code-Replacement-Techniken. Simulationsresultate zeigen, dass die entwickelten Methoden die Performance und Energieeffizienz von arithmetisch-logischen Einheiten (ALU) um 19% beziehungsweise 43% verbessern. Des Weiteren kann der Zuwachs in Performance der optimierten Schaltungen in eine Verbesserung der Zuverlässigkeit umgewandelt werden [5, 6].

Post-Fabrication und Laufzeit-Tuning: Prozess- und Laufzeitvariationen haben einen starken Einfluss auf den Minimum Energy Point (MEP) von NTC-Schaltungen, welcher mit der energieeffizientesten Versorgungsspannung assoziiert ist. Es ist ein besonderes Anliegen, die NTC-Schaltung nach der Herstellung (post-fabrication) so zu kalibrieren, dass sich die Schaltung im MEP-Zustand befindet, um die beste Energieeffizient zu erreichen.

In dieser Arbeit, werden Post-Fabrication und Laufzeit-Tuning vorgeschlagen, welche die Schaltung basierend auf Geschwindigkeits- und Leistungsverbrauch-Messungen nach der Herstellung auf den MEP kalibrieren. Die vorgestellten Techniken ermitteln den MEP per Chip-Basis um den Einfluss von Prozessvariationen mit einzubeziehen und dynamisch die Versorgungsspannung und Frequenz zu adaptieren um zeitabhängige Variationen wie Workload und Temperatur zu adressieren. Zu diesem Zweck wird in die Firmware eines Chips ein Regression-Modell integriert, welches den MEP basierend auf Workload- und Temperatur-Messungen zur Laufzeit extrahiert. Das Regressions-Modell ist für jeden Chip einzigartig und basiert lediglich auf Post-Fabrication-Messungen. Simulationsergebnisse zeigen das der entwickelte Ansatz eine sehr hohe prognostische Treffsicherheit und Energieeffizienz hat, ähnlich zu hardware-implementierten Methoden, jedoch ohne hardware-seitigen Mehraufwand [7, 8].

Selektierte Flip-Flop Optimierung: Ultra-Low-Voltage Schaltungen müssen im nominalen Versorgungsspannungs-Mode arbeiten um zeitliche Anforderungen von laufenden Anwendungen zu erfüllen. In diesem Fall ist die Schaltung von starken Alterungsprozessen betroffen, welche die Transistoren durch Erhöhung der Schwellenspannungen degradieren. Unsere tiefgehenden Analysen haben gezeigt das gewisse Flip-Flop-Architekturen von diesen Alterungserscheinungen beeinflusst werden indem fälschlicherweise konstante Werte (‚0' oder ‚1') für eine lange Zeit gespeichert sind. Im Vergleich zu anderen Komponenten sind Flip-Flops sensitiver zu Alterungsprozessen und versagen unter anderem dabei einen neuen Wert innerhalb

des vorgegebenen zeitlichen Rahmens zu übernehmen. Außerdem kann auch ein geringfügiger Spannungsabfall zu diesen zeitlichen Verstößen führen, falls die betreffenden gealterten Flip-Flops zum kritischen Pfad zuzuordnen sind.

In dieser Arbeit wird eine selektiver Flip-Flop-Optimierungsmethode vorgestellt, welche die Schaltungen bezüglich Robustheit gegen statische Alterung und Spannungsabfall optimieren. Dabei werden zuerst optimierte robuste Flip-Flops generiert und diese dann anschließend in die Standard-Zellen-Bibliotheken integriert. Flip-Flops, die in der Schaltung zum kritischen Pfad gehören und Alterung sowie Spannungsabfall erfahren, werden durch die optimierten robusten Versionen ersetzt, um das Zeitverhalten und die Zuverlässigkeit der Schaltung zu verbessern [9, 10]. Simulationsergebnisse zeigen, dass die erwartete Lebenszeit eines Prozessors um 37% verbessert werden kann, während Leckströme um nur 0.1% erhöht werden.

Während NTC das Potenzial hat große Energieeffizienz zu ermöglichen, ist der Einsatz in neue Anwendungsfeldern wie IoT wegen den zuvor erwähnten Problemen bezüglich der hohen Sensitivität gegenüber Variationen und deshalb mangelnder Zuverlässigkeit, noch nicht durchsetzbar. In dieser Dissertation und in noch nicht publizierten Werken [11–17], stellen wir Lösungen zu diesen Problemen vor, die eine Integration von NTC in heutige Systeme ermöglichen.

Contents

Glossary

AES Advanced Encryption Standard.

AHC Agglomerative Hierarchical Clustering.

ALU Arithmetic Logic Unit.

BRR Bayesian Ridge Regression.

BTI Bias Temperature Instability.

CDF Cumulative Distribution Function.

CMOS Complementary Metal Oxide Semiconductor.

CPS Cyber-Physical-Systems.

CTS Clock Tree Synthesis.

DIBL Drain-induced Barrier Lowering.

DRV Design Rule Violation.

DVFS Dynamic Voltage and Frequency Scaling.

ECO Engineering Change Order.

ECSM Effective Current Source Model.

EDA Electronic Design Automation.

EDP Energy-Delay Product.

EM Electromigration.

FDSOI Fully-Depleted Silicon-On-Insulator.

FPU Floating Point Unit.

GIDL Gate-Induced-Drain-Leakage.

HCI Hot Carrier Injection.

IoT Internet of Things.

ISA Instruction Set Architecture.

LER Line-Edge Roughness.

LUT Look-Up-Tables.

MBU Multiple Bit Upsets.

MEP Minimum Energy Point.

MGG Metal Grain Granularity.

MOS Metal Oxide Semiconductor.

MOSFET Metal-Oxide-Semiconductor Field Effect Transistor.

MTTF Mean Time To Failure.

NTC Near-Threshold Computing.

NTV Near-Threshold Voltage.

OLS Ordinary Least Square.

OTF Oxide Thickness Fluctuation.

PCHIP Piecewise Cubic Hermite Interpolating Polynomial.

PDN Power Delivery Network.

RDF Random Dopant Fluctuation.

RMSE Root Mean Square Error.

RSCE Reverse Short Channel Effect.

RTN Random Telegraph Noise.

S-BTI Static BTI.

SCE Short-Channel Effect.

SGBRT Stochastic Gradient Boosted Regression Trees.

SoC Systems-on-Chip.

SP Signal Probability.

SQP Sequential Quadratic Programming.

SSTA Statistical Static Timing Analysis.

STA Static Timing Analysis.

TDDB Time Dependent Dielectric Breakdown.

VCD Voltage Change Dump.

VLSI Very-Large-Scale Integration.

WPE Well-Proximity Effect.

List of Figures

List of Tables

List of own publications included in this thesis

[1] M. S. Golanbari, S. Kiamehr, M. Ebrahimi, and M. B. Tahoori, "Variation-aware Near-Threshold Circuit Synthesis," in *Design, Automation & Test in Europe Conference (DATE)*, 2016.

[2] M. S. Golanbari, S. Kiamehr, and M. B. Tahoori, "Hold-time Violation Analysis and Fixing in Near-Threshold Region," in *International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2016.

[3] M. S. Golanbari, A. Gebregiorgis, F. Oboril, S. Kiamehr, and M. B. Tahoori, "A Cross-Layer Approach for Resiliency and Energy Efficiency in Near Threshold Computing," in *International Conference on Computer-Aided Design (ICCAD)*, 2016.

[4] M. S. Golanbari, A. Gebregiorgis, E. Moradi, S. Kiamehr, and M. B. Tahoori, "Balancing resiliency and energy efficiency of functional units in ultra-low power systems," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2018.

[5] M. S. Golanbari and M. B. Tahoori, "Design flows for resilient energy-efficient systems," in *International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2017.

[6] M. S. Golanbari and M. B. Tahoori, "Optimizing Datapaths for Near Threshold Computing," in *International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2018.

[7] M. S. Golanbari, S. Kiamehr, F. Oboril, A. Gebregiorgis, and M. B. Tahoori, "Post- Fabrication Calibration of Near-Threshold Circuits for Energy Efficiency," in *International Symposium on Quality Electronic Design (ISQED)*, 2017.

[8] M. S. Golanbari and M. B. Tahoori, "Runtime adjustment of IoT system-on-chips for minimum energy operation," in *Design Automation Conference (DAC)*, 2018.

[9] M. S. Golanbari, S. Kiamehr, M. Ebrahimi, and M. B. Tahoori, "Aging guardband reduction through selective flip-flop optimization," in *IEEE European Test Symposium (ETS)*, 2015.

[10] M. S. Golanbari, S. Kiamehr, M. Ebrahimi, and M. B. Tahoori, "Selective Flip-Flop Optimization for Reliable Digital Circuit Design," submitted to *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*.

List of own publications not included in this thesis

[11] M. S. Golanbari, S. Kiamehr, M. B. Tahoori, and S. Nassif, "Analysis and optimization of flip-flops under process and runtime variations," in *International Symposium on Quality Electronic Design (ISQED)*, 2015.

[12] M. S. Golanbari, S. Kiamehr, and M. B. Tahoori, "Resilient Flip-Flop Design under Process and Runtime Variations," in *SELSE*, 2015.

[13] A. Gebregiorgis, M. S. Golanbari, S. Kiamehr, F. Oboril, and M. B. Tahoori, "Maximizing Energy Efficiency in NTC by Variation-Aware Microprocessor Pipeline Optimization," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 272–277, 2016.

[14] S. Kiamehr, M. Ebrahimi, M. S. Golanbari, and M. B. Tahoori, "Temperature-aware dynamic voltage scaling to improve energy efficiency of near-threshold computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 7, 2017.

[15] S. Kiamehr, M. S. Golanbari, and M. B. Tahoori, "Leveraging aging effect to improve SRAM-based true random number generators," in *Design, Automation & Test in Europe (DATE)*, pp. 882–885, 2017.

[16] M. S. Golanbari, N. Sayed, M. Ebrahimi, M. H. M. Esfahany, S. Kiamehr, and M. B. Tahoori, "Aging-aware coding scheme for memory arrays," in *IEEE European Test Symposium (ETS)*, 2017.

[17] M. S. Golanbari, S. Kiamehr, R. Bishnoi, and M. B. Tahoori, "Reliable memory PUF design for low-power applications," in *International Symposium on Quality Electronic Design (ISQED)*, 2018.

[18] M. Ebrahimi, M. H. Moshrefpour, M. S. Golanbari, and M. B. Tahoori, "Fault injection acceleration by simultaneous injection of non-interacting faults," in *Design Automation Conference(DAC)*, p. 25, 2016.

[19] S. M. Nair, R. Bishnoi, M. S. Golanbari, F. Oboril, and M. B. Tahoori, "VAET-STT: A variation aware estimator tool for STT-MRAM based memories," in *Design, Automation & Test in Europe (DATE)*, pp. 1460–1465, 2017.

[20] S. M. Nair, R. Bishnoi, M. S. Golanbari, F. Oboril, F. Hameed, and M. B. Tahoori, "Vaet-stt: Variation aware stt-mram analysis and design space exploration tool," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 7, pp. 1396–1407, 2018.

[21] A. T. Erozan, M. S. Golanbari, R. Bishnoi, J. Aghassi-Hagmann, and M. B. Tahoori, "Design and evaluation of physical unclonable function for inorganic printed electronics," in *International Symposium on Quality Electronic Design (ISQED)*, pp. 419–424, 2018.

[22] F. Rasheed, M. S. Golanbari, G. C. Marques, M. B. Tahoori, and J. Aghassi-Hagmann, "A smooth EKV-based DC model for accurate simulation of printed transistors and their process variations," *IEEE Transactions on Electron Devices*, vol. 65, no. 2, pp. 667–673, 2018.

[23] A. T. Erozan, G. C. Marques, M. S. Golanbari, R. Bishnoi, S. Dehm, J. Aghassi-Hagmann, and M. B. Tahoori, "Inkjet-Printed EGFET-Based Physical Unclonable Function–Design, Evaluation, and Fabrication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, no. 99, pp. 1–12, 2018.

[24] G. Tshagharyan, G. Harutyunyan, Y. Zorian, A. Gebregiorgis, M. S. Golanbari, R. Bishnoi, and M. B. Tahoori, "Modeling and Testing of Aging Faults in FinFET Memories for Automotive Applications," in *IEEE International Test Conference (ITC)*, 2018.

[25] D. Weller, M. Hefenbrock, M. Golanbari, M. Beigl, and M. Tahoori, "Bayesian Optimized Importance Sampling for High Sigma Failure Rate Estimation," in *Design, Automation & Test in Europe (DATE)*, 2019.

1 Introduction

Since the advent of electronic digital computing, relentless technology scaling has enabled an exponential improvement in computation capability while decreasing the cost and power consumption. Gordon Moore predicted in 1965 that the number of transistors in integrated circuits would double every year[1] to address the ever-increasing demand for higher computation power [36]. Such continuous growth in computation capability over more than 5 decades affected almost all aspects of human life, including but not limited to industry, business, health-care, government, and society which effectively started the Information Age, and made digital computing circuits inseparable part of our everyday life.

Moore's law has faced several technological challenges and has been slowed down in the past decade [34, 37, 38], however, still more transistors can be integrated on every new technology generation down to 3nm node [39, 40]. To avoid exponential growth in the power density, various parameters including supply voltage of digital circuits have been scaled according to Dennard's scaling law [41]. However, the supply voltage did not scale at the same pace for about one decade (since 2005–2006) due to technological challenges associated with nanometer-scale devices [42]. Since then, various architectural and design directions have been actively explored including many-core computation, parallel processing, and 3D integration to provide more computation power [38–40]. However, the main challenges caused by the end of Dennard's scaling are *energy efficiency* and *power density*, which cannot be addressed by such approaches [43–45]. Without proper addressing of the increasing power density due to technology scaling, it is not possible to utilize all the components of a chip at the same time due to overheating, a problem commonly known as Dark Silicon [44, 46], which diminishes the benefits of scaling. Therefore, reducing the power density through improving the energy efficiency is pivotal for future digital circuits.

In fact, *energy-efficient computing* has already become a primary requirement in various application domains. At one end of the spectrum, the growing Internet of Things (IoT) applications, with an expected 20 billion connected devices by 2020 [47–50], are continually looking for more energy efficiency. These IoT devices are expected to operate on limited energy sources such as batteries or energy harvesting sources. High-performance servers and data centers, at the other end, are responsible for a significant[2] amount of consumed electricity worldwide [51]. A large portion of the cost of data centers is directly or indirectly due to the energy consumption of digital circuits [51]; hence, it is necessary to improve the energy efficiency of high-performance computing as well.

Power consumption of digital circuits has two components: *dynamic power* consumption, due to circuit activity and computation, and *leakage power* consumption, due to slight leakage current of transistors. Reducing the power density can be achieved through various approaches such as optimizing design techniques, employing power management strategies, improving the technology, and scaling supply voltage [52]. Each of these approaches aims at reducing dynamic power, leakage power, or both. In this regard, optimizing Instruction Set Architecture (ISA), scheduling methods, pipeline design, and synthesis methodologies have already enabled vast improvements in the energy efficiency [53]. Techniques such as clock-gating and power-

[1] This prediction was adjusted afterwards for a few times to reflect the real progress [34, 35].

[2] About 1.4% of the consumed electricity worldwide in 2011, and growing in much faster speed compared to other electricity consumers

gating are widely used in existing digital circuits to cut down dynamic and leakage power of the idle components [54]. Dynamic Voltage and Frequency Scaling (DVFS) promotes supply voltage and speed adaptation depending on the workload to reduce power consumption under low workload [55, 56]. Aggressive supply voltage reduction down to sub-threshold region is known as an important instrument which reduces power consumption by several orders of magnitude and improves energy efficiency [45, 52, 57–59]. Intel has demonstrated ultra-low power characteristics using such aggressive voltage scaling by its experimental IA32 processor, which can run Windows and Linux on a small solar panel [28]. In addition to extensive power reduction, voltage scaling degrades circuit speed significantly. Many applications have performance constraints that prevent them from operating in the sub-threshold region.

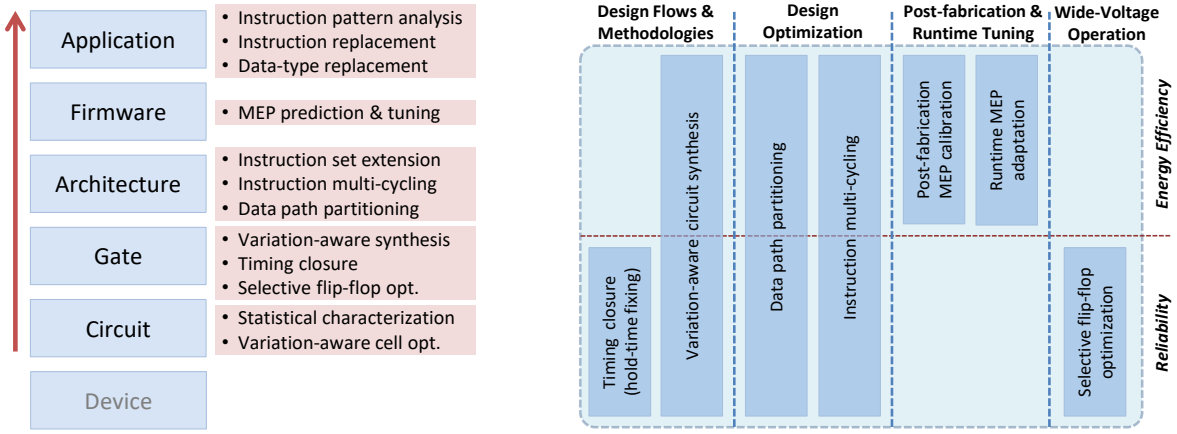
By operating digital circuit at supply voltages close to the threshold voltage of the transistors, which is known as *Near-Threshold Computing (NTC)*, it is still possible to gain very high energy efficiency and achieve enough performance for many applications in the IoT domain [60, 61]. Additionally, many-core computation based on NTC is an attractive approach to improve energy efficiency while satisfying the computational demands[3] of data centers [43, 62–64]. Therefore, NTC is considered an attractive paradigm for improving the energy efficiency in modern technology nodes [65], if the associated reliability challenges are addressed. These reliability challenges are typically caused by enormous sensitivity of NTC circuits to variability sources and complicate NTC circuit design and operation.

1.1 Motivation and objective

In addition to energy efficiency, meeting high-reliability standards is a primary concern for many modern applications of digital circuits including those in the area of Cyber-Physical-Systems (CPS) [43, 66, 67]. As reported by NASA, product wearout and failure rates increase due to technology scaling [68]. Safety standards such as ISO 26262 in the field of automotive industry set high standards for the functional safety of electronic systems used in the produced automobiles. The reliability of digital circuits is threatened by variability sources, such as process and runtime variations, as well as radiation-induced soft-errors [69]. The state of transistors may change due to the accumulated charges generated by particle strikes, which can flip the values stored in memory and the state of logic gates, and may induce transient soft-errors in the circuit output [70]. Such soft-error problems can be avoided by incorporating cross-layer analysis and mitigation methods through advanced error correction and detection techniques [71]. Process variation during circuit fabrication results in a slight difference between the fabricated transistors, which is a major reliability concern in the nanometer era [72]. Moreover, runtime fluctuations in temperature and supply voltage as well as transistor aging alter the behavior of the fabricated components, and may lead to catastrophic failures [73].

A primary barrier towards the NTC is that reliability issues aggravate when the supply voltage is decreased, due to the higher sensitivity to noise and variations at lower supply voltages [59]. Such elevated sensitivity to process and runtime variations brings about several design challenges, including timing issues and high failure rate, which lead to daunting reliability issues. For example, the variation in the delay of a gate in the *Near-Threshold Voltage (NTV)* due to process variation is 20× larger than in the nominal supply voltage [60] and the sensitivity of the circuit speed to temperature variation in the NTV region is shown to be 20× larger than the nominal supply voltage region [74]. In summary, modern NTC circuits are susceptible to even slight process and runtime variations due to nanoscale dimension of transistors and low supply voltage [60, 74].

[3] For highly parallel workloads.



(a) Scope of this thesis and contributions at different abstraction levels

(b) Contributions at different optimization dimensions

Figure 1.1: Scope and contributions of this thesis regarding NTC design challenges.

Even though circuit designs and methodologies have been evolving over past decades to adapt the need for continuous technology scaling, they are still inefficient in addressing variability impacts in the NTV region due to the extent of variations [74–76]. Therefore, designs and methodologies for addressing the reliability issues need to be revisited for resilient and energy efficient NTC circuit design. *The objective of this thesis is to provide a holistic approach for NTC design by tackling the major challenges, in the form of comprehensive design and design automation flow. This includes an in-depth analysis of the reliability issues in the NTV region, followed by optimizing designs and methodologies for better reliability, performance, and energy efficiency.*

1.2 Contributions of this thesis

As stated before, the focus of this thesis is on the reliability challenges associated with the NTC. We address the reliability challenges of NTC circuit design and improve the energy efficiency at different abstraction-levels from circuit-level to application-level, as shown in Figure 1.1. The contributions can be classified into four main categories, which collectively improve the energy efficiency, reliability, and performance: I) Design flows and methodologies, II) Design optimization, III) Post-fabrication and runtime tuning, and IV) Wide-voltage operation. In each of these categories, we target a specific problem and propose methods to address that.

1.2.1 Design flows and methodologies

Electronic Design Automation (EDA) flows and methodologies have enabled large-scale integration of transistors on chips over past decades. They are used for analysis, design, and optimization of digital circuits. These tools are widely used in various chip design stages from system specification until manufacturing and test. Due to variability impacts, fabricated chips may deviate from designated specifications. Some design automation tools can consider variability impacts through corner-case analysis, in which worst and best corners are introduced to the tool. This approach is practical when variability impacts are not significant, hence, a full statistical analysis is typically evaded due to its complexity. However, due to exorbitant variation in characteristics of NTC circuits, design automation tools and methodologies should pay attention to statistical impacts of variability sources and consider them during circuit design

and optimization.

An essential aspect of digital circuits is timing, which is heavily influenced by both process and runtime variation in the NTV region. This results in undesired performance fluctuation among the components, such as gates and flip-flops, which may cause timing violations or functional failures. For that, it is necessary to perform accurate timing analysis and optimize the circuit in order to minimize the timing variations and violations. Existing synthesis and timing optimization tools rely on corner-case Static Timing Analysis (STA) for circuit optimization and timing closure [77]. In addition to that, timing margins are considered to deal with timing variations. However, such corner-case design methodology is very pessimistic in the NTV region, as the gap between worst-case and best-case corners could be huge. Therefore, this approach is either inefficient or too costly in the NTV region due to the extent of variations [60].

This thesis tackles the timing challenge at circuit-level by proposing a synthesis and timing closure methodology, which considers the variability impacts through standard cell library engineering and Statistical Static Timing Analysis (SSTA). We show that such variation-aware methodology is necessary for NTC design and can effectively improve circuit energy efficiency and timing leading to better reliability.

1.2.2 Design optimization

Supply voltage scaling to the NTV region and below changes the assumptions for circuit design and optimization [76]. Therefore, NTC designs need to be particularly optimized based on new paradigms. As an example, leakage power constitutes a significant portion of total power consumption in the NTV region due to the reduced supply voltage [28, 61]. This means that reducing leakage power is more rewarding in NTC compared to the nominal supply voltage.

This thesis proposes a cross-layer approach to optimize NTC processor designs by improving the reliability, energy efficiency, and performance of data paths. A processor data path includes various functional units and data processing units such as Arithmetic Logic Unit (ALU) and consumes a significant amount of power and area [54]. A cross-layer analysis of data paths from application-level to gate-level is performed to reveal the optimization opportunities for NTC operation. For example, instruction pattern analysis at application-level reveals the frequently utilized instructions and rarely utilized instructions, and a timing analysis at circuit-level determines the required time to execute each instruction. Accordingly, we propose *functional unit partitioning* and *instruction multi-cycling* methods which optimize energy efficiency, reliability, and even speed of NTC data paths.

1.2.3 Post-fabrication and runtime tuning

Due to the extent of variability impacts, still large difference between fabricated chip instances and under different temperatures are observed [61, 74]. As an example, the speed of the circuit may change by $\pm 2\times$ under $\pm 50°C$ temperature variation. This will affect the energy efficiency and reliability such that the most efficient operating voltage of an NTC circuit, commonly known as its Minimum Energy Point (MEP), may fluctuate vastly [78]. Therefore, it is necessary to calibrate each fabricated chip to the correct operating condition (supply voltage and frequency at the MEP) and to the variations to improve the energy efficiency and reliability. In this regard, the existing methods can be categorized into analytical methods, which are not effective in NTC, and hardware-implemented closed-loop energy efficiency monitoring methods, which could be too expensive for ultra-low power systems.

This thesis introduces a low-cost post-fabrication calibration and runtime tuning method, with zero to negligible hardware cost. In this method, we predict the most energy-efficient point

for each fabricated chip based on a dataset of previously fabricated test chips. The prediction is done using machine-learning techniques and based on post-fabrication measurements. Our proposed method is implemented at firmware-level and can tune NTC circuits at runtime for the best energy efficiency with high accuracy.

1.2.4 Wide-voltage operation

Ultra-low-voltage circuits may need to operate at super-threshold voltage mode to satisfy specific timing constraints enforced by the running applications. In this case, the circuit is exposed to several runtime variabilities such as transistor aging which degrade the transistors by increasing their threshold voltages. Therefore, it is crucial to perform reliability analysis to evaluate the impact of such variabilities on vulnerable circuit components such as flip-flops. Such reliability analysis determines weak points of a design, i.e., the components which are profoundly impacted by variabilities and are crucial to overall system reliability.

This thesis proposes a selective flip-flop optimization approach. Accordingly, we optimize the reliability of a limited number of flip-flops, with the target of improving overall circuit reliability and lifetime. Our analysis shows that some flip-flops store a constant value, either '0' or '1', for a long time leading to a severe impact of aging. Compared to other components, the flip-flops are also more sensitive to the aging impact and may fail to store a value correctly within the timing constraints. Furthermore, a slight voltage drop could lead to timing violations if the aging-affected flip-flops are parts of the critical paths. These vulnerable flip-flops are then replaced with optimized counterparts, which are more resilient against aging and voltage fluctuation. Simulation results show that this approach can significantly improve the circuit lifetime.

1.3 Structure of this Thesis

This chapter presented the motivation and contributions of this thesis. The rest of the thesis is organized in six chapters:

- Chapter 2 provides the *preliminaries* on reliability challenges associated with energy-efficient computing. There, the most important variability sources for the NTC are explained with an overview of their impacts on NTC circuits. Additionally, the state-of-the-art related to resilient energy-efficient computing is reviewed.

- Chapter 3 presents the first contribution in the domain of design automation flows and methodologies. We start with explaining the detrimental impact of variabilities on circuit timing and present a *variation-aware synthesis and timing closure methodology*, which improves the circuit resiliency under large variability impacts.

- In Chapter 4, several opportunities for NTC design optimization are explored. We perform an analysis on data paths from application-level down to gate-level. Then, we apply a cross-layer methodology to co-optimize resiliency, energy efficiency, and performance of processor data paths.

- We emphasize the need for adaptation of NTC circuits to process and runtime variations in Chapter 5, by analyzing the impacts of variabilities on energy efficiency and reliability of NTC chips. Based on this analysis, a *post-fabrication calibration and runtime tuning method* is proposed. The proposed method exploits post-fabrication measurements data to predict the best operating point for each fabricated chip during runtime, hence, maximizing the energy efficiency with minimum overheads.

- Chapter 6 studies the reliability issues with wide-voltage operation. Some reliability challenges, such as transistor aging and supply voltage fluctuation, do not have any significant impact in the NTV region; however, they are important once the circuit is operating over a wide voltage range. This chapter proposes *selective flip-flop optimization method*, which improves the overall circuit reliability by optimizing the flip-flops severely affected by reliability issues.
- Finally, Chapter 7 concludes the thesis and provides an outlook for future research in this regard.

2 Preliminaries and State-of-the-Art

Numerous challenges have to be addressed to reach the desired resiliency and energy efficiency targets in modern Very-Large-Scale Integration (VLSI) technology. This chapter presents an overview of the fundamentals of VLSI technology and explains the challenges towards resilient energy-efficient computing. Additionally, NTC is studied as a promising design paradigm for energy efficiency in both low-power and high-performance domains and the most relevant approached towards NTC are reviewed.

2.1 VLSI technology: from transistors to circuits

Complementary Metal Oxide Semiconductor (CMOS) technology has been widely used for more than three decades as the primary technology for digital VLSI circuit design. The idea is to use complementary p-type and n-type Metal Oxide Semiconductor (MOS) transistors, namely PMOS and NMOS, to perform logic operations and store bit values. The main advantages of CMOS are low static power (leakage), low noise, and the possibility for massive integration [79].

MOSFET transistors

As the elementary blocks of CMOS technology, Metal-Oxide-Semiconductor Field Effect Transistor (MOSFET) has three terminals, namely source (S), Drain (D), and Gate (G) as shown in Figure 2.1. The channel conductivity between the source and drain terminals is controlled by the voltage applied to the gate terminal. Therefore, the behavior of a MOSFET can be explained based on the voltages applied to these terminals. Three different operating regions (or modes) are commonly considered for a MOSFET:

- *Sub-threshold region*, also known as cut-off region or weak inversion region, when the applied gate to source voltage (V_{gs}) is less than the threshold voltage (V_{th}) of the transistor.

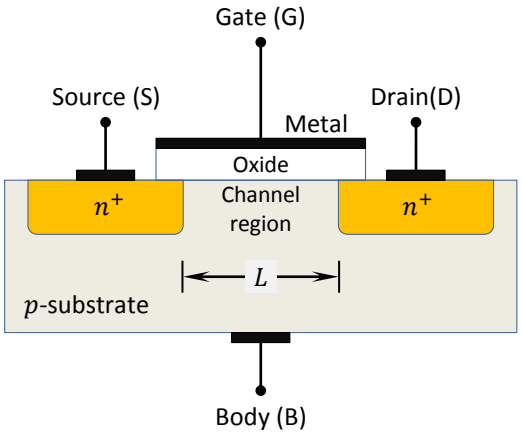


Figure 2.1: MOSFET transistor (NMOS)

- *Linear region*, also known as triode region or ohmic region, when the channel is formed due to $V_{gs} > V_{th}$ and the transistor behaves similar to a resistor. In this region, strong inversion happens near drain because $V_{ds} < V_{gs} - V_{th}$ (or simply $V_{gd} > V_{th}$).

- *Saturation region*, or active region, when the channel is formed due to $V_{gs} > V_{th}$. However, the formed channel comes to a sharp tip close to the drain because $V_{ds} > V_{gs} - V_{th}$ (or simply $V_{gd} < V_{th}$). In this region, the current passing from drain to source (I_{ds}) is not dependent on the drain voltage.

In the sub-threshold region, the transistor is considered to be turned off. However, drain-source leakage current (I_{ds}) is exponentially dependent on V_{gs} as:

$$I_{ds} = I_0 \frac{W}{L} e^{\frac{V_{gs}-V_{th}}{nV_T}} (1 - e^{-\frac{V_{ds}}{V_T}}), \qquad V_T = K_B T/q. \tag{2.1}$$

Here, n is a process-dependent parameter. V_T is the thermal voltage, with Boltzmann constant (K_B), temperature in Kelvin (T), and charge of an electron (q). W is the channel width and L is the channel length of the MOSFET transistor (i.e. transistor dimensions). Therefore, the leakage current is dependent on various process-related parameters as well as operating conditions. In the linear region, I_{ds} of a MOSFET follows:

$$I_{ds} = \mu C_{ox} \frac{W}{L} (V_{gs} - V_{th} - \frac{V_{ds}}{2}) V_{ds}, \tag{2.2}$$

which shows a strong dependency on V_{ds}. However, in the saturation region, the current is mostly independent of V_{ds}:

$$I_{ds} = \mu C_{ox} \frac{W}{L} \frac{(V_{gs} - V_{th})^2}{2}. \tag{2.3}$$

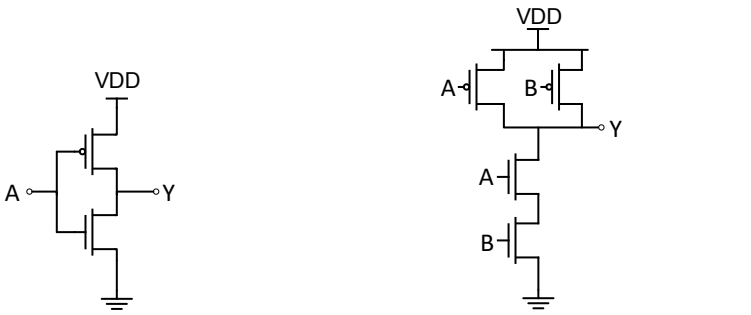
In the above equations, μ represent the carrier mobility in the channel and C_{ox} is the capacitance between gate and channel (per unit area). Please note that in Equations (2.1) to (2.3) the threshold voltage V_{th} also depends on drain voltage and body voltage as follows:

$$V_{th} = V_{th0} - \lambda_{ds} V_{ds} - \gamma_{bs} V_{bs}. \tag{2.4}$$

Various parameters and phenomena, such as Drain-induced Barrier Lowering (DIBL), Gate-Induced-Drain-Leakage (GIDL), Short-Channel Effect (SCE), Reverse Short Channel Effect (RSCE), and Well-Proximity Effect (WPE), affect the characteristics of a transistor, especially in nano-scale transistors. However, the overall characteristics can be explained with the above equations. The behavior of modern devices which have replaced the conventional bulk-MOSFET devices at sub 28nm nodes, such as FinFET and Fully-Depleted Silicon-On-Insulator (FDSOI), can also be explained by the same equation with minor modifications to reflect technology dependent parameters of these new technologies.

CMOS gates

MOSFET transistors are utilized as switches to perform logic operation in CMOS gates, which are the building blocks of modern digital circuits. Figure 2.2 shows the internal circuit of two basic gates: inverter and nand. Typical logic gates employ MOSFET transistors in pull-up and pull-down networks in order to transfer specific input signal combinations to logic level '0', which is the ground voltage level, and logic level '1', which is the supply voltage level.



(a) inverter gate ($Y = \overline{A}$) (b) nand gate ($Y = \overline{A.B}$)

Figure 2.2: Basic inverter and nand gates

Delay and power consumption

A logic gate needs time and energy to transfer input signal combinations to output logic levels. The *propagation delay* of a gate can be generally defined as the required time since an input signal switches until the logic result appears at the gate outputs. As an example, the propagation delay of an inverter is defined as the difference in time between when the input of inverter reaches $V_{dd}/2$ (falling or rising) until the moment that the output crosses $V_{dd}/2$ in the reverse direction (rising or falling). Propagation delay of simple gates can be approximated as follows, for rising $t_{p,LH}$ and falling $t_{p,HL}$ outputs:

$$t_{p,LH} \approx k_f C_L \frac{V_{dd}}{I_{ds}^{PMOS}}, \qquad t_{p,HL} \approx k_f C_L \frac{V_{dd}}{I_{ds}^{NMOS}}. \tag{2.5}$$

Here, C_L is the load capacitor on the output, k_f is a fitting coefficient, and I_{ds}^{PMOS} and I_{ds}^{NMOS} are the on-currents of gate PMOS and NMOS transistors which are pulling-up/-down the output (see Equation (2.3)). According to [80] the coefficient k_f may be around 0.5 depending on the effective drive current of transistors.

When a gate output toggles due to an input transition, it has dynamic and short-circuit power consumption. Dynamic power consumption, due to charging/discharging of the load capacitance, can be calculated as:

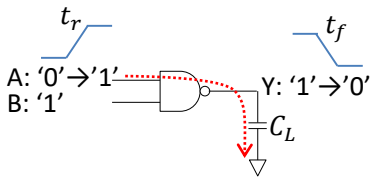
$$P_{dyn,sw} = \alpha f C_L V_{dd}^2, \tag{2.6}$$

where f is the clock frequency of the circuit (in which the inverter is used) and α is the average number of $0 \rightarrow 1$ transitions per clock cycle. During each switching, there could be a time in which both pull-up (consisting of PMOS transistors) and pull-down (consisting of NMOS transistors) networks are partially or fully turned on. This leads to a short-circuit current passing through pull-up and pull-down networks for a short period of time. As a result, there would be a short circuit power as follows [81]:

$$P_{dyn,sc} = \hat{I}_{sc} V_{dd} t_{sc} \alpha f, \tag{2.7}$$

where $\hat{I}_{sc}$ is the average short-circuit current, and t_{sc} is the time in which the short-circuit current passes through pull-up and pull-down networks. When the gate does not execute any transition a small current leaks through transistors leading to a leakage power, which can be calculated as:

$$P_{leak} = I_{leak} V_{dd}, \qquad I_{leak} = I_0 \frac{W}{L} e^{-\frac{V_{th}}{nV_T}} (1 - e^{-\frac{V_{dd}}{V_T}}). \tag{2.8}$$



(a) NAND gate $A \rightarrow Y$ arc at $'1' \rightarrow' 0'$ output transition

	C_{L_1}	C_{L_2}	C_{L_3}
t_{r_1}	$t_{f_{1,1}}$	$t_{f_{1,2}}$	$t_{f_{1,3}}$
t_{r_2}	$t_{f_{2,1}}$	$t_{f_{2,2}}$	$t_{f_{2,3}}$
t_{r_3}	$t_{f_{3,1}}$	$t_{f_{3,2}}$	$t_{f_{3,3}}$

(b) LUT stores output fall transition (t_f) based on input rise transition (t_r) and load capacitance (C_L)

Figure 2.3: nand gate characteristics stored as Look-Up-Tables

For an inverter, the leakage current I_{leak} can be calculated based on Equation (2.1), depending on which transistor is leaking. The above equations approximate the behavior of digital gates for long-channel transistors operating at deep saturation region. However, they are not accurate anymore at modern technology nodes or when V_{dd} is close to V_{th}, i.e. the near-threshold region. In such cases, we need to employ SPICE circuit simulation tools which incorporate accurate transistors models to calculate the circuit behavior.

Digital circuit design

Digital circuits, depending on their size and functionality, may contain up to billions of gates, in various types and sizing, to satisfy the power and timing constraints. In other words, the overall power consumption and delay of a digital circuit can be approximated based on the delay and power consumption of its gates. Therefore, EDA tools and design flows are necessary to facilitate the design and fabrication of such complex circuits. For that, EDA tools are used at different abstraction levels: Technology, Device, Circuit, Gate, RTL, Algorithm, and System. Circuit characteristics at each abstraction level are abstracted and stored in databases to be used by EDA tools. For examples, standard cell libraries contain delay, timing, and power characteristics of gates. Since these parameters depend on the structure and size of gates as well as input signal timing and load capacitance, the standard cell libraries are typically complicated and large. This information is stored in *Look-Up-Tables (LUT)* format, as shown in Figure 2.3. For each transition *arc*, which is a specific signal propagation path from a gate input to a gate output (e.g. $A \rightarrow Y$) and at a specific output transition direction (e.g. $'1' \rightarrow' 0'$), a LUT is created based on input signal transition time t_r and output capacitance (C_L) values. In addition to that, process corner, statistical characteristics such as mean and standard deviation of parameters' distributions, temperature, and supply voltage are also considered into standard cell libraries. These libraries are then used by various EDA tools such as synthesis and STA to design and optimized circuits.

2.2 Variability sources

Process and runtime variations strongly impact the functionality of circuits in the nanoscale technology nodes [82]. Figure 2.4 introduces the most important variation sources. The process of fabricating integrated circuits involves various parameters. Lithography, etching, ion implantation, growth, and almost all other processes bring variations to transistors and make them distinguishable from each other. This translates into variation in important transistor

parameters, such as V_{th}, W_{eff}, and L_{eff}, and eventually impact the circuits by changing the timing and power characteristics. On top of that, the behavior of the fabricated circuit can change due to fluctuation in parameters such as temperature and supply voltage. Additionally, electronic circuits age over time meaning that their behaviors slightly change over time depending on the operating condition.

The combined impact of these variation sources brings about many reliability challenges including yield problems after fabrication and functional failures in the field [82]. In this part of the thesis, the most important variability sources are introduced and their impacts on circuits are studied.

2.2.1 Process variation

As a result of manufacturing process, fabricated components of electronic circuits such as transistors, interconnects, and vias become slightly different from what they were designed. This category of variations is generally called process variation, but also named as intrinsic variation or time-zero variation. Process variation could be due to different sources and differentiates the fabricated devices at wafer, die, and transistor levels. The systematic process variation, which directly affects the yield, has been the subject of research in the past [83]. In modern technology nodes, the random variation is increasingly more important because of strengthened random variability [83].

Random variation could happen between every two fabricated transistors. This may be due to Random Dopant Fluctuation (RDF), Line-Edge Roughness (LER), Metal Grain Granularity (MGG), Oxide Thickness Fluctuation (OTF), and many other reasons. The most important variability source in the plain MOSFET structure is known to be RDF, which is the change in the number and placement of dopant atoms inside the transistor channel [84–86]. However, in modern technologies such as FD-SOI and FinFET, other sources such as MGG and LER have become the dominant source of variability [87–89].

The threshold voltage of a transistor can be explained as a statistical parameter with standard deviation σV_{th}. In case of RDF, σV_{th} can be described based on Pelgrom's model [90], that is:

$$\sigma V_{th} = \frac{A_{vt}}{\sqrt{W_{eff} L_{eff}}}, \tag{2.9}$$

where A_{vt} is a process-dependent coefficient, and W_{eff} and L_{eff} are effective width and length of the transistor [83]. This equation holds for different transistor designs but the parameters in Equation (2.9) need to be adjusted accordingly [91]. For MGG, a fix σV_{th} can be assumed depending on the technology [92]. Finally, the impacts of all random variations can be aggregated as independent random variables into:

$$\sigma V_{th,total}^2 = \sigma V_{th,RDF}^2 + \sigma V_{th,MGG}^2. \tag{2.10}$$

Additionally, LER and OTF affect other parameters of a transistor such as L_{eff} and t_{ox} and can be considered as σL_{eff} or σt_{ox}.

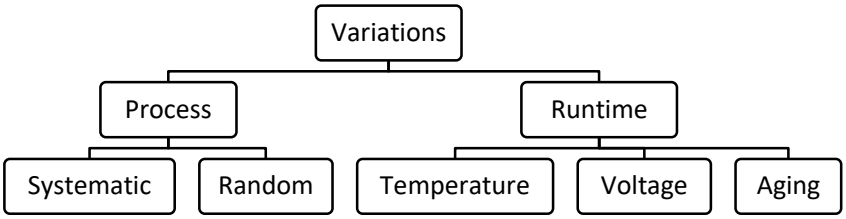


Figure 2.4: Variability sources impacting digital circuits

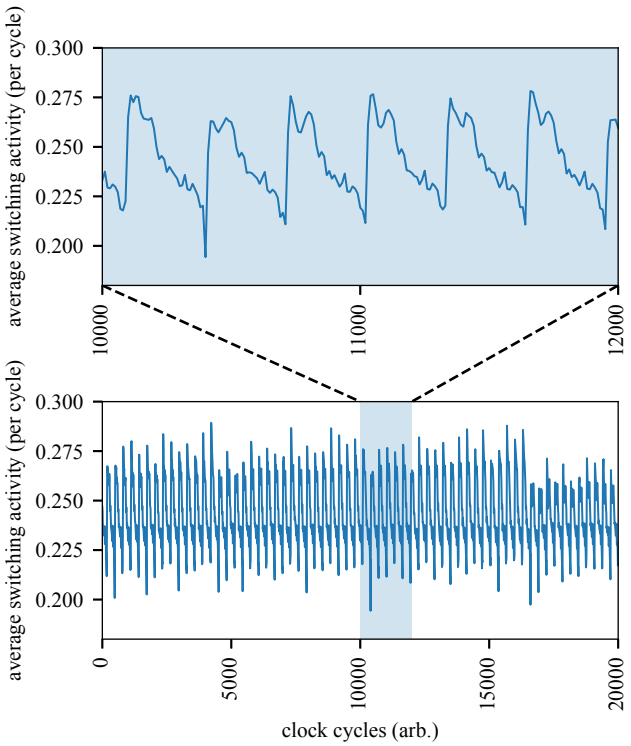


Figure 2.5: Average switching activity of Leon3 flip-flops in different clock cycles, running "bitcount" workload from MiBench benchmark.

2.2.2 Supply voltage fluctuation

Modern digital circuits may have a power consumption in the order of tens of watts, with high power density [93], which is supplied through Power Delivery Network (PDN). Supplying such amount of power under frequently changing workload is associated with various thermal and power supply fluctuation challenges and is considered as a source of variation [94].

Supply voltage fluctuation is caused by power consumption fluctuation of VLSI components at runtime, and is composed of two major factors, *IR drop* and *di/dt noise*. IR drop is the voltage-drop due to the current flow over parasitic resistances of the PDN, whereas di/dt noise is due to the PDN inductive and capacitive properties. Therefore, spatial and temporal changes in current drawn from various PDN nodes result in a supply voltage fluctuation at the sink nodes. In other words, the supply voltage at the circuit's gates and flip-flops fluctuates due to the component activities.

Figure 2.5 shows the average switching activity of the flip-flops of a synthesized Leon3 processor running "bitcount" workload (from MiBench benchmark [95]). As presented, the average switching activity of the flip-flops is changing by about ±20% during the illustrated 20k cycles. Please note that the change in the activity could be much larger at specific locations in the circuit layout. Such runtime variations, shown in the figure, cause fluctuations in the power consumption and the supply voltage at PDN nodes.

As the delay of digital gates is inversely proportional to the supply voltage, transient voltage-drops may cause *supply voltage emergency* conditions [96, 97], when the voltage-drop exceeds a certain threshold. In this situation, the circuit may not operate in the safe condition which results in timing errors. Such errors are typically fixed by optimizing the PDN, e.g. by including decap capacitors in the layout, and by considering *voltage-drop timing margins* in the overall timing margin of the circuit. In fact, modern digital designs require the voltage-drop to stay below 10% of the supply voltage [94]. However, in modern technology nodes,

supply voltage fluctuation aggravates due to the abrupt changes in the power consumption of the components (and as a result of higher speed). Furthermore, fewer resources are available for optimizing the PDN at modern technology nodes [98]. Therefore, it is becoming more important to additionally optimize other parts of the circuit to mitigate the voltage fluctuation effect. For example, various techniques at different abstraction levels have been proposed to prevent such voltage-drops by limiting the change in the power consumption of components [96, 99] or by redistributing the peak current on a longer period [100, 101].

Reducing the supply voltage significantly decreases the power consumption and speed (orders of magnitude lower than above threshold), and diminishes impact of workload on supply voltage fluctuation [76]. Therefore, low-power circuits operating at low supply voltages do not need very sophisticated PDN design, as required for high-performance digital circuits. However, it is still needed to scrutinize the impact of external supply voltage fluctuation in low-power circuits, especially if the circuit is expected to operate over a wide-voltage range.

2.2.3 Temperature variation

The temperature of digital circuits is influenced by both environmental temperature and self-heating due to the dissipation of the consumed electric power. The operating temperature range for digital circuits is quite broad spanning over more than $100°C$. For example, in the automotive, circuits are expected to operate from $-40°C$ to $125°C$. Additionally, on-chip spatial temperature gradient puts different stress on the circuit components. The amount of on-chip spatial temperature difference based on simulation [102, 103], sensor measurements [104], and thermal camera [103] is reported to be up to $\sim 20°C$.

Many physical parameters in digital circuits are fundamentally dependent on temperature. As a general rule, the resistance of materials increases by increasing temperature, affecting interconnects and vias resistance. Unintentional mechanical stress happens due to different thermal expansion coefficient of the materials used in a chip, which may affect the electrical properties of semiconductors as well [105, 106]. Energy band-gap is an important parameter in semiconductors which decreases by increasing temperature. Intrinsic carrier concentration in semiconductor n_i has a very strong dependency on temperature. Carrier mobility μ directly affects transistor current (see Equations (2.1) to (2.3)) and has a complicated dependency on temperature, which is based on various scattering phenomena such as phonon scattering, surface roughness scattering, and Coulombic scattering due to interface charges or ionized impurities [107]. However, due to the opposing effects of scattering phenomena, mobility is reduced at low and high temperatures with a peak at intermediate temperature.

The threshold voltage of MOSFET transistors V_{th} is defined as [108]:

$$V_{th0} = V_{FB} + 2\phi_F + \gamma\sqrt{2\phi_F}, \tag{2.11}$$

where V_{FB} is the flat band voltage, γ is the body effect parameter and ϕ_F is the Fermi energy. In the above equation, V_{th0} is dramatically impacted by temperature. From the above parameters, V_{FB} and ϕ_F vary with temperature depending on thermal voltage V_T and intrinsic carrier concentration n_i. Temperature dependence of threshold voltage is commonly explained as a linear model at common temperature values [108], with a slope as big as 4% per every $10°C$ [109].

According to Equation (2.1), the sub-threshold current of transistors is exponentially dependent on the threshold voltage and temperature, doubling by every $10°C$ increase. Drain-source current in the saturation and linear regime is also dependent on the temperature through mobility and threshold voltage, as presented in Equations (2.2) and (2.3). However, the impact of temperature in the sub-threshold region is much more pronounced compared to the super-threshold region, which is also supported by measurement results [110]. The change in the

current of the transistors is mapped to circuit delay through Equations 2.5. Therefore, temperature has a significant impact on circuit performance, leakage power, and dynamic power. This is confirmed by measurement results, as Intel [110] demonstrated that the variation in maximum frequency of a processor when the supply voltage is below V_{th} is much larger than when the supply voltage is above V_{th}.

In addition, high temperature is responsible for accelerating various reliability issues, such as transistor aging. Due to the issues caused by temperature hot-spots in digital circuits, many methods have been proposed to address the temperature variation issues [111, 112].

2.2.4 Aging

The properties of integrated circuits change over time due to aging. Aging phenomena such as Bias Temperature Instability (BTI), Hot Carrier Injection (HCI), Electromigration (EM), and Time Dependent Dielectric Breakdown (TDDB) have been studied since their discovery as early as the 60s[113–117]. Due to aggressive scaling of device geometries and using new materials and processes such as High-K and metal-gate [117, 118], aging effects pose serious reliability threats in modern integrated circuits. Recent studies have shown that failures can be observed in SRAM memories used in automotive industry as early as one year[1] [24]. Therefore, researchers are actively looking for design methods to make digital circuits more resilient against aging threats.

BTI, HCI, and TDDB impact the transistors in a chip whereas EM affects wires and vias. BTI and HCI alter the threshold voltage of transistors over time leading to degraded performance and noise margin. This eventually causes timing failure in logic circuits or noise margin issues in the SRAM memories. TDDB modifies the gate oxide characteristics and creates a leakage path in the gate-oxide which grows over time, leading to a hard breakdown of MOSFET transistor. EM is caused by continuous contact of high-energy electrons with material atoms due to high current density, which displaces the atoms. As a result of EM, the resistance of a wire increases to the point that it breaks and causes a permanent failure [112].

In the following, these important aging effects are discussed and shown to be highly dependent on the applied electric field and current density, which in fact depend on the supply voltage in digital circuits. In addition, temperature is a very important factor in the aging rate, which is also correlated with the supply voltage and workload due to self-heating and high power dissipation in digital circuits.

Bias Temperature Instability (BTI)

BTI happens in both PMOS and NMOS transistors, known as Negative Bias Temperature Instability (NBTI) and Positive Bias Temperature Instability (PBTI), respectively. Both NBTI and PBTI are important in modern transistors [117, 119]. BTI consist of two phases: I) *stress* and II) *recovery*. During the stress phase, the transistor is turned on and a high electric field is applied to its oxide ($V_{gs} > V_{th}$), which eventually increases the threshold voltage of the transistor. When the transistor is turned off and the electric field is removed, the recovery phase starts in which the threshold voltage is partially recovered towards the original value.

Two prevailing models exist which explain the physical reason behind the threshold voltage change: *Reaction/Diffusion (RD)* and *Trapping/Detrapping (TD)* [120, 121]. The RD model explains the increase and decrease in the threshold voltage based on the generation and diffusion of interface traps due to the perpendicular electric field in the gate oxide. According to this theorem, a strong electric field can break some of the passivated Si-H bonds at the Si-SiO_2 interface and form some Hydrogen related species (atoms or molecules). Then, these

[1] Based on the assumptions in the cited paper

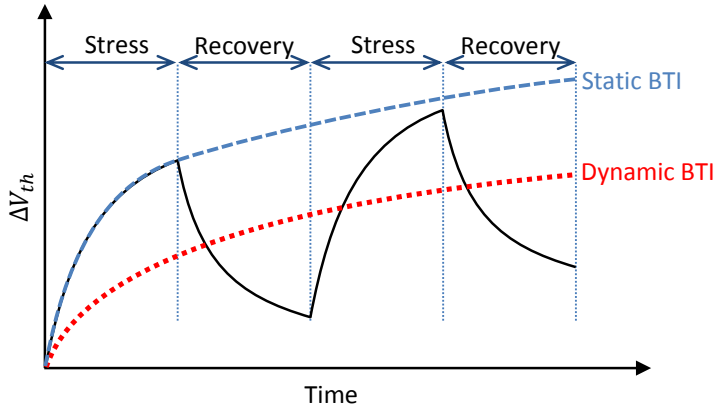


Figure 2.6: Threshold voltage shift (ΔV_{th}) due to Bias Temperature Instability (BTI) under Dynamic (alternating) and Static stress.

species can move into the dielectric and increase the threshold voltage. When the electric field is removed, the Hydrogen related species diffuse back outside the oxide and the threshold voltage is recovered to its original value. However, according to the TD model [121–124], during the device fabrication, some traps are formed inside the oxide. These traps can capture a charge during the stress phase, which increases the threshold voltage. According to this model, the captured charges are released during the recovery phase resulting in the recovery of the threshold voltage. In general, the RD model can correctly predict the long-term behavior [125] while the TD model is suitable to explain the short-term behaviors [123].

BTI impact on the threshold voltage highly depends on temperature, supply voltage, and input stress condition [31]. Figure 2.6 demonstrates the impact of input stress condition on BTI. When the BTI stress is continuously applied to the transistor, its threshold voltage undergoes a significant threshold voltage shift due to the *Static BTI (S-BTI)*. However, under a *Dynamic BTI* condition, i.e. alternating stress and recovery phases, due to a partial recovery of threshold voltage during the recovery phase, the overall impact of BTI is much smaller [126, 127]. The threshold voltage shift caused by S-BTI can be modeled as:

$$\begin{aligned} \Delta V_{th} &= A\Big((1+\delta)t_{ox} + \sqrt{Ct}\Big)^{2n}, \\ &\propto t^n. \end{aligned} \tag{2.12}$$

In the above equation, n is the time exponent determined based on the technology. The vertical electric field E_{ox} in the gate oxide with thickness t_{ox} is also caused by the overdrive voltage ($V_{gs} - V_{th}$):

$$E_{ox} = \frac{V_{gs} - V_{th}}{t_{ox}}. \tag{2.13}$$

Other parameters in Equation (2.12) are explained in Table 2.1. In a long-term Dynamic BTI scenario, alternating stress and recovery phases occur with a BTI stress duty cycle η. This means that in a system with clock period T_{clk}, a stress time equal to ηT_{clk} exists in each clock period. In this situation, the shift in the threshold voltage can be modeled as [126]:

$$\Delta V_{th} = A\Big(\frac{\sqrt{C\eta T_{clk}}}{1-\beta(t)^{1/2n}}\Big)^{2n}, \tag{2.14}$$

where $\beta(t)$ is dependent on time, as shown in Table 2.1. In the long-term, where $t > 1000s$,

Table 2.1: Summary of BTI predictive model parameters [31]

K	$8 \times 10^4 \quad (s^{-0.25}.C^{-0.5}.nm^{-2})$
E_0	$0.335 \quad (\frac{V}{nm})$
E_a	$0.49 \quad (eV)$
δ	0.5
T_o	$10^{-8}(\frac{s}{nm^2})$
n	1/6 for H_2 and 1/4 for H
ξ_1	0.9
ξ_2	0.5
η	Stress duty cycle (Stress duration in each clock period divided by T_{clk})
E_{ox}	$\frac{V_{gs}-V_{th}}{t_{ox}}$
A	$\frac{qt_{ox}}{\epsilon_{ox}}\Big(K^2 C_{ox}(V_{gs}-V_{th})e^{\frac{2E_{ox}}{E_0}}\Big)^{2n}$
C	$\frac{1}{T_o}e^{-\frac{E_a}{K_B T}}$
$\beta(t)$	$1-\frac{2\xi_1 t_e+\sqrt{\xi_2 C(1-\eta)T_{clk}}}{2t_{ox}+\sqrt{Ct}}$
Static BTI	$\Delta V_{th} = A\Big((1+\delta)t_{ox}+\sqrt{Ct}\Big)^{2n}$
Dynamic BTI	$\Delta V_{th} = A\Big(\frac{\sqrt{C\eta T_{clk}}}{1-\beta(t)^{1/2n}}\Big)^{2n} \overset{t>1000s}{\approx} A\Big(\frac{n^2\eta C t_1 t}{\xi_1^2 t_{ox}^2(1-\eta)}\Big)^n, \quad 0<\eta<1$

Equation (2.14) is independent of clock frequency and can be further simplified into [126]:

$$\begin{aligned}\Delta V_{th} &\approx A\Big(\frac{n^2\eta C t}{\xi_1^2 t_{ox}^2(1-\eta)}\Big)^n, \qquad \eta<1 \\ &\propto \Big(\frac{t\eta}{1-\eta}\Big)^n.\end{aligned} \tag{2.15}$$

Equation (2.15) has a strong dependency on the duty cycle η. Therefore, the impact of BTI is much more pronounced at large stress duty cycles, as shown in Figure 2.7. Please note that this equation is only valid when $\eta < 1$, hence, Equation (2.12) should be used for calculating the S-BTI impact. When stress duty factor is less than 1.0, the transistor frequently goes into stress and recovery phase (Dynamic BTI), but when it is 1.0, the transistor is always under BTI stress (i.e. S-BTI). By increasing the stress duty cycle from 0.9 towards 1.0, ΔV_{th} rises rapidly, because the recovery phase becomes very short. As a consequence of such BTI stress, the delay degradation can be much higher.

Hot Carrier Injection (HCI)

When a transistor is turned on, i.e. $V_{gs} > V_{th}$, the charge carriers in the channel move from source to drain (electrons) or vice versa (holes). The lateral electric field in the MOSFET channel due to V_{ds} may give some carriers enough energy such that they can break the interface potential barrier between Si and gate oxide. Such *hot carriers*, may penetrate into the oxide and change the threshold voltage of the transistor [128]. This effect is called HCI and is considered an important reliability issue in modern technology nodes [129]. NMOS transistors are more susceptible to HCI compare to PMOS transistors because of the difference between the mean free paths of carriers and the potential barrier [130].

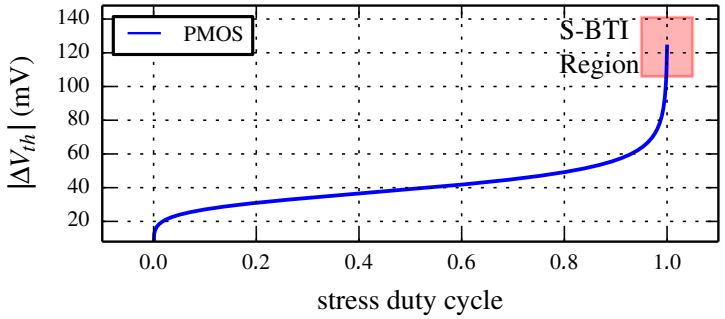


Figure 2.7: The impact of Negative BTI (NBTI) on the threshold voltage of a single PMOS for different stress duty cycles in 5 years.

Table 2.2: Summary of HCI predictive model parameters [31]

K_2	$1.7 \times 10^8 \quad (\frac{nm}{C^{0.5}})$
E_{02}	$0.8 \quad (\frac{V}{nm})$
ϕ_{it}	$3.7 \quad (eV)$
λ	$7.8(nm)$
l	$17(nm)$
n'	0.45
E_m	$\frac{V_{ds}-V_{dsat}}{l}$
HCI	$V_{th,HCI} = \frac{q}{C_{ox}} K_2 \sqrt{C_{ox}(V_{gs}-V_{th})} e^{\frac{E_{ox}}{E_{02}}} \; e^{-\frac{\phi_{it}}{q\lambda E_m}} \; (t.\alpha.f)^{n'}$

In digital circuits, the HCI impact on threshold voltage is dependent on a number of parameters, such as the switching activity at the gate α and the clock frequency f. In addition, the impact of HCI is more significant at higher temperature [129, 131]. The threshold voltage shift due to HCI can be modeled as [31, 132]:

$$\begin{aligned} \Delta V_{th,HCI} &= \frac{q}{C_{ox}} K_2 \sqrt{C_{ox}(V_{gs}-V_{th})} e^{\frac{E_{ox}}{E_{02}}} \; e^{-\frac{\phi_{it}}{q\lambda E_m}} \; (t.\alpha.f)^{n'}, \\ &\propto (t.\alpha.f)^{n'}. \end{aligned} \tag{2.16}$$

Table 2.2 presents the parameters in Equation (2.16) for 65nm technology [31].

It is worth mentioning that the HCI happens only when a current passes through the transistor, however, BTI impact exists whenever the transistor is turned on. Therefore, BTI is a significant effect on both logic and memory, however, the impact of HCI in memory is less significant. Especially, there could be memory cells or logic gates storing a specific value for a long time. In this case, a strong S-BTI is observed, however, HCI has no impact when the stored value is not toggled.

Time-Dependent Dielectric Breakdown (TDDB)

A soft TDDB causes a slight leakage through the gate oxide. However, the leakage increase over time and leads to a hard TDDB, in which the gate oxide breaks and allows a strong current to pass. In this case, the transistor is destroyed and cannot function anymore, which causes a permanent failure [113, 133]. A resistor between gate and drain R_{GS} can emulate the

impact of TDDB as follows [134, 135]:

$$R_{GD} \propto \frac{1}{(\eta.t)^P}, \qquad P \approx 5. \tag{2.17}$$

The resistance decreases over time. Similar to BTI, there is a dependency on η in TDDB as well, which means R_{GD} decreases faster at higher stress duty cycle values. Several models have been proposed to evaluate the Mean Time To Failure (MTTF) due to TDDB [133]. In all the models, there is a strong dependency on oxide field E_{ox} and temperature [133]. Also, Weibull distribution can explain the statistical behavior of TDDB [133].

Electromigration (EM)

Electromigration is the forced movement of the atoms in a conductor material due to a current passing through the material [136]. It is especially important when current density is high and there is a sensitivity to the change in the resistance of the wire. Known for more than 100 years [137], EM impact has extensively increased in the advanced technology nodes, where the interconnects are very small and current density is high. As a result of EM, atoms are depleted from one side of wire creating a void, which causes timing errors and eventually makes an open circuit, and deposited at another side creating a hillock or whisker, which may cause a short circuit. An empirical equation, known as Black's equation, was proposed to model wire MTTF [138] as follows:

$$MTTF = AJ^{-n}e^{\frac{E_{a,em}}{K_B T}}, \qquad J = \frac{I}{t_{wire}W_{wire}}, \tag{2.18}$$

where A is an empirical constant, $E_{a,em}$ is the activation energy, and $n = 2$ is according to [138]. J is the current density which is calculated based on wire thickness t_{wire} and wire width W_{wire}. The values for A and $E_{a,em}$ are presented in [139] for Cu interconnect at 90nm. In addition, EM is highly dependent on the wire shape and temperature, which brings additional complexity to MTTF calculation [113].

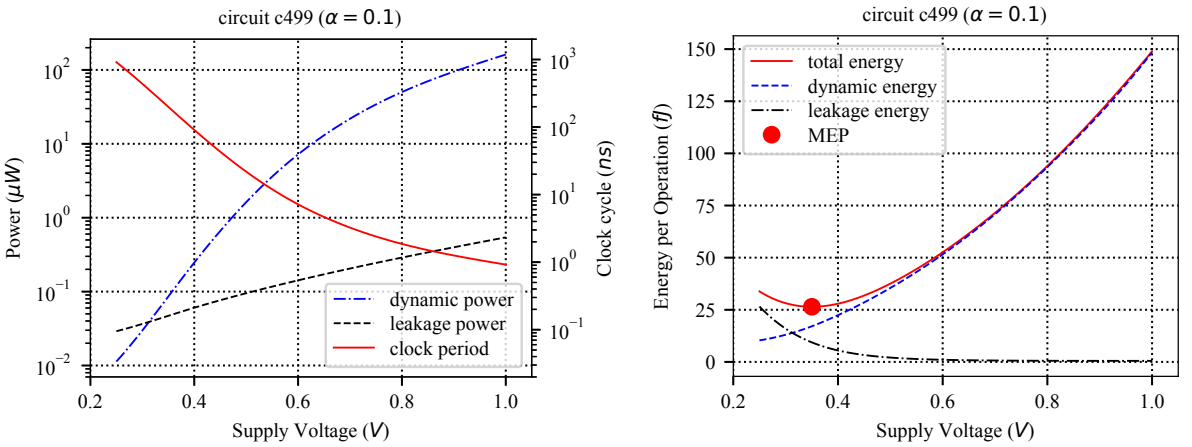
2.2.5 Soft-Errors

Soft-errors are transient errors in memory or logic caused by particle strikes. When an ionizing particle, such as an electron, alpha, or cosmic rays (neutron), hits a sensitive node, it generates some amount of free charge, which may temporarily flip the state of that node. Such transient errors at nodes may propagate to the output of circuit, causing faults in the normal system operation [140]. Soft-errors have been known for a long time as a source of unreliability in circuits, especially in aeronautics applications [141]. The soft-error induced failure rate has been increasing since then due to scaling [142–145], and has become a major reliability challenge in various application domains [16, 146]. Furthermore, soft-error rate increases by supply voltage scaling [144, 145, 147, 148]. The reason is that the critical charge required for a bit-flip reduces by voltage scaling [149].

Soft-errors in memory can be addressed by error detection and correction codes, such as parity bit. However, it is much more difficult to address the errors in logic [71]. Furthermore, more errors appear as Multiple Bit Upsets (MBU) in advanced technology nodes [150–152]. Therefore, methods with the capability to detect and correct MBUs should be employed to prevent catastrophic in field failures [71].

2.3 Near-Threshold Computing

As explained in Chapter 1, various methodologies at different abstraction-levels have been proposed and employed [52–56, 153–155] to improve the energy efficiency and overcome the



(a) Circuit power and clock period versus supply voltage (V_{dd})

(b) Minimum Energy Point (MEP)

Figure 2.8: MEP exploration for circuit c499 from ISCAS'85 benchmark [26, 27]. Minimum energy per operation is achieved where V_{dd} is close to V_{th}.

power wall challenge caused by the end of Dennard's scaling [42–45]. Aggressive supply voltage scaling down to sub-threshold region [45, 52, 57–59] is also presented as an effective way to reduce the power consumption by several orders of magnitude, however, the speed of digital circuits at that supply voltage range is poor. In the following, we present a promising approach towards supply voltage scaling, called NTC, which also retains enough performance for many applications.

Near-Threshold Computing is a paradigm in which the supply voltage is reduced close to the threshold voltage of transistors to gain large energy efficiency while retaining enough performance for many applications. However, there are various challenges towards NTC mainly in the area of reliability and energy efficiency, which can nullify the benefits of NTC if not addressed correctly. *This thesis focuses on the NTC and its challenges.*

From a circuit-level point of view, the total power consumption of a circuit can be calculated as:

$$P_{total} = P_{dyn,sw} + P_{dyn,sc} + P_{leak}, \tag{2.19}$$

where $P_{dyn,sw}$, $P_{dyn,sc}$, and P_{leak} are dynamic switching power, dynamic short-circuit power, and leakage power, respectively, which can be calculated based on Equations (2.6) to (2.8). In the nominal supply voltage, the $P_{dyn,sw}$ is the dominant power consumption, which is quadratically dependent on the supply voltage, as shown in Figure 2.8a. Therefore, reducing the supply voltage can quadratically reduce the overall power consumption. In the same supply voltage regime, the speed of a circuit is linearly dependent on the supply voltage, based on Equations (2.5), (2.3), and (2.2). The energy consumption is calculated based on:

$$E_{total} = P_{total} \times T_{clk} = E_{dyn,sw} + E_{dyn,sc} + E_{leak} \tag{2.20}$$

Therefore, by slightly reducing the supply voltage from the nominal voltage, it is possible to reduce the energy consumption, linearly. As shown in Figure 2.8b, this energy improvement holds as long as the rate of total power consumption reduction is higher than the rate of speed degradation. When V_{dd} goes below V_{th}, the speed degradation becomes exponential because the current has an exponential relation with supply voltage according to Equation (2.1). Therefore, energy consumption starts to increase due to higher rate of speed degradation. As a result, the supply voltage leading to minimum energy consumption is somewhere close the threshold

voltage of transistors. Such operating condition leading to the best energy efficiency, i.e. the lowest energy consumption, is commonly known as the MEP and is shown in Figure 2.8b for c499 circuit from ISCAS'85 benchmark circuits [26, 27].

MOSFET model in the Near-Threshold Voltage region

The conventional three-region long-channel MOSFET model, presented in Equations (2.1), (2.2), and (2.3), as well as the alpha-power model [156] are piece-wise models with a discontinuity at the threshold voltage of transistors, which makes them inappropriate for NTC circuit analysis. However, it is possible to explain the characteristics of MOSFET based on continuous models such as EKV [157–159]. Accordingly, a simplified trans-regional model for digital NTC CMOS circuits is proposed in [160] which facilitates analytical analysis. Based on this model, the MOSFET on-current can be obtained based on the overdrive voltage V_{ov} as follows:

$$I_{ds,NTC} = I_x k_0 e^{k_1 \frac{V_{ov}}{nV_T} + k_2 \left(\frac{V_{ov}}{nV_T}\right)^2}, \qquad V_{ov} = V_{gs} - V_{th}, \tag{2.21}$$

where I_x depends on process parameters and transistor dimensions (W, L), whereas k_0, k_1, and k_2 are process independent fitting parameters [160]. In the above equation, it is assumed that $V_{ds} \gg V_T$, therefore, the term depending on V_{ds} is eliminated (see Equation (2.1)). However, it is possible to consider the threshold voltage dependency on V_{ds} and body-biasing according to Equation (2.4). Based on this model, the propagation delay of a gate in the NTV is obtained as:

$$t_{p,NTC} = \frac{k_f C_L V_{dd}}{I_x k_0} e^{-k_1 \frac{V_{dt}}{nV_T} - k_2 \left(\frac{V_{dt}}{nV_T}\right)^2}, \qquad V_{dt} = V_{dd} - V_{th}. \tag{2.22}$$

Similarly, energy and power consumption of digital circuits can be calculated in the NTV region [160].

Process, Voltage, and Temperature Variation in NTC

The impacts of process, supply voltage, and temperature variations (PVT) are more pronounced in the NTV region [60, 74, 161]. Authors of [162] showed that the sensitivity of the drain-source current of a transistor (I_{ds}) to changes in V_{th} and V_{dd} increases by about 10× when the supply voltage is reduced from the super-threshold region to the sub-threshold region. Equation (2.22) also demonstrates that propagation delay in the NTV region is exponentially dependent on supply and threshold voltages. Intel [74] reported that while the process and temperature variations cause 18% and 5% performance variation in the super-threshold region, their impacts aggravate to 2× performance variation in the NTV region.

The power consumption of NTC circuits is orders of magnitude smaller than the super-threshold region. As a result, runtime supply voltage fluctuation caused by power consumption also decreases by the same scale, which makes it insignificant in NTC circuits, even considering the large sensitivity to fluctuations. Moreover, the temperature of NTC circuits is solely determined by the ambient temperature since the power dissipation is very small and has a negligible impact on circuit temperature fluctuation [14].

2.3.1 NTC challenges

Reliability

Aggressive supply voltage scaling to the NTV region has benefits and drawbacks in terms of reliability. Reducing the supply voltage to the NTV region greatly reduces internal electric

fields and current density compared to the super-threshold region, which helps to protect the circuits from some aging phenomena and their associated reliability problems.

Nevertheless, the exponential sensitivity to PVT in the NTV region severely affects the circuit behavior and may lead to large performance variation or reliability issues, in terms of functional failure or timing violations. As presented in [74], $\pm 2\times$ performance variation is observed between different fabricated NTC cores only due to process variation, whereas this variation was limited to $\pm 18\%$ at the nominal voltage. Techniques such as adaptive body biasing [60] and supply voltage scaling [110] are presented to address such performance fluctuation. Similarly, the maximum clock frequency of an NTC circuit may change by $\pm 2\times$ due to $110°C$ temperature fluctuation, while the impact of such temperature fluctuation on performance is only 5%[2] at the nominal voltage [74]. An unwanted performance fluctuation at gate-level may lead to timing violations. Setup-time violations can be addressed by increasing the clock period T_{clk} (i.e. reducing clock speed f), which inflicts performance and energy efficiency loss. Various timing error detection and correction methods have also been proposed to address setup-time issues [73, 163–166], which may be costly as the number of timing errors increases rapidly in the NTV region. However, a hold-time violation cannot be fixed by changing the clock speed and leads to a functional failure. As an example, the number of hold-time violations increases by up to $16\times$ when operating in the NTV region compared to the nominal voltage [2]. In order to fix these timing violations, many buffers have to be inserted into the violating paths to delay the arrival times of such short paths. Therefore, the overhead of buffer insertion in the NTV region is significantly larger than that of the super-threshold region.

Reduced noise margin due to voltage scaling is a major challenge in storage component design for NTC. Conventional 6T SRAM memory cells cannot operate correctly in the NTV region without redesigning [167], and 8T or 10T SRAMs are preferred due to better noise margin and resiliency to variations [168–171]. In addition, due to low activity of memory cells, the optimum supply voltage may be considerably higher than core logic. Therefore, voltage level converters may be needed to interface memory and core cells which brings additional complexity to routing and PDN design. Similarly, flip-flops have issues due to reduced noise margin and high sensitivity to variations [60, 74]. Therefore, various flip-flop designs have been studied for low-voltage operation and are optimized for NTC [60, 61, 74, 172–174]. The soft-error rate of storage components is dependent on the critical charge, which decreases by $5\times$ in the NTV region [147]. Therefore, more soft-errors are observed in the NTV region in both logic and memory components. Various methods have been proposed at different abstraction levels from device to architecture-level to address the faults caused by soft-errors for NTC [175–182].

Energy efficiency

The energy efficiency of NTC circuits is highly dependent on design, process variation, and runtime parameters. Scaling down the supply voltage changes the ratio of dynamic and leakage power consumption significantly, causing a paradigm shift in design and optimization. Figure 2.9 presents the power consumption of an IA32 processor. The contribution of logic dynamic power decreases from 81% in the super-threshold region to 4% in the sub-threshold region, whereas the contribution of logic leakage power increases from 11% to 33% [28]. Therefore, leakage power reduction techniques at architecture-level (e.g. by power-gating schemes) down to device-level (e.g. by incorporating advanced technologies such as FinFET) are more rewarding in the NTV region compared to the super-threshold region, in which leakage power has a smaller contribution.

[2]The measurement is done for a 65nm typical die.

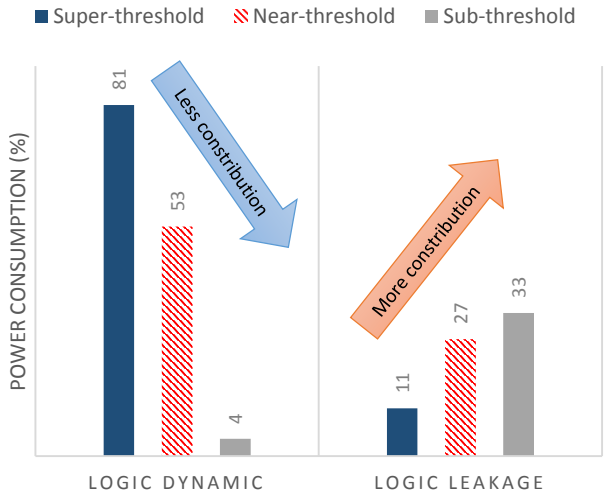


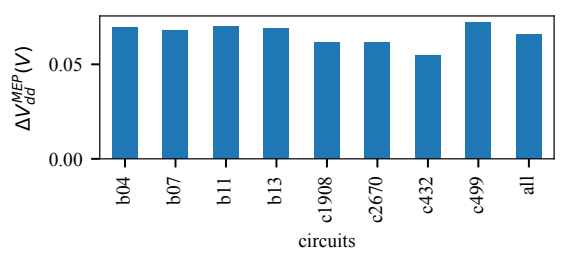
Figure 2.9: Power breakdown of the IA32 processor presented in [28] highlights a significant increase in the relative contribution of the leakage power of logic components to the total power consumption, when supply voltage is reduced towards near-threshold and sub-threshold regimes. The rest of the power consumption is due to the memory (8%, 20%, and 63% in the super-threshold, near-threshold, and sub-threshold region, respectively.)

The methods used for addressing the variation-induced reliability challenges affect the energy efficiency as well. Some of the methods commonly used in the super-threshold region for controlling the impact of variations are too expensive in the NTV region. As an example, adding timing margin to compensate for variation-induced timing fluctuation is energy inefficient in the NTV region due to the extent of variations. Therefore, it is necessary to design the circuits to be more resilient against timing variation, e.g., by variation-aware circuit synthesis.

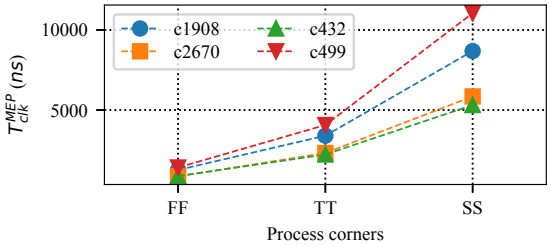
The MEP of a circuit points to a specific supply voltage V_{dd}^{MEP} and the maximum speed f^{MEP} (inversely proportional to clock period $T_{clk}^{MEP} = 1/f^{MEP}$) at that supply voltage. A number of factors impact the MEP of a circuit including technology, internal activity, workload, and process and runtime variations. FinFET technology offers close to 60mV/dec sub-threshold slope leading to better Ion/Ioff ratio, which is very useful in reducing the MEP and improving the energy efficiency. It has been shown that the MEP may move from sub-threshold voltage region to super-threshold voltage region depending on the circuit structure and circuit internal switching activity [183]. For example, the contribution of leakage power to the total power, i.e. P_{leak}/P_{total}, is typically larger in SRAM arrays compared to the core logic. Therefore, the MEP of SRAM arrays is higher than core logic [183]. This also means that workloads which cause higher internal switching activity (high dynamic power), will reduce the MEP of a circuit. Process and runtime variations significantly contribute to MEP fluctuation. Therefore, design optimization and runtime tuning are also necessary to achieve high energy efficiency in the NTV region.

The average impact of process variation on V_{dd}^{MEP} of the benchmark circuits is displayed in Figure 2.10a. The shift in V_{dd}^{MEP} due to process variation is on average 66mV for the benchmark circuits. This shift in V_{dd}^{MEP} may lead to significant performance variation and energy overheads. Moreover, the speed (T_{clk}^{MEP}) of some circuits is also highly affected by process variation. Figure 2.10b demonstrates that T_{clk}^{MEP} of a circuit may change by about one order of magnitude when it is under different process variation impacts.

The impact of temperature variation is shown in Figure 2.11, where the MEP is plotted for different temperatures ranging from $-25°C$ to $100°C$. The clock period corresponding to the MEP (T_{clk}^{MEP}) is also greatly affected by the change in the circuit temperature, with an exponential dependency. Comparing Figure 2.10 and Figure 2.11 reveals that the impact of

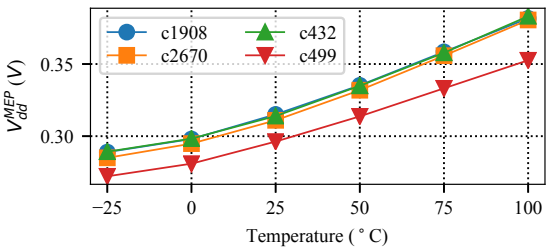


(a) Process variation can change V_{dd}^{MEP} on average by 66mV.

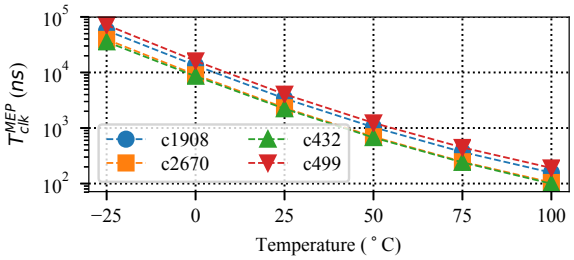


(b) Process variation can change T_{Clk}^{MEP} by up to an order of magnitude over different process corners.

Figure 2.10: Process variation impact on the MEP of some ISCAS'85 benchmark circuits.



(a) Up to 80mV variation in V_{dd}^{MEP} is observed over 125°C temperature change.



(b) More than 2 orders or magnitude variation in T_{clk}^{MEP} is observed over 125°C temperature change.

Figure 2.11: Temperature variation impact on the MEP of some ISCAS'85 benchmark circuits.

temperature could be much stronger than the impact of the process variation. This is also in line with the reported process and temperature variation sensitivities as in [60, 61, 74, 110].

Figure 2.12 shows the change in V_{dd}^{MEP} for circuit c499 of ISCAS'85 benchmark due to change in the internal switching activity. The x-axis of the figure represents the ratio of the dynamic energy to the leakage energy of the circuit (Dynamic/Leakage), when the circuit is operating at the nominal supply voltage (V_{dd}^{NOM}). The Dynamic/Leakage value has an inverse relation with V_{dd}^{MEP} as demonstrated in Figure 2.12. According to this figure, if the input switching activity α changes from 0.01 to 0.1, the V_{dd}^{MEP} can vary from 0.49V down to 0.35V as Dynamic/Leakage increases. In a pure combinational circuit such as c499, the impact of workload variation could be very high as the dynamic power consumption P_{dyn} is dependent on the input switching activity α. However, in sequential circuits, a large portion of dynamic power consumption is related to the clock network. In such cases, P_{dyn} variation due to α fluctuation is typically less than in combinational circuits. Therefore, workload variation can also have a significant impact on energy efficiency, depending on the circuit architecture.

In summary, the energy efficiency of NTC circuits is highly affected by PVT. Design optimization methods can effectively improve energy efficiency by reducing the leakage power. Runtime tuning methods can adapt the circuit to operate at the MEP which fluctuates at runtime due to temperature and workload fluctuation.

Wide-voltage resiliency

Some NTC circuits are required to operate at different supply voltage modes, due to runtime MEP tuning, ultra-low power requirements, or speed constraints (super-threshold). When a task with specific timing constraints executes on the NTC circuit, the supply voltage may be increased to the super-threshold region to meet the performance constraints. However, this

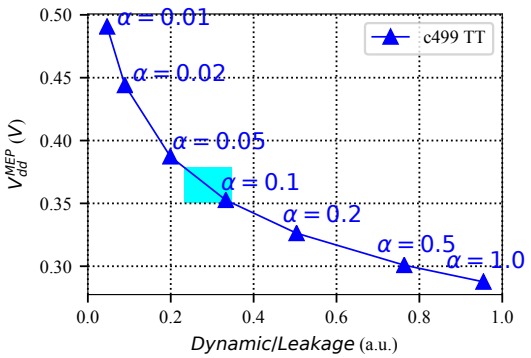


Figure 2.12: Workload variation impact on the MEP. The change in the V_{dd}^{MEP} versus the ratio of dynamic to leakage energy at nominal V_{dd}. The realistic change in the input switching activity α is displayed by the light blue box ($0.062 \leq \alpha \leq 0.108$) according to Section 5.2.1.

brings reliability issues associated with super-threshold voltage, such as aging, into consideration as the circuit may operate in that mode for some time. For example, an IoT edge device may be assigned to process some data within a specific amount of time and send them to an IoT gateway. Therefore, it is necessary to take the aging issues into consideration if a device is expected to operate over a wide-voltage range as this may have deteriorating impact on the device operation when it is switched back to the NTV region or even in the super-threshold voltage region.

Other challenges

STA tools are typically utilized to evaluate circuit timing. Conventional logic synthesis tools leverage STA to further optimize digital circuits and resolve timing issues. They rely on a corner analysis, which means that the circuit timing is evaluated for best and worst process and temperature corners. However, given the extent of variation in the NTC circuits, this approach is too pessimistic. Therefore, Statistical STA (SSTA) should be used to determine the timing of NTC circuits. Conventional SSTA tools propagate random variables, i.e. such as threshold voltage and transistor dimensions, to extract the distribution of output timing. This can be done based on Monte-Carlo analysis, which is extremely time-consuming, or analytical analysis. Based on Equation (2.22), given that V_{dd} or V_{th} are statistical parameters with a normal (Gaussian) distribution, the resulting delay distribution would be log-normal [160]. Therefore, the SSTA tools used for NTC circuits may need to consider such exponential sensitivity by propagating log-normal distributions, which makes the SSTA tools quite complicated.

2.4 State-of-the-art in resilient energy-efficient computing

As explained, keeping up with reliability requirements while gaining high energy efficiency is a major challenge for NTC design. Many researchers have focused on these issues to enable widespread use of NTC in various application domains. In this regard, the state-of-the-art can be divided into:

- Design flow and methodology optimization: Due to high integration and the complexity associated with circuit design in modern technologies, EDA tools are extensively used to enable circuit design and optimization. Hence, efforts have been focused on improving the EDA tools for NTC. Kaul et al [74] provided a list of challenges for NTC circuit design, from device modeling to test, and desired design technologies. Recent works have addressed some of the NTC challenges in device modeling [87, 160], variation modeling

[157, 184, 185], synthesis [186], leakage power management [186, 187], and system-level modeling [75]. There are still major shortcomings in variation-aware library characterization, verification, and testing.

- Design optimization for NTC: Various methods have been proposed to improve the design for NTC, from device-level to architecture-level. At circuit-level, robust standard cell libraries [28], memories [167, 169–171] and flip-flops [60, 74] are optimized for NTC. Level shifter designs have been vastly studied for interfacing between different voltage islands [173], and clock networks are redesigned for low-voltage operation [184]. Additionally, timing error correction methods such as [163, 188, 189] can be used to mitigate timing variations at runtime. At architecture-level, caches [190], processor pipeline [13, 191–193] and ISA [194] can be optimized, and other leakage power reduction methods are also investigated [28]. However, there are still many opportunities for cross-layer design optimization.

- Runtime optimization and tuning: Since the MEP is dependent on process and runtime variations, runtime optimization and tuning could be required depending on the running application and the operating environment. Therefore, various runtime tuning methods have been studied, mostly based on a closed-loop hardware-implemented monitoring circuit. It is proposed by many researchers to measure the circuit power online and take actions to maximize the energy efficiency by adapting the supply voltage in a closed-loop feedback [78, 195–198]. Supply voltage and threshold voltage tuning is also recommended to balance the speed of different cores [74, 185, 199]. However, closed-loop adaptation techniques are associated with additional circuitry for measuring the circuit power and applying the adaptation strategies, which may be too costly for NTC circuits. Therefore, low-cost adaptation methods are highly desirable for NTC.

- Wide-voltage reliability challenges: The impact of aging and supply voltage fluctuation (due to internal activity) is negligible in the NTV region due to low electric field, low current density, and low power consumption; however, operating over a wide-voltage range, from the near-threshold region to the super-threshold region may be required to satisfy performance constraints. In this case, aging phenomena affect the circuit in the super-threshold voltage region, which deteriorates the reliability. Therefore, it is necessary to address such aging challenges when applicable.

2.5 Summary

This chapter provided preliminary information regarding CMOS technology and energy-efficient computing. We started by explaining transistor and gates in digital circuits and explained the impact of voltage scaling on the energy efficiency of digital circuit. The challenges associated with voltage scaling towards NTV region were further discussed and the most important variability sources in NTC circuits were reviewed. Additionally, we provided an overview of the existing work and open challenges.

3 Variation-aware circuit synthesis and timing closure

Design and optimization of circuits are done using EDA tools. Due to the extensive impact of variability in the NTC circuits, EDA tools and methodologies for NTC should also consider the variabilities and their associated impacts. Unfortunately, the design automation methods which are typically used in the nominal voltage range are inefficient for NTC circuit design, as they are not meant to deal with such extreme variations. Therefore, in most cases, a statistical analysis is required. In this chapter, we approach this problem from the point of view of circuit timing and propose a *variation-aware circuit synthesis and timing closure* methodology [1, 2] to improve the reliability, energy efficiency, and performance.

The rest of this chapter is organized as follows. Section 3.1 presents a short introduction of the timing issues of NTC circuits, motivates the proposed method, and briefly explains the contributions. Section 3.2 reviews the most relevant work in this regard and the corresponding design automation challenges. Section 3.3 studies circuit timing in the NTV region under variability impacts. The variation-aware circuit synthesis and timing closure is presented in Section 3.4, and Section 3.5 proposes optimized buffers for fixing hold-time violation in NTC. The results of the proposed methodology are discussed in Section 3.6. Finally, Section 3.7 concludes the chapter.

3.1 Introduction, motivation, and contributions

The performance variation of NTC circuits due to global process variation is up to 20× larger than that of the nominal voltage range [60], while temperature variation can lead to as large as ±2× fluctuation in circuit speed[74]. Such escalated sensitivity to variabilities at reduced supply voltages forces the designers to add very conservative and expensive timing margins to achieve acceptable yield and reliability, which in fact erodes the benefits of the NTC.

In order to deal with huge performance variations, statistical information regarding the variation of circuit elements has to be considered during the logic synthesis phase. In the library pruning technique proposed in [28, 200], cells which exhibit higher performance variation are eliminated from the standard cell library in order to reduce the delay variation of synthesized circuits. However, by applying such a technique, cells with higher energy efficiency such as minimum sized cells are eliminated from the library for the sake of reducing the performance variation (as proposed by [28]). Hence, the energy efficiency of a circuit is limited when synthesized with a pruned library. Gate-sizing is another technique to improve the energy efficiency of the circuits in the near-threshold region [201]. However, gate-sizing techniques work on a fixed circuit topology after logic synthesis and technology mapping, and because of this limitation, their impact could be limited. In summary, a generic synthesis approach to address the huge variations in NTC is still missing.

Evaluation and fixing of hold-time violations is an important challenge for the NTC [76]. In a synchronous circuit, a hold-time violation occurs when a flip-flop is not able to correctly capture the input signal because the signal is not kept stable for long enough. This happens when the signal changes too early in a short path ending at the flip-flop [202]. It is well accepted that the sub-threshold circuit operation imposes a lot of hold-time violations due to

the large extent of variations [184, 203–206]. Our hold-time analysis in the NTV region shows that hold-time violations are also significant in the NTV region, similar to the sub-threshold region. Due to the magnified impact of process variation on delay in the NTV region, the clock tree buffers would have different propagation times which results in a considerably large clock skew [206, 207]. In addition to that, the hold-time and the propagation delay of the circuit flip-flops, as well as the delay of the short paths, vary significantly [208, 209]. Therefore, performing a corner analysis, which considers the maximum and minimum delays under process variation, as typically done in the super-threshold region for finding and fixing the hold-time violations in the NTV region leads to too much pessimism which may negate the benefits of the NTC.

This chapter presents our variation-aware synthesis and timing closure methodology for NTC circuits. We show that it is inefficient to perform synthesis and timing violation fixing without accurate statistical information about the impact of variability sources. Therefore, the proposed design flow optimizes the statistical delay of NTC circuits and finds and fixes timing violations by incorporating statistical delay variation of the circuit into account. The main idea is to apply additional timing constraints for both synthesis and timing violation fixing based on the information collected by performing an SSTA aiming at reducing the variation impact and improving the reliability, energy efficiency, and performance. An SSTA based on properly characterized standard cell library for the NTC can accurately extract the circuit timing by considering the statistical distribution of the delays. The cost of applying the SSTA might be too high if a Monte-Carlo technique is applied [208] because too many samples are required to make a reasonable approximation of the tails of the distributions. However, much faster block-based SSTA techniques could be used with better tractability at the expense of slight inaccuracy [210]. By comparing the results of the conventional approaches and the SSTA, we show that the SSTA is necessary to correctly find the timing violations in the NTV region, without imposing much pessimism. Then, the proposed circuit synthesis and timing closure methodology is applied to improve circuit timing and reliability, and find and fix timing violations with minimum overhead (energy, area) and runtime. The proposed method is wrapped around the standard (commercial) logic synthesis and timing closure tools and utilizes them in an iterative flow. Since the internal optimization engine of the synthesis tool is exploited, the proposed flow will automatically include all possible synthesis optimization techniques such as gate sizing, path restructuring, and logic borrowing.

Hold-time violations are typically fixed by increasing the path delay, e.g., by inserting delay elements in the short paths. However, due to the large number of hold-time violations in the NTV region, fixing them by buffer insertion imposes significant overhead due to the inefficiency of the conventional buffers originally designed for the super-threshold region [211]. This necessitates the use of efficient buffers for violation fixing in the NTV region. Transmission-Gate (TG) based cells offer reasonable energy efficiency as well as variation resiliency in sub-threshold circuits [212]; however, their efficiency in the NTV region has not been studied so far. Therefore, we optimize and compare several TG-based buffers for the NTV region, and evaluate the impact of using different buffer designs at the circuit level.

Optimization results show that the proposed flow can reduce the delay variation by 86.6%, which translates into 24.9% better performance, and improve energy efficiency by 7.4%, at the expense of modest 4.8% area overhead. It is also possible to trade off energy efficiency and performance in this variation-aware flow. Additionally, we demonstrate the advantages of using TG-buffers, since these buffers reduce the energy overhead of the hold-time violations fixing by 43.3% on average, compared to only using the standard CMOS buffers.

3.2 Related Work

Logic synthesis

The performance variation in NTC could be addressed by several techniques [74, 184, 186]. Since our proposed design flow lies within the synthesis and hold-time fixing, we review the techniques which are at the same abstraction-level.

The main idea of *library pruning* methods [28] is to remove the cells with high-performance variation from the standard cell library, which eventually results in a smaller circuit performance variability. For example, it is suggested to remove the cells with high sensitivity to supply voltage variation [200]. In this technique, cells which exhibit a performance variation beyond a certain limit are excluded from the library. Another approach towards library pruning is to eliminate cells based on their size and structure [28], as the delay variation of a cell is correlated with its size and structure. As a result, small cells which exhibit higher variation are eliminated from the cell library. However, the eliminated small cells are beneficial, as they consume considerably less power compared to the larger cells. These cells could be used in shorter paths of a circuit to save the energy, without affecting the performance of the circuit determined by the timing of critical paths. Therefore, this method cannot exploit the full capabilities of a circuit in terms of performance and energy efficiency.

Gate-sizing is a well-known technique to reduce the energy consumption of a circuit. A specific gate-sizing technique is proposed in [201], which employs As-Soon-As-Possible/As-Late-As-Possible (ASAP/ALAP) analysis to identify the cells which are not timing-critical. However, the improvement from this method is limited, as it does not incorporate other synthesis techniques, such as restructuring.

In addition to the aforementioned methods which aim at improving the circuit via synthesis, there are other techniques for improving the performance or energy efficiency of a circuit in NTC such as cell library optimization ([213, 214]), dual-V_{dd} operation ([186]) and body biasing ([185]). Since these methods analyze and improve a circuit from perspectives other than synthesis, they are orthogonal to the library pruning and gate-sizing and can be applied in parallel.

Hold-time fixing

Hold-time violation fixing in the super-threshold region has been studied, extensively. Majority of the work in the super-threshold region identify the hold-time violations by considering minimum and maximum path delays, i.e. corner analysis. Since the variability is not huge in the super-threshold region, this is an acceptable assumption. Therefore, the focus has been on reducing the number of inserted buffers [203, 215–219]. For example, linear programming techniques or a combined hold-time/setup-time fixing approach are proposed to minimize the number of the required buffers. However, variability impacts are much stronger in the NTV region and the benefits of these methods vanish due to performing corner analysis for finding the violations. In the NTV region, the impact of variation is much larger compared to the super-threshold region. All the delay distributions have long tails due to the exponential dependency of the transistor current on V_{th}, as explained by Equation (2.21). Therefore, the corner analysis is not an acceptable option as it leads to too much pessimism [208, 220]. Efforts have been made to minimize the power overhead or meet power requirements without considering the statistical delay distributions [204]. Additionally, alternative methods for designing robust clock-tree networks are proposed in order to minimize the timing violations without explicitly fixing remaining timing violations [206, 221]. A Monte-Carlo hold-time finding and fixing flow is proposed in [208], with the objective of reducing the overhead of Monte-Carlo simulations.

However, since delay distributions have long tails, still many samples are required to make a good approximation of the tails.

3.3 Circuit timing in the NTV region

In this section, we begin by explaining variation impacts on NTC circuits, and discuss our proposed approach which mitigates the impacts. For that, we first need to define some key terms. Figure 3.1 demonstrates the impact of performance variation on the delay of a path. The *nominal delay* of a path is the delay of that path without considering any variation. When considering a statistical variation, a path delay forms a statistical distribution. In this case, the path delay should be considered in such a way that the required yield is satisfied. For example $\mu+3\sigma$ of the delay distribution can be considered as the worst case which satisfies the yield constraints (μ and σ are the mean and the standard deviation of the delay distribution, respectively). We refer to this value as the *corner delay* of the path. The difference between corner delay and nominal delay is considered as the *delay variation*.

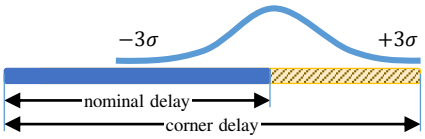


Figure 3.1: Path delay w/o considering the impact of process variation i.e. nominal delay (blue), and considering the variation (hatched area).

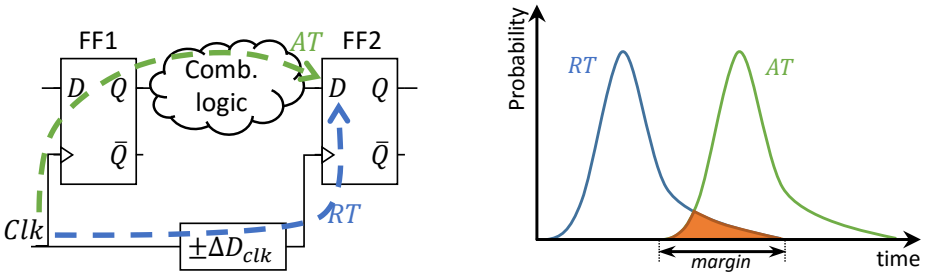
Delay variation impacts short and long paths, resulting in hold-time violation and circuit delay variation. Circuit delay is determined based on the delay of the critical paths, i.e., long paths. Therefore, considerable circuit delay variation imposes performance degradation, because longer clock period is required to avoid timing violations. Accordingly, addressing the impacts of variation on long paths reduces circuit delay variation and improves the speed. On the other hand, hold-time violations are critical from the reliability perspective, as they lead to permanent failures. Therefore, it is necessary to find and fix all the hold-time violations during the design time. As a simple illustration, in a circuit with two flip-flops and a combinational logic in between (Figure 3.2a), the hold-constraint for the second flip-flop can be defined as:

$$\begin{aligned} &\text{actual arrival time} && AT = D_{clk-q1} + D_{comb.}, \\ &\text{required arrival time} && RT = D_{hold2} + \Delta D_{clk}, \\ &\text{hold-slack} && H = AT - RT \geq 0, \end{aligned} \tag{3.1}$$

where D_{hold2} is the hold-time of the second flip-flop (FF2), D_{clk-q1} is the propagation delay of the first flip-flop (clock-to-q), ΔD_{clk} is the clock skew, and $D_{comb.}$ is the delay of the combinational logic (depicted in Figure 3.2a). In an extreme case, where there is no combinational logic (e.g. shift register), the hold-slack depends on the hold-time and propagation delay of the flip-flops and the clock-skew.

3.3.1 Impact of variation on hold-time constraints

According to Equation (3.1), any increase in the required time (RT) or decrease in the arrival time (AT) reduces the hold-slack, which can lead to a functional failure in the circuit. Therefore, an additional *margin* has to be considered to prevent such failure (i.e. by increasing D_{comb}). Since a single-corner nominal hold-time analysis is not able to capture the entire distribution of the variables, it cannot find all the hold-time violations.



(a) Flip-flop timing constraints (b) Timing constraints in NTV

Figure 3.2: (a) Flip-flops hold-time constraints for correct circuit operation. (b) Distributions of AT and RT in the NTV region.

In the super-threshold region, it is possible to find the required margins for fixing the hold-time violations with a *corner analysis* (i.e. considering the smallest AT and the largest RT). This is because the variations in AT and RT are not significant, and considering the hold-slack of flip-flop i as:

$$H_i = AT_{i,min} - RT_{i,max} \tag{3.2}$$

would not impose much pessimism. Therefore, a hold-time violation happens when $H_i < 0$. When the variation is not significant, the parameters of Equation (3.1) can be extracted by a corner analysis, as given in Equation (3.2).

In the NTV region, the amount of timing variation is much larger than that of the super-threshold region [60, 76]. The clock-to-q and the hold-time of the flip-flops also vary significantly due to the supply voltage reduction [209]. Another well-known timing variation is the large clock-skew in the circuits operating in low supply voltage [221]. All of these variations dramatically affect the hold constraints. Due to the large extent of the variations, the corner analysis is not applicable anymore, as it imposes large pessimism to the design by putting all the cells in either fast or slow corners [76].

The impact of large variation on Equation (3.1) is shown in Figure 3.2b, where parameters AT and RT are statistical variables. The shaded area demonstrates the region in which the hold-slack is negative, thus the circuit would fail without fixing hold-time violations. By considering AT and RT as statistical variables, the mean and the standard deviation of the hold-slack (H) can be calculated based on the mean and the standard deviation of AT and RT:

$$\mu_H = \mu_{AT} - \mu_{RT} \tag{3.3}$$

$$\sigma_H^2 = \sigma_{AT}^2 + \sigma_{RT}^2 - 2\rho\sigma_{AT}\sigma_{RT} \tag{3.4}$$

$$\Rightarrow \text{hold-slack} \quad H_{var} = \mu_H - 3\sigma_H > 0 \tag{3.5}$$

where ρ represents the correlation between AT and RT.

A corner analysis to extract the hold-slacks, as presented by Equation (3.2), considers the worst cases of both distributions ($AT_{i,min}$ and $RT_{i,max}$). This means that in a corner analysis, the correlation between AT and RT is pessimistically assumed to be -1, which translates into the maximum standard deviation of the hold-slack in Equation (3.4). However, we can even argue that the correlation coefficient ρ is a positive number, because both AT and RT are affected by the transition time at FF1 input. In the NTV region, the output transition times of the cells are large due to the supply voltage reduction. This negatively

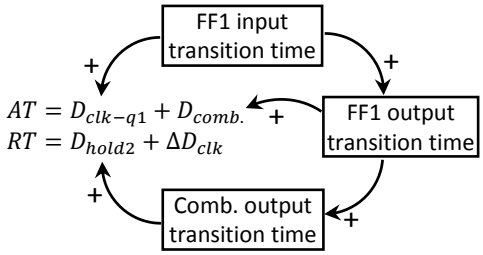


Figure 3.3: Dependencies between different variables leading to a positive correlation between AT and RT.

affects the consecutive cells, by increasing the timing parameters such as propagation delay or hold-time/setup-time. Figure 3.3 explains the impacts of FF1 input transition time on the parameters of Equation (3.1) ($\{D_{clk-q1}, D_{comb.}, D_{hold2}\}$). Please note that, since hold-time violations typically happen at short paths which contain small combinational logic (or even no logic), the impact of the input transition time at the start of these paths (e.g. FF1 output transition time) would be reflected at the end of these paths (combinational output transition time). Therefore, Figure 3.3 describes the positive correlation between RT and AT.

Due to the large amount of variation in the NTV region (large σ_{AT} and σ_{RT}) and the non-negative correlation between AT and RT, the corner analysis imposes significant pessimism to the design, and is not efficient anymore in the NTV region. To reduce the amount of pessimism and its associated costs, it is required to conduct statistical analysis by SSTA to extract the parameters presented in Equations (3.3) to (3.5). In order to be more realistic in our analysis, we calculate the hold-slacks of all flip-flops by considering $\rho = 0$ in Equation (3.4).

3.3.2 Hold-time analysis results

Comparison of nominal analysis, corner analysis, and SSTA: We have extracted the number of hold-time violations with nominal analysis, corner analysis, and SSTA for some large circuits in the super-threshold and near-threshold region. The violations are calculated after *Clock Tree Synthesis (CTS)* with the method proposed in [221]. As reported in Table 3.1, the SSTA finds 30.7x more violations compared to nominal analysis in the NTV region. Due to the large variation in this region, the violations not detected by nominal analysis lead to field failures. On the other hand, the corner analysis reports more violations compared to the SSTA (38.1x versus 30.7x) including many false violations, which are due to the pessimistic approach. Fixing these false hold violations imposes large overhead to the design. For this reason, we only consider the violations extracted by the SSTA.

Impacts of NTV operation on violations: Increased variability in the circuit timing leads to more timing violations. Since the distributions of AT and RT widen due to the increased delay variation, more constraints fall into the shaded region of Figure 3.2b, leading to more timing violations. Figure 3.4 demonstrates the increase in the number of hold-time violations for several benchmark circuits. The number of violations in the NTV region ($V_{dd} = 0.45$) is on average 7x more than in the nominal supply voltage range ($V_{dd} = 1.1$). In order to fix the large number of violations, many buffers have to be inserted into the violating paths to delay the arrival times (AT) of short paths. Therefore, the overhead of buffer insertion in the NTV region is significantly higher than that of the super-threshold region.

In addition to the increased number of violations, the worst negative slack (i.e. the largest negative hold-slack of all the constraints) also grows by reducing the supply voltage as shown in Figure 3.5. In order to effectively address this issue, buffers with large propagation delay are required. Without such buffers, several buffers with smaller propagation delay have to be

Table 3.1: Comparison of the number of the hold-time violations obtained by nominal analysis, corner analysis, and SSTA in the NTV region ($V_{dd} = 0.45$). The number are normalized to the number of violations from the corresponding nominal analysis.

circuit	# violations (normalized to nominal)	
	SSTA	corner analysis
b18	60.0×	83.7×
b19	73.5×	83.8×
Leon3	15.1×	17.1×
OR1200	3.2×	3.7×
DMA	1.9×	2.0×
average	**30.7**×	**38.1**×

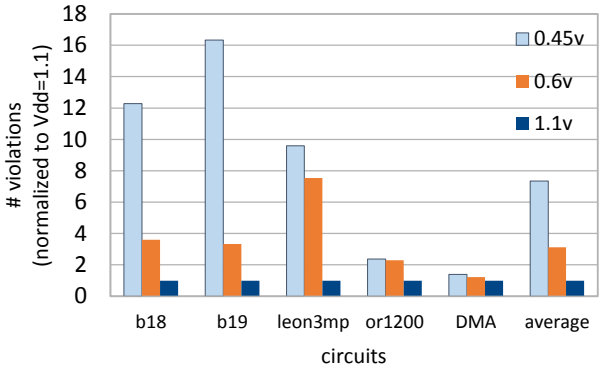


Figure 3.4: The number of violations for $V_{dd} \in \{0.45, 0.6, 1.1\}$ extracted based on SSTA.

inserted to provide the required delay for hold-time fixing, which increases the overhead of hold-time fixing.

Considering *the considerable number of violations* and *the large worst negative slack* in the NTV region, many buffers should be inserted to fix hold-time violations, which inflicts significant energy and area overheads. This clearly shows the need for the design and use of energy-efficient buffers for hold-time violation fixing in the NTV region. When the *Energy per Delay* (E/D) of a buffer is lower, the overhead of hold-fixing with that buffer would be less, as it provides a larger delay at the cost of smaller energy. Figure 3.6 demonstrates the E/D for different buffers of a 65nm commercial library operating in the super-threshold region. The numbers are normalized to the corresponding values of a CMOS buffer cell from the same library. As shown, the buffers have large E/D specifically for small delay values, but the buffers with larger delays have lower E/D. These buffers are not optimized for the NTV region, nevertheless, the overhead of hold-fixing with these buffers is still high due to the large E/D (specifically for lower delays).

Several techniques have been proposed for different voltage ranges to reduce the overhead of hold-time fixing by minimizing the number of inserted delay elements [204, 215]; however, it is essential to design and use the buffers which have lower E/D for NTC. In the next section, we study different buffer structures for near-threshold hold-time fixing, and optimize buffers for the NTV region.

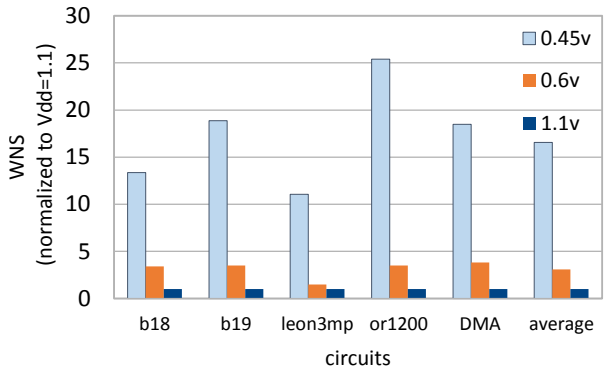


Figure 3.5: The hold-time worst negative slack (WNS) for $V_{dd} \in \{0.45, 0.6, 1.1\}$ extracted based on SSTA.

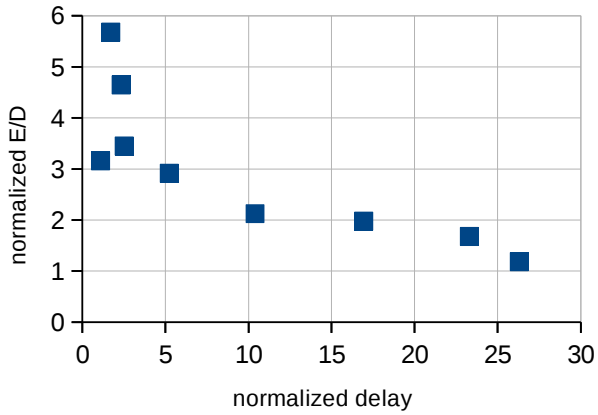


Figure 3.6: Energy per Delay (E/D) vs delay for the buffers of a commercial 65nm library in the super-threshold region.

3.4 Variation-aware logic synthesis and timing closure methodology

The amount of timing variation of a path is dependent on the number of cells in the path and their types, and therefore, the timing variation can be considerably different from one path to another. A balanced circuit timing at the design time can become unbalanced due to variations (see Figure 3.7a).

When no statistical information regarding the delay variation of the cells is provided to the synthesis tool, it tries to balance all path delays by loosening the shorter paths and tightening the critical paths according to their *nominal delays* measured by STA. In this case, since the synthesis tool is not aware of the *corner delay* of the paths (considering the delay variations), it cannot effectively improve the actual circuit timing, considering the delay variations. By incorporating the accurate information regarding the path delays extracted by SSTA, the synthesis tool is able to save area and energy by loosening the short paths and spend the saved area and energy on the critical paths to improve the circuit performance. By doing so, the resulting timing of the synthesized circuit would improve as shown in Figure 3.7b.

Accordingly, we propose an iterative variation-aware synthesis and hold fixing flow shown in Figure 3.8, which reduces the impact of variation on the circuit through better circuit synthesis, and reduces the overhead of hold-violation fixing by considering the statistical information of the circuit. The flow consists of three main parts: library characterization, variation-aware logic synthesis flow, and hold analysis and fixing flow, which are explained as follows.

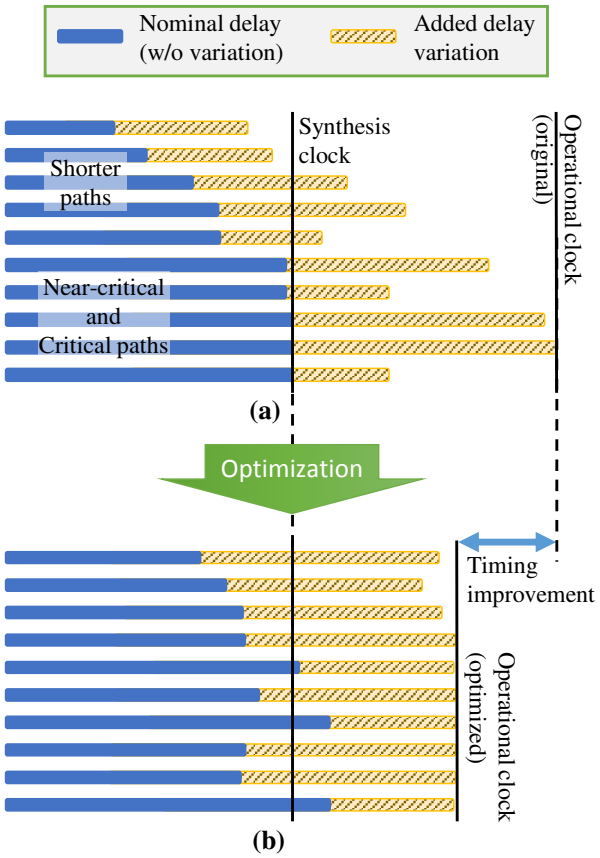


Figure 3.7: **a)** Paths' delays of a circuit synthesized without variation information, and the impact of process variation on each path. **b)** Paths' delays of the same circuit after variation-aware synthesis.

3.4.1 Cell library engineering

As shown in Figure 3.8, gates and flip-flops from Nangate cell library [222] are characterized based on their SPICE netlist and variation parameters, such as A_{vt} in Pelgrom's Model as explained in Equation (2.9), to obtain the *variation library*. The cells with more than three stacking transistors and specific cells which do not operate correctly in the NTV region are excluded from the cell library [28]. For this, we employ Cadence Variety tool [223] and Synopsys HSpice [224] and perform the characterization in the NTV region ($V_{dd} = 0.45$). The variation library contains nominal as well as statistical information of the cell delays, i.e. mean μ and standard deviation σ. Therefore, the characterized library is created in the format of Effective Current Source Model (ECSM) [225], which is capable of storing both nominal and statistical information. The SSTA tool uses this variation library to find the nominal and the statistical behaviors of the synthesized circuit. Although all variation information is available in the variation library, the synthesis tool is not aware of the performance variation because it uses the nominal delay values as the metric. Therefore, we generate a *combined library*, which contains the statistical information, for the purpose of logic synthesis. The combined library, which is solely used for synthesis, contains a weighted sum of the mean and standard deviation of the delay values (i.e. $\mu + \beta\sigma$) instead of the nominal delay values. The flow for obtaining the cell libraries is illustrated in Figure 3.8.

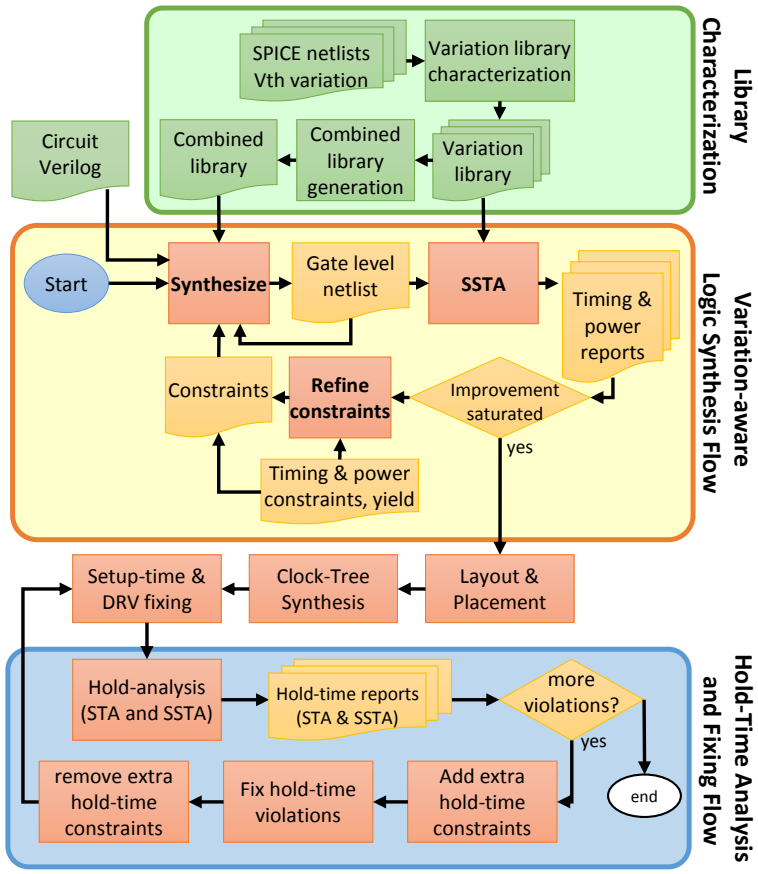


Figure 3.8: Iterative variation-aware synthesis and hold-time violation fixing flow consists of three parts: 1) library characterization, 2) iterative synthesis optimization loop, 3) iterative hold-time fixing.

3.4.2 Logic synthesis

In the proposed iterative variation-aware synthesis flow, two techniques are employed to inform the synthesis tool about the delay variation of the cells: First, the combined library is used for synthesis, which contains the variation information of all the cells in the standard cell library. Second, SSTA is used to evaluate the performance of the synthesized circuits. Afterwards, the variation information from the SSTA is fed back to the synthesis tool by adjusting the timing constraints. In our approach, the synthesis tool is free to use any cell that operates correctly in NTC to improve the performance or the energy efficiency. However, by carefully adjusting the timing constraints and providing variation information to the synthesis tool, the overall timing of the circuit (corner delay including variation) is improved.

A large circuit might have millions of paths. Therefore, it is not practical to apply constraints path by path. However, the paths end at circuit endpoints which are primary outputs, or flip-flop inputs. Fortunately, the number of endpoints is several orders of magnitude smaller than the number of paths. Therefore, instead of setting constraints for specific paths, only the endpoints of the circuits are constrained. We can define the *nominal delay* and *corner delay* for each endpoint similar to the way they are defined for a path.

Figure 3.8 presents the proposed iterative variation-aware synthesis flow. The goal of each iteration is to make the path delays more balanced considering the delay variations. In each iteration:

1. The circuit is synthesized with the combined library considering the constraints determined by the previous iteration (or no constraint for the first iteration).
2. The performance of the synthesized circuit under process variation is evaluated using SSTA. This gives the timing of all the circuit endpoints, i.e. $d_{i,SSTA}$.
3. Constraints are refined according to the extracted timing of each endpoint (based on SSTA). The constraints of those endpoints which have a positive slack are relaxed while the constraints of endpoints with a negative slack are tightened for the next iteration.

This loop continues until timing improvement saturates in several subsequent iterations. More details regarding each step are provided below.

Synthesis with the combined library

Synthesis tools implement an abstract circuit design with logic gates. The abstract design is commonly specified using VHDL or Verilog languages. During the process of synthesis, the circuit is also optimized for area and power consumption based on a set of constraints (such as timing and area). For this purpose, the synthesis tool takes the circuit specification, the associated constraints, and a standard cell library, and tries to find the best possible implementation by applying various optimization algorithms such as gate sizing, logic restructuring, time borrowing [226]. It reads the delay and power information as well as the load capacitance of the cells from the library, but it does not consider the corresponding variations. Based on this information, it decides which cell should be used and where. One way of providing the variation information to the synthesis tool is, by providing min-max libraries each containing minimum and maximum delays of each cell, respectively. This method is effective in the super-threshold region, where the amount of variation is small. However, all NTC cells have huge variations, and therefore, the result of min-max synthesis (and analysis) would give either too optimistic or too pessimistic results, which are not very helpful.

In order to effectively provide the variation information to the synthesis tool, we use the combined library during synthesis. The combined library feeds a combination of the mean and standard deviation of the delay values instead of the nominal delay values to the synthesis tool in order to guide it to choose more appropriate cells. Accordingly, we create the combined library by replacing each delay value in the cell library with $\mu + \beta\sigma$, where β is a coefficient specifying the contribution of variation in the delays of the combined library, and μ and σ are the mean and the standard deviation of the respective delay value from the *variation library*. In order to clarify, suppose that $\mu = 100ps$ and $\sigma = 20ps$ for a specific cell in the lookup table inside the *variation library*. If $\beta = 1$, the delay value which appears at the same location in the *combined library* would be $100ps + 1 \times 20ps = 120ps$. Please note that the combined library is only used for synthesis, as it contains information regarding variation. The reason is that, during synthesis, if the library contains no information regarding the variation of each cell (nominal library), the decisions of the synthesis tool are made based on the nominal cell timings. Therefore, the resulting performance variation of a circuit synthesized with such a library is not known. However, by providing the combined library (which already contains the effect of the delay variation of each cell) to the synthesis tool, the synthesis tool can make more informed decisions on the choice of cells and the resynthesis strategies.

Performance evaluation with SSTA

After each synthesis iteration, an SSTA is carried out. The SSTA uses the variation library for timing analysis and produces accurate variation-aware timing reports regarding the performance of the circuit. The statistical information from the SSTA will be used to adjust the

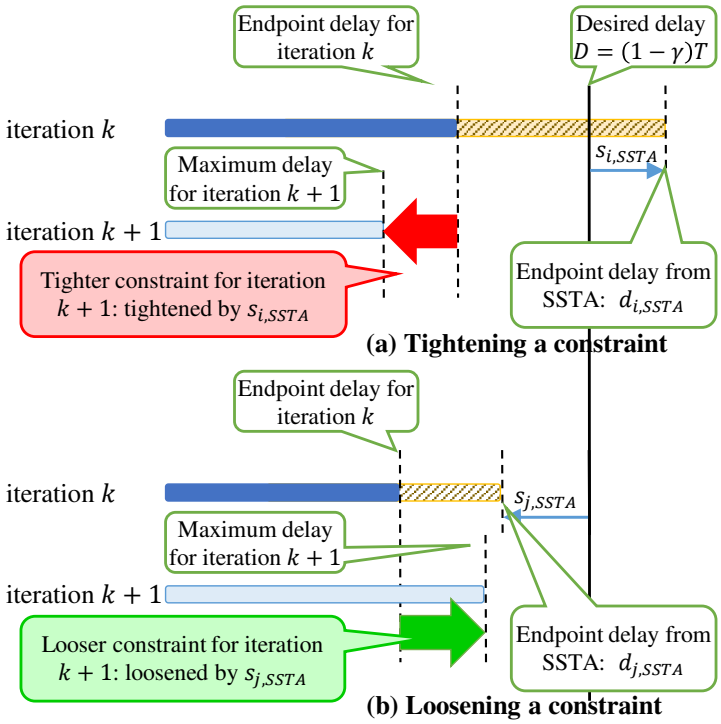


Figure 3.9: **a)** According to the timing of endpoint i calculated by the SSTA ($d_{i,SSTA}$) in iteration k, a tighter constraint is applied to this endpoint for iteration $k+1$, **b)** According to the timing of endpoint j calculated by the SSTA ($d_{j,SSTA}$) in iteration k, a looser constraint is applied to this endpoint for iteration $k+1$.

timing constraints for the next iteration. The *circuit delay* is the maximum of the corner delays of all the endpoints, and the slack of each endpoint is calculated as the difference between the corner delays of the circuit and the given endpoint.

Timing constraints optimization

In each iteration, the target is to improve the timing slightly (i.e., reduce circuit corner delay to *target delay*). Based on the target delay, the constraints are updated for the next iteration, such that the timing and power are improved in the next iteration. These constraints are then fed back to the synthesis tool again. This loop continues until the improvement saturates i.e. no improvement is achieved in several subsequent iterations. Please note that, although spending more time on the synthesis phase might lead to a better result due to its heuristic nature, the limited time budget for finding the optimum result necessitates the termination of the optimization loop when the improvement saturates.

The method to refine the constraints for the circuit endpoints is demonstrated in Figure 3.9. Timing constraints are adjusted based on the SSTA timing reports for all endpoints. Suppose that T is the *circuit corner delay* calculated using the SSTA. This is calculated as the maximum of all the endpoints' corner delays (considering the variations). We want to reduce T by a factor of γ in each iteration (improving the performance). Therefore, the *target delay* (D) for the next iteration would be:

$$D = (1-\gamma)T. \tag{3.6}$$

Based on D and the endpoint delays calculated by the SSTA, we can update the endpoint slacks for the next iteration:

$$s_{i,SSTA} = D - d_{i,SSTA} \tag{3.7}$$

where $d_{i,SSTA}$ is the delay of endpoint i calculated by the SSTA, and $s_{i,SSTA}$ is the statistical slack of endpoint i, respectively. When in iteration k, the slack of an endpoint is negative ($d_{i,SSTA} > D$) as in Figure 3.9a, the paths ending to that endpoint should be tightened. However, if the slack is positive ($d_{i,SSTA} < D$) as in Figure 3.9b, the paths ending to the endpoint should be relaxed. Therefore, the constraints of all the endpoints for iteration $k+1$ are tightened or relaxed by the corresponding amount of $s_{i,SSTA}$, respectively.

3.4.3 Hold-time analysis and fixing flow

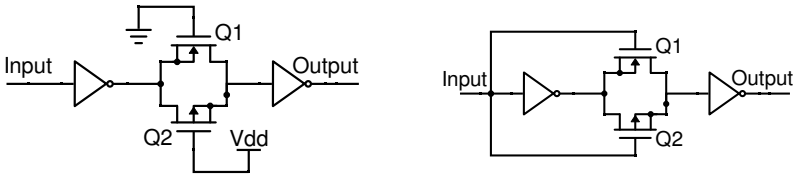
After the placement of the synthesized circuit and performing CTS, several violations appear. In a typical design flow, the hold-time violations are addressed after fixing the setup-time violations [220]. In NTC circuits, the number and amount of hold-time violations is vastly increased due to large clock-skew variation as well as combinational gates and flip-flops delay variation, as shown in Section 3.3.2. Here, we present an iterative hold-time fixing flow (shown in Figure 3.8) developed around existing static hold-violation fixing methods which further improves them by considering accurate statistical variation during the analysis and fixing.

As shown in Figure 3.8, the circuits are initially synthesized for NTV operation based on the aforementioned methodology in Section 3.4.2, and the placement and the CTS are performed using Cadence Encounter Digital Implementation (EDI) [227] afterwards. The CTS policy that worked out the best is to use the least clock buffer levels while using large cells to enhance the driving current, similar to the proposed methods for the ultra-low voltage circuits [221]. After the CTS, the circuit is analyzed and the setup-time and Design Rule Violation (DRV) are fixed. Then, the hold-time analysis and fixing flow is executed which is explained in details in the next subsection. Fixing the violations changes the netlist and the timing of the circuit, therefore it might generate new violations (setup-time/hold-time/DRV), which necessitates to repeat the violation finding and fixing steps.

The existing hold-fixing methodologies (such as [203, 204, 215]) do not perform SSTA to calculate the hold-slacks. As shown in Section 3.3.1, without such statistical approach, the hold-fixing would be too pessimistic or too optimistic, leading to either large overhead or functional failures. The proposed flow for fixing the violations incorporates impacts of statistical variation into the existing hold-time fixing methodologies. The technique is to modify the timing constraints of the circuit in the STA, such that the STA timing reports reflect the timing slacks reported by SSTA, and then apply a hold-violation fixing method. In other words, by modifying the timing constraints we guide a typical hold-violation fixing method, which is designed to fix the violations discoverable by STA, to also fix the hold-time violations which are only discoverable by SSTA. The variation library, which contains both nominal and variation information, is used to perform both analyses (STA and SSTA). The iterative SSTA-aware hold-time analysis and fixing flow is presented in Figure 3.8. The goal of each iteration is to fix the hold-time violations considering the statistical variation. In each iteration:

1. The hold-time slacks of all the endpoints of the circuit are extracted with STA and SSTA ($h_{i,STA}$, $h_{i,SSTA}$ for i^{th} endpoint). The statistical hold-time slacks ($h_{i,SSTA}$) are calculated using Equation (3.5).

2. The differences between the hold-slacks in the STA report and the SSTA report ($\Delta s_i = s_{i,STA} - s_{i,SSTA}$) are calculated for all the endpoints. Variable Δs_i declares the additional delay required at i^{th} endpoint to ensure the correct operation of the circuit under variation. All paths ending to this endpoint should be delayed by the corresponding Δs_i value (new constraints).



(a) TG1 (b) TG2

Figure 3.10: Different buffer structures. a) TG1: an always-on TG between two inverters [29]. b) TG2: a TG controlled by the input signal [30].

3. These new constraints for all the endpoints are applied to the design using `set_min_delay` command in the STA tool. As a result, the hold-time slacks in the STA report are updated according to the SSTA analysis.

4. The hold-time violations are fixed by running a conventional hold-time fixing technique (e.g. [203, 204, 215]) which inserts several buffers into the circuit to satisfy the timing constraints.

5. The added constraints (by `set_min_delay`) are removed from the design, and the circuit timing is restored to the original state.

A variation-unaware fixing method does not consider the statistical behavior of the inserted buffers; however, the buffers are also subjected to large variation in the NTV region. Therefore, it is required to check the timing to confirm the correct timing closure. Performing SSTA at this point determines the remaining violations. In case violations exist in the design, the flow is executed again from setup-time and DRV fixing.

3.5 Buffer optimization for NTC

A hold-time violation can be fixed by increasing the delay of the violating path, and buffer insertion is one of the common techniques to do that [203]. However, it imposes energy and area overhead to the circuit. In order to further enhance the effectiveness of the method, we optimize buffers specifically for hold-time violation fixing. In this section, we optimize the existing buffer designs [29, 211, 228] for the NTV region by appropriate transistor sizing.

A buffer is more efficient for hold-time fixing when it provides a large delay at the expense of small energy and area overhead. Therefore, we use energy per delay as a metric of efficiency. In addition to that, fixing the hold-time violations should have minimum impact on the rest of the circuit. However, many short paths might be parts of larger paths [215, 228] and inserting buffers into the short paths might deteriorate the timing of the critical paths. In order to minimize such a negative impact, the output transition time of the buffer should be as small as possible, to reduce the negative aspects of buffer insertion on the consecutive gates. Accordingly, the characteristics of the buffers are compared by 1) *delay*, 2) *output transition time*, and 3) *energy per delay* (E/D). Since the previous studies [29, 211] did not consider the change in the transistor model for the NTV region [160], the result of their analysis is not accurate for this region.

Review of the existing buffers

A *cascaded inverter buffer* (CMOS buffer), which is the most common buffer structure, is created by cascading two (or more) CMOS inverters. In the super-threshold region, this buffer

is commonly used because it provides a good trade-off between energy and delay and has a good output transition time and driving capabilities.

In the sub-threshold region, *Transmission Gate (TG)* based gates are advantageous in terms of energy efficiency, area, and variation resilience [212]. Unlike a CMOS gate, a TG-based gate does not have a direct leakage path from the supply line to the ground line, hence, it consumes less power compared to a CMOS gate. Moreover, balancing the size of PMOS and NMOS transistors of TG is not mandatory because both transistors are always operating in parallel. Therefore, they can be of the same size [212]. These properties make TG-based gates a suitable candidate for the NTV region, where delay variation is large and the energy efficiency is important. The schematics of two TG-based buffers, namely TG1 [29] and TG2 [30], are presented in Figure 3.10. TG1 buffer adds an always-on TG between the two inverters of a CMOS buffer, and thus increases the delay by increasing the time constant of the RC network. Although this is an efficient method to increase the delay, the output transition time also increases for the same reason [211] which would have a negative impact on the delay of the consecutive gates [215]. TG2 buffer [30] uses a structure similar to TG1 but since both transistors are controlled with the input signal, only one transistor is turned on in each state (input low or input high). As a result, the resistance is further increased, leading to a larger delay compared to TG1. Similar to TG1, this buffer also has a large output transition time.

Buffer optimization method

Cells are normally optimized to have minimum Energy Delay Product (EDP). The EDP optimization target is to minimize both energy and delay, hence, the cells optimized for the EDP would have small delay which is not desirable for hold-time fixing. As discussed in the previous section, a buffer is more efficient for hold-time fixing when it provides the maximum delay at the cost of minimum energy (and area). Furthermore, an optimized buffer in the super-threshold region is not efficient for hold-timing fixing in the NTV region due to two reasons: 1) it is optimized for a minimum EDP not a minimum E/D, and 2) it is optimized for the super-threshold region while the optimum ratio of the widths of PMOS to NMOS transistors depends on the supply voltage [212]. We used Predictive Technology Model (PTM) 45nm transistors and optimized the size of the buffer transistors by a non-linear least square optimization algorithm (Levenberg-Marquardt) to achieve the *minimum E/D* as well as the *minimum output transition time* in the NTV region ($V_{dd} = 0.45$) [229]. All buffers are designed to have the same area to be fair in our comparison.

Figure 3.11 compares different parameters of all buffers extracted from accurate SPICE simulations. The CMOS_ST buffer is optimized for the super-threshold region and the EDP, while other buffers are optimized for the NTC and E/D. We normalized all the parameters to the CMOS buffer for easy comparison. In order to evaluate the energy overhead of buffer insertion, we insert a buffer into a chain of inverters, and the reported total energy in Figure 3.11 is the effective energy overhead caused by the buffer insertion. The CMOS_ST has higher E/D compared to the other buffers optimized for the NTV region, which shows that the buffers optimized for the super-threshold region are not efficient in the NTV region. However, the results of our study show that the CMOS buffer which is optimized for the NTC is also not as energy-efficient as other types of buffers (see Figure 3.11 and Table 3.2), and thus, *cascaded inverter buffer* might not be the best choice for the NTC.

The energy of NTV-optimized TG-based buffers is very close to that of CMOS buffer. However, TG1 and TG2 provide significantly larger delay and better energy efficiency (lower E/D) compared to the CMOS buffer. The main weakness of the TG-based buffers is their large transition time; however, using the TG-based buffers in combination with the CMOS buffers which have better transition time will mitigate this issue. The characteristics of the

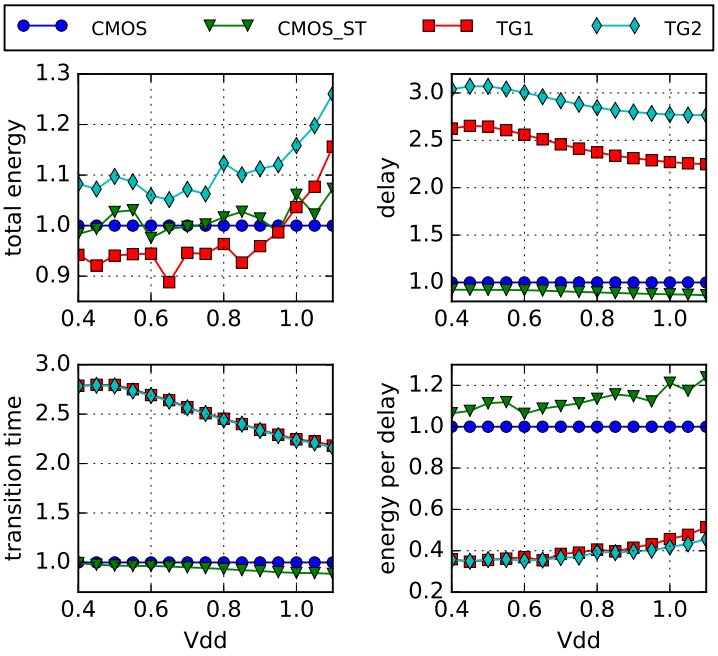


Figure 3.11: Properties of different buffers versus supply voltage. Total energy, delay, transition time and E/D are normalized to the corresponding values of an optimized CMOS buffer for the NTC.

Table 3.2: Buffer comparison in the NTV region. (Normalized to CMOS buffer)

buffer	near-threshold, Vdd=0.45			
	delay	energy	energy per delay	transition time
TG1	2.65×	0.92×	0.35×	2.80×
TG2	3.07×	1.07×	0.35×	2.79×

TG buffers are summarized in Table 3.2 for a specific supply voltage ($V_{dd} = 0.45$).

3.6 Results and discussion

In order to show the effectiveness of our proposed flow, we applied it to different circuits from IWLS 2005 benchmark[1] [234]. We have also compared it with the library pruning technique presented in [200].

3.6.1 Simulation setup

The cells from Nangate 45nm Open Cell Library are characterized for different supply voltages, ranging from 0.45V to 1.1V with Cadence Virtuoso Variety [223] and Synopsys HSpice [224]. The threshold voltages of NMOS and PMOS transistors of Nangate library are 471mV and -423mV, respectively. Therefore, the specified supply voltage range covers both super-threshold and near-threshold regions. The *variation library* is created in the ECSM [225]. This library is then converted into the *combined library* as described in Section 3.4.1. The optimum β

[1]IWLS 2005 benchmark contains 84 circuits from ISCAS'85 [26], ISCAS'89 [230], ITC'99 [231], Faraday technology [232], Gaisler Research [233], and OpenCores.

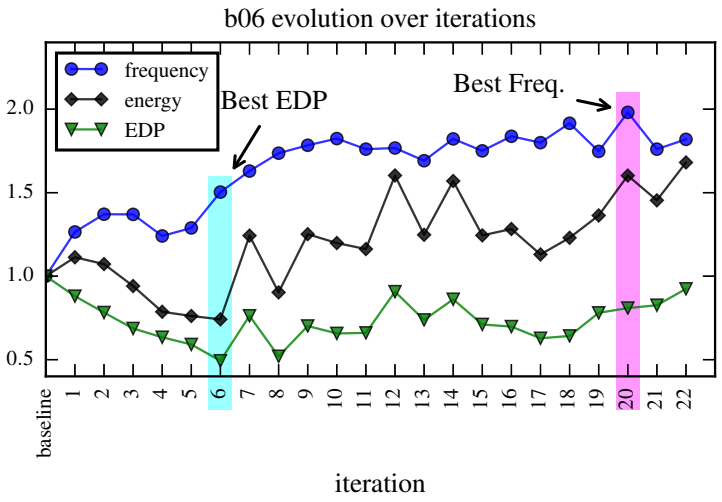


Figure 3.12: b06 circuit evolution over iterations: from baseline to iteration 22 ($V_{dd} = 0.45$, $\gamma = 0.05$). All values are normalized to the baseline.

coefficient for creating such a library is dependent on the circuit and the cell library. We tried a range of coefficients and empirically determined $\beta = 0.5$ to work the best for our synthesis. Synopsys Design Compiler is used for synthesis, and Cadence Encounter Timing System [227] is used for SSTA. The *energy per cycle* is then calculated based on the reported leakage power, dynamic power, circuit nominal delay, and circuit corner delay (from SSTA).

3.6.2 Logic synthesis results

This section explains the detailed result for a single circuit. Thereafter, the results for all circuits are presented in less detail for brevity.

Detailed results for a single circuit

Figure 3.12 shows the evolution of b6 benchmark circuit over the optimization iterations. Frequency, energy (per clock cycle) and Energy-Delay Product (EDP) are normalized with respect to the *baseline* in order to easily follow the improvement compared to the baseline. The *baseline* is the synthesized circuit using the traditional flow (with a normal library).

As shown in this figure, the maximum frequency of the circuit follows a rising trend over iterations, which means the performance of the circuit is improving compared to the baseline. On the other hand, the energy initially drops, but after some iterations it increases again. This means that by applying new constraints in each iteration, the synthesis tool can reduce the corner delay of the critical paths while retaining the power from shorter paths. However, after a certain iteration the synthesis tool has to increase the energy of the circuit in order to apply the new constraints. Therefore, there is an optimum point considering both energy and timing. This point can be discovered by calculating the EDP of the circuit. At the best EDP iteration shown in Figure 3.12, the maximum frequency is improved by 50.3% and the energy is reduced by 25.8%.

Another optimization target marked in this figure represents the iteration with the best performance i.e. best frequency. Although this leads to a 100% performance improvement, the energy overhead is also 50% which is not in line with the energy efficiency expected in NTC. For this circuit, we can conclude that iteration 6 is the optimal result for this circuit. Our target is to optimize both energy efficiency and performance i.e. optimize the EDP.

Results for all benchmarks

The optimization results for all ITC'99 benchmark circuits are presented in Table 3.3. We also implemented the library pruning method proposed in [200] for the sake of comparison. For each benchmark circuit, frequency, area, and energy per cycle of all circuits before optimization (i.e. baseline) and after optimization are extracted. For the optimized circuits, these values are presented as improvement percentages i.e. frequency (performance) improvement, area improvement, and energy improvement over the baseline synthesis. The presented variation improvement is calculated as:

$$\text{variation improvement } (\%) = \left(1 - \frac{\text{corner delay}_{\text{optimized}} - \text{nominal delay}_{\text{baseline}}}{\text{corner delay}_{\text{baseline}} - \text{nominal delay}_{\text{baseline}}}\right) \times 100 \quad (3.8)$$

For the presented simulation setup, the main improvement of the library pruning technique is the energy efficiency (9.7%), since it is not able to effectively improve the timing (just 2.5%). Our investigation reveals that by applying the library pruning technique, some fast cells are eliminated from the cell library, which prevents the synthesis tool from achieving better performance. In addition to that, when the library has fewer cells (as in the pruned library), the synthesis tool has fewer options for improvement. In contrast, our proposed method reduces the variation by 86.6% which corresponds to 24.9% better performance, as well as 7.4% better energy efficiency on average.

We have found that our proposed variation-aware synthesis flow can reduce the corner delay to be even smaller than the nominal delay achieved by the conventional synthesis flow for the largest benchmarks (i.e. b14, b17-b22), where the variation improvement calculated by Equation (3.8) is more than 100%, as shown in Table 3.3. The reason is that the synthesis tool puts a limited effort to optimize a circuit, and this effort is distributed over different endpoints to satisfy the constraints. In case the circuit is small, a considerable effort is put on each endpoint during the synthesis, which fiercely optimizes the circuit from the very first iteration. Therefore, for small circuits the synthesis flow usually saturates in a few iterations (b01-b13). For circuit b03 and b12, since no improvement is observed over iterations, the synthesis flow sticks to the baseline. The synthesis effort is initially distributed among all the endpoints which leads to a limited improvement when just one synthesis is performed (i.e. the baseline or the library pruning method). By defining constraints for the endpoints, we direct the synthesis tool to put more efforts on the critical endpoints (considering variations). As a result, the improvement from the iterative variation-aware synthesis is considerably large. Especially for large circuits which have a wide variety of short and critical paths, the improvement is comparatively higher. This highlights the scalability of the proposed approach and its effectiveness for large circuits. In summary, the energy efficiency of NTC is maintained while improving the performance.

The runtime overhead of the iterative variation-aware synthesis flow is on average 4.5× more than the conventional synthesis. Since the incremental synthesis feature of the synthesis tool is exploited, the runtime overhead is considerably smaller than the number of the required iterations to find the optimum EDP point. In each optimization iteration, the full state of the synthesis environment is loaded from the previous optimization iteration, and an incremental synthesis is performed with the new constraints.

As presented in Table 3.3, the number of the iterations and consequently the runtime is higher for larger circuits. Because these circuits are large, there might be many opportunities for improving their performance or energy efficiency. As a result, the synthesis tool is able

Table 3.3: Optimization results for ITC'99 benchmark circuits($V_{dd} = 0.45$, $\gamma = 0.05$).

benchmarks						improvements from library pruning [200]				improvements from proposed method					
circuit	gates	flip-flops	freq. (MHz)	area (μm^2)	energy per cycle (fJ)	variation (%)	freq. (%)	area (%)	energy (%)	best EDP iteration	variation (%)	freq. (%)	area (%)	energy (%)	runtime overhead
b01	65	5	358.68	61.71	0.31	27.3	10.5	-9.5	-7.0	7	36.6	14.5	2.2	12.2	5.0×
b02	32	4	444.84	28.20	0.10	5.3	2.1	-15.1	-3.8	2	36.0	16.6	-1.9	-3.2	1.1×
b03	185	30	331.24	193.12	1.03	-7.1	-2.3	12.0	11.4	0	0	0	0	0	0.8×
b04	1071	66	115.82	1070.92	11.50	-16.1	-3.8	15.0	16.6	2	14.2	3.6	8.9	11.0	2.0×
b05	2176	34	80.47	2107.25	26.70	23.8	5.6	5.3	13.8	15	106.2	31.4	-3.3	10.3	7.4×
b06	74	9	273.60	74.48	0.38	63.4	28.1	9.3	12.9	6	96.8	50.3	10.7	25.8	4.0×
b07	560	44	174.03	577.22	3.72	27.7	8.8	-3.7	1.8	7	55.9	19.6	-9.5	-0.4	4.0×
b08	204	21	253.10	219.98	1.28	-29.3	-7.7	11.5	10.5	2	-14.3	-3.9	8.22	6.02	1.0×
b09	224	28	243.90	239.40	1.59	-33.8	-10.1	22.4	22.7	9	56.3	23.2	-22.0	-11.2	6.5×
b10	227	17	222.22	223.71	1.24	-11.6	-3.4	11.2	8.5	7	82.4	32.8	-31.5	-14.1	2.8×
b11	865	30	152.91	929.67	7.17	-11.4	-3.4	10.3	15.7	2	29.1	10.0	0.9	7.5	2.7×
b12	1561	120	172.03	1523.91	9.30	-25.2	-7.0	15.5	19.9	0	0	0	0	0	0.5×
b13	364	48	267.88	376.39	1.75	24.9	9.3	8.1	9.6	2	17.2	6.2	8.5	13.1	2.0×
b14	12530	215	23.16	12681.02	488.33	67.4	13.3	2.2	14.6	9	143.7	33.3	-9.1	8.5	3.3×
b15	9215	417	70.49	9287.66	107.84	-36.4	-7.7	5.8	4.1	12	67.1	18.2	-5.3	5.1	3.5×
b17	26509	1317	63.26	27356.24	340.88	-38.7	-7.9	3.8	-0.2	27	102.1	29.3	-8.5	7.9	6.4×
b18	89049	3020	19.39	90167.08	3471.53	-1.6	-0.3	2.7	4.3	28	181.6	43.5	-9.0	15.6	10.0×
b19	164552	6042	17.91	168536.00	6900.49	34.7	5.8	3.5	10.3	24	234.2	58.0	-7.0	23.7	13.4×
b20	26226	430	20.72	26660.91	1179.30	41.1	7.1	3.8	12.7	19	206.7	50.4	-11.2	14.6	6.3×
b21	25847	430	21.29	26330.81	1137.99	79.0	15.0	3.5	16.8	25	195.4	47.6	-12.4	12.1	7.4×
b22	39451	613	22.01	39982.73	1667.53	5.5	0.9	3.9	8.8	12	171.0	37.5	-8.9	10.4	4.4×
average						9.0	2.5	5.8	9.7	10.3	86.6	24.9	-4.8	7.4	4.5×

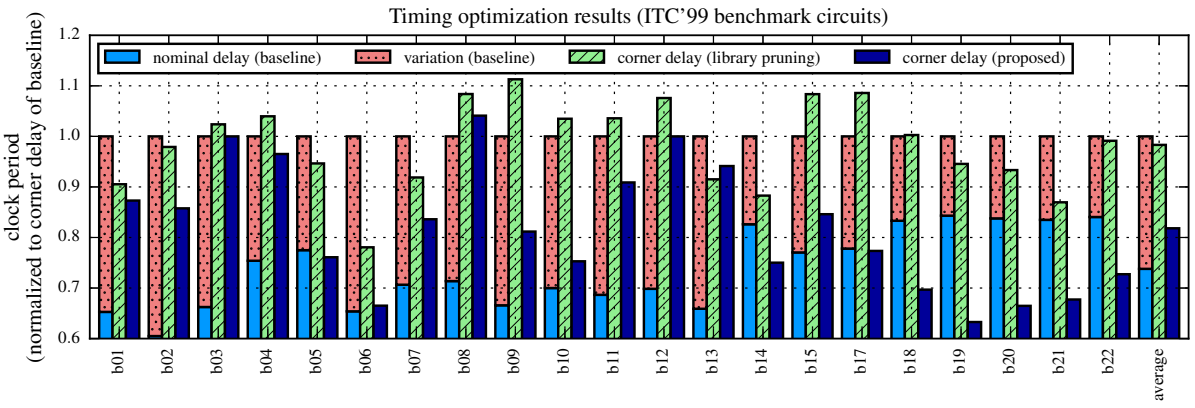


Figure 3.13: *Clock period* comparison of ITC'99 benchmark circuits with different methods ($V_{dd} = 0.45$, $\gamma = 0.05$). The contributions of nominal delay and variation for baseline are also presented.

to continue the optimization process for many iterations. This can be fixed by considering a higher γ coefficient e.g. 0.1, which can effectively decrease the number of iterations. However, since for the other circuits in Table 3.3, the results are reported for $\gamma = 0.05$, for the sake of consistency, the results for large circuits are also reported for the same γ.

Figure 3.13 compares the timing improvement of all ITC'99 benchmark circuits. The corner delay of the library pruning and our proposed flow are also shown in this figure. As illustrated, the corner delays of b17-b21 circuits optimized with the proposed flow are even smaller than the nominal delay of the respective baselines (i.e. without considering the variations). Although for these circuits the flow imposes an area overhead, the performance and energy improvements are also significant (see Table 3.3).

Improvement of the proposed method with V_{dd} reduction

Figure 3.14 presents the average of the improvements (performance and energy) of the benchmark circuits for different supply voltages. The average performance improvement is maximized at $V_{dd} = 0.45$, where the amount of variation is higher comparatively. The energy improvement is also better for the same supply voltage. When the amount of variation is larger, the difference between the corner delays of the endpoints is also larger. Thus, by improving a smaller number of critical endpoints (in terms of timing), a large performance improvement can be achieved. At the same time, the energy efficiency can be improved by loosening the shorter paths without affecting the performance of the circuit. However, when the amount of variation is smaller, the synthesis flow needs to deal with many critical or near-critical endpoints. This means that the synthesis tool needs to distribute the optimization effort among many paths, which usually leads to a smaller improvement. Therefore, we can conclude that on average the improvement of our method increases as the supply voltage reduces.

3.6.3 Hold-time fixing results

The major variability sources which affect the hold-time violations are flip-flop hold-time variation, path delay variation, and the clock skew. Since the clock skew is significant only in large circuits, we choose large benchmark circuits for our analysis (b17, b18, b19, DMA controller, Leon3 processor, and OpenRISC 1200 processor) which have many flip-flops (>1K) and large clock-tree structures. Table 3.4 reports the number of gates and flip-flops, the number of hold-time violations, and the total power consumption before performing any hold-time violation fixing.

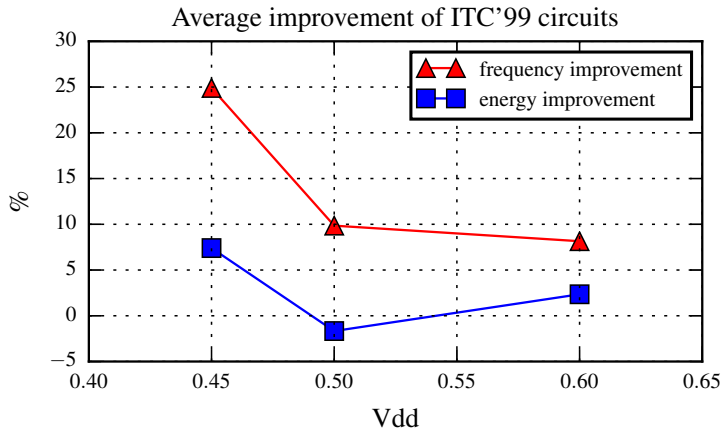


Figure 3.14: Average improvement of ITC'99 benchmark circuits over different supply voltages.

We execute the flow presented in Figure 3.8 for different types of buffers. The proposed hold-time violation fixing flow was able to successfully fix all hold-time violations irrespective of the type of buffers used. During the execution of the flow for a given buffer type, the hold-time violation fixing flow is forced to use only that type of buffer. Therefore, the impact of each buffer on the hold-time violation fixing is evaluated separately. Additionally, we run the flow with all types of buffers allowed to evaluate the impact of using all types of buffers together. Table 3.4 presents the results of these experiments, containing the power and area overheads due to the fixing.

As discussed in Section 3.5 and shown in Table 3.4, CMOS buffer is not efficient for hold-time fixing in NTC due to the energy inefficiency. Hold-time violation fixing with this buffer imposes comparatively larger energy and area overhead. Although a CMOS buffer has much better transition time compared to the other buffers, its energy inefficiency negates its benefits. The overheads of hold violation fixing with any of the presented TG buffers are smaller than those of CMOS buffers. Therefore, we can conclude that using TG buffers for fixing the hold-time violations is beneficial in the NTV region due to energy and area efficiency. The last column of this table demonstrates the result when all buffer types are used together. Allowing the violation fixing flow to use all types of buffers would provide the opportunity to leverage the benefits of all buffers while avoiding their drawbacks. A large output transition time of a buffer has a negative impact on the timing of the consecutive cells in the path. However, using energy-efficient TG-buffers in combination with CMOS buffer which has better transition time proves to be advantageous. In our experiment, this leads to slightly better results. In a real world scenario, this would be the preferred approach since it also provides more opportunities for fixing other types of violation (setup-time/DRV).

Table 3.4: Hold-time violations of the circuits before fixing, and fixing overheads for different buffers.

circuits	Before Fixing					Hold violation fixing using different buffer structures							
						CMOS		TG1 [29]		TG2 [30]		all buffers	
	endpoints	flip-flops	gates	power (μW)	violations	energy overhead (%)	area overhead (%)	energy overhead (%)	area overhead (%)	energy overhead (%)	area overhead (%)	energy overhead (%)	area overhead (%)
b17	1413	1317	33119	46.2	719	6.51	3.40	4.93	2.51	5.30	2.51	4.33	2.43
b18	3042	3020	104143	130.1	2161	4.00	2.88	2.92	2.40	2.92	2.40	2.69	2.31
b19	6072	6042	206219	249.7	5289	6.17	4.48	3.64	3.12	7.46	3.12	3.56	3.11
Leon3	3399	3074	49173	49.1	2715	15.25	13.50	8.37	9.12	10.46	9.12	8.12	7.45
OR1200	3054	2628	43491	67.2	2296	17.75	14.99	9.54	9.89	11.17	9.89	10.20	10.24
DMA	2393	2138	16614	43.7	2111	23.39	21.02	13.61	15.51	16.72	15.51	12.53	13.42
average	–					12.18	10.05	7.17	7.09	9.00	7.09	6.91	6.49
improvement over CMOS	–					–		41.1%	29.4%	26.0%	29.4%	43.3%	35.4%

3.7 Summary

This chapter presented optimized design automation methodologies for circuit timing improvement to address various NTC design challenges. The proposed synthesis and timing closure methodology can successfully reduce the impact of variations and improve the performance of NTC designs. The results show that the proposed method can reduce the variation by 86.6% (which means 24.9% better performance), and additionally improve energy efficiency by 7.4% with a small area overhead (4.8%). Furthermore, we showed that hold-time violations are much more severe in the NTV region compared to the super-threshold region, due to the increased effect of variations and the increased clock-skew, and the overhead of traditional hold violation fixing is significant which necessitates the use of SSTA for the hold-time violation analysis and fixing. Additionally, we optimized TG-based buffers and studied their applicability for fixing the hold-time violation in NTC, and proposed an iterative hold-time violation fixing flow for NTC based on SSTA. The simulation results demonstrate that incorporating TG-buffers reduces the energy overhead of hold-time fixing by 43.3% compared to traditional CMOS buffers.

4 Cross-layer reliability, energy efficiency, and performance optimization of data paths

Operating in the NTV region changes the circuit design paradigm as the assumptions for performance, power consumption, and variation impacts in this operating region are totally different from those of the nominal operating voltage [76]. Therefore, circuits should be specifically optimized for this operating region to satisfy the required energy-efficiency, speed, and reliability constraints. In this chapter, we discuss cross-layer design optimization techniques for NTC with a focus on processor data paths [3, 4, 6]. A processor data path comprises of several functional units such as ALU, Floating Point Unit (FPU), and multipliers. Therefore, we evaluate the effectiveness of the proposed methods on a 64-bit ALU, as a case study.

The rest of this chapter is organized as follows. Section 4.1 introduces the challenges of circuit design in the NTV region, motivates the proposed methods, and briefly explains the contributions. Section 4.2 reviews the state-of-the-art and their shortcomings. The optimization approaches based on instruction multi-cycling [3] and functional unit partitioning [4] are presented in Section 4.3, while Section 4.4 discusses the optimization results. Finally, Section 4.5 concludes the chapter.

4.1 Introduction, motivation, and contributions

Along with the enormous energy benefits, NTC comes with a variety of design challenges. The most obvious one is performance reduction of 10× compared to the super-threshold domain, which may limit the applicability of NTC [60]. In addition, the escalated sensitivity to variability (such as process and voltage variation) at reduced supply voltages forces designers to add very conservative and expensive timing margins to achieve acceptable yield and reliability [74]. Moreover, in the NTV region leakage energy becomes comparable to dynamic energy, thus approaches to minimize leakage are of uttermost importance for NTC designs. Hence, although conventional designs can technically operate in the NTC domain, due to these challenges, new design paradigms have to be developed for NTC to harness its full potentials.

Data paths are core components of any processing element such as processor cores and accelerators. A data path consists of various functional units, such as ALU and Floating Point Unit (FPU), whose timing and power consumption characteristics significantly impact the overall performance of the processor. Therefore, optimizing the reliability, energy efficiency, and performance of data paths is of decisive importance.

To save leakage energy, it is important to reduce the time a circuit is powered on but idle, i.e., the time a circuit is performing no operation. For this purpose, the timing slack under all operation conditions has to be trimmed. This is a very challenging task, in particular for functional units, which are fundamental components of data paths. In functional units, different operations have different timing criticalities and slacks. For instance, in an ALU, some operations need only a fraction of a clock cycle (e.g.,logic operations), whereas others require a full clock cycle (e.g.,addition operation with `long` operands). Consequently, whenever an operation of the first category is executed energy is "wasted" due to leakage, as the clock period is defined according to the slowest operation. Instead, it is much more efficient to

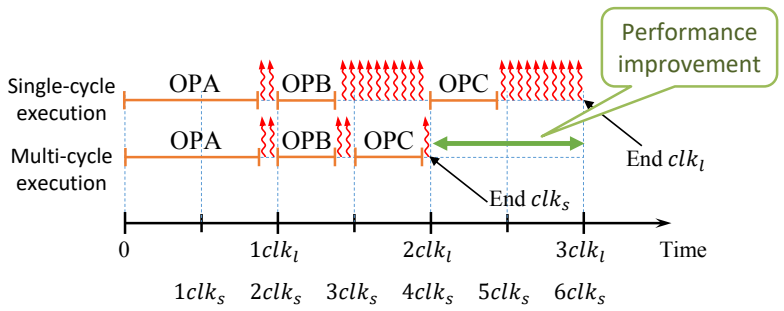
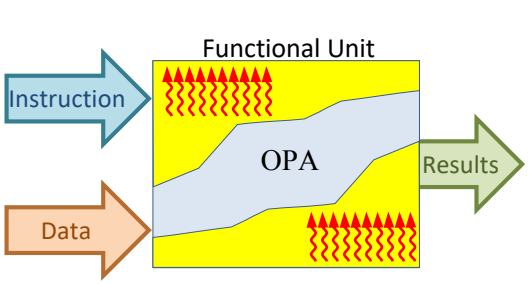
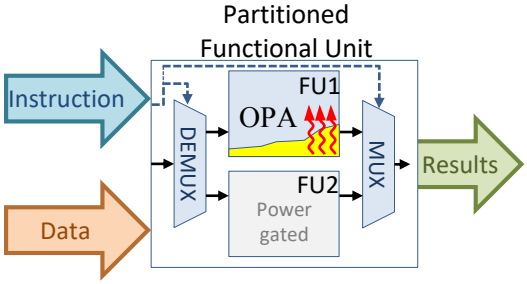


Figure 4.1: Conceptual illustration of the impact of a short clock period (clk_s) and multi-cycle operations (e.g. `OPA`) on runtime and "wasted" leakage. Leakage is illustrated by ≀.



(a) Executing instruction `OPA` on original functional unit, and the associated leakage dissipation by unexercised components



(b) Executing instruction `OPA` on the partitioned functional unit and power-gating of smaller units

Figure 4.2: Conceptual illustration of the impact of partitioning on a functional unit executing `OPA` instruction, and its impact on "wasted" leakage. Leakage is illustrated by ≀.

execute operations of the second category at multiple cycles, which reduces the overall idle time of the functional unit, as conceptualized in Figure 4.1.

Additionally, when an instruction is being executed by a functional unit, some parts of the functional unit are idle as they are not exercised by the executing instruction. To tackle this issue and improve the overall energy efficiency and reliability, we revisit the structure of the functional unit by partitioning it into multiple smaller and simpler units, to enable fine-grain power gating of unexercised functional units. If a particular functional unit is not utilized by a long sequence of instruction stream, it can safely be power-gated to save leakage energy, as shown in Figure 4.2. Proper clustering of the instructions into several smaller functional units allows maximizing the power-down intervals of multiple functional units, and hence reducing the leakage energy. At the same time, simplifying the functional unit can reduce timing uncertainties and improve a reliable operation of NTC processors under process and runtime variations.

This chapter presents two cross-layer functional units optimization opportunities based on 1) *instruction multi-cycling* and 2) *functional unit partitioning*, which improve energy-efficiency, reliability, and performance. We evaluate our methodology by using an ALU implemented for Alpha ISA. Our results show that the proposed approach can effectively improve energy efficiency and reliability. For example, instruction multi-cycling effectively removes the extra timing slacks and improves the energy efficiency of a circuit by up to 34%. Furthermore, the functional unit partitioning approach can improve the energy efficiency of an ALU by 43.4% while having positive impacts on the reliability and performance as well.

4.2 Related work

Adjusting the supply voltage of the circuit to its MEP depending on the process and runtime variations can improve energy efficiency. However, these techniques do not reduce the idle time of the circuit, when it is not doing any specific task but is wasting leakage power. As the leakage power constitutes a considerable portion of the total power consumption, it is crucial to reduce the idle time of circuits to improve energy efficiency.

Although many methods have been proposed to analytically find the MEP [59, 235] or track it during the runtime [78, 196, 197], finding the MEP point does not necessarily lead to the maximum energy efficiency as a global supply voltage is assigned to the whole circuit (coarse-grained supply voltage assignment). Fine-grained supply voltage assignment methods always lead to better improvement though imposing overheads [186, 236].

Performance variations in the NTV region can be addressed by using conservative techniques such as structural duplication, voltage and frequency margining [237]. However, such approaches can impose significant area and energy overhead. Moreover, the conservative timing margin will increase the idle period of the structures and potentially reduce the energy benefits of the NTC. Soft edge clocking [238, 239] and body biasing [240] are two other well-known approaches to deal with process variation at NTC. Several methods have been already proposed [1, 186], which incorporate synthesis techniques to improve the circuit characteristics for NTC.

Dynamic input vector variation is another source of variation [241]. By applying different input vectors, different parts of the circuits are activated, leading to different propagation delay from inputs to outputs. There are so-called *better-than-worst-case* design techniques to improve circuit performance or energy considering dynamic input vector variations [73, 163, 164]. In these approaches, the timing margin of the circuit is reduced to a value smaller than the conservative worst-case margin, and possible timing errors are detected and corrected on-the-fly to achieve error resiliency on top of improved performance and energy. However, traditional designs try to balance the circuit and hence there is a critical point called "path walls" which reducing the delay beyond this point leads to a huge number of timing failures. In order to reduce timing errors, and hence further improving the efficiency of *better-than-worst-case* designs, re-timing techniques to avoid path walls are introduced in the literature [165, 166]. However, such techniques are not effective in NTC because of the increased amount of timing errors.

For some circuits, such as ALU, there is a difference between the execution time of slow instructions (SI) and fast instructions (FI) due to dynamic input vector variation. Based on this fact, an aging-aware instruction scheduling is proposed in [242, 243] to execute each instruction group (FI or SI) with its own specialized functional units to improve lifetime.

In NTC, the delay difference between FI and SI is more pronounced. This means that if the clock cycle is set according to SI propagation delay, there is a considerable waste of leakage energy during the execution of FIs. Based on this, we propose a technique in which the clock cycle is significantly reduced close to the propagation delay of FI instructions to reduce the wasted leakage energy. In this approach, SIs are executed in multiple cycles to avoid possible timing failures.

At super-threshold domain, various coarse-grained power-gating techniques have been proposed to turn-off idle cores [244–246]. The cores are turned-off when determined idle time is observed. However, such approaches require state preservation mechanism and typically impose a long wake-up latency which is costly at NTC. Furthermore, two methods based on the power-gating of execution units are proposed in [247]. These techniques allow the power gating of an entire execution unit. However, some of the instructions of an execution unit are utilized rarely by the running applications. Therefore, analyzing the instruction utilization

pattern could reveal opportunities for fine-grained power-gating.

4.3 Cross-layer data path optimization

Overview

As explained in Chapter 2, leakage and dynamic energy are comparable in the NTV region. Therefore, it is crucial to control the amount of leakage power to leverage the benefits of the NTC. This section presents cross-layer methodologies to optimize energy efficiency, performance, and reliability of NTC functional units based on reducing the idle time.

The leakage power can be reduced by reducing the idle time of a circuit. A circuit may become idle within a clock cycle, for example when it has extra timing slack, or over consecutive clock cycles. The former may be avoided by modifying the circuit design to reduce the timing slack for all conditions, and the latter can be avoided by power-gating the circuit when possible.

The timing-slack minimization can be done by circuit synthesis techniques which are tailored for NTC circuits [1, 186]. However, these approaches are not much efficient when dealing with functional units (e.g., adders, multipliers, or complete ALUs) as the timing slacks of the instructions may be widely different [3]. For example, a simple instruction such as a `Bitwise-AND` only needs a fraction of the clock cycle, whereas an `ADD` instruction could use a large portion of the clock cycle. As the delays of these instructions are intrinsically different, the NTC synthesis techniques are not able to effectively balance the delay of these two instructions. We propose to execute the slow instructions in multiple cycles and the fast instructions in a single clock cycle (*instruction multi-cycling*) as the solution to this problem [3]. Applying other optimization techniques such as opportunistic circuit synthesis, instruction replacement, and data type manipulation can further improve the effectiveness of the proposed method.

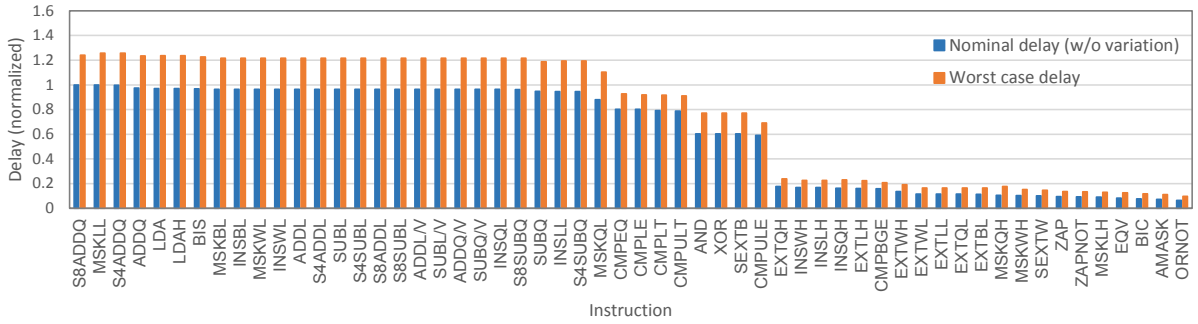
In a real-world scenario, the running application may utilize only a fraction of the instructions implemented inside a functional unit. This means that during the execution of such applications, those gates of the functional unit which are exclusively used by the non-exercised instructions of the functional unit are not utilized. The leakage power from these gates contributes to the *wasted* energy of the system. On the other hand, it is not feasible to power off the entire functional unit because any of the instructions can be called at a time. We propose to address this by redesigning the entire functional unit to allow power-gating [4]. In this approach, a large functional unit such as an ALU is partitioned into several smaller functional units such that each unit can be power-gated separately (*functional unit partitioning*). For this purpose, the instructions need to be clustered properly into different groups. A number of parameters such as the instruction utilization pattern, the temporal distance between the instructions inside an application instruction stream, and intrinsic similarity between the instructions need to be considered for a proper clustering. Accordingly, the instruction stream of various applications has to be analyzed to extract the required information for clustering [4].

4.3.1 Instruction multi-cycling

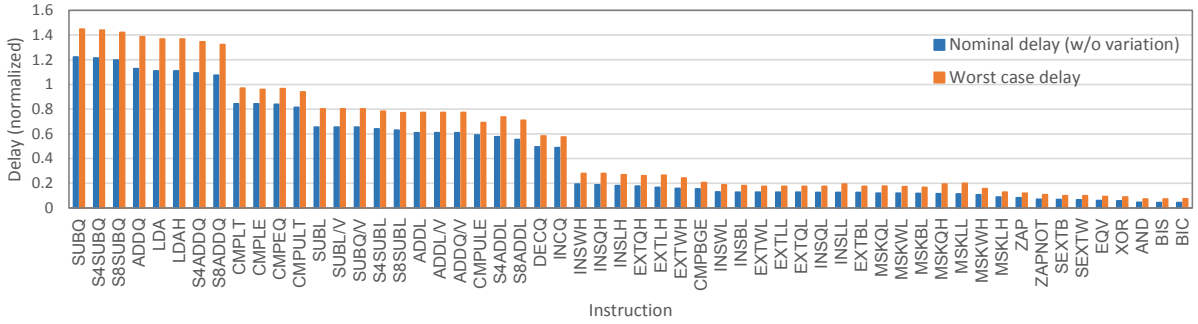
This scheme consists of four main ideas to fully enable the potential of NTC:

1. We classify the instructions into *slow* (with little timing slack) and *fast* (with large timing slack), based on the time required to execute the instructions. The slow instructions are executed in multiple clock cycles, to reduce the leakage power while executing fast instructions within one clock cycle.

2. In contrary to the super-threshold regime, we use relaxed timing constraints for the synthesis of functional units in the NTV region. While this increases the delay of the



(a) An ALU synthesized with typical timing constraints (Tight ALU).



(b) An ALU synthesized with loose timing constraints (Loose ALU).

Figure 4.3: The nominal case delay without considering the impact of process variation and the worst case delay (with process variation) for the instructions of an ALU synthesized with typical and loose timing constraints. Worst case delay is extracted by SSTA as $\mu + 3\sigma$ of the instruction delay. All the numbers are normalized to the maximum nominal delay of the Tight ALU which is the nominal delay of `S8ADDQ` in (a). ($V_{dd} = 0.5V$)

most critical path, it avoids that all instructions have similar delays and belong to the same category. By that means, the scheme of point 1 is much more efficient.

3. Finally, the clock period is not only set according to points 1 and 2 but also by considering the sensitivity of the functional unit to variations. As a result, it is possible to co-optimize energy, performance, and reliability.

4. In order to reduce the number of "slow" multi-cycle operations, we propose to employ instruction- and compiler-level optimization approaches to replace some of these instructions in the code with fast single-cycle instructions, which further improves energy and performance.

Energy improvement through instruction multi-cycling

The propagation delay of functional units is typically dependent on input vectors. For example in an ALU, the instructions can be categorized into slow instruction and fast instructions and the delay gap between fast and slow instructions might be significant. The logical operations such as `inversion` can be executed very fast while the execution of arithmetic operations such as `addition` and `subtract` requires more time. Figure 4.3a shows the normalized delay of various instructions of an ALU at 0.5V (according to the simulation setup presented in Section 4.4). The synthesis tool balances as many paths as possible to optimize the energy given the timing constraints. In this case, 26 instructions are balanced to be critical or near-critical in terms of timing (slow instructions); however, the rest of the instructions require less time for completion (fast instructions).

In a traditional design approach, the delay of the unit is determined by the delay of the slowest instruction. Therefore, during the execution of the fast instructions, the ALU finishes the execution early and leaks for the rest of the clock cycle. The "wasted" leakage energy is more important in the NTC because 1) the leakage energy is significant and constitutes a significant portion of the total energy, 2) the delay gap between the fast instructions and the slow instructions is even more pronounced due to the impact of process variation. As shown in Figure 4.3a, the amount of variation induced delay (the difference between the nominal delay and the worst-case delay) is larger for the slow instructions which significantly increase the delay gap between the slow and fast instructions in the near-threshold regime.

Our proposed idea is to reduce the leakage energy by using a shorter clock period and executing the slow instructions in multiple cycles. For example in the presented ALU in Figure 4.3a, it is possible to reduce the clock period to half of the delay of the slowest instruction and execute the slow instructions (from `S8ADDQ` to `CMPULE` - totally 35 instructions) in two cycles and the rest of the instructions in one cycle. Therefore, once a fast instruction is executed the amount of the wasted leakage energy would be much smaller compared to the traditional approach. This would also improve performance because fast instructions require less time for execution.

For the execution of the slow instructions which may require more than one cycle, there is no need to insert any flip-flops or latches into the circuit. During the execution of such instructions, the inputs of the circuit are kept unchanged in order to allow the circuit to complete the execution of the slow instructions in more than one clock cycle. Therefore, it is needed to make some modifications in the microprocessor as discussed in Section 4.4.4.

We propose a set of cross-layer techniques from logic synthesis to compiler level in order to leverage the maximum benefits from the proposed method by moving more executed instructions to the "fast" category.

In order to show the impact of the logic synthesis on the ALU, we synthesized the same ALU with two different strategies. First, the ALU is synthesized with tight timing and area constraints which is the default strategy in the super-threshold region. From now on, we refer to this synthesized ALU as *Tight ALU*. As the results depicted in Figure 4.3a show, the synthesis tool balanced the delay of many instructions (mostly arithmetic instructions) to maximize the energy efficiency and performance. However, there are still many instructions (mostly logic instructions) which are too short to be balanced similar to the slow instructions. Furthermore, there is a sharp transition from the delay of slow instructions to the fast instructions.

Then, the same ALU is synthesized with loose timing and area constraints (*Loose ALU*). The results of this synthesis are shown in Figure 4.3b. Since the synthesis tool is not under tight constraints, the delays of the slow instructions are larger compared to the Tight ALU, and there is a smooth transition from the delay of the slow instructions to the fast instructions.

The synthesis results show that the performance of the Tight ALU is better than the Loose ALU by 13% considering the variations; however, due to the wide spectrum of the delays of the instructions in the Loose ALU, there are more opportunities for improving the ALU with the envisioned multi-cycling technique. The reason is that fewer instructions are critical in terms of timing and by cleverly choosing the clock period we can gain in terms of energy and performance as shown later in Section 4.4.

High-level optimization techniques

From the above discussion, we observe that some ALU instructions are slower with longer execution time, while other instructions are faster with shorter execution time. Therefore, the aim is to exploit high-level optimization techniques such as data type conversion and instruction replacement to replace slower instructions of an application by fast ones wherever possible,

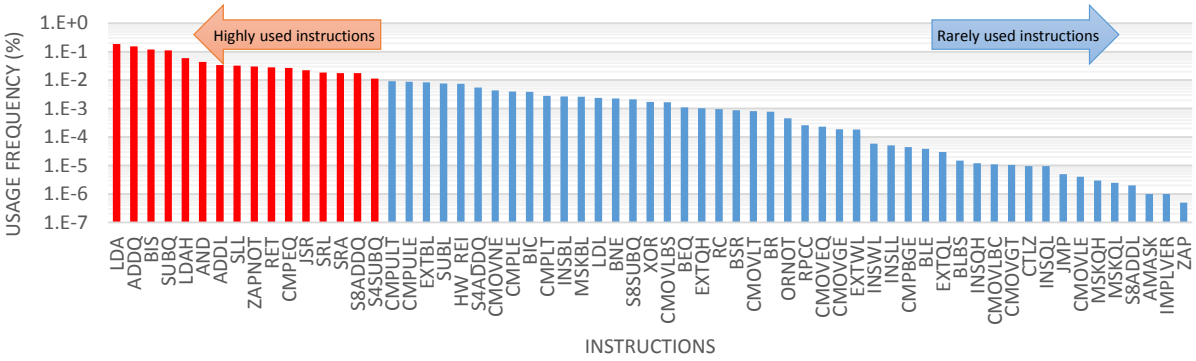


Figure 4.4: Instruction usage frequency in a 64-bit ALU for gzip workload. There are orders of magnitude difference between utilization frequency of *'highly used instructions'* on the left and *'rarely used instructions'* on the right.

so that the overall energy demand will be reduced further. Since the slower instruction are executed in multiple clock cycles, replacing them with faster instruction can also improve the performance by reducing the Cycle Per Instruction of the applications.

Data type conversion The selection of data types for variables has a huge impact on the instruction binary generated by the compiler. The data types such as `short` (1-byte), `char` (2-byte), `int` (4-byte) and `long` (8-byte) does not only specify the storage space size of variables but also the type of operation on the variables (e.g., 64-bit arithmetic such as `ADDQ` vs. 32-bit arithmetic such as `ADDL`). This affects the occurrence rate of slower and faster instructions within an application. Hence, in addition to memory optimization, a clever data type selection will help to save energy by using fast instructions (e.g., `ADDL` in Figure 4.3b) instead of the slow ones (e.g., `ADDQ` in Figure 4.3b).

Instruction replacement Many applications such as sorting and matrix multiplication algorithms spend most of the execution time in loops. Hence, simple instructions such as increasing loop counters and array indexes significantly contribute to the overall instruction count. If such instructions are not assigned appropriate data type and operation, they can impose significant performance and energy overhead. For instance, since modern ALUs have a dedicated increment/decrement circuitry, the usage of `increment/decrement` instructions plays a vital role in improving the performance and energy efficiency by reducing the number of slower instructions. This and other instruction replacements (e.g. shift instead of multiplication) can be achieved either by the programmer or using different compiler optimization techniques.

4.3.2 Functional unit partitioning

Fine-grained power-gating has been known as an effective solution for reducing the leakage power consumption of a system by cutting the supply lines of the idle components [247]. However, it is costly to power-on and power-gate the components in terms of execution time as each power-gating cycle mandates a minimum time between power-on and power-off cycles. Once a component is put to sleep, its functionality cannot be used. Therefore, the components should to be carefully tailored towards power-gating.

One opportunity for power-gating of the components is to determine the unused parts of a circuit based on the running application and power-gate those specific parts instead of the entire component. For ALUs executing several instructions, unused parts of the ALU corresponding to the instructions not executed for a long time could be safely power-gated.

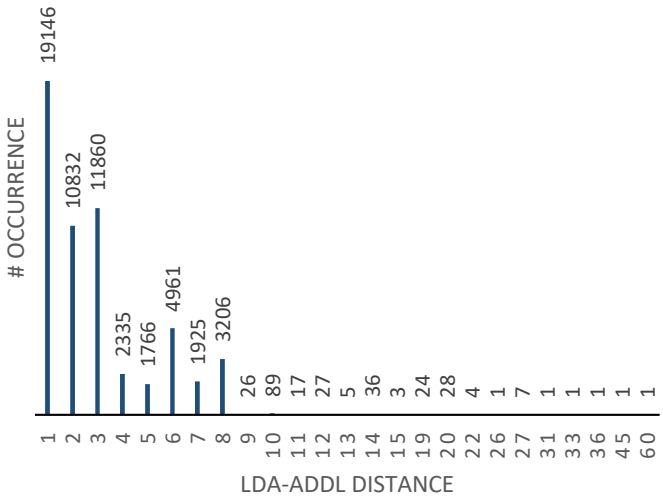


Figure 4.5: The temporal distance between `LDA` and `ADDL` instructions in "bzip2" workload (simulation for 2 million cycles). There are 19146 cases that the `ADDL` instruction appeared right after `LDA`. The average distance is 2.97.

Figure 4.4 presents the usage frequency of a 64-bit ALU (for Alpha ISA). As shown in the figure, there are orders of magnitude difference between the usage frequency of a highly used instruction such as `LDA` and a rarely used instruction such as `ZAP`. If both of the mentioned instructions are implemented in a single ALU, the circuitry for `ZAP` is leaking most of the time while being at idle state. In other words, the gates which are implemented exclusively for the rarely used instructions are in the idle state most of the time and are contributing to the total leakage of the ALU.

Our proposed solution is to partition a functional unit like an ALU into multiple smaller ALUs. This allows us to power-gate some of the ALUs based on the utilization of their instructions by turning off the ALUs which are not used at the moment. The original ALU could be partitioned based on different parameters such as the instruction utilization frequency, instruction similarity, and instruction temporal proximity.

In summary, this scheme synergistically exploits NTC in conjunction with a fine-grained power-gating of functional units to enable energy-efficient operation of devices designed for IoT applications. A hierarchical clustering algorithm groups the instructions into smaller functional units by considering the frequent instruction sequences, obtained by application profiling, in order to maximize power-gating intervals. Accordingly,

1. We characterize the instruction flow of representative workloads and analyze the instruction stream in order to obtain instructions' utilization frequency and temporal distance.

2. The instructions are then partitioned into several groups according to the metrics extracted from the instruction stream analysis as well as the inherent similarity of the instruction.

3. Each set of instructions residing in the same partition will be implemented by a dedicated functional unit, and they form a complete functional unit altogether.

4. To reduce the leakage power, only the functional unit corresponding to the running instruction partition is activated while other units are power-gated.

Instruction pattern analysis

A careful analysis of the instruction patterns on a set of representative workloads provides information regarding the ALU partitioning. For this purpose, we first simulate the execution of a set of representative workloads by an architectural simulation tool as explained in Section 4.4, and then based on the extracted instruction streams the *utilization frequency* and the *temporal distance* of different instructions are extracted.

Instruction utilization frequency The instruction utilization frequencies presented in Figure 4.4 shows a significant difference in utilization among the instructions. The instruction utilization frequency has an inverse relation with the power-gating feasibility for the ALUs associated with the instructions. For example, the ALUs which contain instructions `ADDQ` and `BIS` are less likely to be power-gated because these instructions appear in the instruction buffer on average every 10 cycles. However, it is more likely to power-gate the ALUs implementing the rarely used instructions on the right side of Figure 4.4 such as `S8ADDL`.

The utilization frequency of an instruction can be simply defined as the number of cycles in which the instruction is executed divided by total cycles. For a given instruction stream $\mathbf{S}$, which is a sequence containing $N = |\mathbf{S}|$ instructions, we can define the frequency of instruction `A` as:

$$Freq_{\texttt{A}} = \frac{\#S_i \in \mathbf{S} \;\; such\ that\ S_i = \texttt{A}}{N}$$

where S_i is the i-th element (instruction) of $\mathbf{S}$. If an instruction is used rarely, it can be easily grouped into any existing ALUs. However, we need to be more cautious in grouping frequently used instruction because the grouping strategy might improve or deteriorate the power-gating capability of the ALUs. Based on this analysis, we can define the *frequency distance metric* between two instructions `A` and `B` as the geometric mean:

$$dist^{freq}_{\texttt{A-B}} = \sqrt{Freq_{\texttt{A}} * Freq_{\texttt{B}}}.$$

Such definition facilitates the partitioning of rarely used instructions into a single ALU.

Instruction temporal distance Some instructions are more likely to appear next to each other in an application. For example, from "bzip2" workload, we observed that `ADDL` appears after `LDA` on average every 2.97 instructions (see Figure 4.5). In these cases, it could be beneficial to group these neighboring instructions inside one ALU in order to improve the power-gating interval for other ALUs.

The temporal distance between two instructions `A` and `B` can be extracted based on the number of cycles between any occurrences of `A` and `B`. For example, according to the results presented in Figure 4.5, there are 19146 `LDA` instructions which are directly followed by an `ADDL` instruction, and there are 11860 `LDA` instructions which are followed by an `ADDL` instruction after three cycles. Based on the workload analysis, a distribution is extracted for every instruction pairs `A-B` which explains the percentage of the `A-B` pairs that are far from each other by k cycles. The Survival Function (SF) [1] of these *temporal distance* distributions is useful in the clustering problem definition in Section 4.3.2.

We define a *temporal distance set* which contains the indexes of consequent instructions `A` and `B` and their corresponding distances (i, k):

$$Temporal_{\texttt{A-B}} = \big\{(i,k) \;\big|\; 0 < i,\ 0 < k, S_i = \texttt{A},\ S_{i+k} = \texttt{B}, \nexists j,\ 0 < j < k,\ S_{i+j} \in \{\texttt{A}, \texttt{B}\}\big\}.$$

[1] is defined as $1 - CDF$ of a distribution

Accordingly, the Probability Mass Function (PMF) based on distance k is calculated as:

$$PMF_{\texttt{A-B}}(k) = \frac{\left|\{(i,r)|(i,r) \in Temporal_{\texttt{A-B}},\ r = k\}\right|}{|Temporal_{\texttt{A-B}}|}.$$

The extracted PMF is then used to find the SF as follows:

$$CDF_{\texttt{A-B}}(k) = \sum_{i=1}^{k} PMF_{\texttt{A-B}}(i),$$
$$SF_{\texttt{A-B}}(k) = 1 - CDF_{\texttt{A-B}}(k).$$

In a fictitious scenario where only instructions `A` and `B` exist, and they are divided into two ALUs, the SF can explain the power-gating possibility. In this case, if the minimum number of cycles required to perform a power-gating (power-gating threshold) is $PGTH$, then $SF_{\texttt{A-B}}(PGTH)$ obtains the power-gating probability. Therefore,

$$dist^{temporal}_{\texttt{A-B}} = SF_{\texttt{A-B}}(PGTH)$$

can be used as the *temporal distance metric* between two instructions `A` and `B`.

Instruction similarity Many instructions share some gates in ALU mostly due to their similarity. For example, an ALU could have different addition and subtraction instructions which are inherently similar. As a result, these instructions share a large portion of gates in the synthesized netlist. Therefore, implementing these instructions in separate ALUs would impose redundant structures leading to undesirable leakage and area overhead. Therefore, it is preferable to group such instructions into one ALU to reduce the associated overheads.

We introduce *dissimilarity metric* defined as the structural dissimilarity between instructions ($dist^{dissimilarity}_{\texttt{A-B}}$). For example, $dist^{dissimilarity}_{\texttt{ADDL-ADDQ}} = 0.0$ as both addition instructions implement similar functionality. However, $dist^{dissimilarity}_{\texttt{ADDL-ORNOT}} = 1.0$ as the corresponding instructions implement two completely different logic structures. The dissimilarity values are assigned based on the knowledge we have about the logic implementation of different instructions.

Some of the above parameters may lead to contradictory grouping of instructions into ALUs. For example, instructions `S8ADDL` and `ADDL` should be grouped into one ALU because of *similarity*; however, according to their *utilization frequencies* they should be placed into different ALUs to allow power-gating. In the next section, we define a formal clustering problem considering the aforementioned parameters and solve it to find the best instruction grouping strategy.

Instruction clustering problem definition

The problem of partitioning a large ALU into smaller ALUs can be defined as a clustering problem, in which the distance between the instructions is explained by temporal proximity, utilization frequency, and similarity of instructions. The goal of such clustering algorithm is to maximize the distance between ALUs while minimizing the distance between instructions of each ALU. This allows us to increase the overall power-gating likelihood of ALUs which leads to lower leakage and better energy efficiency.

For this purpose we apply Agglomerative Hierarchical Clustering (AHC) algorithm [248] to cluster the instructions into several groups, each group implemented in one ALU. AHC is suitable for our problem because we can provide pairwise distances between each and every two instructions. We create the *pairwise distance matrix* needed for the AHC algorithm based on the frequency distance metric ($dist^{freq}_{\texttt{A-B}}$), temporal distance metric ($dist^{temporal}_{\texttt{A-B}}$), and

structural similarity ($dist^{similarity}_{\texttt{A-B}}$) introduced in the previous section. Finally, the elements of the pairwise distance matrix ($pdist$) are obtained as (Cartesian distance on a 3D space):

$$pdist^2_{\texttt{A-B}} = (\beta.dist^{freq}_{\texttt{A-B}})^2 + (\gamma.dist^{temporal}_{\texttt{A-B}})^2 + (\lambda.dist^{dissimilarity}_{\texttt{A-B}})^2. \quad (4.1)$$

Here, β, γ, λ are coefficients to scale all the metrics into the same scale.

We consider *single-linkage* clustering method on the AHC. In a single-linkage method, the linkage function $D(X, Y)$, which is the distance between two clusters X and Y, is defined as the minimum distance between every two members of the clusters:

$$D(X, Y) = \min_{\texttt{A}\in X, \texttt{B}\in Y} pdist_{\texttt{A-B}}.$$

Therefore, maximizing the distance between clusters X and Y will allow the maximum power gating possibility of the corresponding ALU implementations and improves the energy efficiency.

Fine-grained power-gating prediction

Once the inactive phase for a component is detected at architecture-level, the component can be power-gated by asserting a sleep signal on the header/footer sleep transistors. Although the power-gating can effectively reduce the wasted leakage energy, it has to be done when the functional unit is not utilized for a minimum number of cycles ($PGTH$) to break-even the associated overheads. This value is estimated to be around 10 cycles for a typical technology [247].

The inactive phase of a functional unit can be predicted based on several techniques at runtime. In order to determine the inactive interval of each functional unit and decide whether to power gate it or not, it is possible to monitor the instruction buffer for a number of upcoming instructions while considering the branch prediction buffer. However, the power-gating signal for a given functional unit can be mispredicted due to branch misprediction. As a result of misprediction, the entire pipeline may be needed to be flushed to reload the correct instructions. This provides some time to properly power up the required functional units without imposing much overhead due to pipeline stall. It is worth mentioning that sub-threshold and near-threshold processors typically have a deeper pipeline to benefit in terms of performance and energy efficiency [249]. In such processors, there is enough time for functional unit power up after misprediction due to deeper pipeline design.

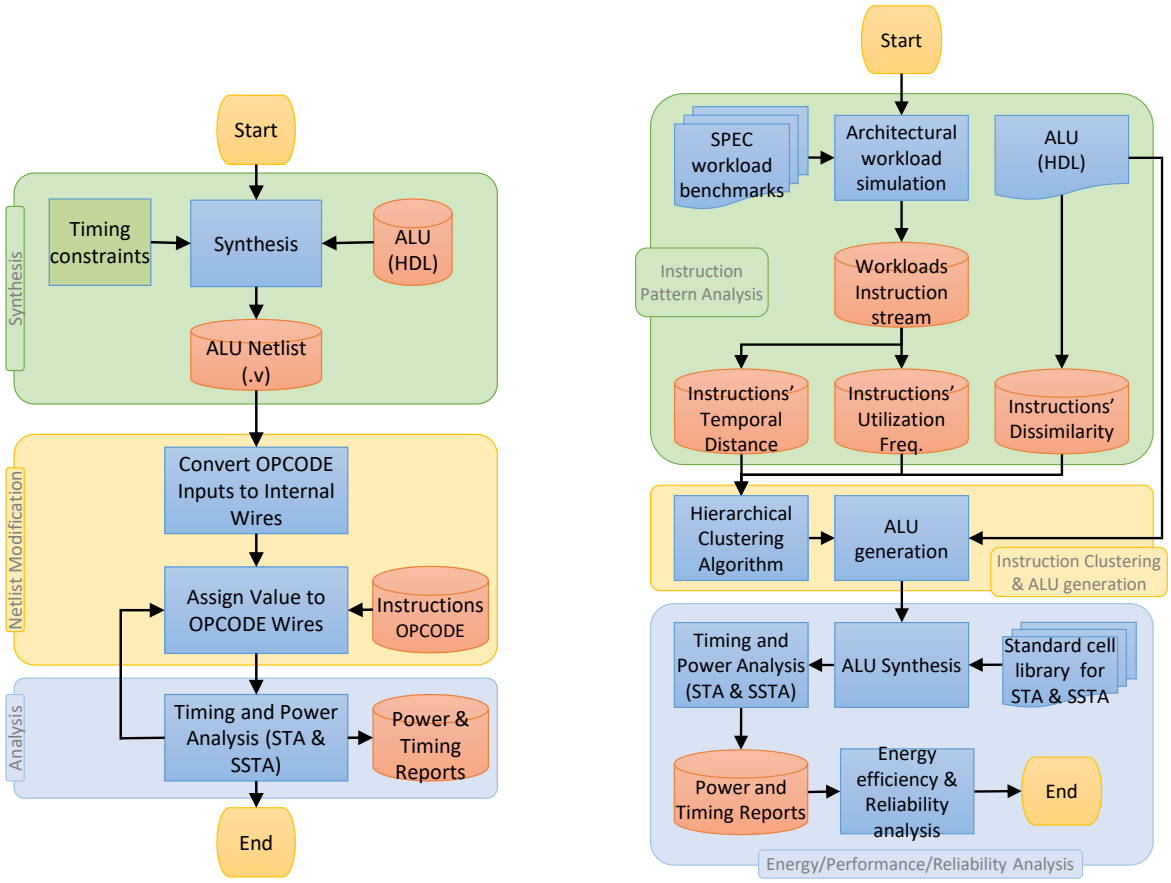
4.4 Results and discussion

This section presents the methodology and simulation setup for the proposed data path optimization approaches. We evaluate the effectiveness of the proposed approaches by applying them to an ALU.

4.4.1 Implementation flow

Input vector dependent timing and power analysis As explained in Section 4.3.1, the time and power required by a functional unit to complete an operation execution is different from one instruction to another. Here, we use the flow illustrated in Figure 4.6a to extract the detailed timing and power information for each instruction. Therefore, for an ALU as a case study:

1. *Synthesis*: The synthesis step of Figure 4.6a is executed with timing constraints to obtain the gate level netlist of the synthesized ALU.



(a) Implemented flow for extracting the timing and power information of each ALU instruction.

(b) Flow of the proposed functional unit paritioning for ALU optimization.

Figure 4.6: Implementation flows of a) Instruction multi-cycling, b) Functional unit partitioning applied to an ALU.

2. *Netlist modification*: The synthesized netlists of the ALUs are modified such that the OPCODE input signals which determine the instruction to be executed are changed to internal wires. For each instruction, the OPCODE signals are assigned to associated values inside the Verilog netlist. This will effectively deactivate the rest of the instruction paths in the ALU and force the STA tool to evaluate only the paths associated with the execution of the given instruction. This is because the rest of the paths (which belong to other instructions) are deactivated, and any change in the output pins of the ALU is only due to the paths belonging to the given instruction.

3. *Timing and power analysis*: The timing and power analyses are performed on the modified netlists, and the delay of the instructions considering the variation as well as the dynamic power and the leakage power for each instruction are extracted.

The change in the leakage power from one instruction to another is negligible because even the inactive gates which are not part of the propagation paths are leaking. However, if the execution time is different from one instruction to another, the amount of the leakage energy would be different proportionally. The dynamic energy for each instruction is also calculated based on the dynamic power value.

Functional unit partitioning flow Figure 4.6b shows the overall flow of the proposed functional unit partitioning method applied to an ALU. The flow consists of three distinct steps:

1. *Instruction Pattern Analysis:* In this step, the instruction stream extracted from running representative workloads are analyzed to extract instruction temporal distance and utilization frequency. Instruction dissimilarity is also defined based on the field knowledge about the implementation of logic units.

2. *Instruction Clustering and ALU generation:* With the help of the information collected from the previous step, instructions are grouped into n clusters, and each cluster is implemented as a new ALU. Moreover, all the partitioned ALUs are combined with a circuit for multiplexing their outputs to form a *union ALU*, as shown in Figure 4.2b.

3. *Energy, Performance and Reliability Analysis:* The union ALU generated from the previous step is evaluated in terms of timing and performance, and finally, its reliability is evaluated when it is used instead of the original ALU.

4.4.2 Reliability analysis

Various sources of delay variation, e.g. process variation, aging, temperature and voltage variations, can potentially lead to timing failures. Therefore, timing failure is presented as a stochastic metric which is dependent on the value of additional timing margin and hence the allocated clock period. Therefore, statistical information regarding the circuit delay can be used to evaluate the reliability. Accordingly, *Reliability* is the probability of not having a timing failure due to variation effects.

In a functional unit such as an ALU, the Failure Probability of a circuit caused by timing issues can be modeled as a function of the allowed time for instruction execution T, based on the delay distributions of the instructions:

$$Reliability = CDF_{ALU}(T) = \prod_{\texttt{INST}}^{\{\text{all instructions}\}} CDF_{delay,\texttt{INST}}(T), \tag{4.2}$$

where $CDF_{delay,\texttt{INST}}$ is the Cumulative Distribution Function (CDF) of delay of instruction `INST`. Accordingly, the *Failure Probability* is obtained as:

$$Failure\ Probability = 1 - Reliability. \tag{4.3}$$

There is a trade-off between reliability and performance as explained in the above equation. A larger clock period (T) results in higher reliability and lower failure probability at the cost of speed.

In a multi-cycling scenario, the allowed time for instruction execution T is dependent on the number of cycles allocated by each instruction. For a single-cycle instruction T is equal to the clock period T_{clk}; however, a two-cycle instruction is allowed to be executed for $2T_{clk}$. Therefore, Equation (4.2) is modified as follows:

$$CDF_{ALU}(T_{clk}) = \prod_{\texttt{INST}}^{\{\text{all instructions}\}} CDF_{delay,\texttt{INST}}(n_{\texttt{INST}} \times T_{clk}). \tag{4.4}$$

In the above equation, $n_{\texttt{INST}}$ is the number of cycles allocated for instruction `INST` obtained based on the distribution of the instruction delays:

$$n_{\texttt{INST}} = \lceil \frac{d_{\texttt{INST}}}{T_{clk}} \rceil. \tag{4.5}$$

$d_{\texttt{INST}}$ is the instruction delay considering the variation, i.e a point in the tail of the delay distribution referring to very low failure probability. Please note that slow instructions have large logic depth, i.e. there are many gates in the critical paths of these instructions. According to the *Central Limit Theorem* [250], the delay distribution of such instructions is approximately normal (*Gaussian*). In such case, we can use parameters such as the mean (μ) and standard deviation (σ) of instruction delay to approximate its CDF function. Therefore, we may choose μ+3σ of the instruction delay as $d_{\texttt{INST}}$ which corresponds to less than 0.135% failure probability.

We perform SSTA to evaluate the impact of process variation [83] on the timing of the circuit. The SSTA tool reads variation information from the variation library (see Section 3.4.1) containing variation information at cell-level. The SSTA extracts accurate delay distribution for each instruction of the ALU represented by their *CDF* (i.e. $CDF_{delay,\texttt{INST}}$). Based on these distribution functions and Equation (4.4), we extract the reliability of the ALU.

4.4.3 Simulation setup

The proposed approaches are applied to a 64-bit ALU. For this purpose, we synthesize a 64-bit ALU with loose and tight timing constraints and characterize it for the NTC by a Statistical Static Timing Analysis (SSTA) to extract the power and delay of each instruction as well as the reliability as explained in Section 4.4.1. The same methodology is applied to synthesize and analyze the partitioned ALUs generated by the functional unit partitioning method.

Additionally, we extract the instruction stream for SPEC2000 benchmark workloads using gem5 architectural simulator [251]. The result of the architectural simulation is used to evaluate the impact of the proposed *instruction multi-cycling* approach and to perform instruction pattern analysis and clustering in the proposed *functional unit partitioning* approach. The results of both approaches are compared to the *baseline*, which is a conventional functional unit design, i.e. conventional synthesis with tight constraints.

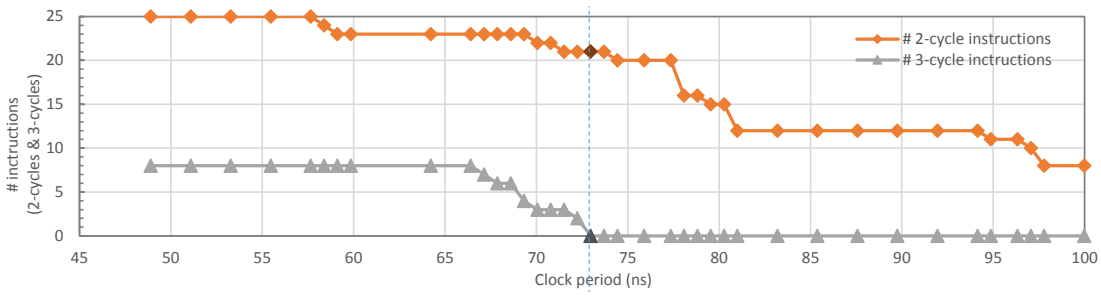
4.4.4 ALU multi-cycling results

Multi-cycling improvement

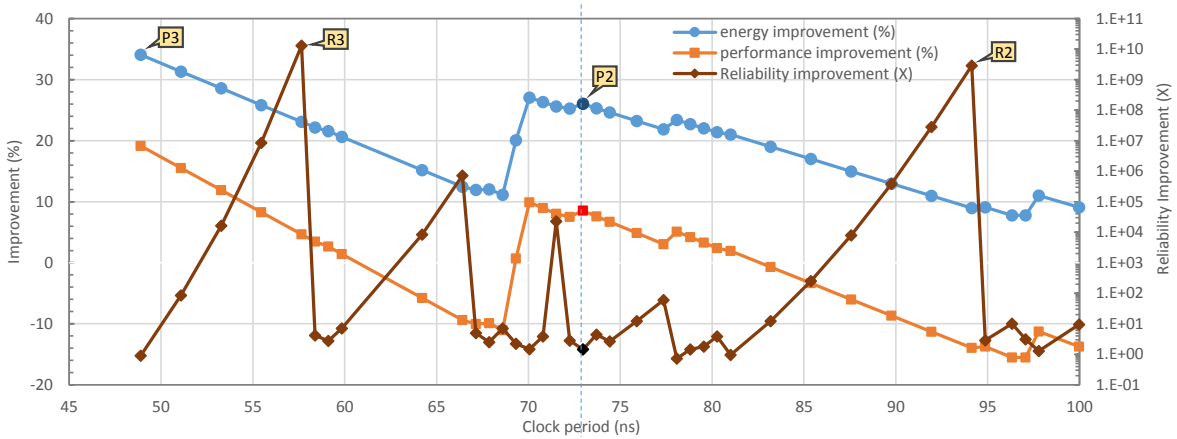
By reducing the clock period below the delay of the slowest instruction, some instructions need to be executed in multiple clock cycles. The number of the required clock cycles can be calculated from Equation (4.5). The delay of the slowest instruction is 146ns, hence, the minimum clock period for running all instructions in one cycle is also 146ns.

We changed the clock period of the Loose ALU and extracted the instructions which should be executed in multiple cycles for each clock period. Here, the clock period is swept from 49ns (one-third of the delay of the most critical instruction) to 100ns to explain the impact of clock period on the circuit characteristics. As shown in Figure 4.7a, some instructions need to be executed in three cycles when the clock period is below 73ns (which is half the delay of the most critical instruction - marked by a vertical dashed line in the Figure).

The energy and performance improvement of the ALU executing workload "equake" are plotted in Figure 4.7b for various clock periods. The improvement numbers are calculated in comparison with the Tight ALU which executes all instructions in one clock cycle. When the clock period is reduced, the energy and performance improvements increase because the delay slacks of the instructions are trimmed. However, when any change in the sets of 1-cycle, 2-cycle or 3-cycle instructions happens due to the clock period reduction, i.e. some 1-cycle instructions become 2-cycle instructions or some 2-cycle instructions need to be executed in three cycles, the improvements suddenly drop. The maximum energy improvement (34%) and performance improvement (19%) are achieved when the clock period is 49ns.



(a) Number of 2-cycle and 3-cycle instructions for different clock period values



(b) Energy, performance and reliability improvements for different clock period values

Figure 4.7: Modifying the clock period of the ALU changes the set of 2-cycle and 3-cycle instructions which impacts the energy, performance and reliability improvements significantly. The results are extracted for Loose ALU + INC/DEC running workload "equake"[2] at $V_{dd} = 0.5V$. The improvement values are relative to the Tight ALU. Four pareto points are marked on the graph: P2 (R2) provides the best energy and performance (best reliability) when ALU is limited to execute all instructions in two clock cycles, and P3 (R3) provides the best energy and performance (best reliability) when some instructions can be executed in three clock cycles.

Impact of workload on the improvement ratio

Executing a slow instruction has no performance or energy benefits because it utilizes the ALU for several clock cycles. However, executing a fast instruction which can be executed in fewer clock cycle(s) will save some energy and improve performance. Since each workload execution specific set of instructions, the amount of improvement is highly dependent on the profile of the instructions utilized by a workload. Workloads which frequently execute fast instructions will have better improvements using the proposed approach. We measured the amount of energy improvement for different workloads as shown in Figure 4.8. The clock period is once set to 73ns which means all instructions are executed in at most two clock cycles, and then set to 49ns to evaluate the results for when the instructions can occupy three clock cycles. The hatched bars in this figure show the amount of energy improvement in different workloads compared to the baseline. The improvement is smaller for workloads which frequently execute slow instructions (applu, mgrid, swim), and larger for workloads with more executions of fast instructions (equake, mesa, mcf, twolf). For example, 76% of the executed instructions in "applu" are among the 8 slowest instructions (`addq, lda, ldah, s4addq, s4subq, s8addq,`

[2]For other workloads the improvement plots show similar trend.

Figure 4.8: Energy improvement over the baseline (Tight ALU). The additional improvement is also calculated for when `addition` / `subtract` are replaced by `increment` / `decrement` when possible (clock period is 73ns and 49ns).

`s8subq, subq` as depicted in Figure 4.3b). However, this is only 45% for workload "equake". The average energy improvement and the performance improvement over all workloads are 20.8% and 1.7%, respectively.

High-level optimization improvements

As explained, when the fast instructions are more utilized by the workload the improvements are even higher. This concept can be incorporated at higher level e.g. application and compiler optimization, to improve energy efficiency and performance further. This can be done in various ways as discussed in Section 4.3.1.

Our analysis of the ALU instruction stream shows that in some workloads a number of `add/subtract` instructions can be replaced by `increment/decrement`. In workloads such as "bzip2" and "gzip" which utilizes `add` and `subtract` a lot, it is possible to change up to 11% of all instructions to `increment/decrement`. Since `increment/decrement` instructions are faster compared to `add/subtract` instructions, the improvement in energy and performance is considerable for these types of workloads. The energy improvements and the boost from applying the instruction replacement are depicted in Figure 4.8 for two clock periods: 73ns (all instructions are executed in at most two clock cycles) and 49ns (three clock cycles). For 49ns, the energy and performance improvements are 3% and 4.3% higher for "bzip2" and "gzip" when instruction replacement technique is applied. The technique is still applicable to other workloads; however, the improvements are less (approximately 1%). Applying the instruction replacement technique increases the average energy and performance improvements to 29.3% and 12.8% when the clock period is 49ns (21.4% and 2.4% on average when the clock period is 73ns).

In order to show the benefits of data type conversion, we executed a simple "matrix manipulation" application which calculates $M1 + M2 * 2$ for given matrices $M1$ and $M2$. The corresponding results are presented in Table 4.1 for two clock periods: 73ns and 49ns. With a clock period of 49ns and 2-byte data type used for "matrix manipulation", the energy and performance improvements of the multi-cycled Loose ALU over the baseline are 26.5% and 8.6%, respectively. However, changing the data type to a 1-byte data type increases the energy and

Table 4.1: Energy and performance improvements of executing the "matrix manipulation" workload with different data types

		baseline		proposed multi-cycling approach			
clock period	data type	energy (nJ)	time (μs)	energy (nJ)	time (μs)	energy improvement	performance improvement
73ns	`short` (2-bytes)	**27.9**	502	22.2	502	20.3%	0.1%
	`char` (1-byte)	22.5	404	**17.6**	396	21.5%	2%
	Overall improvement (baseline - `short` → multi-cycling - `char`)					36.9%	21.1%
49ns	`short` (2-bytes)	**27.9**	502	20.5	459	26.5%	8.6%
	`char` (1-byte)	22.5	404	**15.9**	356	29.0%	12%
	Overall improvement (baseline - `short` → multi-cycling - `char`)					42.9%	29.2%

performance improvements to 29.0% and 12% over the baseline.

Furthermore, the 1-byte data type is inherently more energy-efficient compared to the 2-byte data type (22.5 nJ vs 27.9 nJ). Therefore, the cumulative energy improvement of changing the data type from 2-byte data type to 1-byte data type is 42.9% (going from 27.9nJ to 15.9nJ). Additionally, the performance also improves by 29.2% (going from 502μs to 356μs).

Energy/performance/reliability trade-off

In the near-threshold voltage region, each instruction has a much wider delay distribution compared to the super-threshold region with a longer tail. Therefore, changing the clock period affects the tail of the distribution contributing to failures. We calculate the failure probability for each clock period according to Equation (4.3) and obtain the *reliability improvement* as the ratio of the failure probability between the baseline and the optimized ALU:

$$\textit{Reliability improvement}\ (T_{clk}) = \frac{\textit{Failure Probability of baseline ALU}\ (T_{clk})}{\textit{Failure Probability of optimized ALU}\ (T_{clk})}. \tag{4.6}$$

The reliability improvement is depicted in Figure 4.7b versus the clock period. There are points where the reliability improvement worsens which are mostly points with very high energy and performance improvement. This is due to the fact that for these points of clock periods the timing margins of most of the instructions are very small, leading to a significant probability of failure even larger than the baseline. However, there are several points with orders of magnitude better reliability. For example, the reliability improvement ratio is $\approx 1.3 \times 10^{10}$ when the clock period is 57.7ns or 22891 when the clock period is 71.5ns. The performance and energy improvement is also significant in these points. The reason for such large reliability improvements is that the baseline has many critical or near-critical instructions with zero or minimal slack. However, the multi-cycling strategy is able to provide enough timing margin for many instructions such that the provided time for the execution of each instruction marks very high sigma values on the tails of all the delay distributions. Therefore, the designer is

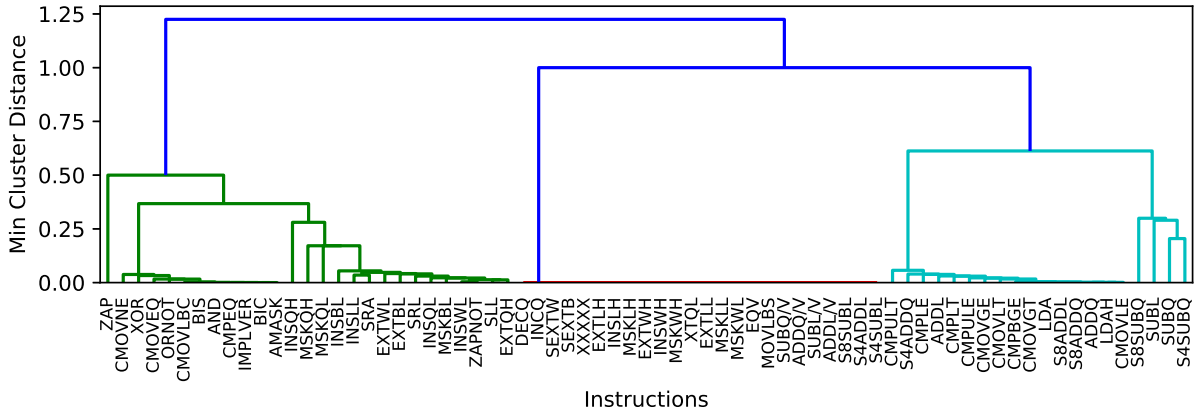


Figure 4.9: Dendrogram illustration of the proposed AHC method for some of the instructions. The instructions are merged bottom-up to form larger clusters.

able to find a good trade-off among energy efficiency, performance and reliability according to the design requirements.

Discussion

It is evident from the results analyzed before that the multi-cycling approach can provide considerable energy and performance benefits, as well as reliability improvements. In that regard it is important to note that these results represent the ideal case, where the clock period can be adjusted without any constraints. However, in a real processor (or an ASIC), there may be other timing critical components, limiting the freedom for modifying the clock period. Hence, another operation point in Figure 4.7b might be chosen. Moreover, in such a scenario, to exploit a short clock period for the ALU, the entire processor has to operate at a higher frequency. In fact, this may require additional design tweaks for other parts of the processor. For this purpose, timing critical components of the processor can be split up into multiple "short-cycle" components. Consequently, this results in a deeper pipeline, requiring more control logic and more registers. However, this overhead should be compensated by the overall leakage savings of the entire processor due to the runtime benefits of our novel multi-cycling approach. In case a multi-cycle instruction has to be executed by the ALU, the pipeline frontend of the microprocessor is stopped (through using a no-operation instruction, or clock gating), such that the ALU can work for multiple cycles without change of the input vector.

An alternative solution is to execute multiple short instructions in the ALU per clock cycle. By that means, no major modifications of other processor components are necessary. For instance, to execute two instructions in a clock cycle, the high and low levels of the clock signal can be exploited as it was done in the Intel Pentium 4 [252]. On the other hand, to execute more than two instructions in one cycle, additional shifted clock signals are required. Hence, this scheme is only feasible for processors featuring reservation stations to execute multiple instructions per clock cycle.

4.4.5 ALU partitioning results

Clustering results

The distance metrics are extracted based on the analysis of the SPEC benchmark workloads. The temporal distance metric is extracted for $PGTH = 100$. The dendrogram in Figure 4.9 illustrates the results of the AHC method. As shown in this figure, most of the rarely used

Table 4.2: Energy improvement results for 3-ALU and 4-ALU over Original ALU for 14nm PTM [32]

	Area overhead (%)	Performance improvement (%)	$V_{dd}(V)$	Energy Saving for Power-Gating Threshold ($PGTH$) in %				
				10 cycles	20 cycles	50 cycles	100 cycles	500 cycles
3-ALU	17	11	0.80 (super-V_{th})	22.8	19.7	19.5	19.4	17.7
		7.0	0.35 (near-V_{th}[3])	24.6	21.4	21.2	21.1	19.2
		5.5	0.25 (sub-V_{th})	27.7	24.4	24.1	24.0	21.8
4-ALU	19	11	0.80 (super-V_{th})	43.1	38.6	32.1	27.6	15.1
		7.0	0.35 (near-V_{th})	**43.4**	38.9	33.0	28.9	17.3
		5.5	0.25 (sub-V_{th})	44.1	39.6	34.5	31.1	20.6

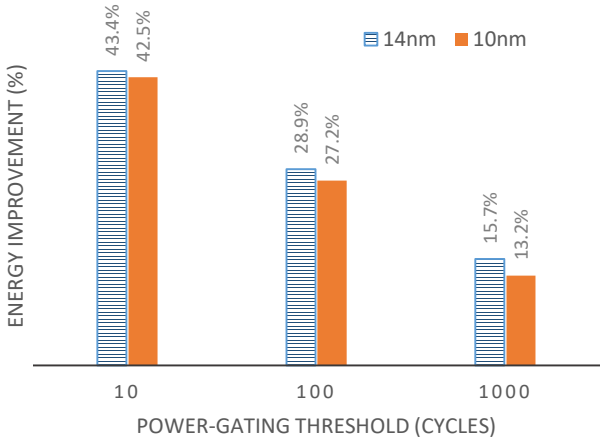


Figure 4.10: Energy improvement of functional unit partitioning, on an ALU partitioned in to 4 smaller units (4-ALU), for 10nm and 14nm technology nodes.

instruction in Figure 4.4 are either grouped together or grouped with other closely similar structures.

The original ALU can be partitioned into n smaller ALUs according to the dendrogram. One can determine the best n value, based on *inconsistency coefficients* [253]. However, as the overhead of hardware implementation increases with the number of clusters, we only have to limit n to have at most 4 clusters (ALUs). Here, we present results for partitioning the ALU into three and four smaller ALUs as the energy improvement for two ALUs vanishes away as the power-gating threshold ($PGTH$) increases.

Circuit-level results

Based on the clustering method, the original ALU is partitioned into three ALUs (3-ALU) and four ALUs (4-ALU). Table 4.2 reports the area overhead and performance improvement of the 3-ALU and 4-ALU compared to the original ALU at 14nm. In the performance improvement calculation, we also considered the delay of the extra multiplexer/demultiplexer circuit for both 4-ALU and 3-ALU designs. Both 3-ALU and 4-ALU designs occupy more area compared to the original ALU as expected. However, they perform faster because the logic implemented in each of the partitioned ALUs is simpler and more coherent. By reducing the supply voltage, the performance gain diminishes slowly, because the impact of process variation on the shorter

[3]In this technology V_{th} is close to 0.35V.

critical paths of smaller ALU is more than the long critical path in the original ALU.

Additionally, the overall energy improvement achieved by the proposed ALU partitioning method is extracted and reported in Table 4.2. This is done by calculating the percentage of the time that each smaller ALU can be power-gated based on a given power-gating threshold ($PGTH$) and for each workload. As presented in Table 4.2, there is a significant energy improvement even at super-threshold region ($V_{dd} = 0.80V$). The reason is that one of the partitioned ALUs is mostly in sleep mode because its instructions are rarely used (see Figures 4.9 and 4.4). The percentage of the time an ALU is power-gated for a specific $PGTH$ is the same for all supply voltages; however, the energy improvement by the proposed ALU partitioning technique is slightly more at lower supply voltages. The reason is that the leakage power contribution to the overall power consumption grows significantly by reducing the supply voltage. Therefore, reducing the same amount of leakage results in a larger percentage of energy saving at lower supply voltages.

Figure 4.10 compares the energy improvement results for 4-ALU at near-threshold region ($0.35V$) in two technology nodes: 10nm and 14nm. As shown, the energy improvement results for these technology nodes resemble closely. A similar trend is observed for the rest of the results.

Performance and reliability trade-off

The performance improvement of the functional unit partitioning, shown in Table 4.2, could be traded for reliability at low supply voltages, similar to what explained for instruction multi-cycling in Section 4.4.4. According to the results, it is possible to enjoy maximum performance improvement (for example 11% at $0.80V$) or invest the achieved speed on reliability to get up to 10^9 times lower failure probability. At the near-threshold supply voltage ($0.35V$) the maximum achievable reliability improvement is 11.5× (by trading 7% performance improvement from Table 4.2) because the delay distribution of the union ALU becomes wider than the original ALU due to its shorter critical path.

4.5 Summary

The assumptions and optimization targets for NTC circuit design are different from the conventional super-threshold design, due to large impact of variabilities as well as comparable contributions of leakage power and dynamic power. This chapter presented two cross-layer design optimization approaches for NTC circuits to co-optimize energy-efficiency, reliability, and performance.

In the *instruction multi-cycling* approach, the idle time of the functional units is reduced by executing slow instructions in multiple clock cycles and fast instructions in one clock cycle. This approach consists of circuit redesign for smoother slack distribution across instructions (circuit level), multi-cycle execution of slow instructions (architecture level) and code replacement (compiler level). Our experimental results show that this approach achieves significant energy (34%) and performance (19%) improvement while providing orders of magnitude reduction of timing failure rate.

The proposed *functional unit partitioning* approach improves the energy efficiency and reliability of functional units by exploiting fine-grained power-gating. For this purpose, a large functional unit like an ALU is partitioned into several smaller (and faster) units based on the instruction usage pattern of the running applications and the instructions inherent similarity. As a result, the smaller functional units can be power-gated whenever they are not used for a long time. Our simulation results show that the energy efficiency of an ALU can be improved

by up to 43.4% in the NTV region. Additionally, the performance can be improved by at least 7.0%, or the reliability can be improved by 11.5 times in the NTV region.

Therefore, by revisiting the design of functional units as important components of data paths and utilizing cross-layer approaches, it is possible to optimize the energy-efficiency, performance, and reliability of NTC designs.

5 Post-fabrication calibration and runtime tuning for energy efficiency

The best energy-efficiency is achieved when a circuit operates in the NTV region; however, the best operating point for maximum energy-efficiency could vary depending on post-fabrication parameters (i.e., process corner) and runtime parameters (e.g., operating temperature and workload). In addition, the power-gating state (power modes) of the components at runtime affect the energy-efficiency of the Systems-on-Chip (SoC). Hence, the best operating point must be determined after the fabrication and at runtime for each individual SoC.

This chapter proposes post-fabrication calibration and runtime tuning approaches based on machine-learning to tune NTC circuits for MEP on a per-chip basis by considering process and runtime variations [7, 8]. The presented methods do not require costly power measurement circuitry on-chip or extra hardware, which makes them suitable for IoT edge chips. The simulation results show that the proposed method has high MEP prediction accuracy and achieves near-optimal energy-efficiency with low overhead while considering the runtime performance and reliability constraints.

This chapter is organized as follows. We continue to Section 5.1 by presenting an introduction, and motivating the necessity of post-fabrication calibration and runtime tuning in NTC. Afterward, Section 5.2 and Section 5.3 present the proposed methods for individual NTC circuits as well as SoCs, respectively. The effectiveness of the proposed methods is studied in Section 5.4, by discussing the simulation results. Section 5.5 concludes this chapter, subsequently.

5.1 Introduction, Motivation, and Contributions

The supply voltage leading to the minimum energy consumption (best energy-efficiency), which is known as MEP, is largely dependent on process and runtime variations as well as the functionality of the circuit, as explained in Section 2.3.1. As an additional constraint, supply voltage scaling can only be done to the level that the performance requirements allow because operating the circuit at MEP drops the performance drastically [254]. Therefore, determining the MEP of a circuit and the correct voltage scaling range is a challenging runtime decision as it requires considering impacts of process and runtime variations.

Several methods have been proposed for obtaining the MEP point during design-time [235] or runtime [78, 195, 197]. These methods are either inaccurate, because they cannot consider all the important parameters affecting the MEP, or they are not very efficient, due to additional hardware overhead, especially for NTC circuits with stringent energy-efficiency constraints.

Additionally, the impact of power-gating on the MEP, when the circuit is composed of several components like SoCs, is mostly overlooked. In the IoT domain, SoCs are gaining popularity as they integrate commonly used components such as general purpose computing, accelerators (e.g., multi-media or security accelerators), digital signal processors, communication and transmission units, and analog/digital converters on a single chip with power management features. Such integration brings additional complexity to the MEP tuning on SoCs because of the additional parameters associated with the power management of the components such as power-gating, clock-gating, and voltage-frequency islands [255]. Each SoC component has

a unique power consumption profile while some components could share the same supply voltage. Depending on the requirements of the running application, some SoC components are active whereas the rest are inactive or powered down. Additionally, supply voltage scaling range of a SoC is constrained by the performance, functionality, and reliability requirements of the running application [254]. Therefore, the system-wide MEP depends on the MEP of the individual components and their states (power-gated, idle, fully utilized, etc.) and could vary significantly at runtime.

This chapter presents two methods for optimizing the energy-efficiency, by accounting for process and runtime variations with low cost:

- *Post-fabrication calibration and runtime MEP adaptation method* for individual circuits [7]: This method can determine the MEP of a circuit with specific functionality, which has a single supply voltage and clock frequency.

- *Runtime adjustment* of SoCs [8]: This method is applicable when various circuits are operating together with different power modes on different voltage islands.

These methods complement each other at different levels, one for individual NTC circuits, and the other one for SoCs containing several NTC circuits on different voltage islands.

In the *post-fabrication calibration and runtime MEP adaptation* method, we propose to predict the MEP based on a limited number of measurements performed during post-fabrication test and calibration (e.g., speed and power consumption) using a trained machine-learning model, and at runtime based on the operating temperature. Simulation results and experimental measurements [74] show that temperature fluctuation is the most influential runtime variability which significantly affects the MEP of NTC circuits. Therefore, this method addresses the MEP variation caused by temperature with minimum overhead to the circuit.

At the circuit level, the idea is to account for both process and temperature variation by using a calibrated LUT which maps each runtime condition (i.e., temperature) to a specific MEP (i.e., operating supply voltage and clock frequency). This LUT is generated using a trained regression model during post-fabrication tests and stored on the chip. The calibrated LUT is used to tune the supply voltage and clock frequency of the circuit to its MEP at runtime, accounting for both process and temperature variations.

Due to the dependency of the system-wide MEP on numerous parameters, designing a single model for all SoCs is not effective. Therefore, we propose a method for *runtime adjustment of IoT SoCs*: to determine the MEP at the firmware-level using a trained machine-learning model. The main advantage of the proposed technique is that it can be applied to any existing SoC with voltage scaling capability without hardware overhead. Moreover, it is lightweight and at the same time accurate enough to achieve a high energy-efficiency.

For the runtime adjustment of SoCs, we first need to evaluate the impact of various runtime parameters such as temperature variation and power-gating of the components on the system-wide MEP of SoCs. Then, a lightweight regression model is trained for predicting the MEP of each SoC at the firmware-level during runtime. The prediction is done based on various parameters including the power management states, sensors, and performance counters which are already available on most SoCs. Finally, the SoC is tuned to the predicted MEP while complying to the performance and reliability constraints.

The proposed methods can be implemented either in the hardware or software (as a part of the firmware) and directly predict the MEP based on the temperature and the LUT. Therefore, they have the flexibility of software (firmware) level, to be able to meet application performance needs and components reliability constraints, and at the same time, are as accurate as hardware-level monitoring, due to their detailed learning-based models. Therefore, the MEP is

determined without implementing any closed-loop energy monitoring and supply voltage tuning hardware as done in the state-of-the-art, which eliminates the costly additional circuitry. The simulation results show that the proposed methods can tune circuits and SoCs to closely track their MEP, resulting in a near-optimal energy-efficiency under the impact of runtime variations.

5.1.1 Related work

The properties of the MEP have been studied by analytical models in the literature [59, 192, 235, 256, 257]. Such models cannot find the MEP accurately because they do not consider all the aforementioned parameters affecting the MEP such as process and runtime variations.

In order to consider the impact of runtime variability sources, it is proposed in [78, 195–198, 258, 259] to measure the circuit power online and take actions to maximize the energy efficiency by adapting V_{dd} in closed-loop feedback. Despite demonstrating a high accuracy in supply voltage tuning, all the closed-loop V_{dd} adaptation techniques are associated with costly additional circuitry for measuring the circuit power and applying the adaptation strategies. In addition, the resolution of online power measurement may not be high enough for fine-grained MEP tracking. Furthermore, the impact of local variation cannot be correctly captured in the methods that perform measurements on a limited circuit replica [195, 258]. Therefore, it is crucial to have a low overhead solution for finding the MEP accurately on a per-chip basis, considering the impacts of variabilities.

5.2 Post-fabrication calibration and runtime MEP adaptation of NTC circuits

This section presents a method to calibrate each individual circuit to its MEP based on post-fabrication measurements, and tune it at runtime based on temperature variation.

5.2.1 MEP analysis

We carried out experimental simulations to evaluate the impact of variability sources on the MEP based on commercial 40nm standard cell library. Process variation is applied to MOSFET transistors based on the Pelgrom's law in Equation (2.9), and the process-dependent coefficient is extracted from [83].

Standard cell library preparation

From the standard cell library, we used the cells which are suitable for ultra-low voltage operation, i.e., cells with less than 3 cascading transistors as explained in [28], and characterized them for NTC operation, using Cadence Liberate [260] and according to characterization setup in Table 5.1 [1]. Therefore, the selected cells are characterized at supply voltages from 350mV to 1.0V, temperatures between -25°C and 100°C, and at different process variation corners: $\{typical, fast, slow\}$. A *typical* corner represents the average delay (μ_{delay}) of a cell when process variation is applied, whereas *slow* and *fast* corners correspond to $\pm 3\sigma_{delay}$ tails of the delay distribution.

Since it is not feasible to characterize the entire library for all other temperature and voltage values, we used Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) to extract all the intermediate values similar to [261]. The PCHIP interpolation is applied on the logarithm

[1] In total, $3 \times 14 \times 9 \times 6 = 2268$ characterization rounds executed.

Table 5.1: Library characterization setup for MEP analysis

Corners	TT, FF, SS (all transistor at "typical", "fast", and "slow" corners)	
Cells	inverter (4 cells), nand (2 cells), nor (2 cells), xor (2 cells), flip-flop (2 cells), and latch (2 cells): total 14 cells	
Characterization conditions	Voltages (V)	0.35, 0.40, 0.45, 0.50, 0.55, 0.60, 0.65, 0.8, 1.0
	Temperature ($^{\circ}C$)	-25, 0, 25, 50, 75, 100

Table 5.2: Circuit characterization setup for MEP analysis

Circuits	c432, c499, c1908, c2670, b04, b07, b11, b13	
Corners	$N = 1000$ Monte-Carlo Samples (representative sampling)	
Characterization conditions	Voltages (V)	0.35, 0.40, 0.45, 0.50, 0.55, 0.60, 0.65, 0.8, 1.0
	Temperature ($^{\circ}C$)	-25, 0, 25, 50, 75, 100
	Input Switching Activity (α)	0 (idle), 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1
Characterization results	Dynamic Power	extracted using Synopsys Design Compiler
	Leakage Power	
	Clock period (fastest possible)	

of the measured values, which significantly improves the interpolation quality for the NTC libraries due to the wide scale of variations in the measured values.

Monte-Carlo analysis of benchmark circuits

The benchmark circuits for our simulation analysis are chosen from ISCAS'85 [26] and ITC'99 [231] and are synthesized using Synopsys Design Compiler based on the characterized cell library. Then, we perform a statistical analysis to collect information regarding the impact of variability sources on the MEP.

To do that, we follow a Monte-Carlo approach in which the samples (circuit instances) required for the analysis are generated based on a *Representative Sampling* strategy. In this approach, each cell of a circuit netlist is drawn from a few representative corners, namely "typical", "fast", and "slow". We follow these steps to generate a circuit instance using representative sampling:

1. In the characterized standard cell libraries for different corners, supply voltages, and temperatures, we add postfixes to the name of the cells according to the characterization corner. For example, we will have `NAND2_T`, `NAND2_F`, and `NAND2_S` instead of a single `NAND2`.

2. Each cell instance inside the netlist of the circuit is replaced with the same cell type but from a random corner, based on the sampling strategy. For example, the following line from the netlist file:

```
NAND2 U1 (.A1(N3), .A2(N6), .Y(n1));
```

can be replaced with one of these lines:

```
NAND2_T U1 (.A1(N3), .A2(N6), .Y(n1));
NAND2_F U1 (.A1(N3), .A2(N6), .Y(n1));
NAND2_S U1 (.A1(N3), .A2(N6), .Y(n1));
```

Then, the generated circuit instance is characterized[2] at different supply voltages, temperatures, and input switching activity as explained in Table 5.2, and its power and timing characteristics are extracted using Synopsys Design Compiler.

There are two reasons behind using such sampling strategy instead of a truly random sampling, in which each cell is chosen from a truly random population. First, choosing the cells only from the representative corners reduces the number of standard cell characterizations significantly. Since the process of standard cell characterization is time-consuming, we have to limit the cell samples population to a few representative corners to be timely feasible. Second, it reduces the number of circuit instances required for reaching important samples which reflect the tails of the distribution [262]. Since in this sampling we choose the cells from the most important categories corresponding to average and tails of the distribution, it is more likely to acquire important circuit instances instead of many circuit instances close the population average. As a result, comparatively fewer circuit instances are required with such sampling strategy compared to the normal sampling.

For the Monte-Carlo simulation, N circuit instances (netlists) are generated based on the representative sampling strategy and are characterized under the conditions mentioned in Table 5.1 and stored in a dataset. Since MEP is a function of process and runtime variation, PCHIP interpolation is used to obtain the MEP of each circuit (to consider process variation) under various runtime conditions (temperature and workload variations) by sweeping V_{dd} in the range of $0.35 \leq V_{dd} \leq 1.0$. Then a dataset is created which maps circuits and operating conditions to MEP:

$$circuit,\ instance,\ T,\ \alpha \longrightarrow V_{dd}^{MEP}, T_{clk}^{MEP}$$

This dataset and the interpolating PCHIP functions for each circuit are eventually used in Section 5.2.2 to create the training and evaluation dataset for the proposed method.

Please note that we are giving equal weights to "typical", "fast", and "slow" corners in the representative sampling strategy. However, in a truly random sampling, we would observe abundant number samples close to "typical" and infrequent samples close to the "slow" and "fast" tails. Therefore, the distribution of the circuits generated using the representative sampling is much wider compared to random sampling. Although the distribution is perturbed by such representative sampling, it is beneficial for our analysis, since we are also interested in exploring the rare corners thoroughly.

MEP variations

The results of this MEP analysis show that the MEP fluctuates depending on the process and runtime variations. As presented in Section 2.3.1, process, temperature, and workload variations induce MEP fluctuation.

[2]Since 1000 samples for each of these 8 circuits are characterized at 9 voltages, 6 temperatures, and 8 activity values, we executed in total $1000 \times 8 \times 9 \times 6 = 432000$ timing characterization rounds and $1000 \times 8 \times 9 \times 6 \times 8 = 3456000$ power characterization rounds with Design Compiler

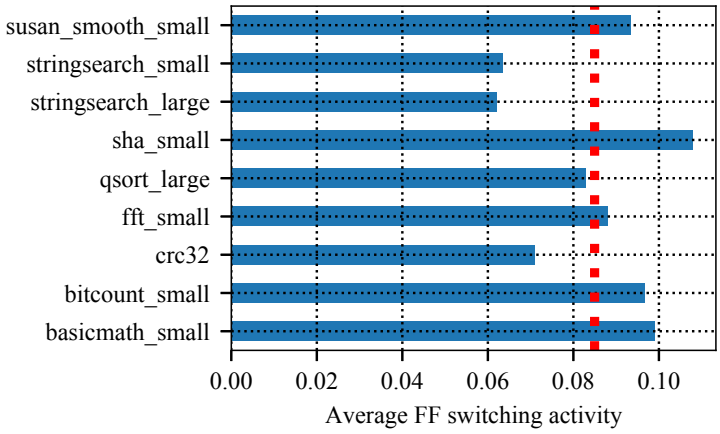


Figure 5.1: Average flip-flop switching activity of Leon3 under different MiBench workloads changes in the range of $0.062 \leq \alpha \leq 0.108$, with $\mu_\alpha = 0.085$ (in red). This range is used as the range of input switching activity variation ($\pm 27\%$).

Process variation The amount of process variation is dependent on technology, circuit structure, and cell library. Circuits with large logic depth experience less process variation, leading to less MEP variation from chip to chip. Using gates with smaller size also contributes to higher process variation. In Chapter 3, we proposed a methodology to reduce the impact of process variation using better EDA flow.

Temperature variation The temperature of NTC circuits is solely determined by ambient temperature[3]. The range of temperature variation could vary significantly, depending on the application and in-field conditions. For example, an implanted device has a very narrow temperature fluctuation while the temperature fluctuation in automotive and environmental sensing applications could vary in a wide range. Consequently, the impact of temperature on the MEP could be small or large depending on the application. This encourages on-chip or on-board temperature sensing and adapting the circuit to the temperature for circuits experiencing large temperature variation. Please note that only one sensor is sufficient as there is no temperature gradient due to power dissipation on the chip. Here, the temperature is considered to change in the range of $-25^{\circ}C \leq T \leq 100^{\circ}C$, as it is completely dependent on the ambient condition, i.e., there is no cooling/heating system to control the temperature.

Workload variation As explained in Section 2.3.1, the impact of workload is dependent on the circuit structure as well as input switching activity α. To grasp a correct understanding of the range of α variation in a real scenario, we executed several MiBench workloads [95] on a Leon3 processor [233], which matches the characteristic of an IoT processor for low power applications. The average input switching activity of the processor can be represented by the switching activity of the flip-flops, which is shown in Figure 5.1 for different workloads. According to this observation, the average input switching activity for different workloads can vary by at most $\pm 27\%$ over different workloads. This means that in most cases, the change in the MEP due to the dynamic to leakage ratio is not significant compared to the process and temperature variations.

Figure 5.2 compares the relative impacts of process, temperature, and workload variations on the V_{dd}^{MEP} of various benchmark circuits. The results are extracted based on the aforemen-

[3]The ambient temperature does not necessarily relate to the air temperature. Especially in many-core systems, if an NTC core is fabricated on the same die next to a high-performance core with high power consumption, the temperature of the NTC core could be correlated to the activity of that high-performance core.

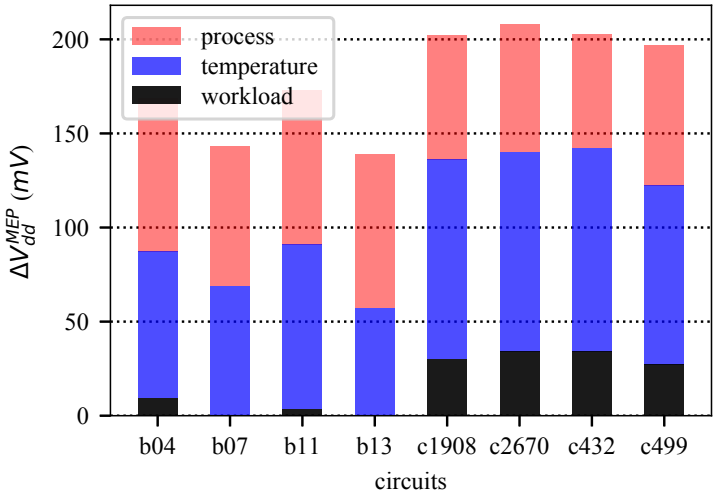


Figure 5.2: The impact of process, temperature, and workload variations on V_{dd}^{MEP} for different benchmark circuits under full temperature range $[-25°C, 100°C]$ and the reasonable activity range.

tioned Monte-Carlo simulation setup and under *reasonable input switching activity variation*, i.e., $\pm 27\%$ α variation. As shown, temperature variation has the highest impact on the MEP, followed by process variation. The workload variation impact in all circuits is much less than the cumulative impacts of temperature and process variation.

Overall, due to the lower impact of the workload on the MEP as well as the high cost of implementing a dedicated circuit for measuring the activity, in compact and low-power chips designed for IoT applications, we can establish our MEP adaptation method only based on process and temperature variations.

5.2.2 MEP tuning based on process and temperature variations

Considering the impact of workload variation for MEP adaptation imposes extra hardware overhead because it is necessary to measure the amount of activity at runtime [78, 195–198, 258, 259]. The idea of the proposed method in this section is to achieve the best efficiency without imposing such hardware overhead by predicting the MEP of an NTC circuit at runtime only based on process and temperature variations.

We collect accurate information regarding the process corner of a fabricated chip and its sensitivity to temperature during the post-fabrication measurements and test. Then, a LUT is created and stored on the chip to facilitate the MEP tuning at runtime. This LUT maps each temperature to a specific MEP condition and is created with the help of a pre-trained machine-learning model. Therefore, it is possible to predict the MEP by reading the temperature at runtime, easily.

Most modern IoT chips have temperature sensors or have facilities to collect temperature information, such as Analog-to-Digital Converter (ADC). Therefore, the proposed method does not impose extra circuit overhead for measurements. The temperature→MEP mapping logic can be implemented in either hardware or software. Therefore, we eliminate the need for an embedded closed-loop energy monitoring system, resulting in much less hardware overhead.

Since the proposed method relies on supervised machine-learning, we need to prepare a training dataset. The training dataset is extracted by characterizing many instances of the same circuit and includes important characteristics of the circuit instances such as speed and power as well as the MEP. Once the machine-learning model is trained on the training dataset, it can predict the MEP of any given instance of the circuit based on the circuit characteristics,

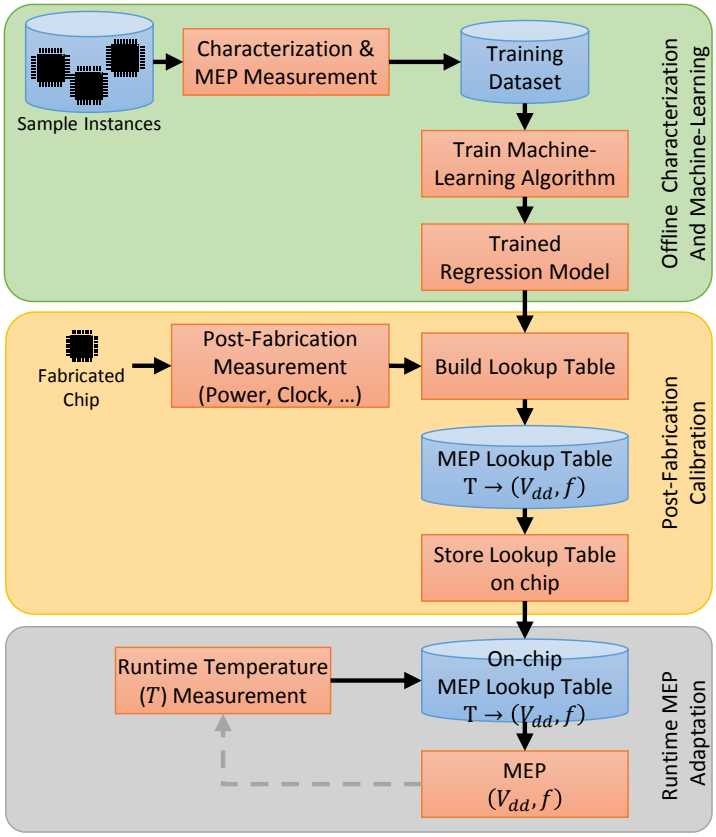


Figure 5.3: Overall flow of the proposed MEP prediction method for NTC circuits. The flow consists of three main parts: 1) Offline characterization and machine-learning, 2) Post-fabrication calibration, and 3) Runtime MEP adaptation.

which can be obtained using a limited number of measurements during the manufacturing tests on a per-chip basis. Afterward, we create the LUT which maps the temperature to the MEP using the trained machine. The LUT is finally stored on the chip to be used for MEP adaptation at runtime. Eventually, the MEP tuning is done at runtime based on the calibrated LUT.

The proposed method leverages the flattening of the energy-voltage curve around the MEP (see Figure 2.8b). We will show in Section 5.4 that the energy inefficiency, caused by a small inaccuracy in the MEP prediction due to ignoring the impact of workload variation, is negligible (see Figure 5.8).

The overall flow of the proposed method is presented in Figure 5.3. The flow consists of three steps (1) Offline characterization and Machine-learning training, (2) Post-fabrication calibration, and (3) Runtime MEP adaptation, explained in the following.

Offline characterization and machine-learning training Enough number of randomly chosen *sample circuits* are needed to make a training dataset which contains sample circuit characteristics and the corresponding MEPs. Then, by training a regression model on the generated training dataset, it is possible to predict the MEP (both V_{dd}^{MEP} and T_{clk}^{MEP}) based on any set of input parameters.

Sample circuits: could be generated using a Monte-Carlo simulation (pre-silicon), or could be fabricated chips (post-silicon). Here, we employ the sampling method presented in Section 5.2.1 to create the circuit sample instances. These circuits are characterized at different temperatures and voltages as mentioned in Table 5.1, and their power and timing properties

Table 5.3: Regression parameters for MEP prediction

Process Variation Inputs ($Predictors$)	
Parameters	Leakage power ($Pleak$), Dynamic power (P_{dyn}), Clock period (T_{clk})
at supply voltages	Near-threshold (V_{dd}^{NT}), Nominal(V_{dd}^{NOM})
at temperatures	Low (TL), High (TH)
Regressions	$V_{dd}^{MEP} = f(Predictors, \boldsymbol{T})$ $T_{clk}^{MEP} = g(Predictors, \boldsymbol{T})$

are extracted.

Training dataset: maps *predictors* (or inputs) to *targets* (or outputs). According to the analysis presented in Section 5.2.1, temperature and process variation are the most important factors affecting the MEP. Therefore, the predictors are the parameters which represent process variation and temperature of the circuits. The targets are circuit MEP parameters, which are: $\{V_{dd}^{MEP}, T_{clk}^{MEP}\}$.

The process variation impact can be explained by: Dynamic power (P_{dyn}), Leakage power (P_{leak}), and Clock period (T_{clk}). However, we need to measure these parameters at two different supply voltages $V_{dd} \in \{V_{dd}^{NT}, V_{dd}^{NOM}\}$ and two different temperatures (Low TL and High TH) to contain information about the sensitivity of the MEP to both supply voltage and temperature in the dataset. Table 5.3 summarizes the predictor parameters stored in the dataset and used for building a regression model for MEP prediction.

In addition to the predictors, the target parameters are also included in the training dataset. For each sample circuit, the MEP is discovered by sweeping V_{dd} from nominal voltage to below threshold range, and by measuring the power and speed of the sample circuit. This process is also done at different temperatures and input switching activity values. Finding the MEP of the sample circuits is an exhaustive process; however, this step is a one-time process and can be done offline.

Post-fabrication calibration In this step, the trained machine-learning model from the previous step can predict the MEP of any instance of the chip based on the predictor parameters shown in Table 5.3. Therefore, for each fabricated chip, we need to measure these parameters during the post-fabrication tests.

It is important to keep the measurement time as short as possible at this step because of the corresponding costs. During a typical test phase, many of the parameters mentioned in Table 5.3 are measured at different temperature and voltages in order to assure the functionality of the circuit and to evaluate the specification of the chip. Therefore, the required predictor measurements do not impose much overhead to the test phase.

Please note that the accuracy of the measurements during the post-fabrication testing is rather high considering the use of external measurement tools and Automated Test Equipment (ATE), and is sufficient for the MEP prediction.

Once all the predictors are measured, the regression models in Table 5.3 are used to obtain the MEP for different temperature values ($\boldsymbol{T}$). Accordingly, a LUT is created which maps different $\boldsymbol{T}$ values to the corresponding MEP values. In the end, the generated LUT is stored on the chip to be used for MEP tuning at runtime. As we ignore the fluctuation in the workload while creating the LUT, for each circuit we assume the average activity $\hat{\alpha}$ and the LUT is generated only for a single $\hat{\alpha}$ value.

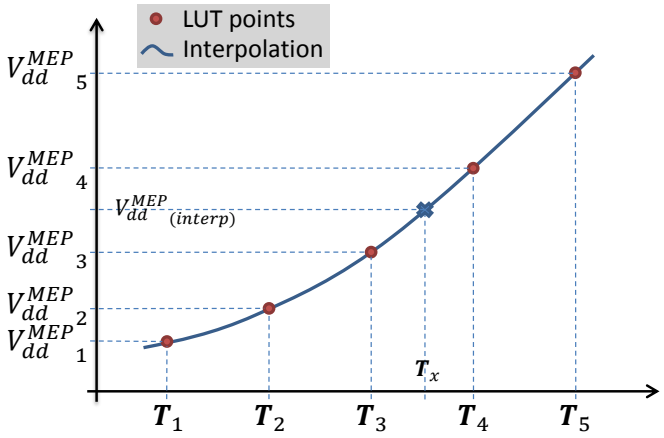


Figure 5.4: Interpolation is used to find the MEP values for any intermediate temperature value ($\boldsymbol{T_x}$) which does not exist in the stored LUT on chip.

Runtime MEP adaptation At runtime, the temperature is measured, and with the help of the LUT, the MEP is obtained. We use PCHIP linear interpolation to find the MEP for any given value which does not exist in the LUT, as shown in Figure 5.4. We apply the PCHIP interpolation on the logarithm of the values in order to improve the interpolation quality for T_{clk}^{MEP} interpolation (i.e., log-linear interpolation). Then, the supply voltage and frequency are adapted according to the obtained MEP.

The runtime adaptation task does not need to be running all the time. Since the MEP change is only significant due to temperature variation, the runtime adaptation is only invoked when temperature changes significantly to predict the MEP and adapt circuit V_{dd} and T_{clk} accordingly.

Creating the Look-Up-Table for MEP prediction

The LUT is a simple mapping of temperature to V_{dd}^{MEP} and T_{clk}^{MEP}. However, it needs to consider the impact of the process variation as well. Therefore, we first train regression models based on the training dataset, and then we use this regression model to extract the LUT. We are interested in finding the regression models f and g, shown in Table 5.3, which predict the MEP ($V_{dd}^{MEP,pred}$, $T_{clk}^{MEP,pred}$) based on the measured predictors at two different temperatures (TL, TH) and two different supply voltages ($V_{dd}^{NT}, V_{dd}^{NOM}$).

In order to create the LUT, we put the measured predictors into the regression models f and g to create two new regression models which are solely dependent on temperature ($\boldsymbol{T}$):

$$V_{dd}^{MEP} = F(\boldsymbol{T}), \tag{5.1}$$

$$T_{clk}^{MEP} = G(\boldsymbol{T}). \tag{5.2}$$

The new regression models are in fact the LUT. Therefore, we evaluate F and G regression models for a number of temperature values and store them on the chip as the LUT. At runtime, a PCHIP interpolation method will find the values for all temperature values.

Machine-learning model building We use Bayesian Ridge Regression (BRR) to make the regression models [263]. BRR is a linear model similar to an Ordinary Least Square (OLS) fitting; however, it has properties which makes it suitable for our MEP prediction problem. It is mathematically provable that a *Ridge regression* is more suitable for the problem with collinearity [264, 265], i.e. when a predictor variable can be predicted from other predictors

with good accuracy. Since we most probably have such relation between the predictors (the measured parameters), it is necessary to use a regression model which is able to handle such problems. Despite an OLS which finds a set of parameters which optimizes the error between predicted target values and real target values, a *Bayesian regression model* treats the parameters as random variables and tries to find a set of hyper-parameters that optimizes the posterior distribution. Therefore, it would lead to less over-fitting and better cross-validation scores, which results in good interpolating and extrapolating characteristics. The fitting quality of BRR for ill-posed problems is also better than OLS, due to penalization of the size of coefficients.

The prediction performance of the regression model can be substantially improved by using *logarithmic* and *polynomial* transformation. *The logarithmic transformation* that we use for this purpose is log-linear as follows:

$$\log(V_{dd}^{MEP}) = f(Predictors, \boldsymbol{T}),$$
$$\log(T_{clk}^{MEP}) = g(Predictors, \boldsymbol{T}),$$

This improves the accuracy of the regression model because it comprehends the exponential change in circuit parameters in the NTV region [160]. Moreover, we can incorporate the interdependency between the parameters with higher order polynomials. Therefore, we transform the parameters space using a second order polynomial. Finally, we perform 5-fold cross-validation to reduce the over-fitting.

5.3 Runtime adjustment of IoT SoCs

5.3.1 Voltage and frequency islands in SoC

Among different techniques to manage the power consumption of the SoCs, Voltage/Frequency Island (VFI) is popular because it allows the designer to maintain performance while controlling the power consumption and temperature [255]. Depending on the running workload, some SoC components are utilized whereas the unused components are power-gated. However, having multiple VFI increases the design complexity of SoC integration, as it necessitates to design multiple Power Delivery Networks (PDN) and to add extra components such as voltage regulators, PLLs, and voltage level shifters. Therefore, in a typical IoT SoC, the number of VFIs is typically limited. As the overhead of having multiple VFI could be high in ultra-low power SoCs, several SoC components could reside on the same voltage island, sharing the same supply voltage. This means that changing the supply voltage of a VFI affects several components. Additionally, each SoC component could have a unique power consumption pattern. For example, the leakage power of a cache memory dominates its power consumption, while on the contrary, the dynamic power is predominant in a cryptography co-processor. The voltage scaling capabilities of the components are also widely different; Memories and communication components are known to have a limited voltage scaling whereas the supply voltage of the combinational circuits can be scaled down easier. Moreover, the performance requirements for each component, dictated by the running application demands or functional constraints, could impede the VFI voltage-scaling.

As a result, any envisioned methodology for voltage-scaling of the VFIs has to consider a considerable number of parameters, which overly complicates the problem if the voltage-scaling method is going to be implemented as a hardware component.

Table 5.4: Problem definition of the machine-learning model to find the most energy-efficient system-wide voltage

Objective: Find the energy-efficient system-wide $V_{dd}^{MEP,chip}$ given the SoC parameters and constraints	
Parameters affecting $V_{dd}^{MEP,chip}$ of SoC (also called "features")	
$S = [s_i], \quad s_i \in \{0, 1\}$	Power-gating states of the SoC components
$U = [u_i], \quad 0 \leqslant u_i \leqslant 1$	Utilizations of the SoC components
$\boldsymbol{T}$	Temperature of the SoC
Constraints from application (performance) or functional requirements	
$V_{dd}^{min} = [V_i^{min}]$	Minimum voltage of the SoC components
Approach: Predict $V_{dd}^{MEP,chip}$ using a machine-learning based regression model trained on the parameters of the SoC: $V_{dd}^{MEP,chip} = \underset{V_{dd}}{\operatorname{argmin}}\, Energy(V_{dd}) \quad \text{subject to} \quad \{S, U, \boldsymbol{T}, V_{dd}^{min}\}$	

5.3.2 Regression model for runtime adjustment of SoC

In this section, we propose the method for tuning the supply voltage of SoCs for the best energy efficiency.

Problem definition

The objective is to find the best supply voltage for each voltage island, which leads to the best energy-efficiency for the entire SoC based on the SoC operating temperature and the utilization of the SoC components. A SoC consists of different components, some of which have the same supply voltage. They could also be fully/partially utilized, idle, or completely power-gated during the runtime. We assume that the information regarding the *power-gating state* and the *utilization* of the components is available at the firmware-level through hardware registers, performance counters, and from the running application. Table 5.4 summarizes the problem definition.

We propose to solve this problem by determining the MEP during the runtime at firmware-level with the help of a supervised machine-learning model, which is trained offline on the input parameters (S, U, T). Then, the supply voltage of the SoC is adapted to the determined MEP value $V_{dd}^{MEP,chip}$.

Without the loss of generality, we solve the problem for a SoC with one voltage island and only for one process corner. The same approach can be applied to SoCs with several voltage islands and different process corners[4] by repeating the same methodology for individual voltage islands.

Input parameters

The parameters and the constraints presented in Table 5.4 are accessible at firmware-level.

- **The power-gating states of the SoC components** ($s_i \in \{0, 1\}$) describe whether a SoC component is power-gated ($s_i = 0$) or turned-on ($s_i = 1$). This information is accessible via power-management registers.

[4]The process corner is typically extracted during the post-manufacturing tests [266].

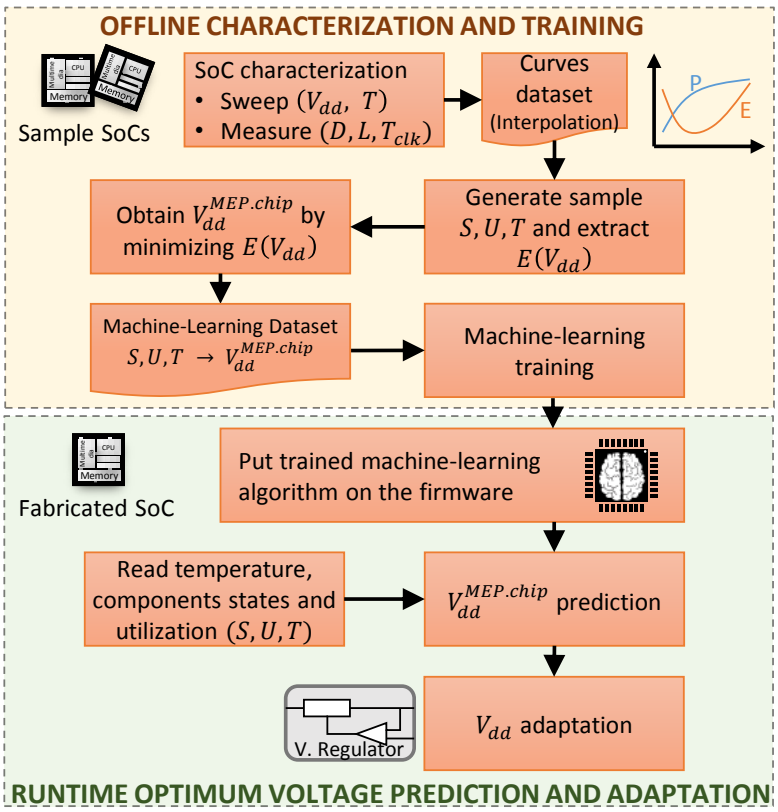


Figure 5.5: Overall flow of the proposed runtime adjustment of SoCs to minimum energy operation.

- **The temperature of the chip (T)** can be obtained from on-chip or off-chip sensors. Most of the conventional IoT SoCs, which are designed to operate at NTC, contain a temperature sensor or provide facilities to read it from an off-chip sensor because temperature significantly affects NTC operation.

- **The utilization of the SoC components** under the running workload ($0 \leqslant u_i \leqslant 1$) could range from idle ($u_i = 0$) to fully utilized ($u_i = 1$). There is no single recipe to derive the utilization of the SoC components with an acceptable accuracy, as the utilization value depends on the circuit type and structure. As an example, the average power consumption of an Advanced Encryption Standard (AES) encoder when utilized is not significantly dependent on the input data. Therefore, the utilization value of the AES encoder is 1.0 when it is used. However, in the case of a processor, the utilization highly depends on the input data and the running workload. Fortunately, there exist numerous methods for extracting the utilization of various circuits, such as [267, 268] for microprocessors, or [269] for utilization prediction of other circuits.

- **Minimum supply voltage constraints** (V_{dd}^{min}) are also applied to the SoC depending on the internal structure of SoC components, or due to the application performance requirements.

5.3.3 Implementation flow

The overall flow of the proposed method is presented in Figure 5.5. The flow consists of two steps:

1. *Offline characterization and training*, in which samples of the SoC from the same process corner are characterized, the training dataset is prepared based on the characterization data, and the machine-learning model is trained.

2. *Runtime optimum voltage prediction and adaptation*: The trained machine-learning model is put on the firmware of the fabricated chips. The optimum supply voltage is calculated by the machine-learning model during runtime, according to the power-gating state and utilization of the SoC components and the temperature. Afterward, the chip is adapted to that voltage by tuning the voltage regulator.

Offline characterization and training

We collect samples of the SoC from the same process corner and characterize them to obtain the dynamic power P_{dyn}, the leakage power P_{leak}, and the speed T_{clk} curves of the individual components on the SoC. For this, the supply voltage of the voltage island is swept from the above-threshold region to the sub-threshold region. For each supply voltage value, the dynamic power P_{dyn}, the leakage power P_{leak}, and the speed T_{clk} of the SoC components are measured. The temperature also needs to be swept over the operating range, and the aforementioned curves are extracted. As it could be too costly to perform the characterization on many different supply voltages and temperatures on each SoC, we perform the measurements on a limited number of temperatures and supply voltage levels, and the rest of the points are extracted by PCHIP interpolation [270]. Finally, representative dynamic power $P_{dyn}(V_{dd}, \boldsymbol{T})$, leakage power $P_{leak}(V_{dd}, \boldsymbol{T})$, and speed $T_{clk}(V_{dd}, \boldsymbol{T})$ curves are extracted as the average of the curves of the measured SoCs. This process is an offline characterization and has to be done only once to obtain the *curves dataset*.

The representative curves are used along with the power-gating states (S), the utilization parameters (U), and the temperature ($\boldsymbol{T}$) to approximate the power consumption of the SoC chip:

$$P_{leak,chip}(V_{dd}, \boldsymbol{T}) = \sum_{i=1}^{N} s_i.P_{leak,i}(V_{dd}, \boldsymbol{T}),$$

$$P_{dyn,chip}(V_{dd}, \boldsymbol{T}) = \sum_{i=1}^{N} u_i.P_{dyn,i}(V_{dd}, \boldsymbol{T}).$$

Here, $P_{dyn,chip}$ and $P_{leak,chip}$ are dynamic and leakage power consumption of the SoC chip. $P_{dyn,i}$, $P_{leak,i}$, s_i, and u_i are dynamic power, leakage power, power-gating state, and the utilization of component i of the SoC chip. Furthermore, we can approximate the number of operations performed by the SoC as:

$$f_{chip}(V_{dd}, \boldsymbol{T}) = \sum_{i=1}^{N} w_i.f_i(V_{dd}, \boldsymbol{T}),$$

where f_{chip} is the number of operations done by the chip at a unit of time and f_i is the frequency of component i. The number of operations performed by a component is typically proportional to the frequency of the component; however, it is not always the same value. Therefore, the scale factor w_i is introduced for each component to scale the frequency to the number of operations. Consequently, the energy (per-cycle) for each temperature and supply voltage is obtained as:

$$Energy(V_{dd}, \boldsymbol{T}) = \frac{P_{leak,chip}(V_{dd}, \boldsymbol{T}) + P_{dyn,chip}(V_{dd}, \boldsymbol{T})}{f_{chip}(V_{dd}, \boldsymbol{T})}. \tag{5.3}$$

The optimum supply voltage of the SoC ($V_{dd}^{MEP,chip}$) is calculated by minimizing $Energy(V_{dd}, T)$ in Equation (5.3), for a given power-gating state vector S, utilization vector U, and temperature $\boldsymbol{T}$. Consequently, we make the *machine-learning dataset* as a LUT which maps all possible combinations of power-gating state vector S, utilizations vector U, and temperature $\boldsymbol{T}$ to the optimum supply voltage of the SoC ($V_{dd}^{MEP,chip}$).

$$\text{Machine-Learning Dataset} : \begin{cases} \text{Features:} & (S, U, \boldsymbol{T}) \\ \text{Target:} & V_{dd}^{MEP,chip} \end{cases}$$

This dataset is used to train and validate a machine-learning model for predicting $V_{dd}^{MEP,chip}$ based on the features $(S, U, \boldsymbol{T})$. Therefore, this dataset is partitioned into a *training dataset* and a *validation dataset*, where the former is used for training the machine-learning model, and the latter is used for the validation of the trained machine-learning model and to extract the accuracy results presented in Section 5.4.

In this work, we used Stochastic Gradient Boosted Regression Trees (SGBRT) [271] for predicting $V_{dd}^{MEP,chip}$ during the runtime. SGBRT performs well for the model fitting problems, which is the case for our problem, whereas a polynomial regression model fits poorly on our dataset. It is important to control the evaluation cost of the trained SGBRT model because the evaluation process is done at the firmware-level during runtime. The evaluation cost of SGBRT is dependent on the number of trees and the depth of the tree. For a tree with depth k, we only need $k-1$ comparisons to get to the leaf value. Therefore, for an SGBRT model containing m regression trees, we need at most $(k-1) \times m$ comparisons and $m-1$ additions.

Better prediction performance can be achieved by *feature engineering* techniques such as considering the inter-relation between the features described in Table 5.4. In this regard, we introduce additional features by considering the multiplications of every two features (second order polynomial feature transformation). Moreover, a *feature selection* method based on information criterion is applied to obtain the most important features out of the original (i.e., $(S, U, \boldsymbol{T})$) and the engineered features. Therefore, only a small subset of all features are used for prediction in the final model. This significantly improves the performance of the machine-learning model considering the cross-validation results, while shrinking the dimensions of the regression trees.

To prevent over-fitting and under-fitting, the hyper-parameters of the SGBRT (e.g., k and m) should be optimized. We perform a grid search on the hyper-parameter space and perform 5-fold cross-validation to evaluate the best hyper-parameters set, which does not overfit the data[5]. We also consider the implementation overhead during the hyper-parameter optimization by limiting the upper bounds of k and m.

Runtime optimum voltage prediction and adaptation

The trained machine-learning model from the previous step is stored on the firmware after SoC fabrication. During the runtime, the machine-learning model on the firmware is called to determine the $V_{dd}^{MEP,chip}$. The predicted $V_{dd}^{MEP,chip}$ is then checked versus the minimum supply voltage constraints of the active components. The supply voltage, to which the circuit is tuned, should be greater than the minimum supply voltage of all active components:

$$V_{dd} = \max\left\{V_{dd}^{MEP,chip}, \max_{1 \le i \le n} u_i.V_i^{min}\right\}$$

[5] A good hyper-parameter set maximizes the prediction accuracy in all 5 cross-validation folds and minimizes the difference among the accuracy of the cross-validation folds [272].

Then, the SoC is adapted to the V_{dd} obtained from the above equation by setting the predicted voltage on the voltage regulator. As a result, the proposed methodology considers the performance requirements of the applications being executed on the SoC.

5.4 Results and discussions

5.4.1 Circuit results

In this section, we evaluate the accuracy of MEP prediction in the proposed method and report the corresponding energy-efficiency. The experiment is done on circuits c432, c499 c1908, and c2670 from ISCAS'85 benchmark and circuits b04, b07, b11, and b13 from ITC'99 benchmark [26, 231]. $N = 1000$ samples are generated for each benchmark circuit using the Monte-Carlo method presented in Section 5.2.1, and 30% of all samples are put aside for the validation purpose (validation dataset) while the rest of them are used for training with cross-validation (training dataset). The log-linear BRR model is then fitted on the training data. The prediction results are evaluated on the validation dataset for different input switching activity levels to extract the energy-efficiency of the proposed method.

We compare the state-of-the-art runtime MEP tuning methods with our proposed post-fabrication calibration and runtime MEP adaptation methods. The *post-fabrication calibration* method [7] is suitable for circuits which are under limited temperature variation; however, the proposed *runtime MEP adaptation* method is applicable to the circuits which are under large temperature variation.

MEP prediction accuracy

The predicted MEP values for each sample circuit are compared to the values from timing and power reports extracted by Synopsys Design Compiler. The coefficient of determination (r^2 score) and *Root Mean Square Error (RMSE)* are reported in order to present the quality of the predictions [273]. The coefficient of determination explains the portion of the variance in the dependable parameters (V_{dd}^{MEP} and T_{clk}^{MEP}) that is predictable from the predictor variables (inputs in Table 5.3). Therefore, in our regression problem, a score close to 1.0 means that the changes in the dependent variables (here V_{dd}^{MEP} and T_{clk}^{MEP}) can be better predicted based on the predictor variables (i.e., the measured parameters during post-fabrication tests and temperature).

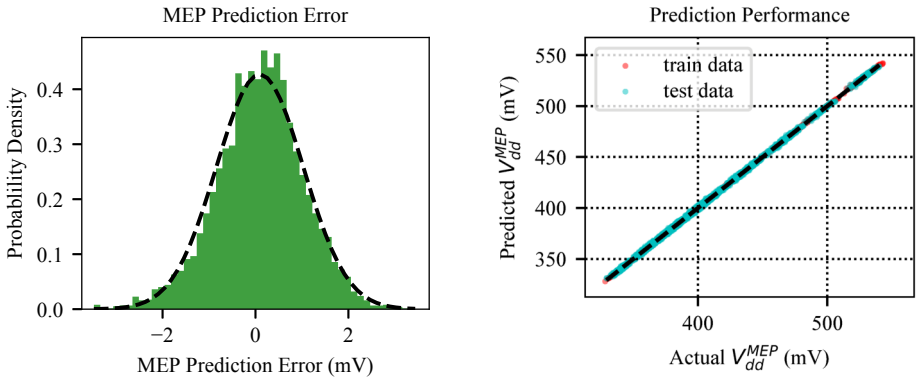
The *RMSE* metric quantifies the prediction power of a model and can be calculated as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2}, \qquad e_i = y_{i,predicted} - y_{i,actual}. \tag{5.4}$$

The *RMSE* is the standard deviation of the residuals (e_i). We only report the *RMSE* for V_{dd} because the range of variation in T_{clk} is very large and a single *RMSE* value is not meaningful without considering the real T_{clk} for each condition. For T_{clk} we rely on the r^2 score.

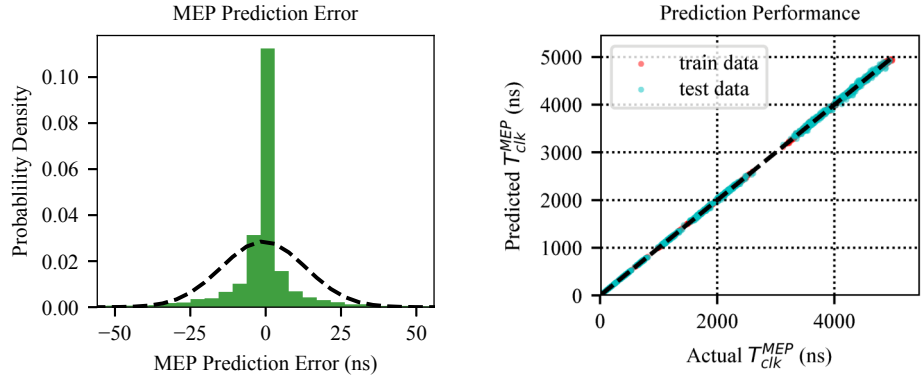
The fitting results for benchmark circuit b04 from ITC'99 are demonstrated in Figure 5.6. The r^2 score for predicting V_{dd}^{MEP} is 0.997 and for T_{clk}^{MEP} is 0.999 on the validation set, which explains an excellent fitting. The *RMSE* for V_{dd}^{MEP} is less than 1.4mV and the prediction has less than 0.3% relative error. High r^2 fitting scores and the low *RMSE* confirm the effectiveness of the approach.

Table 5.5 summarizes the prediction accuracy of the proposed method for the tested benchmark circuits. The average prediction scores of $r^2 = 0.997$ and the average $RMSE = 1.1mV$ (relative error of less than 0.3%) show a very high fitting quality. The prediction error is



(a) V_{dd}^{MEP} prediction error

(b) V_{dd}^{MEP} prediction performance



(c) T_{clk}^{MEP} prediction error

(d) T_{clk}^{MEP} prediction performance

Figure 5.6: (a), (b) V_{dd}^{MEP} prediction performance for circuit b04 ($r^2 = 0.997$, $RMSE = 1.4mV$). (c), (d) T_{clk}^{MEP} prediction performance for circuit b04 ($r^2 = 0.999$).

$\sim 10\times$ and $\sim 50\times$ smaller than the V_{dd} step-size in [78] and [197] (1.1mV compared to 10mV and 50mV), respectively, which means far better accuracy and energy-efficiency.

Temperature measurement error impacts

Runtime temperature measurement is typically associated with some inaccuracies, which leads to inaccuracy in the MEP prediction. Moreover, since the LUT is generated based on an assumption on the circuit activity level, any deviation from that assumption may lead to additional MEP prediction inaccuracy. Therefore, it is necessary to evaluate the impact of such inaccuracies on the MEP prediction and energy-efficiency in a realistic scenario.

Figure 5.7 reports the impacts of such inaccuracies for c2670 circuit. As shown, a high measurement error in temperature ($\geq 5°C$) leads to relatively high V_{dd}^{MEP} prediction error. Furthermore, additional inaccuracy in the circuit activity assumption ($\Delta\alpha = \pm 25\%$) could deteriorate the MEP prediction quality, when the temperature measurement error is also high.

However, under a reasonable measurement temperature accuracy ($\leq 2°C$), the MEP prediction error is limited to below 20mV, which does not significantly affect the energy-efficiency. This is due to the small sensitivity of the energy-efficiency to the V_{dd} variations around the MEP (see Figure 2.8b). Therefore, a fair temperature measurement accuracy helps to keep the predicted V_{dd} close to the real V_{dd}^{MEP}.

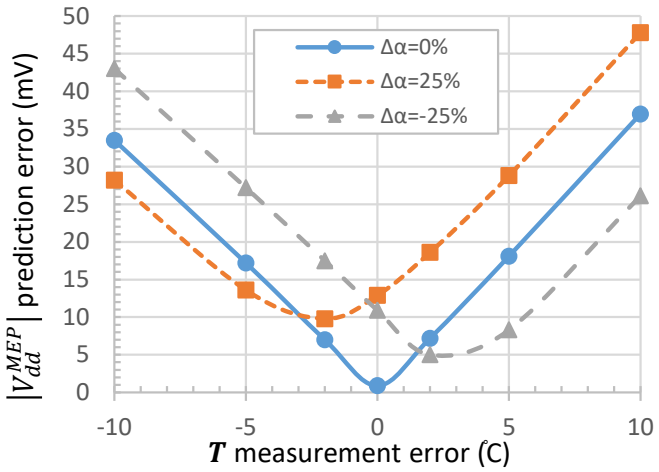


Figure 5.7: The impact of measurement error in temperature (up to $\pm 10°C$) and circuit activity fluctuation (up to $\pm 25\%$ α error) on c2670 MEP prediction accuracy.

Energy consumption of sub-optimal supply voltage

Operating a circuit at a supply voltage other than its V_{dd}^{MEP} is associated with sub-optimal energy. We refer to the amount of energy consumption due to the inaccurate prediction of the MEP by ΔV_{dd}^{MEP} as *Imprecision Energy (IE)*, which is presented as a relative percentage number. Therefore, a *lower* IE corresponds to a *better* energy-efficiency and overall effectiveness of the method.

Figure 5.8 explains the impact of ΔV_{dd}^{MEP} on imprecision energy for selected ISCAS'85 and ITC'99 benchmark circuits. According to this figure, the imprecision energy associated with $\Delta V_{dd}^{MEP} = \pm 150mV$ could be as high as 276% on average with a standard deviation of 151% (for c499). Therefore, the IE could be very large if the supply voltage is not tuned to V_{dd}^{MEP}. However, due to the flattening of energy curves at the MEP, a slight difference between the supply voltage and the V_{dd}^{MEP} does not lead to much IE:

$$|\Delta V_{dd}^{MEP}| \leq 50mV \longrightarrow IE_{average} \leq 14\%,$$
$$|\Delta V_{dd}^{MEP}| \leq 25mV \longrightarrow IE_{average} \leq 3\%.$$

Therefore, a slight inaccuracy in the V_{dd}^{MEP} prediction may be traded for less complexity of the V_{dd}^{MEP} tuning method depending on the application.

We evaluate the imprecision energy of the benchmark circuits by considering the MEP prediction inaccuracies due to temperature measurement error. The values are extracted for two different scenarios:

1. *Small* $\boldsymbol{T}$ *variation*, in which the operating temperature is assumed to be in a limited range, i.e., $25 \pm 10°C$.

2. *Large* $\boldsymbol{T}$ *variation*, in which the operating temperature can change from $-25°C$ to $100°C$.

In either case, we assume a reasonable circuit activity variation of $\pm 25\%$ α, as explained in Section 5.2.1, and a temperature measurement error of $\pm 2°C$ in energy calculations, to reflect the real condition.

The imprecision energies of these scenarios are reported in Table 5.5, for our proposed post-fabrication calibration method [7], and for the runtime MEP adaptation method. The post-fabrication calibration method cannot adapt to the $\boldsymbol{T}$ variation and results in up to 74%

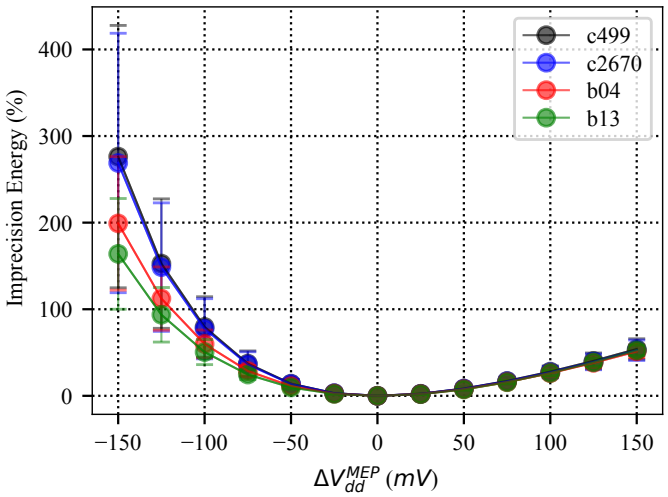


Figure 5.8: The imprecision energy associated with operating a circuit at a non-optimal supply voltage (with ΔV_{dd}^{MEP} shift from optimum V_{dd}^{MEP}) for selected ISCAS'85 and ITC'99 benchmark circuits. The circles demonstrate the average imprecision energy (μ_{IE}) over different operating conditions and the vertical bars demonstrate the standard deviation σ_{IE}.

imprecision energy for combinational circuits (c432 and c2670). However, the runtime MEP adaptation method correctly reflects the impact of temperature on the predicted V_{dd}^{MEP} and can keep the imprecision energy well below 2.6%. Please note that since the runtime MEP adaptation method adapts to $\boldsymbol{T}$ variation, the associated imprecision energy under small and large $\boldsymbol{T}$ variations are similar. Hence, only the results for large $\boldsymbol{T}$ variation scenario are reported in the table.

Hardware overhead

In the NTC circuits, the most important runtime variability, which is ambient temperature variation, changes slowly. Therefore, it is not necessary to monitor and tune the supply voltage with high frequency. As a result, it is favorable to implement the MEP prediction at the software level when possible, which eliminates the entire monitoring and MEP finding hardware overhead. However, the calibrated LUT, which maps the temperature to the MEP, has to be stored somewhere in the permanent memory. The required memory size is dependent on the resolution of temperature, V_{dd}^{MEP}, and T_{clk}^{MEP} values stored the LUT. For example, if we consider $1°C$ resolution in the LUT temperature values, we need to have 126 values of V_{dd}^{MEP} and T_{clk}^{MEP} for a full range of $[-25, 100]°C$. In this case, we may assume one-byte resolution for both V_{dd}^{MEP} and T_{clk}^{MEP} values, which means that the size of the LUT is $2 \times 126 = 252$ bytes. This amount of memory is allocable in almost all modern IoT chips. In case of limited allowable memory for the LUT, it is also possible to have less temperature resolution (e.g., $5°C$) and use the PCHIP interpolation instead, which reduces the size of the LUT to only 50 bytes (for $5°C$ resolution).

Comparison with the state-of-the-art

Table 5.6 compares the proposed method based on post-fabrication calibration and runtime MEP adaptation, and the state-of-the-art hardware-based closed-loop V_{dd} tuning method in different scenarios. The post-fabrication calibration method [7], which extracts a single best MEP on a per-chip basis during the post-fabrication tests, is effective under small $\boldsymbol{T}$ variation.

Table 5.5: MEP prediction scores of the proposed method (higher r^2 scores, lower $RMSE$, and lower imprecision energy values are better).

Circuit	Prediction Scores			Worst-case Imprecision Energy (%)		
	V_{dd}^{MEP}		T_{clk}^{MEP}	post-fabrication calibration [7]		runtime MEP adaptation
	r^2	$RMSE$ (%)	r^2	Small $\boldsymbol{T}$ variation	Large $\boldsymbol{T}$ variation	Large $\boldsymbol{T}$ variation
c432	0.999	0.9mV (0.2%)	0.999	4.5%	74%	2.3%
c499	0.999	0.6mV (0.2%)	0.999	3.0%	40%	1.3%
c1908	0.999	0.9mV (0.2%)	0.999	3.5%	48%	2.6%
c2670	0.999	0.9mV (0.2%)	0.999	4.5%	73%	2.4%
b04	0.997	1.4mV (0.3%)	0.999	0.69%	26%	0.38%
b07	0.997	1.0mV (0.3%)	0.999	0.10%	11%	0.12%
b11	0.995	1.6mV (0.4%)	0.999	0.24%	15%	0.35%
b13	0.994	1.2mV (0.3%)	0.999	0.10%	8.1%	0.97%
average	0.997	1.1mV (0.3%)	0.999	2.1%	37%	1.2%

As this method does not adapt to runtime variations and only considers the impact of process variation, there is no hardware overhead associated with it. Nevertheless, it does not result in high energy-efficiency when temperature variation is significant.

The proposed runtime MEP adaptation method can adapt the MEP to temperature and process variation, resulting in high energy-efficiency in all scenarios. It monitors the temperature and tunes V_{dd} accordingly, which does not impose much hardware overhead. Furthermore, this task can be done at the software level (or firmware level). By implementing the proposed method at the firmware level we have the opportunity to check V_{dd} for performance constraints as well (for applications which demand specific speed).

The existing closed-loop V_{dd} tuning approaches as in [78, 195, 197, 198, 258, 259] constantly monitor the energy-efficiency of the circuit by evaluating the energy consumption of the circuit at different supply voltages in order to find the best supply voltage which maximizes the energy-efficiency. However, they require extra circuitry to measure the power consumption and find the V_{dd}^{MEP}. Additionally, other constraints such as performance constraints cannot be implemented in such techniques.

5.4.2 SoC results

We assume that a SoC contains n different components. Here, we assumed a SoC with four components: 1) a processor core (Leon3mp), 2) a security coprocessor (AES), 3) an FFT core, and 4) the digital part of a communication interface (ZigBee). The details of the components are described in Table 5.7.

Each component is synthesized by Design Compiler. As these components are supposed to operate at near-threshold voltage range, all the cells which are not suitable for NTC (e.g., cells which contain more than three stacking transistors) are eliminated from the library [28]. Furthermore, we characterize the cell library for various supply voltages and temperatures in Table 5.7. These libraries are used to obtain *power* and *timing* reports of the components at the supply voltages and temperatures in Table 5.7 using Synopsys Design Compiler.

In order to extract a more accurate estimation of the power consumption from Synopsys

Table 5.6: Comparison of different V_{dd} tuning methods

Scenario	Small $\boldsymbol{T}$ variation	Large $\boldsymbol{T}$ variation
Proposed post-fabrication calibration method [7]	+ Accuracy + Energy-efficiency + Zero hardware overhead	– Inefficient
Proposed runtime MEP adaptation method	+ Accuracy + Energy-efficiency = Small hardware overhead	+ Accuracy + Energy-efficiency = Small hardware overhead
Closed-loop V_{dd} tuning [78, 195, 197]	+ Accuracy + Energy-efficiency – High hardware overhead	+ Accuracy + Energy-efficiency – High hardware overheads

Table 5.7: Simulation setup for testing the proposed method, including the circuit components and the characterization setup

Circuits

	# Gates	# Flip-flops
Leon3mp	37030	2364
AES	21725	530
FFT	10414	705
ZigBee	6097	480

Characterizations

Supply Voltages (V)	0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.9, 1.05
Temperatures ($^{\circ}C$)	-25, 0, 25, 50, 75

Design Compiler, we execute relevant workloads on each component in ModelSim [274], for example MiBench workloads on the Leon3mp core and several rounds of encryption on the AES module, and calculate the average switching activity of each component in the SAIF format for power calculation.

The collected dataset is divided into a *training set* and a *validation set*. The SGBRT regression model is trained on the training set (including the cross-validations for model hyper-parameters evaluation), and finally, the prediction accuracy is evaluated on the validation dataset.

Similar to circuit results, two statistical metrics explain the accuracy of our results: 1) the coefficient of determination (R^2 score) which defines the ability of the model in explaining the variance in the predicted variable and 2) $RMSE$ which explains the standard deviation of the residual errors [272]. The $RMSE$ of the model is 7mV, which means 99.7% of the predictions are done with the accuracy of $3 * RSME = 21mV$. As explained for the circuit result in Section 5.4.1, the impact of supply voltage miss-prediction on energy-efficiency is very small considering that around the MEP point the energy-efficiency is not compromised by a small amount of error.

Comparison with the state-of-the-art

The energy improvement achieved by the proposed method is compared with three different scenarios:

1. **Offline methods**: No runtime MEP adaptation scheme is used (i.e., one predefined supply voltage is applied to all chips) [235],

2. **Runtime MEP adaptation methods**: A runtime V_{dd} adaptation scheme is applied as in [78, 195, 197] and V_{dd} is adapted to the running workload and environmental conditions,

3. **Golden scenario**: Supply voltage is always set to the MEP.

In order to be fair, we assume that the overhead of the voltage regulator is the same for all methods, hence, we only compare the amount of energy-efficiency achieved based on the accuracy of the V_{dd}^{MEP} finding methods. We calculate the energy-efficiency of the above scenarios and compare them to our method:

- For the offline methods, we determine V_{dd} during design time as the average MEP of all $(S, U, \boldsymbol{T})$ combinations in the training dataset.
- For the runtime adaptation methods, we make a random deviation from the MEP as the *uniform distribution* in range $[-res/2, +res/2]$ where res is the resolution of the runtime MEP finding method.
- For our proposed method, we obtain V_{dd} from our machine-learning model.

Tables 5.8 and 5.9 compare the normalized energy-efficiency of the proposed method to the state-of-the-art at different temperatures and power modes, respectively. The results are presented as the *additional energy overhead compared to the golden scenario*; therefore, lower values are better. As reported, The energy consumption of the offline methods is on average 83% larger than the golden scenario, because the supply voltage is not adapted to the runtime variations (workload and temperature). Our proposed method and the runtime MEP adaptation method in [197] have the lowest overall energy overhead (less than 0.1%), because of high prediction accuracy. Our proposed method outperforms other runtime adaptation methods that have resolutions larger than 10mV (presented in [78, 195]) because the prediction accuracy of the proposed method is much better than those runtime adaptation techniques. In contrast, the runtime adaptation methods suffer from a uniform error distribution.

Table 5.10 compares the features and capabilities of all methods. Despite all hardware-based runtime adaptation methods ([78, 195, 197]), our proposed method comes with no hardware overhead or extra power-management design complexity, as it operates based on the MEP prediction at firmware-level. Another important advantage of our proposed method is that it is implemented in software. As a result, it can easily incorporate voltage scaling constraints, such as those imposed by the running application performance or reliability requirements. However, hardware-based runtime adaptation schemes [78, 195, 197] search for the absolute MEP by adjusting V_{dd} and measuring the energy-efficiency. Hence, they may pick a V_{dd} which is not suitable for some components in terms of reliability (leading to failures). In addition, they might miss the performance targets of the running application since they do not consider such constraints into account.

In practice, some input parameters of the proposed machine-learning model such as temperature, utilization of the components, and process corners are subject to inaccuracies. This will lead to inaccuracy in the predicted MEP; however, the impact of these inaccuracies are not significant as long as the inaccuracy in the input parameters is not too large. This can be attributed to the nature of the proposed machine-learning model in which the prediction is done with the help of many weak regression trees.

Runtime prediction overheads

The implemented machine-learning method is limited to have only 50 regression trees with a maximum depth of 5, i.e., $k = 5, m = 50$. Although this may reduce the accuracy of the

Table 5.8: Normalized energy overhead† of the proposed method and related work at different temperatures

Method	Offline methods [235]	Runtime MEP adaptation V_{dd}^{MEP} resolution:		**Proposed method**
		20mV [195]	10mV [197]	
T=-25°C	49%	0.9%	0.1%	0.1%
T=25°C	38%	0.4%	0.1%	0.1%
T=75°C	163%	0.6%	0.1%	0.1%
overall	83%	0.6%	0.1%	0.1%

† Compared to the **golden scenario** when the circuit is always operating at most the energy-efficient supply voltage. *Lower values are better.*

Table 5.9: Normalized energy overhead at different power modes

	Energy overhead (%) (different power modes)				
Power Mode (active components)	Leon3mp	Leon3mp ZigBee	Leon3mp FFT	Leon3mp ZigBee AES	Leon3mp AES
Runtime MEP [197]	0.1%	0.1%	0.1%	0.1%	0.1%
Proposed method	0.2%	0.2%	0.1%	0.1%	0.2%

prediction to some extent, it can limit the number of operations required for predicting the supply voltage to 200 comparison and 49 addition operations. Considering memory address calculation and load operations, an efficient implementation of the method can evaluate each tree with $5k+4$ operations. Therefore, the optimum V_{dd} can be calculated with approximately 1500 operations on a typical SoC firmware. A trained SGBRT model containing m regression trees with depth k has at most $m*(2^k-1)$ nodes and leaves. Therefore, the overall memory footprint for storing the parameters of our model is about 6KB, which is a reasonable value for common IoT devices.

Table 5.10: Comparison of the proposed method and related work in terms of the features and the capabilities

	Offline methods [235]	Runtime MEP adaptation methods [78, 195, 197]	**Proposed method**
Implemented at	Design-time	Hardware	Firmware
Implementation overheads	-	Extra circuitry (hardware) required	Software
Can adapt to runtime variations?	NO	YES	YES
Performance constraints can be applied?	NO (could lead to application errors)		YES
Reliability constraints can be applied?	YES pessimistic margins	NO (always seeking for the best energy-efficiency)	YES

5.5 Summary

Variability sources such as process variation, workload variation, and temperature affect the Minimum Energy Point (MEP) of circuits operating in the NTV region. For maximum energy-efficiency, it is required to determine the MEP and operate the circuit at the MEP (voltage and frequency). We demonstrated that the temperature and process variations are the dominant factors in determining the MEP of NTC circuits, whereas the energy-efficiency of a SoC and the corresponding optimum supply voltage is mainly dependent on runtime parameters including the operating temperature, power-gating states of the SoC internal components, their utilization, as well as performance requirements of the running application.

Accordingly, we proposed a machine-learning based techniques for tuning the MEP of NTC circuits at runtime using a calibrated LUT, generated during post-fabrication tests, to account for both process and temperature variations. Therefore, the best operating point leading to the maximum energy-efficiency is found considering the process variation on a per-chip basis, and temperature variation at runtime. For SoCs, the LUT is calibrated to also consider power-gating states of the SoC internal components and their utilization.

The hardware implementation overhead of the proposed method is minimal compared to conventional closed-loop supply voltage tuning methods due to the elimination of embedded energy measurement and voltage tuning circuitry. Monte-Carlo analysis shows that the proposed method can effectively track the MEP of NTC circuits to achieve near-optimal energy-efficiency. A firmware level implementation of the proposed method can predict the MEP with high accuracy, and can easily incorporate performance and reliability constraints into voltage prediction flow.

6 Selective flip-flop optimization for circuit reliability

Some NTC circuits are required to operate in a wide supply voltage range in order to achieve both energy efficiency, by operating in the NTV region, or satisfy timing constraints, by operating in the super-threshold voltage region. When utilized in the super-threshold region, runtime variability sources such as aging and supply voltage fluctuation affect both timing and functionality of the circuit. Therefore, it is important to address these effects, as the overall reliability of the circuit is threatened by them.

This chapter proposes a selective flip-flop optimization method [9, 10], in which the timing and reliability of the VLSI circuits are improved by optimizing the timing critical components under severe impact of runtime variations. As flip-flops are vulnerable to aging and supply voltage fluctuation, it is necessary to address these reliability issues in order to improve the overall system lifetime. In the proposed method, we first extend the standard cell libraries by adding optimized versions of the flip-flops designed for better resiliency against severe BTI impact and/or supply voltage fluctuation. Then, we optimize the VLSI circuit by replacing the aging-critical and voltage-drop critical flip-flops of the circuit with optimized versions to improve the timing and reliability of the entire circuit in a cost-effective way. Simulation results show that incorporating the optimized flip-flops in a processor can prolong the circuit lifetime by 36.9%, which translates into better reliability.

This chapter is organized as follows. Section 6.1 introduces wide-voltage operation reliability issues and motivates the proposed selective flip-flop optimization approach. The impacts of runtime variations on flip-flops are explained in Section 6.2. Consequently, Section 6.3 presents cell-level optimization of the flip-flops. The proposed selective flip-flop optimization methodology is described in Section 6.4, and optimization results are discussed in Section 6.5. Finally, Section 6.7 concludes the chapter.

6.1 Introduction, motivation, and contributions

VLSI circuits are influenced by several sources of process and runtime variabilities [39]. Among them, supply voltage fluctuation and transistor aging due to BTI are the most important factors [275–277]. They degrade the performance of VLSI circuits by increasing the delay, and consequently deteriorate lifetime.

The impacts of both voltage-drop and aging are significant on sequential elements such as flip-flops and latches. Due to particular aspects of flip-flops, such as the internal feedback structure, degradation of the transistors of a flip-flop as well as supply voltage fluctuation may lead to serious timing degradation or even functional failure (inability to capture the input independent of timing) [278]. Furthermore, many flip-flops are on the critical paths of a circuit because logic synthesis tools balance the delays of circuit paths to achieve the best performance, area, and power. Therefore, it is necessary to employ design-time mitigation techniques to consider and control such gradual degradation, e.g. by adding appropriate timing margins (*aging and voltage-drop guardband*) [279, 280].

Our analysis shows that in a typical digital design such as a microprocessor, based on the functionality of different components, some flip-flops operate under static or near-static BTI

stress, irrespective of the workload. These flip-flops experience large timing degradation because the flip-flop input Signal Probability (SP) is very close to 0.0 or 1.0. Being subject to severe BTI stress, the aforementioned flip-flops degrade faster, imposing a large *aging guardband* to the entire circuit. Flip-flops also experience a large temporally localized voltage-drop, because they are synchronized with the clock edge and supposedly operate at the same time (at clock edge), hence, drawing substantial current leading to a significant voltage-drop over PDN [97]. Moreover, recent studies have shown that the voltage-drop impact gets more severe by technology scaling [98, 277, 281]. Therefore, in a conventional design flow, costly *voltage-drop timing guardband* is considered for reliable circuit operation [97].

In this chapter, we explore methods to improve circuit reliability by addressing the timing degradation of flip-flops under severe aging[1] and voltage-drop, i.e. *selective flip-flop optimization*. The idea is to find timing-critical flip-flops under high aging and/or voltage-drop impact, and selectively re-optimize them for operating under such stress by appropriately sizing their transistors. This effectively improves the reliability and lifetime of circuits without imposing much overhead, because these flip-flops constitute a small portion of all flip-flops.

Simulation results obtained by applying the proposed method to a processor show that the flip-flops optimized with the proposed method exhibit much less delay degradation, while imposing less than 0.1% leakage power overhead to the processor. As a result, the required timing guardband of the processor using the proposed method is significantly less compared to the original processor. Therefore, given a specific clock period, the optimized processor design with the proposed method has 36.9% longer lifetime and better reliability compared to the original processor design.

6.2 Variability impact on flip-flops

6.2.1 Flip-flop timing

Flip-flop timing metrics such as setup-time (U), hold-time (H), clock-to-q (D_{CQ}), and data-to-q (D_{DQ}) are well discussed in [282, 283]. When the setup-time is large enough, the clock-to-q value is almost constant, but further reduction of the setup-time will increase the clock-to-q value monotonously until a value after which the flip-flop is unable to capture and latch the input [282]. Based on this, the *optimum setup-time* is defined as the setup-time value which causes the clock-to-q value to increase by 10% from its minimum value [284]. Moreover, each flip-flop has two internal paths; one for transferring the input state 'zero' to the output i.e. High-to-Low (HL) input transition, and the other for transferring the input state 'one' to the output i.e. Low-to-High (LH) input transition. Basically, the timing parameters for these two internal paths can be different [278] as shown in Figure 6.1, meaning that there are two sets of timing parameters for internal LH and HL paths of a flip-flop:

$$\begin{aligned} &\{U_{LH}, D_{CQ_{LH}}, D_{DQ_{LH}}\} && \text{for LH transition,} \\ &\{U_{HL}, D_{CQ_{HL}}, D_{DQ_{HL}}\} && \text{for HL transition.} \end{aligned}$$

Flip-flop delay should be defined such that the correct functionality of the flip-flop will be guaranteed, disregard of the transition. Therefore, we define the flip-flop delay as the *summation of the worst setup-time and the worst clock-to-q of both transitions* as shown in Figure 6.1.

$$delay = max\{U_{LH}, U_{HL}\} + max\{D_{CQ_{LH}}, D_{CQ_{HL}}\}. \tag{6.1}$$

This guarantees that in both transitions the input signal is correctly captured and propagated to the flip-flop output.

[1] We consider the impact of NBTI on PMOS transistors, and PBTI on NMOS transistors.

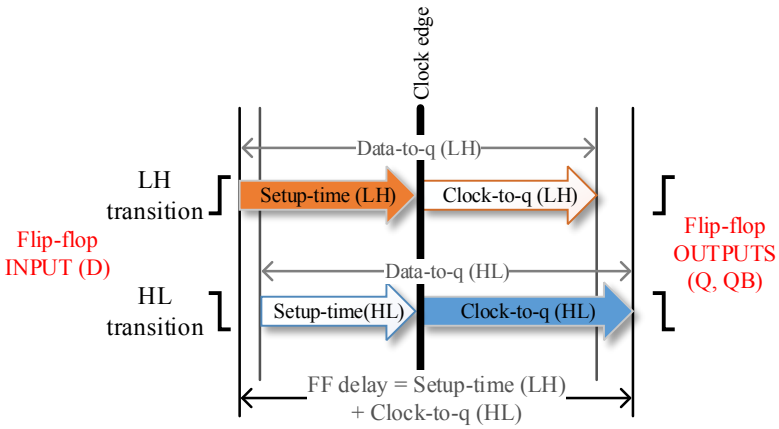
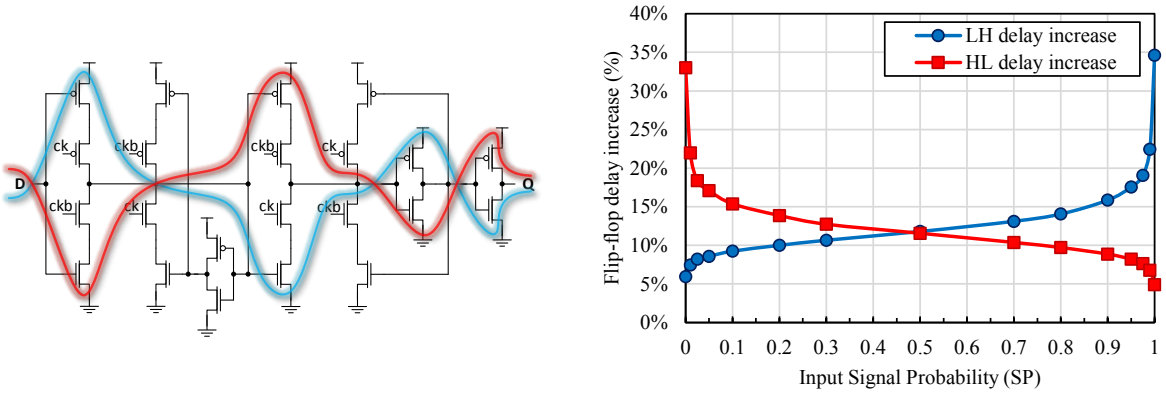


Figure 6.1: Different flip-flop timing parameters. The correct functionality is guaranteed by considering the flip-flop delay as illustrated.



(a) Internal LH (red), and HL (blue) paths of a C2MOS flip-flop [282]

(b) Delay of internal LH and HL paths of an aged C2MOS flip-flop after 5 years

Figure 6.2: Separate internal LH/HL paths of the flip-flop (a), and delay of internal LH/HL paths of an aged C2MOS flip-flop (optimized for PDP in the fresh state) for different input SPs (b).

6.2.2 Runtime variation impacts on flip-flops

Several parameters such as supply voltage, workload, and temperature affect the performance of flip-flops in a circuit. Parameters such as temperature and supply voltage affect all the transistors of a flip-flop in the same way, whereas the impact of the input SP is different for the transistors of a flip-flop [135]. This results in an asymmetric aging of transistors according to their stress duty cycles. Therefore, the delay degradation of internal LH and HL paths inside an aged flip-flop depends on the input SP [278].

In the C2MOS flip-flop[2] depicted in Figure 6.2a, the internal LH and HL paths consist of two separate groups of transistors, which makes the aging of these two paths independent according to the input SP. Figure 6.2b illustrates the delay of LH and HL transitions of an aged C2MOS flip-flop [282] for different input SPs. When the flip-flop is aged under input $SP = 0.0$ (SP0), the worst delay degradation happens on the flip-flop HL path, however, the delay of the flip-flop LH path is only slightly affected. On the other hand, an aging under input $SP = 1.0$ (SP1) greatly degrades the delay of the flip-flop LH path while slightly affecting the

[2]A C2MOS flip-flop design is a master-slave flip-flop built of two connected C2MOS latches. It is one of the commonly used flip-flops in modern processor designs [282].

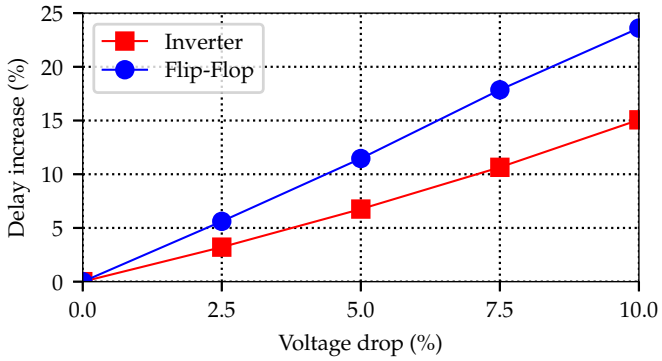


Figure 6.3: Comparison between the voltage-drop induced delay degradation of a flip-flop and an inverter, which are aged under same condition (Aging under SP1 for 5 years).

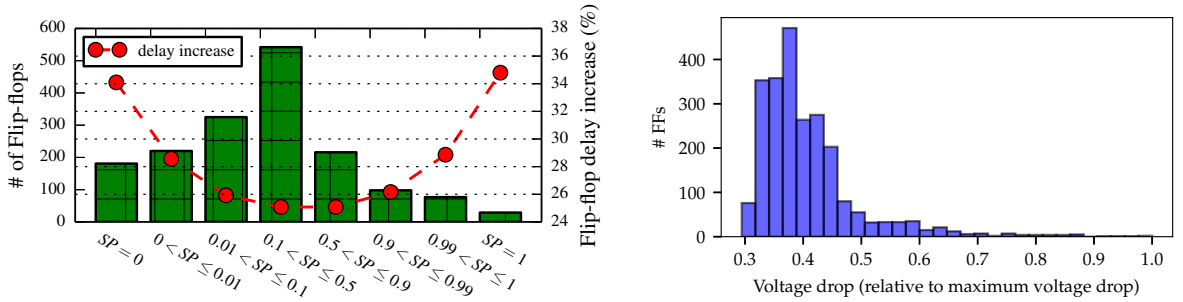
delay of the flip-flop HL path. For moderate aging condition, i.e. $0.1 < SP < 0.9$, the delay degradation of both LH and HL paths is moderate. The reason is that under SP0 and SP1 conditions, S-BTI asymmetrically alters the threshold voltages leading to unbalanced aging of LH and HL paths of the flip-flop as the stress duty cycle of some transistors is 1.0, i.e., always under BTI stress. However, in moderate aging condition, the transistors can partially recover as the stress duty cycle is less than 1.0.

The impact of supply voltage fluctuation on the flip-flops of a circuit depends on the workload variation and dynamic power consumption of the circuit. Therefore, each flip-flop may experience a specific amount of voltage-drop. A voltage-drop causes performance degradation of the flip-flops, which is typically larger than the degradation of simpler combinational gates in the standard cell library. Figure 6.3 compares the impact of a voltage-drop up to 10% on the delay of an aged flip-flop and an aged inverter. Compared to a no-voltage-drop condition, the delay of the flip-flop increases by 23.6% whereas the delay of the inverter is increased by 15%.

Moreover, the flip-flops of a circuit generally experience higher amount of voltage-drop compared to combinational gates [285]. As a result of temporally localized switching of flip-flops at the positive (or negative) edge of clock signal, the instantaneous current drawn from PDN at the synchronized clock edge is comparatively high. This leads to high voltage-drop at the clock-edge, when the flip-flops are processing their input signals. This peak current consumption is damped over the rest of the clock period, when the combinational cells are active. Therefore, in this work we focus on dealing with the impact of voltage fluctuation on the flip-flops.

Temporal and spatial temperature variations can also affect the circuit performance. The temporal temperature change could be rather high and has been the subject of research since it affects the reliability of the VLSI circuits. It is demonstrated in [61] that the circuit performance can be changed by up to 10% for $110°C$ temperature variation. Therefore, in order to meet the reliability constraints, the circuit timing should be adjusted according to the worst temperature corner, which is typically at high temperature. On-chip spatial temperature gradient puts different stress on circuit components across a chip. The amount of on-chip spatial temperature difference (only on cores) based on simulation [102, 103], sensor measurements [104], and thermal camera [103] is reported to be up to $\sim 30°C$. Since the delay change is approximately 4% for every $40°C$ [61, 286], the overall difference between the delay degradation of core flip-flops due to such spatial temperature gradient is expected to be less than 3%, and hence, much smaller compared to voltage-drop variation [287].

The combined impact of voltage-drop and aging significantly degrades the performance of flip-flops. As an example, the delay of a fresh flip-flop optimized with balanced HL/LH delay increases from $98.5ps$ to $165.7ps$ due to the combined impact of voltage-drop (10%) and S-



(a) The average flip-flop SPs during the execution of some MiBench workloads on Leon3, and the corresponding delay degradation in 5 years. In order to be fair in the analysis, the flip-flops which are not exercised by the workloads are excluded from this analysis.

(b) The distribution of voltage-drop on the Leon3 flip-flops. The voltage-drop values are normalized to the maximum voltage-drop across the processor.

Figure 6.4: (a) Input signal probabilities and (b) voltage-drop analysis of Leon3 flip-flops executing MiBench Workloads

BTI (5-years under SP0). This is equivalent to 68% delay increase. If such a flip-flop is in a critical path of the circuit, a large timing guardband is required for timing closure considering the reliability constraints. Therefore, it is necessary to find such flip-flops at design-time and optimize them for operating under such conditions.

6.2.3 Significance of flip-flops in circuit reliability

In a properly designed circuit, the timing of circuit paths are balanced during the synthesis process. Therefore, many flip-flops are *timing-critical* as they lie on the circuit critical paths. Studies [9, 285] have shown that in VLSI circuits, some flip-flops are under severe static BTI leading to a large timing degradation over time. Furthermore, the impact of voltage-drop on flip-flops could be very high as a result of localized power consumption at a specific time (e.g. positive clock-edge) or at a specific location on the circuit layout.

The large impact of S-BTI and voltage-drop on flip-flops has a significant impact on the reliability of a circuit when such flip-flops are timing-critical. In order to investigate the likelihood of having such a scenario in a typical digital design, we use the flow presented in Section 6.4 to extract the voltage-drop and the aging of the Leon3 flip-flops by executing six MiBench workloads [95] namely stringsearch, qsort, basicmath, bitcount, fft, and crc32 on Leon3 processor [233]. In order to be fair, we excluded the flip-flops belonging to the parts which are not exercised by the employed workloads such as interrupt handler, timers, and UART controller. The synthesized netlist of the Leon3 processor has 2,352 flip-flops, but the results demonstrated in this section contain only 1,686 flip-flops belonging to the parts which are exercised by all employed workloads.

Figure 6.4a demonstrates the input SP distribution of the aforementioned 1,686 flip-flops. The results show that 181 flip-flops always experience input SP0, whereas 29 flip-flops are under input SP1. Our analysis shows that the flip-flops with such behavior typically belong to either the error checking and exception handling registers or higher bits of address registers which are constant due to temporal and spatial locality of the executed instructions. Besides, the SP of a considerable number of flip-flops is very close to either 0.0 or 1.0. Please note that the results reported in Figure 6.4a are the average of six employed workloads, and hence, the flip-flops with SP=0 or SP=1 have such SP across all executed workloads. Similar experiment has been carried out in [288] to study the impact of workload in real systems, which shows that some flip-flops are always under S-BTI across different workloads.

Figure 6.4b shows the distribution of the maximum voltage-drop impacting the flip-flops of Leon3 processor compared to the peak voltage-drop across all the executed workloads. Please note that it is necessary to consider the maximum voltage-drop over the execution of all workloads, because it eventually impacts the flip-flop characteristics. A significant portion of flip-flops experience on average 41% of the maximum amount of voltage-drop, however, there are flip-flops at the right side tail of the distribution which experience large voltage-drop comparable to the maximum voltage-drop in the circuit.

According to the observations in Figure 6.4, there are flip-flops experiencing both S-BTI and high voltage-drop which leads to high degradation. If such flip-flops are on a critical path of the processor (i.e. timing-critical flip-flops), the degradation of the flip-flops should be reflected in the timing guardband of the circuit. Timing-critical flip-flops can be categorized into different groups base on the impact of voltage-drop and aging as follows:

- low voltage-drop and low aging,
- low voltage-drop but S-BTI aging (SP0/SP1)*
- high voltage-drop but typical aging*
- high voltage-drop and S-BTI aging (SP0/SP1)*

Therefore, we propose to generate flip-flops specifically optimized for such high-degradation conditions (marked by *) and add them to the standard cell library. Using the proposed flow in Section 6.4, we determine such high-degradation and timing-critical flip-flops and replace them with the optimized versions to improve the timing and reliability of the circuit.

6.3 Reliability-aware flip-flop design

In a typical reliability-aware circuit design, one should consider the delay of the elements under variation impacts to ensure the correct functionality of the circuit during the expected lifetime. Therefore, higher delay degradation of timing-critical flip-flops imposes a large timing guardband. In our proposed methodology, we create optimized versions of the flip-flops for different stress conditions based on aging and voltage fluctuations, and use these optimized versions only when a flip-flop is timing-critical and subject to such stress conditions to avoid unnecessary over design. This means that in the cell library, we add the following resilient versions of the flip-flops:

- Aging-resilient flip-flops, optimized for different aging corners (SP0 and SP1),
- Voltage-drop resilient flip-flops, optimized to have lower performance degradation under voltage fluctuation,
- Aging and voltage-drop resilient flip-flops.

6.3.1 Aging resilient flip-flop design

When the fresh delays of internal paths of a flip-flop (i.e., LH and HL paths) are designed to be similar (depicted as solid lines in Figure 6.5), the internal path with higher degradation rate eventually becomes dominant and determines the total delay of the flip-flop. In this case, a significant aging in flip-flop characteristics is observed over time (corresponding to the internal path with higher degradation). On the other hand, if the internal path with higher degradation rate is initially faster (by design) than the internal path with lower degradation

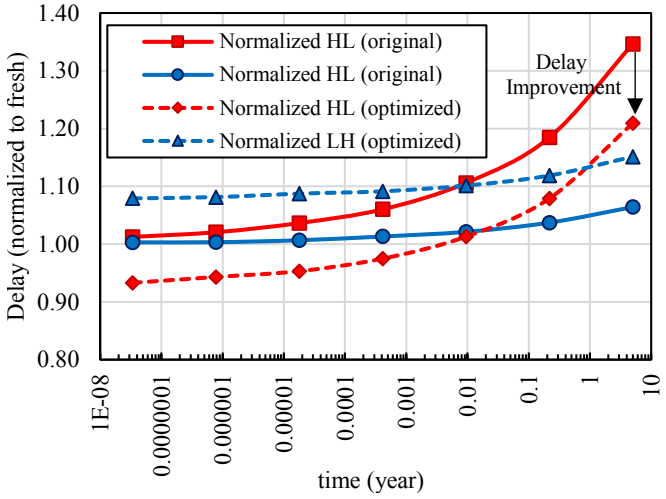


Figure 6.5: Delay of a C2MOS flip-flop which is aged under SP=0 over 5 years for LH/HL transitions, compared to the flip-flop optimized for SP=0 showing how the unbalanced aging of internal LH/HL paths worsens the degradation in original flip-flop.

rate, the dominant internal path would be the slower one, and hence the higher degradation rate of the faster internal path is masked. Consequently, the overall aging of the flip-flop would be rather small. The delay of the optimized flip-flop, shown in Figure 6.5 by dashed lines, exhibits such characteristics. The post-aging delay of the optimized flip-flop would increase by ∼10 ps, which is much lower than ∼40 ps increase in the delay of the original flip-flop. We exploit this method for designing aging resilient flip-flops.

In order to decrease the overall BTI-induced aging inflicted to a flip-flop, our proposed method balances the delay of internal HL and LH paths of the flip-flop for *post-aging state of the flip-flop*, by resizing the transistors of internal HL and LH paths. In other words, the proposed method increases the fresh delay ($t = 0$) of the flip-flop internal path which has lower degradation rate in order to compensate the overall degradation of the flip-flop after aging. Although the fresh delay of the optimized flip-flop might be slightly larger compared to the fresh delay of the original flip-flop, the overall delay of the optimized flip-flop considering the aging-induced timing margin would be smaller than those of the original flip-flop since the aging rate is much smaller.

Please note that this method reduces the degradation for a given SP, but inevitably worsens the aging at the other corners of SP. For example, if we optimize the flip-flop for SP0, the degradation would be much higher if the optimized flip-flop operates at SP1. Nevertheless, these flip-flops under S-BTI will not operate at other SP corners, because their SP is determined by the circuit structure and functionality. Therefore, we only optimize for the given SP corner. This means that we intentionally sacrifice other corners, which never occur due to the specific functionality of the circuit, to gain a larger improvement.

6.3.2 Voltage-drop resilient flip-flop design

Other than aging, which affects each flip-flop transistor based on the input signal probability, a drop in the supply voltage of the flip-flop slows down all flip-flop transistors in the same way. However, a slight up-sizing of specific transistors can compensate the degradation in the flip-flop timing. Therefore, we evaluate the delay of the flip-flop when operating under the impact of voltage-drop, and optimize the flip-flop with the goal of improving the delay. Consequently, the optimized flip-flop would have better timing at the cost of higher power consumption.

6.3.3 Aging and voltage-drop resilient flip-flop design

The degradation in the flip-flop timing due to both S-BTI and voltage-drop is very large. Such timing degradation may not be effectively compensated by resizing the transistors within a flip-flop area without upsizing the entire flip-flop. Therefore, in addition to targeting for better timing under the impact of the aging and voltage-drop, we allow the optimization algorithm to increase the area of the flip-flop by a small percentage. Please note that an extra Engineering Change Order (ECO) might be needed to replace the original flip-flop with the optimized version in this case. However, since there exist only a few flip-flops under such degradation it would not be an issue to perform an ECO on placement.

6.3.4 Problem formulation for flip-flop resiliency optimization

The delay of a flip-flop under a specific working condition (including temperature, voltage, and input SP) can be presented as a function of the transistors' widths:

$$delay = f\,(W)\,,\quad W = [w_i], \tag{6.2}$$

where $[w_i]$ is a vector containing the width of flip-flop transistors. Here, *delay* is the delay (Data-to-q) of the flip-flop, according to Equation (6.1), under variation impact, which could be S-BTI stress, voltage-drop, or both depending on the optimization approach.

The delay function f is a complicated function of transistors' widths. Our experimental results for flip-flops with different sizing show that f cannot be presented with any general linear function. Therefore, we use Sequential Quadratic Programming (SQP) which is a non-linear programming technique [289]. In SQP, the problem is converted into quadratic sub-problems and solved in order to find a better sizing in each iteration. For this purpose, we follow an iterative approach in order to minimize the delay of Equation (6.2). Given an initial sizing, the delay function f is approximated with a quadratic function:

$$f(W) \approx f(W_0) + \bigtriangledown f(W_0)^T \cdot (W - W_0) + \frac{1}{2}(W - W_0)^T \cdot H_f(W_0) \cdot (W - W_0), \tag{6.3}$$

where $\bigtriangledown f(W)$ and $H_f(W)$ are the gradient and the Hessian of the delay function f, respectively. Minimizing the quadratic approximation of Equation (6.3), with respect to some constraints, which will be discussed later in this section, yields an optimized transistor sizing. Thereafter, the obtained sizing is used as the initial sizing, and a new iteration is launched. This cycle continues until the optimization reaches the required precision, i.e. the difference between the optimized delays of two consecutive iterations becomes smaller than a predefined threshold ϵ_{delay}. Therefore, the solver continues by checking the precision of the resulting delay:

$$|delay_{i-1} - delay_i| < \epsilon_{delay} \tag{6.4}$$

where $delay_i$ represents the delay of i$^{\text{th}}$ iteration.

Another reason to use the quadratic approximation is that the optimum result of a linear problem always lies on the boundaries, while the optimum result of a quadratic problem can be any point within the boundaries as well as the boundaries themselves. In Section 6.5, we demonstrate that the optimum result does not necessarily lie on the boundaries, and hence a non-linear programming technique is needed to find a better result. Table 6.1 summarizes the optimization problem.

Several constraints are applied to the optimization problem, relating to transistors size, flip-flop area, and leakage. The first constraint shown in Table 6.1 limits the minimum size of transistors. The second constraint limits the area of the optimized flip-flop. In case of optimizing for S-BTI or voltage-drop, we consider $\lambda = 0$ to keep the flip-flop area within the

Table 6.1: Flip-flop Optimization Method Summary

Parameters	$W = (w_1, \ldots, w_n)$	
Initial guess	W_0 = optimized W for PDP (fresh)	
Constraints	$w_i \geq w_{min}$ $\sum_1^n w_i \leq (1+\lambda) \sum W_0$ power$(W) \leq (1+\beta)$ *power*(W_0)	
Target	minimize: $delay = f(W)$	
Constants	w_{min}	minimum size
	λ	acceptable excessive area
	β	acceptable excessive leakage

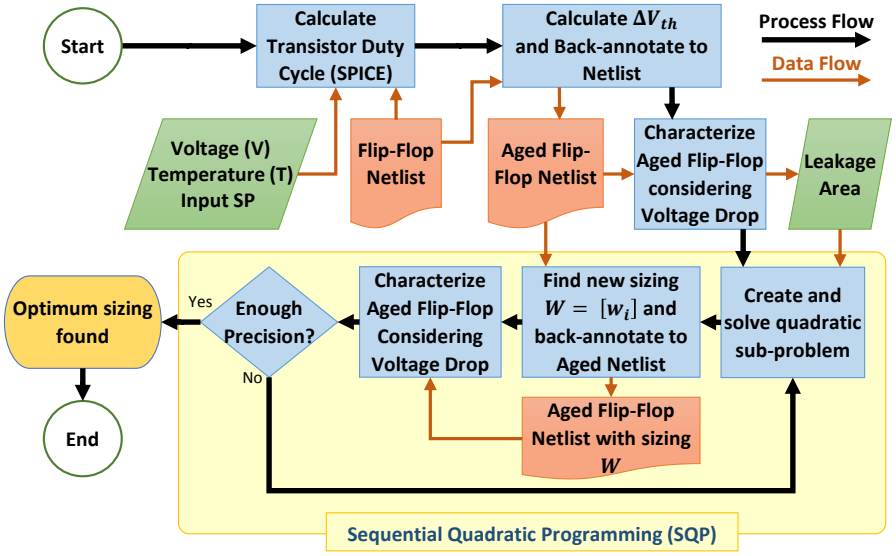


Figure 6.6: Overall flow to find the optimum flip-flop sizing for under S-BTI stress and voltage-drop at a specific working corner (voltage, temperature).

area of the original flip-flop which also facilitates keeping the aspect ratio almost equal to the aspect ratio of the original flip-flop. This way, the optimized flip-flop can easily replace the original flip-flop without any layout modifications at the circuit level. This is achieved by limiting the summation of transistor widths w_i. However, for flip-flops under S-BTI and voltage-drop, we assume a $\lambda > 0$ value to compensate the delay degradation better. The third constraint sets an upper limit for the excessive leakage of the flip-flop by parameter β. This constraint is applied to the optimization problem to limit the leakage power of the optimized flip-flops within an acceptable range. The initial guess of optimization W_0 is the optimum sizing for minimum PDP in the fresh state.

6.3.5 Reliability-aware flip-flop optimization flow

Figure 6.6 presents our proposed reliability-aware optimization flow. For a given input SP, the SP of all transistors are once calculated using SPICE simulations. Afterwards, based on the extracted SP for transistors and the operating corner of the flip-flop (temperature, supply voltage, etc.), the BTI-induced threshold voltage shifts of all transistors (ΔV_{th}) are obtained.

Then, the ΔV_{th} values are back annotated into the original flip-flop SPICE netlist, and the SPICE netlist of aged flip-flop is generated.

In each SQP iteration, the quadratic sub-problems are created and solved to generate further improved flip-flop sizing. Subsequently, the new sizing is back-annotated into the aged flip-flop netlist extracted before, and a new aged flip-flop with the given sizing is generated. Then, Cadence Virtuoso Liberate [260] is used to characterize the new flip-flop and extract its delay and power consumption. When the improvement is small enough and the condition in Equation (6.4) is met, the SQP method terminates and returns the last sizing as the optimum solution for the problem.

As the process is executed at a specific supply voltage (V_{dd}), it can inherently be used to optimize for a voltage-drop as well, when the given supply voltage includes the impact of the voltage-drop. We can also create voltage-drop resilient version of a flip-flop for typical aging, by considering input SP of 0.5. Therefore, we execute the flow presented in Figure 6.6 for these conditions in order to create variation-resilient versions of the flip-flop, assuming a supply voltage of V_{dd} and a maximum voltage-drop of $R\%$:

	Supply Voltage (V)	Aging Condition
Aging	V_{dd}	S-BTI (SP0, SP1)
voltage-drop:	$(1-\frac{R}{100})V_{dd}$	Typical aging (SP=0.5)
Aging and voltage-drop:	$(1-\frac{R}{100})V_{dd}$	S-BTI (SP0, SP1)

After optimization process, it is necessary to re-characterize the flip-flops for different supply voltage ($(1-\frac{R}{100})V_{dd}$ to V_{dd}) and aging conditions ($0 \leq SP \leq 1$). The characterization results are then used to obtain overall circuit timing under supply voltage fluctuation and aging impacts.

6.4 Selective flip-flop optimization

This section explains how the optimized flip-flops in Section 6.3 can be employed to improve the reliability of a circuit. The idea is to find the flip-flops affected by S-BTI and/or voltage-drop impacts which are also influential on circuit reliability, i.e. the timing-critical flip-flops, and replace them with the optimized versions. The reason for this *selective flip-flop optimization* is that the reliability-aware flip-flop optimization is costly in terms of leakage overhead per flip-flop. Therefore, flip-flop replacement should be done only for the timing-critical flip-flops which experience S-BTI and/or large voltage-drop to be cost effective. Since they constitute a small subset of the all flip-flops in the design, the proposed method is able to reduce the overall timing guardband in a cost-effective way.

The overall flow of the proposed selective flip-flop optimization methodology is presented in Figure 6.7. The flow uses the results of the Synthesis and Place & Route steps of a VLSI design flow and is composed of *I) Aging and Voltage-Drop Analysis* and *II) Selective Flip-flop Replacement* steps. The optimization flow updates the gate-level netlist and the circuit layout to improve the reliability of the circuit under voltage-drop and aging impacts. The outputs of the optimization method can be further used in the rest of the VLSI design flow. Therefore, the proposed method is transparent to the VLSI design flow and can be easily integrated into it.

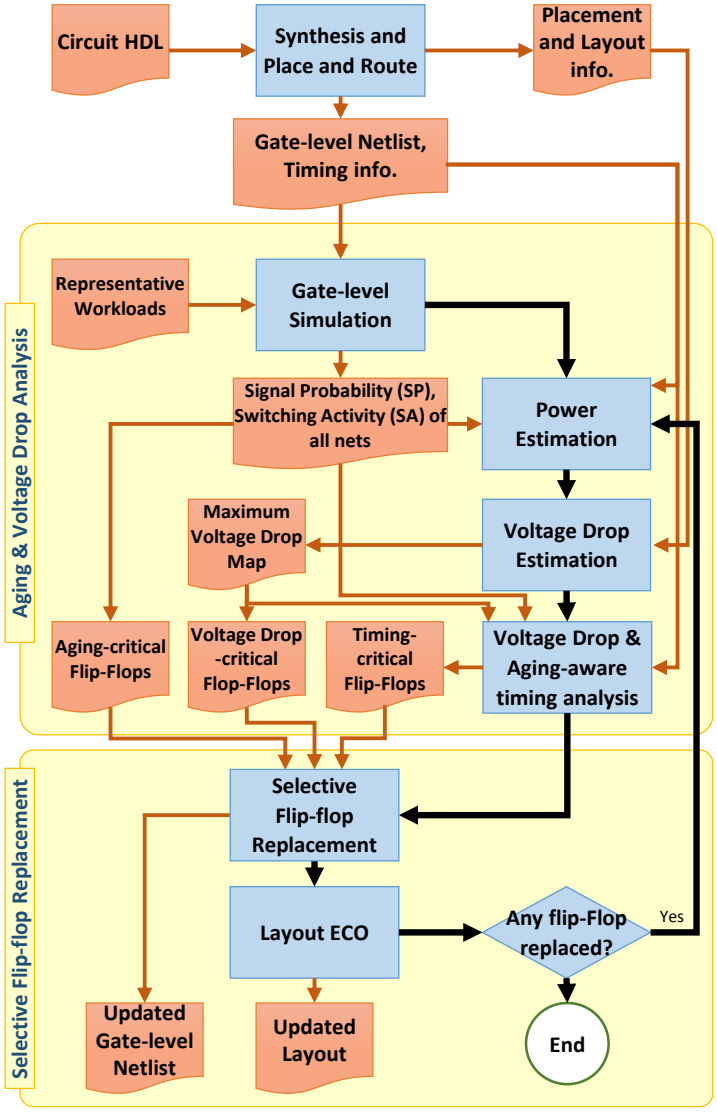


Figure 6.7: Circuit optimization flow using the proposed selective flip-flop optimization method.

6.4.1 Aging and voltage-drop analysis

In this step, the results of Synthesis and Place & Route steps of the VLSI design flow are used to discover the flip-flops which are *aging-critical*, *voltage-drop critical*, and *timing-critical*.

Aging-critical flip-flops are those flip-flops which experience large impact of aging, i.e. flip-flops under S-BTI. To find the aging-critical flip-flops we need to extract the SP of the flip-flops. Therefore, we perform a gate-level simulation running some representative workloads. The representative workloads are pieces of workloads which are typically executed on the circuit. The result of the gate-level simulation is the Voltage Change Dump (VCD) of all nets inside the circuit. Based on this information we can collect SP of all flip-flops and determine the aging-critical flip-flops.

Dynamic power profiles of circuit components can be extracted from the VCD reports. We estimate the dynamic voltage-drop in the circuit based on the power profiles and the layout and packaging of the circuit. This accounts for the resistive and inductive components of the voltage fluctuation. We generate a *voltage-drop map* of the circuit by evaluating the *maximum voltage-drop* of each cell (gates, flip-flops, etc.) over the time and over different workloads. As a

result, we find the maximum amount of voltage-drop that each flip-flop experiences over time. Accordingly, the flip-flops which experience a large amount of voltage-drop are extracted.

Furthermore, the gate-level simulation results are used to perform a *voltage-drop and aging-aware timing analysis* which obtains the delay of circuit paths under variability impacts. We extended the aging-aware timing analysis in [290] by considering voltage-drop related information. This is done by characterizing the cells at two different supply voltages: the nominal V_{dd} and the supply voltage considering the maximum drop $(1 - \frac{R}{100})V_{dd}$. Then, for each gate/flip-flop in the gate-level netlist, based on the amount of voltage-drop on the gate, we perform a linear interpolation among the standard cell library entries for two supply voltages and find the corresponding timing information. The linear interpolation is a valid method under the assumption of limited change in the supply voltage, as shown in Figure 6.3. For a more aggressive voltage fluctuation, it could be necessary to characterize the standard cell libraries for a few intermediate supply voltage values and employ a PCHIP method. Accordingly, we find the timing-critical flip-flops, which are parts of the critical and near-critical paths of the circuit considering the impact of variations.

6.4.2 Selective flip-flop replacement

In the Selective Flip-flop Replacement step, we replace the flip-flops which are timing-critical, aging-critical, or voltage-drop-critical with their optimized counterparts for such aging and/or voltage-drop conditions. Although a small portion of the flip-flops are replaced during the flip-flop replacement process, the circuit layout, timing, and power properties change since the replaced flip-flops are timing-critical and may have different area and power characteristics. Therefore, the proposed flip-flop replacement is an iterative process which replaces a number of flip-flops with the optimized versions in each iteration. The iterative process continues until no flip-flop needs to be replaced by an optimized version anymore.

In iteration i of the method, we assume that the circuit delay is D^i based on timing analysis results, and d_j^i is the maximum delay of the paths terminating at flip-flop j (including the delay of the flip-flop as well). Therefore, in each iteration:

1. we choose the timing-critical flip-flops with a timing slack value of less than $k\%$ of the circuit delay, i.e. when:

$$d_j^i \geq (1 - \frac{k}{100})D^i, \quad j : \text{index of flip-flops.}$$

2. Among these flip-flops, those which are also included in the aging-critical and/or voltage-drop-critical flip-flops, are replaced with the optimized versions.

3. A trial voltage-drop and aging-aware timing analysis is performed and the circuit delay (D^{i++}) is determined considering the replaced flip-flops.

4. We keep the optimized flip-flops only when the corresponding path delay of the flip-flops *before optimization* is larger than a percentage of the evaluated circuit delay (D^{i++}):

$$d_j^i > r \times D^{i++} \Longrightarrow FF_j \rightarrow FF_{j,opt} \tag{6.5}$$

   The rest of the updated flip-flops in this iteration are rolled back to the original versions. Please note that we also consider a ratio $r < 1$ into Equation (6.5) to compensate for the calculation errors due to simulation.

5. The layout and gate-level netlist of the circuit are updated. The layout is only updated if a cell with larger area is used (particularly applicable to the flip-flops under both aging and voltage-drop as explained in Section 6.3).

6. In case any flip-flop is replaced by an optimized version during this iteration, we need to start a new iteration because the timing and power specification of the circuit are modified. This is done by re-executing the aging and voltage-drop analysis, as explained in Section 6.4.1. The gate-level simulation, which is a time consuming process, does not need to be repeated as its results are not affected by the flip-flop replacement.

The above flow replaces minimum number of flip-flops with the optimized versions and impose minimum amount of overhead to the circuit. In our simulations the flow is terminated within a few iterations, since the changes in the circuit layout, power, and timing are not extensive.

6.5 Results and discussions

In this section, we evaluate the efficiency of the proposed selective flip-flop optimization based on simulation results.

6.5.1 Simulation Setup

We applied the method to several flip-flop topologies, namely C2MOS latch, Dynamic/Static Single Transistor Clocked latch (DSTC/SSTC), and Semi-Dynamic flip-flop (SDFF) [282]. The flip-flops are implemented using 45nm Bulk CMOS Predictive Technology Model (PTM) transistors [291]. All flip-flops are initially optimized for the minimum PDP in the fresh state (original design). The aging parameters of the model proposed in [126] are tuned so that the post-aging delay of a Fan-Out 4 (FO4) inverter increases by 10% at SP=0.5 over 5 years. For delay and leakage measurements, the output load of flip-flops is set to FO4, and the cells are characterized at room temperature and at different supply voltages, ranging from 80% to 100% of the nominal supply voltage of the technology node.

We used Leon3 processor as a case study for our proposed method. We used Nangate 45nm open cell library for combinational logic, and aging assumptions are the same as described at the beginning of this section. The processor is synthesized using Synopsys Design Compiler and placement and routing is done using Cadence EDI [227].

We executed various MiBench workloads on the synthesized Leon3 processor and extracted the VCD files. Based on the VCD files, the SP of each node of the synthesized circuit is calculated and the power consumption of the gates and flip-flops are calculated using Synopsys Power Compiler. The voltage-drop map of the processor is also extracted using VoltSpot tool [98], which is able to extract the voltage-drop caused by both resistive and inductive components.

Please note that the proposed technique is not restricted to a specific working condition or flip-flop topology. We proceed with presenting detailed results and analysis for a C2MOS flip-flop. Then, we discuss the results for other types of flip-flops concisely. Afterwards, the dependency of the improvement achieved by the proposed method to the excessive leakage will be investigated. At the end of this section, the impact of using optimized flip-flops on a Leon3 processor lifetime will be demonstrated.

6.5.2 Detailed optimization results of C2MOS flip-flop

We apply the proposed optimization flow presented in Section 6.3.5 (see Figure 6.6) to C2MOS flip-flop design to create optimized flip-flops for aging and voltage-drop resilience. In order to create the aging-resilient versions of the C2MOS flip-flop, we let the optimizer to consider designs with up to 25% more leakage compared to the original flip-flop by setting the coefficient β in Table 6.1 to 0.25. At this point, we limit the area of the flip-flop to the the area of the

original flip-flop, i.e. $\lambda = 0$. Please note that the total overhead of the leakage power for the entire circuit would be negligible since the number of optimized flip-flops in the design would be limited. For example, if according to Section 6.2.3, 12.45% of flip-flops are working under S-BTI, and the leakage overhead of an optimized flip-flop would be less than 25%, the leakage overhead imposed on the flip-flops would be *at most* 3.11% (much less overhead when considering the entire processor design). The aging and voltage-drop resilient version of the C2MOS flip-flop can be created by assuming an extra area up to 20% and more leakage overhead. For this, we assume $\lambda = 0.2, \beta = 1$. Using the extra area, the optimizer is able to find a better design for those flip-flops which are timing-critical and are under large impact of aging and voltage-drop. Since these flip-flops are very rare, but have significant impact on the overall processor lifetime and reliability, it is effective to spend more area for large reliability and lifetime gains.

Table 6.2 compares the characteristics of an original and optimized C2MOS flip-flop (such as setup-time (U), clock-to-q (D_{CQ}), data-to-q (D_{DQ}), delay, and leakage) in three different optimization scenarios:

Scenario 1 *post-aging PDP*, optimized for PDP in post-aging.

Scenario 2 The proposed method (optimized for aging), in which the flip-flop is optimized for aging resiliency, by minimizing its delay for post-aging. The acceptable excessive area and leakage are 0% and 25%, respectively ($\beta = 0.25, \lambda = 0$).

Scenario 3 The proposed method (optimized for aging+vdrop), in which the flip-flop is optimized for aging and voltage-drop resiliency, by minimizing its delay for post-aging and under voltage-drop impact. The acceptable excessive area and leakage are 20% and 100%, respectively ($\beta = 1, \lambda = 0.2$).

The optimization results in Table 6.2 are reported for "fresh" state (no aging or voltage-drop), for "aged" state (under S-BTI aging SP0 for 5 years), and for "aging+vdrop" state (when the flip-flop is aged under S-BTI for 5 years, and when the supply voltage is dropped by 10%). Setup-time, clock-to-q, and data-to-q values are presented for LH/HL transitions and the delay is calculated according to Section 6.2.1. The delay degradation is the *relative post-aging delay increase of a design compared to the fresh delay of the original design* (marked as bold in the table):

$$\text{delay degradation} = \frac{delay_{opt.,aged} - delay_{orig.,fresh}}{delay_{orig.,fresh}}. \tag{6.6}$$

Since the optimized flip-flop will replace the corresponding flip-flop in the design, the delay degradation is compared to the fresh delay of the original flip-flop in order to give a better understanding of how close the aged delay of the optimized flip-flop is to the fresh delay of the original design.

Basically, scenario 1 is similar to the methods proposed in many flip-flop optimization methods such as [11, 33] in the sense that they consider a multiplication of energy and delay (e.g, the PDP or the EDP) as the optimization target. Scenario 1 is able to effectively reduce the PDP by increasing the delay and reducing the leakage, but this may result in an unacceptable timing for S-BTI corners. Table 6.2 shows that due to not considering the flip-flop delay as the optimization target, the PDP methods cannot find the optimum aging-resilient sizing for S-BTI corners.

As presented, for the original flip-flop, the fresh delay of LH and HL paths are almost identical (see $D_{DQ,LH}$ and $D_{DQ,HL}$), but after aging HL path is much slower than LH path.

Table 6.2: C2MOS flip-flop characteristics for 1) Original flip-flop (Optimized for PDP in the fresh state), 2) Optimized flip-flop for PDP in post-aging [33], and optimized by the proposed method for 3) only aging , and 4) for aging+vdrop. The results are reported for "fresh", "aged" and "aged+vdrop" states and under SP0[4] aging.

	Original (optimized for fresh PDP)			Post-aging PDP (similar to [33]) (Scenario 1)			Proposed method					
							optimized for aging (Scenario 2)			optimized for aging+vdrop (Scenario 3)		
Parameters [3]	fresh	aged	aged+vdrop[2]	fresh	aged	aged+vdrop	fresh	aged	aged+vdrop	fresh	aged	aged+vdrop
U_{LH} (ps)	20.2	22.6	29.2	25.6	30.0	32.9	23.0	24.6	30.3	30.0	30.0	37.7
$D_{CQ,LH}$ (ps)	78.3	101.8	123.3	91.8	97.6	117.5	83.3	88.8	103.9	72.5	77.5	92.2
$D_{DQ,LH}$ (ps)	98.5	124.4	152.5	117.4	125.6	150.4	106.3	113.4	134.2	102.5	107.6	129.9
U_{HL} (ps)	16.6	30.8	42.4	13.7	28.2	39.6	16.0	30.6	40.6	11.1	24.6	30.6
$D_{CQ,HL}$ (ps)	78.1	100.5	121.0	82.9	97.9	117.2	75.9	88.5	106.4	65.8	75.6	91.7
$D_{DQ,HL}$ (ps)	94.7	131.3	163.4	96.6	126.1	156.8	91.9	119.1	147.0	76.9	100.2	122.3
delay (ps) (Section 6.2.1)	**98.5**	132.6	165.7	117.4	126.1	157.1	106.3	119.4	147.0	102.5	107.6	129.9
leakage (nW)	44.3	30.7	15.5	42.0	27.9	14.1	46.4	31.4	15.8	67.8	46.1	23.1
PDP	4368	4074	2573	4936	3525	2215	4928	3744	2318	6952	4963	3000
delay degradation, Eq. (6.6) [1]	–	35%	68%	–	28%	60%	–	21%	49%	–	9.2%	32%
excessive leakage	–			-5.2%			4.7%			53%		

[1]The reference for calculating the delay degradation is 98.5 ps (Original, fresh flip-flop).

[2]Measurements are done under 10% delay degradation assumption (Section 6.5.1) and 10% voltage-drop.

[3]Dynamic power is not reported because it is irrelevant for flip-flops under S-BTI as these flip-flops do not change state frequently.

[4]Optimization results for SP1 are much better. For example, the delay degradation of the proposed method for aging is only 11% (for SP0 it is 21%).

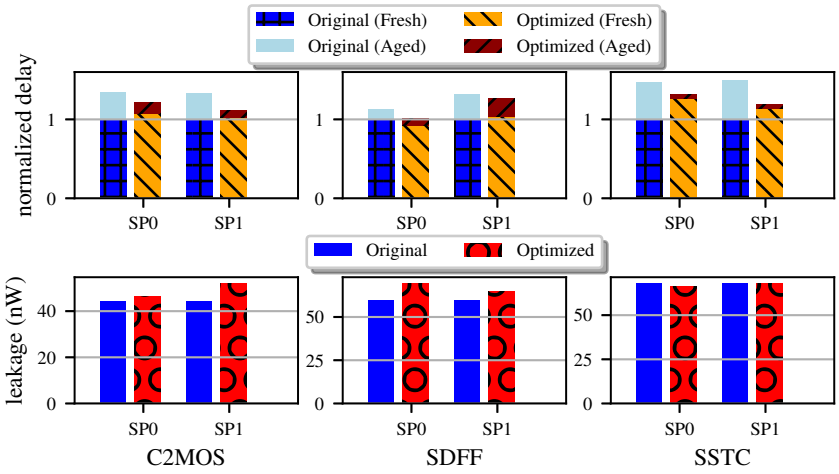


Figure 6.8: Performance of the original flip-flop vs. the flip-flop optimized by the proposed method at SP0 and SP1, before and after aging (5 years).

This leads to 35% delay degradation due to only aging and about 68% when aging and voltage-drop affects the flip-flop. When this flip-flop is optimized for scenario 1, the delay is not reduced well enough because the main concern is PDP not delay. On the other hand, in scenario 2 (proposed method, only for aging), the optimizer alters the sizing to equalize the post-aging delay of the LH/HL paths to achieve the smallest possible post-aging delay with respect to the constraints (119.4 ps). In this case, the post-aging delay is increased by 21% compared to the fresh delay of the original flip-flop. Also the leakage overhead is limited to 4.7%. Since the flip-flop operates in S-BTI zone, the switching rate of the flip-flop is very small. This means that its dynamic power is almost negligible. Therefore, the total power in of flip-flops under S-BTI is determined by the leakage power.

Even though scenario 2's design is much better for flip-flops which are only under the aging impact compared to the original and the state-of-the-art [33] flip-flop designs, the impact of 10% voltage-drop is significant on the delay, i.e. 49% delay degradation. The flip-flop optimization results for scenario 3 show that such flip-flops are more resilient against both aging and voltage-drop impacts. These flip-flops consume about 53% more leakage, however, the delay degradation is only 32% under both aging and voltage-drop. Please note that the number of flip-flops under such condition is very small. Therefore, using flip-flops optimized by scenario 3 has negligible impact on the overall processor power consumption.

6.5.3 Optimization results for other flip-flops

Figure 6.8 provides the optimization results for a set of representative flip-flops. It compares the delay and the leakage of the original and optimized flip-flops, for both fresh and post-aging states. All delay values are normalized to the fresh delay of the corresponding original flip-flops (which are 114.8 ps for C2MOS, 28.5 ps for SDFF, and 71.0 ps for SSTC).

For C2MOS flip-flop, the proposed method reduces the delay degradation in Equation (6.6) to 21%, while the delay degradation of the original design is 35% (14% improvement). This flip-flop has a symmetric structure, which means it can have balanced timing for LH/HL transitions (shown in Figure 6.2b), while some flip-flop topologies such as SDFF, always have an unbalanced timing for LH/HL transitions due to their internal structure. For example, in an SDFF, the delay of HL transition is always smaller than the LH transition. The reason is that, an intermediate precharged node in this flip-flop should be discharged in LH transition in order to transfer the input 'one' to the output, while for the HL transition no such discharging

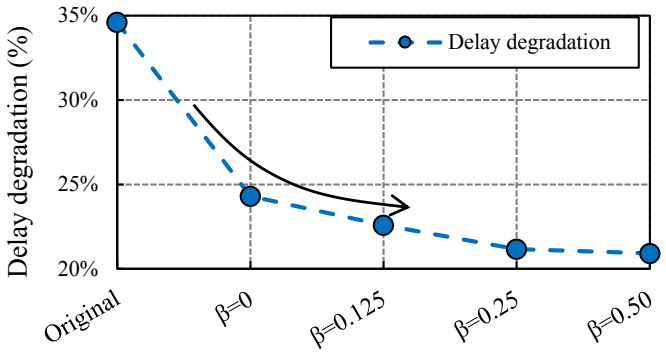


Figure 6.9: Delay of C2MOS flip-flops optimized for SP0 aging using extra leakage (scenario 2). Delay degradation saturates as β increases (after $\beta = 0.25$).

is required. Hence, the slower path is always the LH path. This may worsen the aging if it is coupled with unbalanced aging. For these flip-flops, the optimizer minimizes the delay of the slower path by taking as much area as it can from the faster path, and giving the area to the slower path. For SDFF, this is attained with 15.8% additional leakage at SP0, but it leads to better S-BTI resiliency.

6.5.4 Delay-leakage trade-off

In order to understand the trade-off between additional leakage and delay, we optimized a C2MOS flip-flop with several excessive leakage amounts ranging from 0% to 50% (i.e. $\beta \in \{0, 0.125, 0.25, 0.5\}$). As shown by Figure 6.9, lower delay degradation can be achieved by allowing the optimization method to design flip-flops with higher leakage. However, the improvement saturates as β increases. Hence, providing extra leakage to the optimizer is only beneficial until about 25%, because the improvement in the delay is not significant. Please note that the designed flip-flops with looser leakage constraints, i.e. higher β, do not necessarily have very high leakage. As shown in Table 6.2, the optimized flip-flop in scenario 2 (only aging) has only 4.7% extra leakage while providing much better resiliency against S-BTI aging compared to the original flip-flop and scenario 1 (state-of-the-art work).

6.5.5 Delay-area trade-off

The impact of a small amount of extra area on the resiliency of the flip-flops against both aging and voltage-drop impacts are studied by changing parameter *excessive area overhead* λ (see Table 6.1). We run the optimization flow in Section 6.3 for $\lambda \in \{0, 0.2\}$ values and compare the results to the original flip-flop design. Based on the results shown in Figure 6.10, the flip-flop designs with no extra area, i.e. scenario 2, exhibits good resiliency against aging, however, under the impact of 10% voltage-drop it has up to 49% delay degradation. Under the impact of voltage-drop, the flip-flop designed with 20% extra area exhibit much better characteristics with maximum 32% delay degradation. This observation confirms that using flip-flops with 20% extra area can be beneficial for the cases when both aging and voltage-drop impacts are severe.

6.5.6 Circuit level results

The proposed selective flip-flop optimization method presented in Section 6.4 is applied to Leon3 processor with the setup presented in Section 6.5.1 to evaluate the overall impact on the processor timing and reliability. The “original flip-flop” designs are optimized for different

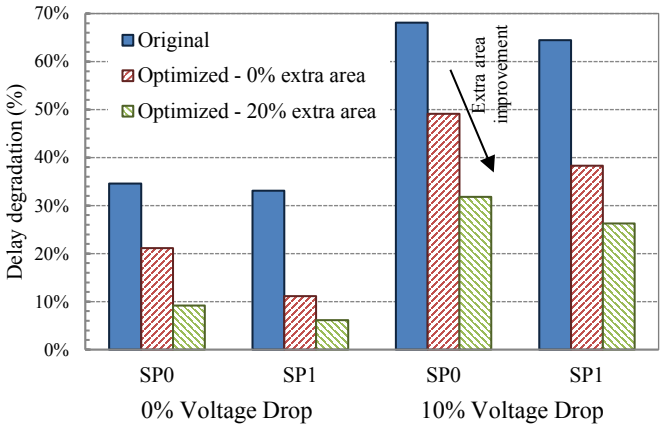
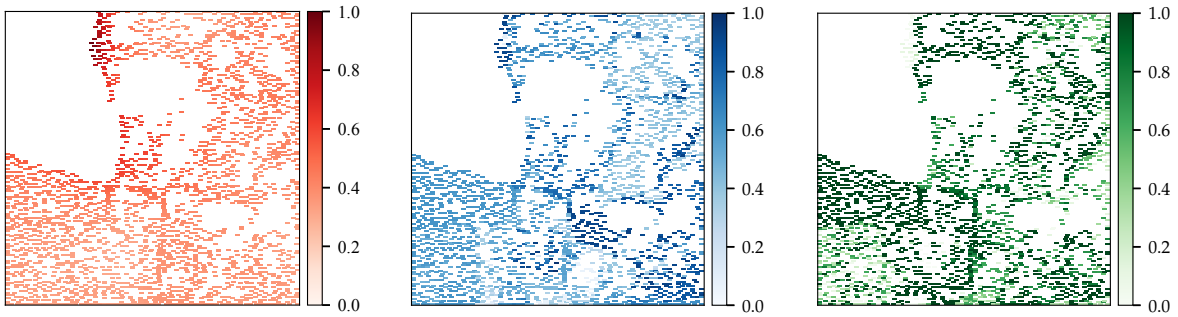


Figure 6.10: Comparison of the aging-induced delay degradation under impact of voltage-drop, for original flip-flop, optimized flip-flop with 0% extra area allowance (scenario 2), and optimized flip-flop with 20% extra area allowance (scenario 3). The voltage-drop induced delay increase may be compensated by 20% upsizing of the flip-flop cell during the optimization.



(a) Relative voltage-drop criticality of flip-flops

(b) Relative timing criticality of flip-flops

(c) Relative aging-criticality of flip-flops

Figure 6.11: The layout map of the Leon3 flip-flops during the execution of some MiBench workloads on Leon3, showing relative voltage-drop criticality, timing criticality, and aging criticality of different flip-flops. Values close to '1' correspond to higher criticality, and values closer to '0' represents the non-critical parts. The top-left part of the processor layout is filled by combinational gates.

output loads for minimum PDP in the fresh state, while the "optimized flip-flop" designs for "aging" and "aging+vdrop" are obtained by applying the proposed method. Therefore, per each original flip-flop design for a specific output load, there are different optimized designs for S-BTI corners SP0 and SP1 as well as no-vdrop and max-vdrop conditions (according to Section 6.3.5).

The timing of Leon3 processor is evaluated using the "aging and voltage-drop analysis" step of the proposed flow (see Figure 6.7). This incorporates using an improved version of an aging-aware timing analysis tool [290] which also considers the impact of supply voltage variation as explained in Section 6.4.1. This timing analysis determines the processor delay under runtime variation impacts.

Figure 6.11 illustrates the timing of Leon3 flip-flops on the processor layout as well as the calculated impacts of voltage-drop and aging on the processor timing. The presented plots are all normalized to the maximum values (maximum voltage-drop, maximum delay, maximum aging) for better visualization. Therefore, higher values (darker colors) represents a critical situation. Figure 6.11a presents voltage-drop of the flip-flops extracted using the "aging and

Table 6.3: Processor delay comparison when 1) using only original flip-flops, and 2) using proposed method

	processor delay in fresh state	processor delay after 7 years + voltage-drop	delay degradation	guardband reduction	equivalent lifetime improvement
Using original flip-flops	1,389.6 ps	1,528.2 ps	9.97 %	–	–
Proposed (only aging)	1,391.3 ps	1,494.8 ps	7.44 %	33.4 ps	30.8%
Proposed (aging + voltage-drop)	1,379.7 ps	1,486.7 ps	7.75 %	41.5 ps	36.9%

voltage-drop analysis" step. The voltage-drop values are normalized to the maximum voltage-drop value extracted during the simulations. As shown, many flip-flops experience at least a moderate voltage-drop during the workload execution. However, the flip-flops on the top-left corner of the layout experience heavy voltage-drop. The timing criticality of the flip-flops is also shown in Figure 6.11b. The flip-flops with lower timing slack have values closer to 1.0 in this figure (darker). Interestingly, some of the flip-flops on the top-left corner are also timing-critical. Additionally, the aging-criticality of the flip-flops is presented in Figure 6.11c. It is shown that many flip-flops which are under S-BTI are also timing-critical. Most importantly, a few timing-critical flip-flops are affected by both aging and voltage-drop impacts.

Table 6.3 presents processor delays obtained in fresh state, i.e. no aging or voltage-drop, and when under aging and voltage-drop impacts. We compare the delay of original processor (before applying the proposed method) with the delay of the optimized processors, under runtime variation impacts (aging and voltage-drop) after 7 years. The results are reported for:

1. "Original processor": using only original flip-flops,

2. "Optimized processor for aging": when only the impact of aging is considered during optimization,

3. "Optimized processor for aging and voltage-drop": when the impacts of aging and voltage-drop are considered during optimization.

The "original processor" is synthesized using the original flip-flops designs in Table 6.2. Then, we apply the proposed selective flip-flop optimization in two modes: I) when only aging is considered, and II) when both aging and voltage-drop are considered. This obtains two versions of the optimized processor, i.e. "Optimized processor for aging" and "Optimized processor for aging and voltage-drop". In the optimization flow presented in Section 6.4.2, we assume $k = 0.15$. Therefore, all flip-flops with a slack value less than 15% of the processor delay are assumed as timing-critical flip-flops. Additionally, we assume $r = 0.95$, which means up to 5% calculation error guardband in the timing analysis method is acceptable. In fact, r value depends on the accuracy of the timing analysis method. After replacing the critical flip-flops according to the proposed method, the processor delay is obtained again using the "aging and voltage-drop analysis" step.

According to the table, delay of the "original processor" is increased by 9.97% after 7 years. This translates into 138.6ps timing guardband for 7 years of circuit operation, i.e. $T_{clk} \geq 1528.2ps$. The "optimized processor for aging" has better delay 1494.8ps under the impacts of aging and voltage-drop which reduces the required timing guardband by 33.4ps for

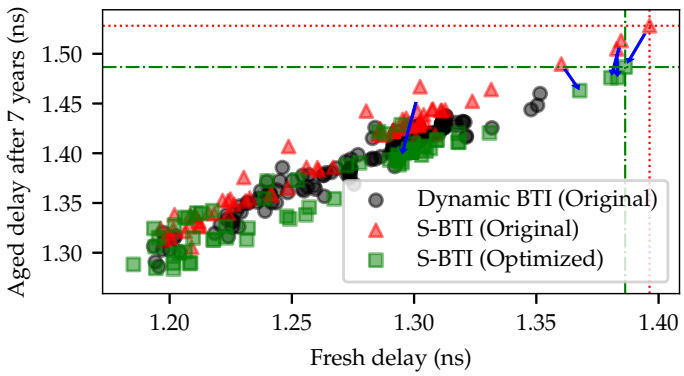


Figure 6.12: Fresh delay (no-aging, no voltage-drop) vs. increased delay (aged and 10% voltage-drop) of critical paths of Leon3 processor. The proposed selective flip-flop optimization method replaces the original flip-flops under S-BTI (red) with the optimized flip-flops (green) and suppresses the aging and voltage-drop degradation of the most critical paths.

7 years of operation, hence optimizing the performance. Therefore, the degradation rate of this optimized processor is such that it can operate for 9.2 years (30.8% lifetime improvement), if it is used with the timing margins of $T_{clk} = 1528.2ps$. Finally, the required timing guardband of "Optimized processor for aging and voltage-drop" is further reduced by $41.5ps$ compared to the original processor. Therefore, the lifetime of the processor is improved by 36.9% (9.6 years).

The reason for the achieved improvements in Table 6.3 is explained by Figure 6.12. Here, we only plotted the delay of timing-critical flip-flops with a slack smaller than 15% of the processor delay (under aging and voltage-drop impacts). With this assumption, there are 261 timing-critical flip-flops. Among the timing-critical flip-flops, 92 flip-flops are under S-BTI impact (i.e. $0 \leq SP < 0.01$ or $0.99 < SP \leq 1$), 235 flip-flops experience at least 33% relative voltage-drop. After applying the selective flip-flop optimization method, 96 flip-flops are replaced with optimized versions, from which 39 flip-flops are upsized (due to both aging and voltage drop impact).

As the optimized flip-flops constitute about 4% of all flip-flops in Leon3, the overall leakage overhead with this method is 0.22% according to power analysis results using Synopsys Design Compiler. Moreover, there is virtually no dynamic power overhead because the replaces flip-flops are mostly under S-BTI impact and they rarely switch. The additional area overhead is also very negligible because only 39 flip-flops are replaced by the upsized versions (less than 0.1% area overhead). The ECO process easily fits these flip-flops into the existing layout by slightly moving other cells. Please note that the impact of the voltage-drop and aging on the driving logic paths is much less compared to the flip-flops. Therefore, these paths are degraded at a much lower rate.

6.6 Comparison with the related work

Various methods have been proposed to address the impact of aging and voltage-drop on flip-flops [11, 33, 135, 292]. For example, [33] proposes a method to improve flip-flop reliability for a set of corners with different working conditions such as temperatures and voltages by altering the sizing of transistors. These studies mostly optimize flip-flops for dynamic BTI stress condition, and flip-flops under static BTI are mostly overlooked. As explained, the traditional optimization techniques such as optimization for the PDP, or EDP cannot effectively address the delay increase of flip-flops under such stress. There are techniques to reduce the overall impact of voltage-drop on VLSI circuits by skewing the clock input of the flip-flops at design-

time in order to reduce the peak current at clock edge [101, 293]. However, these methods are not applicable to flip-flops with zero (or close to zero) timing slack on the critical paths. The techniques at high abstraction level by software-guided thread scheduling [99] or by voltage emergency prediction [96] also impose additional overhead at another abstraction level than circuit-level, in order to address a circuit-level problem.

6.7 Summary

In many cases, NTC circuits are required to operate over a wide voltage range in order to achieve energy efficiency and satisfy performance constraints as needed. Therefore, an NTC circuit may be exposed to reliability issues such as aging and voltage drop which are significant in the super-threshold region.

In this chapter, we discussed that a non-negligible portion of circuit flip-flops may be under severe aging or large voltage-drop impact, which leads to timing and functional failures. Therefore, these flip-flops need to be treated separately and specific stress-tolerant designs should be used in order to improve the reliability and lifetime. Accordingly, we propose a method to selectively optimize the flip-flops operating under severe aging stress and/or voltage-drop conditions. The proposed optimization flow resizes the flip-flop transistors to obtain the variability-resilient cells. Then, flip-flops which are under the impact of aging and/or voltage-drop are determined using a variation-aware static timing analysis tool, and are replaced by the optimized flip-flops which can withstand aging and voltage-drop impacts much better. Simulation results show that the proposed selective flip-flop optimization method can reduce Leon3 processor timing guardband, and improve the lifetime of the processor by 36.9%, with negligible power and area overhead.

7 Concluding Remarks

7.1 Summary

Aggressive downscaling of supply voltage down to the near-threshold voltage region, commonly known as Near-Threshold Computing (NTC), can improve the energy efficiency of circuits by one order of magnitude. This is especially interesting when the circuit is under stringent energy constraints, such as devices which are supposed to run on a limited energy budget.

However, along with the enormous energy benefits, NTC comes with a variety of design challenges. The higher sensitivity to process and runtime variations, caused by aggressive voltage scaling, reduces the reliability of NTC circuits extensively. The traditional methods of dealing with such variabilities which are applied in the nominal voltage range are not efficient in the near-threshold voltage region due to the extent of variations. Additionally, the circuit design and optimization approaches for the nominal voltage range are not applicable to NTC, as the initial assumptions and optimization objectives are different. Therefore, new design and optimization paradigms are required to overcome NTC challenges.

In this thesis, we addressed important challenges of NTC in a comprehensive framework, but from different perspectives. This includes enhancing the design automation flows for NTC circuit design, cross-layer co-optimization of energy/reliability/performance, post-fabrication and runtime calibration of NTC circuits, and optimizing for wide-voltage operation. We presented in-depth analyses of the reliability issues for the NTC and proposed optimization techniques to overcome the challenges associated with the NTC. More specifically, the contributions of this thesis are as follows:

- The timing of circuits is highly impacted when operating in the near-threshold voltage region, leading to large performance variation, due to critical-path delay fluctuation, or functional failures, due to hold-time violations. In Chapter 3, a "variation-aware circuit synthesis and timing closure methodology" is proposed which significantly reduces the impact of variations on NTC circuits and can efficiently find and fix timing violations.

- Careful cross-layer analysis from circuit-level to application-level in Chapter 4 reveals various optimization opportunities for improving circuit in the NTC domain. We exploited these opportunities and presented two optimization schemes, namely "instruction multi-cycling" and "functional unit partitioning," to co-optimize energy efficiency, reliability, and performance of processor data paths.

- We showed that the energy efficiency of NTC circuits is highly dependent on the operating condition and process variation. Therefore, post-fabrication and runtime tuning methods are needed in the NTC domain to adapt the NTC circuit to process and runtime variation. Chapter 5 of this thesis presented a very lightweight approach for tuning NTC circuits to the best operating condition based on machine-learning methods. This approach can achieve near-perfect energy efficiency with minimal hardware overhead, which is applicable to NTC circuits and SoCs in application domains such as the Internet of Things.

- In many cases, NTC circuits are required to operate over a wide voltage range in order to achieve both energy efficiency and performance constraints as needed. Operating in the

super-threshold region introduces various runtime challenges such as aging and supply voltage fluctuation. The "selective flip-flop optimization method", proposed in Chapter 6, addresses the reliability issues of digital circuits, by optimizing the reliability of vulnerable flip-flops, which are under high impact of aging or supply voltage fluctuation. The proposed methodology replaces such flip-flops with variation-resilient versions to improve the overall reliability and lifetime of digital circuits with minimum overheads.

We demonstrate that reliability is a first-class challenge for the NTC, which needs to be considered in circuit design optimization and methodologies. In fact, many applications in the domain of the Internet of Things and Cyber-Physical Systems design can benefit from maximum energy efficiency offered by the NTC, if the associated reliability issues are addressed. The contributions in this thesis facilitate design NTC circuits by improving their reliability, energy efficient, and even performance. Some of the proposed methods can be also utilized to optimize for other reliability issues. For example, the variation-aware synthesis and timing closure methodology can be slightly modified to consider the runtime variations caused by Random Telegraph Noise (RTN), or the selective flip-flop optimization can be extended to address the reliability issues in both gates and flip-flops. Technology scaling trends show that reliability and energy efficiency remain as the two most challenging issues in the upcoming years [39, 40]. Therefore, the methodologies presented in this thesis are useful in addressing both challenges for future technology nodes.

Bibliography

[1] M. S. Golanbari, S. Kiamehr, M. Ebrahimi, and M. B. Tahoori, "Variation-aware near threshold circuit synthesis," in *Design, Automation & Test in Europe Conference (DATE)*, 2016.

[2] M. S. Golanbari, S. Kiamehr, and M. B. Tahoori, "Hold-time Violation Analysis and Fixing in Near-Threshold Region," in *International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2016.

[3] M. S. Golanbari, A. Gebregiorgis, F. Oboril, S. Kiamehr, and M. B. Tahoori, "A Cross-Layer Approach for Resiliency and Energy Efficiency in Near Threshold Computing," in *International Conference on Computer-Aided Design (ICCAD)*, 2016.

[4] M. S. Golanbari, A. Gebregiorgis, E. Moradi, S. Kiamehr, and M. B. Tahoori, "Balancing resiliency and energy efficiency of functional units in ultra-low power systems," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2018.

[5] M. S. Golanbari and M. B. Tahoori, "Design flows for resilient energy-efficient systems," in *International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2017.

[6] M. S. Golanbari and M. B. Tahoori, "Optimizing Datapaths for Near Threshold Computing," in *International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2018.

[7] M. S. Golanbari, S. Kiamehr, F. Oboril, A. Gebregiorgis, and M. B. Tahoori, "Post-Fabrication Calibration of Near-Threshold Circuits for Energy Efficiency," in *International Symposium on Quality Electronic Design (ISQED)*, 2017.

[8] M. S. Golanbari and M. B. Tahoori, "Runtime adjustment of IoT system-on-chips for minimum energy operation," in *Design Automation Conference (DAC)*, 2018.

[9] M. S. Golanbari, S. Kiamehr, M. Ebrahimi, and M. B. Tahoori, "Aging guardband reduction through selective flip-flop optimization," in *IEEE European Test Symposium (ETS)*, 2015.

[10] M. S. Golanbari, S. Kiamehr, M. Ebrahimi, and M. B. Tahoori, "Selective Flip-Flop Optimization for Reliable Digital Circuit Design," submitted to *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*.

[11] M. S. Golanbari, S. Kiamehr, M. B. Tahoori, and S. Nassif, "Analysis and optimization of flip-flops under process and runtime variations," in *International Symposium on Quality Electronic Design (ISQED)*, 2015.

[12] M. S. Golanbari, S. Kiamehr, and M. B. Tahoori, "Resilient Flip-Flop Design under Process and Runtime Variations," in *SELSE*, 2015.

[13] A. Gebregiorgis, M. S. Golanbari, S. Kiamehr, F. Oboril, and M. B. Tahoori, "Maximizing Energy Efficiency in NTC by Variation-Aware Microprocessor Pipeline Optimization," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 272–277, 2016.

[14] S. Kiamehr, M. Ebrahimi, M. S. Golanbari, and M. B. Tahoori, "Temperature-aware dynamic voltage scaling to improve energy efficiency of near-threshold computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 7, 2017.

[15] S. Kiamehr, M. S. Golanbari, and M. B. Tahoori, "Leveraging aging effect to improve SRAM-based true random number generators," in *Design, Automation & Test in Europe (DATE)*, pp. 882–885, 2017.

[16] M. S. Golanbari, N. Sayed, M. Ebrahimi, M. H. M. Esfahany, S. Kiamehr, and M. B. Tahoori, "Aging-aware coding scheme for memory arrays," in *IEEE European Test Symposium (ETS)*, 2017.

[17] M. S. Golanbari, S. Kiamehr, R. Bishnoi, and M. B. Tahoori, "Reliable memory PUF design for low-power applications," in *International Symposium on Quality Electronic Design (ISQED)*, 2018.

[18] M. Ebrahimi, M. H. Moshrefpour, M. S. Golanbari, and M. B. Tahoori, "Fault injection acceleration by simultaneous injection of non-interacting faults," in *Design Automation Conference(DAC)*, p. 25, 2016.

[19] S. M. Nair, R. Bishnoi, M. S. Golanbari, F. Oboril, and M. B. Tahoori, "VAET-STT: A variation aware estimator tool for STT-MRAM based memories," in *Design, Automation & Test in Europe (DATE)*, pp. 1460–1465, 2017.

[20] S. M. Nair, R. Bishnoi, M. S. Golanbari, F. Oboril, F. Hameed, and M. B. Tahoori, "Vaet-stt: Variation aware stt-mram analysis and design space exploration tool," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 7, pp. 1396–1407, 2018.

[21] A. T. Erozan, M. S. Golanbari, R. Bishnoi, J. Aghassi-Hagmann, and M. B. Tahoori, "Design and evaluation of physical unclonable function for inorganic printed electronics," in *International Symposium on Quality Electronic Design (ISQED)*, pp. 419–424, 2018.

[22] F. Rasheed, M. S. Golanbari, G. C. Marques, M. B. Tahoori, and J. Aghassi-Hagmann, "A smooth EKV-based DC model for accurate simulation of printed transistors and their process variations," *IEEE Transactions on Electron Devices*, vol. 65, no. 2, pp. 667–673, 2018.

[23] A. T. Erozan, G. C. Marques, M. S. Golanbari, R. Bishnoi, S. Dehm, J. Aghassi-Hagmann, and M. B. Tahoori, "Inkjet-Printed EGFET-Based Physical Unclonable Function–Design, Evaluation, and Fabrication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, no. 99, pp. 1–12, 2018.

[24] G. Tshagharyan, G. Harutyunyan, Y. Zorian, A. Gebregiorgis, M. S. Golanbari, R. Bishnoi, and M. B. Tahoori, "Modeling and Testing of Aging Faults in FinFET Memories for Automotive Applications," in *IEEE International Test Conference (ITC)*, 2018.

[25] D. Weller, M. Hefenbrock, M. Golanbari, M. Beigl, and M. Tahoori, "Bayesian Optimized Importance Sampling for High Sigma Failure Rate Estimation," in *Design, Automation & Test in Europe (DATE)*, 2019.

[26] F. B. Fujiwara, "H. A Neutral Netlist of 10 Combinational Benchmark Circuits," in *International Symposium on Circuits and Systems (ISCAS)*, pp. 695–698, 1985.

[27] F. Brglez, P. Pownall, and R. Hum, "Accelerated ATPG and fault grading via testability analysis," in *International Symposium on Circuits and Systems (ISCAS)*, pp. 695–698, 1985.

[28] S. Jain, S. Khare, S. Yada, V. Ambili, P. Salihundam, S. Ramani, S. Muthukumar, M. Srinivasan, A. Kumar, S. K. Gb *et al.*, "A 280mV-to-1.2 V wide-operating-range IA-32 processor in 32nm CMOS," in *International Solid-State Circuits Conference (ISSCC)*, pp. 66–68, 2012.

[29] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. USA: Addison-Wesley, 2010.

[30] M. R. Choudhury, Q. Zhou, and K. Mohanram, "Soft Error Rate Reduction Using Circuit Optimization and Transient Filter Insertion," *Journal of Electronic Testing*, vol. 25, no. 2-3, pp. 197–207, 2009.

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

[32] S. Sinha, G. Yeric, V. Chandra, B. Cline, and Y. Cao, "Exploring sub-20nm FinFET design with predictive technology models," in *Design Automation Conference (DAC)*, pp. 283–288, 2012.

[33] H. Abrishami, S. Hatami, and M. Pedram, "Multi-corner, energy-delay optimized, NBTI-aware flip-flop design," in *International Symposium on Quality Electronic Design (ISQED)*, pp. 652–659, 2010.

[34] G. E. Moore *et al.*, "Progress in digital integrated electronics," in *International Electron Devices Meeting (IEDM)*, vol. 21, pp. 11–13, 1975.

[35] D. C. Brock and G. E. Moore, *Understanding Moore's law: four decades of innovation*. Chemical Heritage Foundation, 2006.

[36] G. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, Apr. 1965.

[37] S. Borkar, "Design challenges of technology scaling," *IEEE Micro*, no. 4, pp. 23–29, 1999.

[38] M. Bohr, "The new era of scaling in an SoC world," in *International Solid-State Circuits Conference (ISSCC)*, pp. 23–28, Feb 2009.

[39] International Technology Roadmap for Semiconductors (ITRS). [Online]. Available: http://www.itrs2.net

[40] International Roadmap for Devices and Systems (IRDS). [Online]. Available: https://irds.ieee.org

[41] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, vol. 9, no. 5, pp. 256–268, 1974.

[42] M. Bohr, "A 30 year retrospective on Dennard's MOSFET scaling paper," *IEEE Solid-State Circuits Society Newsletter*, vol. 12, no. 1, pp. 11–13, 2007.

[43] S. Borkar and A. A. Chien, "The Future of Microprocessors," *Communications of the ACM*, vol. 54, no. 5, pp. 67–77, May 2011.

[44] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *International Symposium on Computer Architecture (ISCA)*, pp. 365–376, 2011.

[45] L. Chang, D. J. Frank, R. K. Montoye, S. J. Koester, B. L. Ji, P. W. Coteus, R. H. Dennard, and W. Haensch, "Practical Strategies for Power-Efficient Computing Technologies," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 215–236, Feb 2010.

[46] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Power Limitations and Dark Silicon Challenge the Future of Multicore," *ACM Transactions on Computer Systems (TOCS)*, vol. 30, no. 3, pp. 11:1–11:27, Aug. 2012.

[47] D. Evans, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything," *CISCO white paper*, pp. 1–11, April 2011.

[48] H. Vestburg, "CEO to shareholders: 50 billion connections 2020," *Ericsson PRESS RELEASE*, April 2010.

[49] A. Ericsson, "enabling the internet of things," *Ericsson mobility report: On the pulse of the Networked Society*, p. 10, November 2015.

[50] P. Middleton, "Forecast Analysis: Internet of Things – Endpoints, Worldwide, 2016 Update," Gartner, February 2017. [Online]. Available: https://www.gartner.com/doc/3597469/forecast-analysis-internet-things-

[51] M. Avgerinou, P. Bertoldi, and L. Castellazzi, "Trends in data centre energy consumption under the European code of conduct for data centre energy efficiency," *Energies*, vol. 10, no. 10, p. 1470, 2017.

[52] M. Pedram, "Power minimization in IC design: Principles and applications," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 1, no. 1, pp. 3–56, 1996.

[53] M. Pedram, "Low power design methodologies and techniques: An overview," *Microprocessor Report*, vol. 486, p. 66, 1999.

[54] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel, and F. Baez, "Reducing Power in High-performance Microprocessors," in *Design Automation Conference (DAC)*, pp. 732–737, 1998.

[55] G. Semeraro, G. Magklis, R. Balasubramonian, D. H. Albonesi, S. Dwarkadas, and M. L. Scott, "Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling," in *International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 29–40, 2002.

[56] T. D. Burd, T. A. Pering, A. J. Stratakos, and R. W. Brodersen, "A dynamic voltage scaled microprocessor system," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, pp. 1571–1580, 2000.

[57] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, April 1992.

[58] H. Soeleman, K. Roy, and B. C. Paul, "Robust subthreshold logic for ultra-low power operation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 1, pp. 90–99, 2001.

[59] R. Gonzalez, B. M. Gordon, and M. A. Horowitz, "Supply and threshold voltage scaling for low power CMOS," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 8, pp. 1210–1216, Aug 1997.

[60] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 253–266, 2010.

[61] H. Kaul, M. A. Anders, S. K. Mathew, S. K. Hsu, A. Agarwal, R. K. Krishnamurthy, and S. Borkar, "A 320 mv 56 μw 411 gops/watt ultra-low voltage motion estimation accelerator in 65 nm cmos," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 107–114, 2009.

[62] A. Pahlevan, J. Picorel, A. P. Zarandi, D. Rossi, M. Zapater, A. Bartolini, P. G. Del Valle, D. Atienza, L. Benini, and B. Falsafi, "Towards near-threshold server processors," in *Design, Automation & Test in Europe Conference (DATE)*, pp. 7–12, 2016.

[63] R. Dreslinski Jr, "Near Threshold Computing: From Single Core to Many-Core Energy Efficient Architectures." Ph.D. dissertation, University of Michigan, Ann Arbor, 2011.

[64] V. De, "Fine-grain power management in manycore processor and System-on-Chip (SoC) designs," in *International Conference on Computer-Aided Design (ICCAD)*, pp. 159–164, Nov 2015.

[65] B. Zhai, R. G. Dreslinski, D. Blaauw, T. Mudge, and D. Sylvester, "Energy Efficient Near-threshold Chip Multi-processing," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 32–37, 2007.

[66] E. A. Lee, "Cyber physical systems: Design challenges," in *IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*, pp. 363–369, 2008.

[67] S. K. Khaitan and J. D. McCalley, "Design techniques and applications of cyberphysical systems: A survey," *IEEE Systems Journal*, vol. 9, no. 2, pp. 350–365, 2015.

[68] M. White and Y. Chen, "Scaled cmos technology reliability users guide," Pasadena, CA: Jet Propulsion Laboratory, National Aeronautics and Space Administration, 2010., Tech. Rep., 2010.

[69] J. Henkel, L. Bauer, N. Dutt, P. Gupta, S. Nassif, M. Shafique, M. Tahoori, and N. Wehn, "Reliable On-chip Systems in the Nano-era: Lessons Learnt and Future Trends," in *Design Automation Conference (DAC)*, pp. 99:1–99:10, 2013.

[70] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Mterials Reliability*, vol. 5, no. 3, pp. 305–316, 2005.

[71] M. Ebrahimi, "Cross-layer Soft Error Analysis and Mitigation at Nanoscale Technologies." Ph.D. dissertation, Karlsruhe Institute of Technology, 2016.

[72] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10–16, 2005.

[73] K. Bowman, J. Tschanz, C. Wilkerson, S.-L. Lu, T. Karnik, V. De, and S. Borkar, "Circuit techniques for dynamic variation tolerance," in *Design Automation Conference (DAC)*, pp. 4–7, July 2009.

[74] H. Kaul, M. Anders, S. Hsu, A. Agarwal, R. Krishnamurthy, and S. Borkar, "Near-threshold voltage (NTV) design: opportunities and challenges," in *Design Automation Conference (DAC)*, pp. 1153–1158, 2012.

[75] U. R. Karpuzcu, N. S. Kim, and J. Torrellas, "Coping with parametric variation at near-threshold voltages," *IEEE Micro*, vol. 33, no. 4, pp. 6–14, 2013.

[76] M. Alioto, "Ultra-Low Power VLSI Circuit Design Demystified and Explained," *TCSI*, vol. 59, no. 1, pp. 3–29, 2012.

[77] Synopsys Design Compiler. [Online]. Available: https://www.synopsys.com

[78] Y. K. Ramadass and A. P. Chandrakasan, "Minimum energy tracking loop with embedded DC–DC converter enabling ultra-low-voltage operation down to 250 mV in 65 nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 256–265, 2008.

[79] R. Zimmermann and W. Fichtner, "Low-power logic styles: CMOS versus pass-transistor logic," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 7, pp. 1079–1090, July 1997.

[80] M. Na, E. Nowak, W. Haensch, and J. Cai, "The effective drive current in CMOS inverters," in *International Electron Devices Meeting (IEDM)*, pp. 121–124, 2002.

[81] H. J. Veendrick, "Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits," *IEEE Journal of Solid-State Circuits*, vol. 19, no. 4, pp. 468–473, 1984.

[82] 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 (DAC)*, pp. 338–342, 2003.

[83] K. J. Kuhn, M. D. Giles, D. Becher, P. Kolar, A. Kornfeld, R. Kotlyar, S. T. Ma, A. Maheshwari, and S. Mudanai, "Process technology variation," *IEEE Transactions on Electron Devices*, vol. 58, no. 8, pp. 2197–2208, 2011.

[84] Y. Ye, S. Gummalla, C.-C. Wang, C. Chakrabarti, and Y. Cao, "Random variability modeling and its impact on scaled CMOS circuits," *Journal of computational electronics*, vol. 9, no. 3-4, pp. 108–113, 2010.

[85] G. Roy, A. R. Brown, F. Adamu-Lema, S. Roy, and A. Asenov, "Simulation study of individual and combined sources of intrinsic parameter fluctuations in conventional nano-MOSFETs," *IEEE Transactions on Electron Devices*, vol. 53, no. 12, pp. 3063–3070, 2006.

[86] K. J. Kuhn, "Reducing variation in advanced logic technologies: Approaches to process and design for manufacturability of nanoscale CMOS," in *IEEE International Electron Devices Meeting (IEDM)*, pp. 471–474, 2007.

[87] S. Markov, A. S. M. Zain, B. Cheng, and A. Asenov, "Statistical variability in scaled generations of n-channel UTB-FD-SOI MOSFETs under the influence of RDF, LER, OTF and MGG," in *IEEE International SOI Conference (SOI)*, pp. 1–2, 2012.

[88] X. Wang, A. R. Brown, B. Cheng, and A. Asenov, "Statistical variability and reliability in nanoscale FinFETs," in *IEEE International Electron Devices Meeting (IEDM)*, pp. 5–4, 2011.

[89] G. Leung and C. O. Chui, "Interactions between line edge roughness and random dopant fluctuation in nonplanar field-effect transistor variability," *IEEE Transactions on Electron Devices*, vol. 60, no. 10, pp. 3277–3284, 2013.

[90] M. J. Pelgrom, A. C. Duinmaijer, and A. P. Welbers, "Matching properties of MOS transistors," *IEEE Journal of Solid-state circuits*, vol. 24, no. 5, pp. 1433–1439, 1989.

[91] L. Gerrer, S. M. Amoroso, S. Markov, F. Adamu-Lema, and A. Asenov, "3-D statistical simulation comparison of oxide reliability of planar MOSFETs and FinFET," *IEEE Transactions on Electron Devices*, vol. 60, no. 12, pp. 4008–4013, 2013.

[92] V. B. Kleeberger, H. Graeb, and U. Schlichtmann, "Predicting future product performance: Modeling and evaluation of standard cells in FinFET technologies," in *Design Automation Conference (DAC)*, p. 33, 2013.

[93] W. Huang, M. R. Stan, S. Gurumurthi, R. J. Ribando, and K. Skadron, "Interaction of scaling trends in processor architecture and cooling," in *IEEE Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM)*, pp. 198–204, 2010.

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

[95] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," in *IEEE International Workshop on Workload Characterization*, pp. 3–14, 2001.

[96] V. J. Reddi, M. S. Gupta, G. Holloway, G.-Y. Wei, M. D. Smith, and D. Brooks, "Voltage emergency prediction: Using signatures to reduce operating margins," in *International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 18–29, 2009.

[97] S. Nithin, G. Shanmugam, and S. Chandrasekar, "Dynamic voltage (IR) drop analysis and design closure: Issues and challenges," in *International Symposium on Quality Electronic Design (ISQED)*, pp. 611–617, 2010.

[98] R. Zhang, K. Wang, B. H. Meyer, M. R. Stan, and K. Skadron, "Architecture implications of pads as a scarce resource," in *International Symposium on Computer Architecture (ISCA)*, pp. 373–384, 2014.

[99] V. J. Reddi, S. Kanev, W. Kim, S. Campanoni, M. D. Smith, G.-Y. Wei, and D. Brooks, "Voltage smoothing: Characterizing and mitigating voltage noise in production processors via software-guided thread scheduling," in *International Symposium on Microarchitecture*, pp. 77–88, 2010.

[100] V. Reddy, J. Carulli, A. Krishnan, W. Bosch, and B. Burgess, "Impact of negative bias temperature instability on product parametric drift," in *Interational Test Conference (ITC)*, pp. 148–155, 2004.

[101] J. P. Fishburn, "Clock skew optimization," *IEEE Transactions on Computers*, vol. 39, no. 7, pp. 945–951, July 1990.

[102] J. Denney and C. Ramsey, "Comparison of finite-difference and spice tools for thermal modeling of the effects of nonuniform power generation in high-power CPUs," *The Hewlett-Packard J.*, vol. 50, pp. 37–45, 1998.

[103] H. Amrouch, T. Ebi, J. Schneider, S. Parameswaran, and J. Henkel, "Analyzing the thermal hotspots in FPGA-based embedded systems," in *International Conference on Field Programmable Logic and Applications (FPL)*, pp. 1–4, 2013.

[104] C. Tradowsky, E. Cordero, T. Deuser, M. Hübner, and J. Becker, "Determination of on-chip temperature gradients on reconfigurable hardware," in *International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, pp. 1–8, 2012.

[105] S. Marella, "Performance variations due to layout-dependent stress in VLSI circuits," Ph.D. dissertation, University of Minnesota, 2015.

[106] S.-Y. Kim, Y. M. Kim, K.-H. Baek, B.-K. Choi, K.-R. Han, K.-H. Park, and J.-H. Lee, "Temperature dependence of substrate and drain–currents in bulk FinFETs," *IEEE transactions on Electron Devices*, vol. 54, no. 5, pp. 1259–1264, 2007.

[107] K. Chain, J.-h. Huang, J. Duster, P. K. Ko, and C. Hu, "A MOSFET electron mobility model of wide temperature range (77-400 K) for IC simulation," *Semiconductor science and technology*, vol. 12, no. 4, p. 355, 1997.

[108] S. M. Sze and K. K. Ng, *Physics of semiconductor devices*. John wiley & sons, 2006.

[109] C.-H. Tsai and S.-M. S. Kang, "Standard cell placement for even on-chip thermal distribution," in *International Symposium on Physical Design (ISPD)*, pp. 179–184, 1999.

[110] R. K. Krishnamurthy and K. Himanshu, "Ultra-low voltage technologies for energy-efficient special-purpose hardware accelerators," *Intel Technology Journal*, vol. 13, no. 4, pp. 102 – 117, 2009.

[111] Y.-K. Cheng, P. Raha, C.-C. Teng, E. Rosenbaum, and S.-M. Kang, "ILLIADS-T: An electrothermal timing simulator for temperature-sensitive reliability diagnosis of CMOS VLSI chips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 8, pp. 668–681, 1998.

[112] D. Wolpert and P. Ampadu, "Temperature effects in semiconductors," in *Managing temperature effects in nanoscale adaptive systems*. Springer, 2012, pp. 15–33.

[113] E. Maricau and G. Gielen, "CMOS reliability overview," in *Analog IC Reliability in Nanometer CMOS*. Springer, 2013, pp. 15–35.

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

[115] B. E. Deal, M. Sklar, A. S. Grove, and E. H. Snow, "Characteristics of the surface-state charge (Qss) of thermally oxidized silicon," *Journal of the Electrochemical Society*, vol. 114, no. 3, pp. 266–274, 1967.

[116] A. Goetzberger, A. Lopez, and R. Strain, "On the Formation of Surface States during Stress Aging of Thermal Si-SiO2 Interfaces," *Journal of the Electrochemical Society*, vol. 120, no. 1, pp. 90–96, 1973.

[117] S. Pae, M. Agostinelli, M. Brazier, R. Chau, G. Dewey, T. Ghani, M. Hattendorf, J. Hicks, J. Kavalieros, K. Kuhn *et al.*, "BTI reliability of 45 nm high-K+ metal-gate process technology," in *IEEE International Reliability Physics Symposium (IRPS)*, pp. 352–357, 2008.

[118] G. Bersuker, J. Sim, C. S. Park, C. D. Young, S. V. Nadkarni, R. Choi, and B. H. Lee, "Mechanism of Electron Trapping and Characteristics of Traps in HfO_2 Gate Stacks," *IEEE Transactions on Device and Materials Reliability*, vol. 7, no. 1, pp. 138–145, 2007.

[119] 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*, pp. 23–25, 2006.

[120] A. W. Strong, E. Y. Wu, R.-P. Vollertsen, J. Sune, G. La Rosa, T. D. Sullivan, and S. E. Rauch III, *Reliability wearout mechanisms in advanced CMOS technologies*. John Wiley & Sons, 2009, vol. 12.

[121] T. Grasser, *Bias temperature instability for devices and circuits*. Springer Science & Business Media, 2013.

[122] K. B. Sutaria, J. B. Velamala, C. H. Kim, T. Sato, and Y. Cao, "Aging statistics based on trapping/detrapping: Compact modeling and silicon validation," *IEEE Transactions on Device and Materials Reliability*, vol. 14, no. 2, pp. 607–615, 2014.

[123] 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.

[124] J. B. Velamala, K. B. Sutaria, T. Sato, and Y. Cao, "Aging statistics based on trapping/detrapping: Silicon evidence, modeling and long-term prediction," in *IEEE International Reliability Physics Symposium (IRPS)*, pp. 2F–2, 2012.

[125] V. Huard, C. Parthasarathy, C. Guerin, T. Valentin, E. Pion, M. Mammasse, N. Planes, and L. Camus, "NBTI degradation: From transistor to SRAM arrays," in *IEEE International Reliability Physics Symposium (IRPS)*, pp. 289–300, 2008.

[126] S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, and S. Vrudhula, "Predictive modeling of the NBTI effect for reliable design," in *Custom Integrated Circuits Conference (CICC)*, pp. 189–192, 2006.

[127] W. Wang, S. Yang, S. Bhardwaj, R. Vattikonda, S. Vrudhula, F. Liu, and Y. Cao, "The impact of NBTI on the performance of combinational and sequential circuits," in *Design Automation Conference (DAC)*, pp. 364–369, 2007.

[128] K.-L. Chen, S. A. Saller, I. A. Groves, and D. B. Scott, "Reliability effects on MOS transistors due to hot-carrier injection," *IEEE Transactions on Electron Devices*, vol. 32, no. 2, pp. 386–393, 1985.

[129] A. Bravaix, C. Guerin, V. Huard, D. Roy, J. Roux, and E. Vincent, "Hot-carrier acceleration factors for low power management in DC-AC stressed 40nm NMOS node at high temperature," in *IEEE International Reliability Physics Symposium (IRPS)*, pp. 531–548, 2009.

[130] E. Takeda, C. Y. Yang, and A. Miura-Hamada, *Hot-carrier effects in MOS devices*. Academic Press, 1995.

[131] W.-K. Yeh, W.-H. Wang, Y.-K. Fang, and F.-L. Yang, "Temperature dependence of hot-carrier-induced degradation in 0.1 μm SOI nMOSFETs with thin oxide," *IEEE Electron Device Letters*, vol. 23, no. 7, pp. 425–427, 2002.

[132] F. Oboril and M. B. Tahoori, "Cross-layer approaches for an aging-aware design of nanoscale microprocessors: Dissertation summary: IEEE TTTC EJ McCluskey doctoral thesis award competition finalist," in *IEEE International Test Conference (ITC)*, pp. 1–10, 2015.

[133] J. McPherson, "Time dependent dielectric breakdown physics–Models revisited," *Microelectronics Reliability*, vol. 52, no. 9-10, pp. 1753–1760, 2012.

[134] B. Kaczer, R. Degraeve, P. Roussel, and G. Groeseneken, "Gate oxide breakdown in FET devices and circuits: From nanoscale physics to system-level reliability," *Microelectronics Reliability*, vol. 47, no. 4-5, pp. 559–566, 2007.

[135] C. Nunes, P. F. Butzen, A. I. Reis, and R. P. Ribas, "BTI, HCI and TDDB aging impact in flip-flops," *Microelectronics Reliability*, vol. 53, no. 9-11, pp. 1355–1359, 2013.

[136] H. Ceric and S. Selberherr, "Electromigration in submicron interconnect features of integrated circuits," *Materials Science and Engineering: R: Reports*, vol. 71, no. 5-6, pp. 53–86, 2011.

[137] P. S. Ho and T. Kwok, "Electromigration in metals," *Reports on Progress in Physics*, vol. 52, no. 3, p. 301, 1989.

[138] J. R. Black, "Electromigration—A brief survey and some recent results," *IEEE Transactions on Electron Devices*, vol. 16, no. 4, pp. 338–347, 1969.

[139] C.-K. Hu, L. Gignac, B. Baker, E. Liniger, R. Yu, and P. Flaitz, "Impact of Cu microstructure on electromigration reliability," in *International Interconnect Technology Conference*, pp. 93–95, 2007.

[140] R. Aitken, E. H. Cannon, M. Pant, and M. B. Tahoori, "Resiliency challenges in sub-10nm technologies," in *VLSI Test Symposium (VTS)*, pp. 1–4, 2015.

[141] T. C. May and M. H. Woods, "A new physical mechanism for soft errors in dynamic memories," in *Annual Reliability Physics Symposium*, pp. 33–40, 1978.

[142] R. Baumann, "The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction," in *International Electron Devices Meeting (IEDM)*, pp. 329–332, 2002.

[143] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *International Conference on Dependable Systems and Networks (DSN)*, pp. 389–398, 2002.

[144] N. Seifert, B. Gill, S. Jahinuzzaman, J. Basile, V. Ambrose, Q. Shi, R. Allmon, and A. Bramnik, "Soft error susceptibilities of 22 nm tri-gate devices," *IEEE Transactions on Nuclear Science*, vol. 59, no. 6, pp. 2666–2673, 2012.

[145] N. Seifert, P. Slankard, M. Kirsch, B. Narasimham, V. Zia, C. Brookreson, A. Vo, S. Mitra, B. Gill, and J. Maiz, "Radiation-induced soft error rates of advanced CMOS bulk devices," in *IEEE International Reliability Physics Symposium Proceedings (IRPS)*, pp. 217–225, 2006.

[146] J.-L. Autran and D. Munteanu, *Soft Errors: from particles to circuits*. CRC Press, 2015.

[147] V. Chandra and R. Aitken, "Impact of technology and voltage scaling on the soft error susceptibility in nanoscale CMOS," in *IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*, pp. 114–122, 2008.

[148] J. Tonfat, J. R. Azambuja, G. Nazar, P. Rech, C. Frost, F. L. Kastensmidt, L. Carro, R. Reis, J. Benfica, F. Vargas *et al.*, "Analyzing the influence of voltage scaling for soft errors in SRAM-based FPGAs," in *European Conference on Radiation and Its Effects on Components and Systems (RADECS)*, pp. 1–5, 2013.

[149] F. L. Kastensmidt, J. Tonfat, T. Both, P. Rech, G. Wirth, R. Reis, F. Bruguier, P. Benoit, L. Torres, and C. Frost, "Voltage scaling and aging effects on soft error rate in SRAM-based FPGAs," *Microelectronics Reliability*, vol. 54, no. 9-10, pp. 2344–2348, 2014.

[150] E. Ibe, H. Taniguchi, Y. Yahagi, K.-i. Shimbo, and T. Toba, "Impact of scaling on neutron-induced soft error in SRAMs from a 250 nm to a 22 nm design rule," *IEEE Transactions on Electron Devices*, vol. 57, no. 7, pp. 1527–1538, 2010.

[151] N. Seifert, B. Gill, K. Foley, and P. Relangi, "Multi-cell upset probabilities of 45nm high-k+ metal gate SRAM devices in terrestrial and space environments," in *IEEE International Reliability Physics Symposium (IRPS)*, pp. 181–186, 2008.

[152] A. D. Tipton, J. A. Pellish, R. A. Reed, R. D. Schrimpf, R. A. Weller, M. H. Mendenhall, B. Sierawski, A. K. Sutton, R. M. Diestelhorst, G. Espinel *et al.*, "Multiple-bit upset in 130 nm CMOS technology," *IEEE Transactions on Nuclear Science*, vol. 53, no. 6, pp. 3259–3264, 2006.

[153] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou, "Precomputation-based sequential logic optimization for low power," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 426–436, 1994.

[154] L. Benini, G. De Micheli, and E. Macii, "Designing low-power circuits: practical recipes," *IEEE Circuits and Systems magazine*, vol. 1, no. 1, pp. 6–25, 2001.

[155] L. Wei, Z. Chen, M. Johnson, K. Roy, and V. De, "Design and optimization of low voltage high performance dual threshold CMOS circuits," in *Design Automation Conference (DAC)*, pp. 489–494, 1998.

[156] T. Sakurai and A. R. Newton, "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas," *IEEE Journal of solid-state circuits*, vol. 25, no. 2, pp. 584–594, 1990.

[157] D. Markovic, C. C. Wang, L. P. Alarcon, T. Liu, and J. M. Rabaey, "Ultralow-Power Design in Near-Threshold Region," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 237–252, Feb 2010.

[158] C. C. Enz, F. Krummenacher, and E. A. Vittoz, "An analytical MOS transistor model valid in all regions of operation and dedicated to low-voltage and low-current applications," *Analog integrated circuits and signal processing*, vol. 8, no. 1, pp. 83–114, 1995.

[159] M. Bucher, A. Bazigos, F. Krummenacher, J.-M. Sallese, and C. Enz, "EKV3. 0: An advanced charge based MOS transistor model. A design-oriented MOS transistor compact model," in *Transistor Level Modeling for Analog/RF IC Design*. Springer, 2006, pp. 67–95.

[160] S. Keller, D. M. Harris, and A. J. Martin, "A compact transregional model for digital CMOS circuits operating near threshold," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 10, pp. 2041–2053, 2014.

[161] N. Drego, A. Chandrakasan, and D. Boning, "Lack of spatial correlation in MOSFET threshold voltage variation and implications for voltage scaling," *IEEE Transactions on Semiconductor Manufacturing*, vol. 22, no. 2, pp. 245–255, 2009.

[162] S. Hanson, B. Zhai, K. Bernstein, D. Blaauw, A. Bryant, L. Chang, K. K. Das, W. Haensch, E. J. Nowak, and D. M. Sylvester, "Ultralow-voltage, minimum-energy CMOS," *IBM Journal of Research and Development*, vol. 50, no. 4.5, pp. 469–490, 2006.

[163] D. Ernst, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, N. S. Kim, and K. Flautner, "Razor: circuit-level correction of timing errors for low-power operation," *IEEE Micro*, vol. 24, no. 6, pp. 10–20, 2004.

[164] B. Greskamp and J. Torrellas, "Paceline: Improving single-thread performance in nanoscale CMPs through core overclocking," in *International Conference on Parallel Architecture and Compilation Techniques*, pp. 213–224, 2007.

[165] S. G. Ramasubramanian, S. Venkataramani, A. Parandhaman, and A. Raghunathan, "Relax-and-retime: A methodology for energy-efficient recovery based design," in *Design Automation Conference (DAC)*, p. 111, 2013.

[166] Y. Liu, R. Ye, F. Yuan, R. Kumar, and Q. Xu, "On logic synthesis for timing speculation," in *International Conference on Computer-Aided Design (ICCAD)*, pp. 591–596, 2012.

[167] B. Zhai, S. Hanson, D. Blaauw, and D. Sylvester, "A variation-tolerant sub-200 mV 6-T subthreshold SRAM," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 10, p. 2338, 2008.

[168] L. Chang, R. K. Montoye, Y. Nakamura, K. A. Batson, R. J. Eickemeyer, R. H. Dennard, W. Haensch, and D. Jamsek, "An 8T-SRAM for variability tolerance and low-voltage operation in high-performance caches," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 4, pp. 956–963, 2008.

[169] L. Chang, D. M. Fried, J. Hergenrother, J. W. Sleight, R. H. Dennard, R. K. Montoye, L. Sekaric, S. J. McNab, A. W. Topol, C. D. Adams *et al.*, "Stable SRAM cell design for the 32 nm node and beyond," in *Symposium on VLSI Technology*, pp. 128–129, 2005.

[170] J. P. Kulkarni, K. Kim, and K. Roy, "A 160 mV, fully differential, robust schmitt trigger based sub-threshold SRAM," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 171–176, 2007.

[171] B. H. Calhoun and A. Chandrakasan, "A 256kb sub-threshold SRAM in 65nm CMOS," in *International Solid-State Circuits Conference (ISSCC)*, pp. 2592–2601, 2006.

[172] J. Hu and X. Yu, "Low voltage and low power pulse flip-flops in nanometer CMOS processes," *Current Nanoscience*, vol. 8, no. 1, pp. 102–107, 2012.

[173] N. Pinckney, D. Blaauw, and D. Sylvester, "Low-power near-threshold design: Techniques to improve energy efficiency," *IEEE Solid-State Circuits Magazine*, vol. 7, no. 2, pp. 49–57, 2015.

[174] H. Fuketa, K. Hirairi, T. Yasufuku, M. Takamiya, M. Nomura, H. Shinohara, and T. Sakurai, "12.7-times energy efficiency increase of 16-bit integer unit by power supply voltage (V DD) scaling from 1.2 V to 310mV enabled by contention-less flip-flops (CLFF) and separated V DD between flip-flops and combinational logics," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 163–168, 2011.

[175] E. H. Cannon, D. D. Reinhardt, M. S. Gordon, and P. S. Makowenskyj, "SRAM SER in 90, 130 and 180 nm bulk and SOI technologies," in *IEEE International Reliability Physics Symposium Proceedings (IRPS)*, pp. 300–304, 2004.

[176] P. Hazucha, T. Karnik, S. Walstra, B. A. Bloechel, J. W. Tschanz, J. Maiz, K. Soumyanath, G. E. Dermer, S. Narendra, V. De *et al.*, "Measurements and analysis of SER-tolerant latch in a 90-nm dual-V/sub T/CMOS process," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 9, pp. 1536–1543, 2004.

[177] R. Naseer and J. Draper, "DF-DICE: a scalable solution for soft error tolerant circuit design." in *International Symposium on Circuits and System (ISCAS)*, pp. 3890–3893, 2006.

[178] M. Zhang, S. Mitra, T. Mak, N. Seifert, N. J. Wang, Q. Shi, K. S. Kim, N. R. Shanbhag, and S. J. Patel, "Sequential element design with built-in soft error resilience," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 12, pp. 1368–1378, 2006.

[179] Q. Zhou and K. Mohanram, "Transistor sizing for radiation hardening," in *International Reliability Physics Symposium (IRPS)*, pp. 310–315, 2004.

[180] M. Ebrahimi, P. M. B. Rao, R. Seyyedi, and M. B. Tahoori, "Low-cost multiple bit upset correction in SRAM-based FPGA configuration frames," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 3, pp. 932–943, 2016.

[181] I. J. Chang, J.-J. Kim, S. P. Park, and K. Roy, "A 32 kb 10T sub-threshold SRAM array with bit-interleaving and differential read scheme in 90 nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 2, pp. 650–658, 2009.

[182] X. Fu, T. Li, and J. A. B. Fortes, "Soft error vulnerability aware process variation mitigation," in *International Symposium on High Performance Computer Architecture (HPCA)*, pp. 93–104, Feb 2009.

[183] B. Zhai, R. G. Dreslinski, D. Blaauw, T. Mudge, and D. Sylvester, "Energy efficient near-threshold chip multi-processing," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 32–37, 2007.

[184] M. Seok, G. Chen, S. Hanson, M. Wieckowski, D. Blaauw, and D. Sylvester, "CAS-FEST 2010: Mitigating variability in near-threshold computing," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 1, pp. 42–49, 2011.

[185] S. Hanson, B. Zhai, M. Seok, B. Cline, K. Zhou, M. Singhal, M. Minuth, J. Olson, L. Nazhandali, T. Austin *et al.*, "Performance and variability optimization strategies in a sub-200mV, 3.5 pJ/inst, 11nW subthreshold processor," in *IEEE Symposium on VLSI Circuits*, pp. 152–153, 2007.

[186] M. R. Kakoee, A. Sathanur, A. Pullini, J. Huisken, and L. Benini, "Automatic synthesis of near-threshold circuits with fine-grained performance tunability," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 401–406, 2010.

[187] J. Myers, A. Savanth, R. Gaddh, D. Howard, P. Prabhat, and D. Flynn, "A subthreshold ARM cortex-M0+ subsystem in 65 nm CMOS for WSN applications with 14 power domains, 10T SRAM, and integrated voltage regulator," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 1, pp. 31–44, 2016.

[188] S. Kim, I. Kwon, D. Fick, M. Kim, Y.-P. Chen, and D. Sylvester, "Razor-lite: A side-channel error-detection register for timing-margin recovery in 45nm SOI CMOS," in *International Solid-State Circuits Conference (ISSCC)*, pp. 264–265, 2013.

[189] T. Austin, V. Bertacco, D. Blaauw, and T. Mudge, "Opportunities and challenges for better than worst-case design," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 2–7, 2005.

[190] A. Gebregiorgis, R. Bishnoi, and M. B. Tahoori, "A Comprehensive Reliability Analysis Framework for NTC Caches: A System to Device Approach," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

[191] B. Zhai, S. Pant, L. Nazhandali, S. Hanson, J. Olson, A. Reeves, M. Minuth, R. Helfand, T. Austin, D. Sylvester *et al.*, "Energy-efficient subthreshold processor design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 8, pp. 1127–1137, 2009.

[192] B. Zhai, S. Hanson, D. Blaauw, and D. Sylvester, "Analysis and mitigation of variability in subthreshold design," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 20–25, 2005.

[193] A. Gebregiorgis and M. B. Tahoori, "Fine-Grained Energy-Constrained Microprocessor Pipeline Design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 3, pp. 457–469, 2018.

[194] L. Nazhandali, B. Zhai, J. Olson, A. Reeves, M. Minuth, R. Helfand, S. Pant, T. Austin, and D. Blaauw, "Energy optimization of subthreshold-voltage sensor network processors," *ACM SIGARCH Computer Architecture News*, vol. 33, no. 2, pp. 197–207, 2005.

[195] Y. Ikenaga, M. Nomura, Y. Nakazawa, and Y. Hagihara, "A circuit for determining the optimal supply voltage to minimize energy consumption in LSI circuit operations," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 4, pp. 911–918, 2008.

[196] L. Koskinen, M. Hiienkari, J. Mäkipää, and M. J. Turnquist, "Implementing minimum-energy-point systems with adaptive logic," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 4, pp. 1247–1256, 2016.

[197] N. Mehta and K. A. Makinwa, "Minimum energy point tracking for sub-threshold digital CMOS circuits using an in-situ energy sensor," in *International Symposium on Circuits and Systems (ISCAS)*, pp. 570–573, 2013.

[198] N. Mehta and B. Amrutur, "Dynamic supply and threshold voltage scaling for CMOS digital circuits using in-situ power monitor," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 5, pp. 892–901, 2012.

[199] A. Wang, A. P. Chandrakasan, and S. V. Kosonocky, "Optimal supply and threshold scaling for subthreshold CMOS circuits," in *IEEE Computer Society Annual Symposium on VLSI*, pp. 7–11, 2002.

[200] J. Crop, R. Pawlowski, N. Moezzi-Madani, J. Jackson, and P. Chaing, "Design automation methodology for improving the variability of synthesized digital circuits operating in the sub/near-threshold regime," in *International Green Computing Conference and Workshops*, pp. 1–6, 2011.

[201] N. Conos, S. Meguerdichian, S. Wei, M. Potkonjak *et al.*, "Maximizing yield in Near-Threshold Computing under the presence of process variation," in *International Workshop on Power and Timing Modeling, Optimization and Simulation*, pp. 1–8, 2013.

[202] V. Stojanovic and V. G. Oklobdzija, "Comparative analysis of master-slave latches and flip-flops for high-performance and low-power systems," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 4, pp. 536–548, Apr 1999.

[203] G. Neuberger, G. Wirth, and R. Reis, *Protecting Chips Against Hold Time Violations Due to Variability*. Springer, 2014.

[204] W.-P. Tu, C.-H. Chou, S.-H. Huang, S.-C. Chang, Y.-T. Nieh, and C.-Y. Chou, "Low-Power Timing Closure Methodology for Ultra-Low Voltage Designs," in *International Conference on Computer-Aided Design (ICCAD)*, pp. 697–704, 2013.

[205] Y. Zhang and B. H. Calhoun, "Hold Time Closure for Subthreshold Circuits Using a Two-Phase, Latch Based Timing Method," in *S3S*, pp. 1–2, 2013.

[206] X. Zhao, J. R. Tolbert, S. Mukhopadhyay, and S. K. Lim, "Variation-Aware Clock Network Design Methodology for Ultralow Voltage (ULV) Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 8, pp. 1222–1234, 2012.

[207] D. Bol *et al.*, "Robust and Energy-Efficient Ultra-Low-Voltage Circuit Design under Timing Constraints in 65/45 nm CMOS," *Journal of Low Power Electronics and Applications*, vol. 1, no. 1, pp. 1–19, 2011.

[208] J. Kwong, Y. K. Ramadass, N. Verma, and A. P. Chandrakasan, "A 65 nm sub-microcontroller with integrated SRAM and switched capacitor DC-DC converter," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 115–126, 2009.

[209] N. Lotze, M. Ortmanns, and Y. Manoli, "Variability of Flip-Flop Timing at Sub-Threshold Voltages," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 221–224, 2008.

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

[211] N. R. Mahapatra, A. Tareen, and S. V. Garimella, "Comparison and analysis of delay elements," in *Midwest Symposium on Circuits and Systems (MWSCAS)*, vol. 2, pp. II–II, 2002.

[212] N. Reynders and W. Dehaene, *Ultra-Low-Voltage Design of Energy-Efficient Digital Circuits*. Springer, 2015.

[213] J. Zhou, S. Jayapal, B. Büsze, L. Huang, and J. Stuyt, "A 40 nm dual-width standard cell library for near/sub-threshold operation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 11, pp. 2569–2577, 2012.

[214] Q. Xie, X. Lin, Y. Wang, M. J. Dousti, A. Shafaei, M. Ghasemi-Gol, and M. Pedram, "5nm FinFET standard cell library optimization and circuit synthesis in near-and super-threshold voltage regimes," in *IEEE Computer Society Annual Symposium on VLSI*, pp. 424–429, 2014.

[215] P.-C. Wu, M. D. Wong, I. Nedelchev, S. Bhardwaj, and V. Parkhe, "On timing closure: Buffer insertion for hold-violation removal," in *Design Automation Conference (DAC)*, pp. 1–6, 2014.

[216] S. Chowdhury and J. Lillis, "Repeater insertion for concurrent setup and hold time violations with power-delay trade-off," in *International Symposium on Physical Design (ISPD)*, pp. 59–66, 2007.

[217] S. Held *et al.*, "Timing closure in chip design," Ph.D. dissertation, Research Institute for Discrete Mathematics, University of Bonn, 2008.

[218] S.-H. Huang, C.-H. Cheng, C.-M. Chang, and Y.-T. Nieh, "Clock period minimization with minimum delay insertion," in *Design Automation Conference (DAC)*, pp. 970–975, 2007.

[219] Y.-M. Yang, K. H. Tam, and I. H.-R. Jiang, "Criticality-dependency-aware timing characterization and analysis," in *Design Automation Conference (DAC)*, p. 167, 2015.

[220] A. B. Kahng, "New game, new goal posts: A recent history of timing closure," in *Design Automation Conference (DAC)*, pp. 1–6, 2015.

[221] M. Seok, D. Blaauw, and D. Sylvester, "Robust clock network design methodology for ultra-low voltage operations," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 2, pp. 120–130, 2011.

[222] NanGate FreePDK45 Open Cell Library. [Online]. Available: http://www.nangate.com

[223] Cadence Virtuoso Variety Statistical Characterization Solution. [Online]. Available: http://www.cadence.com

[224] HSPICE: The Gold Standard for Accurate Circuit Simulation. [Online]. Available: https://www.synopsys.com/verification/ams-verification/hspice.html

[225] ECSM Specification. [Online]. Available: https://www.si2.org

[226] G. D. Micheli, *Synthesis and Optimization of Digital Circuits*, 1st ed. McGraw-Hill Higher Education, 1994.

[227] Cadence Encounter Timing System. [Online]. Available: http://www.cadence.com

[228] S. Chowdhury and J. Lillis, "Repeater Insertion for Concurrent Setup and Hold Time Violations with Power-delay Trade-off," in *International Symposium on Physical Design (ISPD)*, pp. 59–66, 2007.

[229] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of applied mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[230] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *International Symposium on Circuits and Systems (ISCAS)*, pp. 1929–1934, 1989.

[231] F. Corno, M. S. Reorda, and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results," *IEEE Design & Test of computers*, vol. 17, no. 3, pp. 44–53, 2000.

[232] Faraday Technology Corporation - Silicon IP. [Online]. Available: https://www.faraday-tech.com

[233] A. Gaisler and S. Göteborg, "Leon3 multiprocessing cpu core," *Aeroflex Gaisler, February*, 2010.

[234] C. Albrecht, "IWLS 2005 benchmarks," in *International Workshop for Logic Synthesis (IWLS)*, 2005.

[235] B. H. Calhoun, A. Wang, and A. Chandrakasan, "Modeling and sizing for minimum energy operation in subthreshold circuits," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 9, pp. 1778–1786, 2005.

[236] U. R. Karpuzcu, A. Sinkar, N. S. Kim, and J. Torrellas, "EnergySmart: toward energy-efficient manycores for near-threshold computing," in *International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 542–553, 2013.

[237] S. Seo, R. G. Dreslinski, M. Woh, Y. Park, C. Charkrabari, S. Mahlke, D. Blaauw, and T. Mudge, "Process variation in near-threshold wide SIMD architectures," in *Design Automation Conference (DAC)*, pp. 980–987, 2012.

[238] M. Wieckowski, Y. M. Park, C. Tokunaga, D. W. Kim, Z. Foo, D. Sylvester, and D. Blaauw, "Timing yield enhancement through soft edge flip-flop based design." in *Custom Integrated Circuits Conference (CICC)*, pp. 543–546, 2008.

[239] V. Joshi, D. Blaauw, and D. Sylvester, "Soft-edge flip-flops for improved timing yield: design and optimization," in *International Conference on Computer-Aided Design (ICCAD)*, pp. 667–673, 2007.

[240] G. Gammie, A. Wang, M. Chau, S. Gururajarao, R. Pitts, F. Jumel, S. Engel, P. Royannez, R. Lagerquist, H. Mair *et al.*, "A 45nm 3.5 g baseband-and-multimedia application processor using adaptive body-bias and ultra-low-power techniques," in *International Solid-State Circuits Conference (ISSCC)*, pp. 258–611, 2008.

[241] E. Krimer, R. Pawlowski, M. Erez, and P. Chiang, "Synctium: a near-threshold stream processor for energy-constrained parallel applications," *IEEE Computer Architecture Letters*, vol. 9, no. 1, pp. 21–24, 2010.

[242] 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.

[243] F. Oboril, F. Firouzi, S. Kiamehr, and M. Tahoori, "Reducing NBTI-induced processor wearout by exploiting the timing slack of instructions," in *CODES+ISSS*, pp. 443–452, 2012.

[244] M. Annavaram, "A case for guarded power gating for multi-core processors," in *International Symposium on High-Performance Computer Architecture (HPCA)*, 2011.

[245] J. Leverich, M. Monchiero, V. Talwar, P. Ranganathan, and C. Kozyrakis, "Power management of datacenter workloads using per-core power gating," *Computer Architecture Letters*, 2009.

[246] P. Bose, A. Buyuktosunoglu, J. A. Darringer, M. S. Gupta, M. B. Healy, H. Jacobson, I. Nair, J. A. Rivers, J. Shin, A. Vega *et al.*, "Power management of multi-core chips: Challenges and pitfalls," in *Design, Automation & Test in Europe Conference (DATE)*, 2012.

[247] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural techniques for power gating of execution units," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 32–37, 2004.

[248] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.

[249] D. Jeon, M. Seok, C. Chakrabarti, D. Blaauw, and D. Sylvester, "A super-pipelined energy efficient subthreshold 240 ms/s fft core in 65 nm cmos," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 1, pp. 23–34, 2012.

[250] M. H. DeGroot and M. J. Schervish, *Probability and statistics*. Pearson Education, 2012.

[251] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[252] G. Hinton, D. Sager, M. Upton, D. Boggs, D. P. Group, and I. Corp, "The Microarchitecture of the Pentium 4 Processor," *Intel Technology Journal*, vol. 1, p. 2001, 2001.

[253] P.-N. Tan *et al.*, *Introduction to data mining*. Pearson Education India, 2007.

[254] M. Severson, K. Yuen, and Y. Du, "Not so fast my friend: Is near-threshold computing the answer for power reduction of wireless devices?" in *Design Automation Conference (DAC)*, pp. 1164–1166, 2012.

[255] D. E. Lackey, P. S. Zuchowski, T. R. Bednar, D. W. Stout, S. W. Gould, and J. M. Cohn, "Managing power and performance for system-on-chip designs using voltage islands," in *International Conference on Computer-Aided Design (ICCAD)*, pp. 195–202, 2002.

[256] A. Wang and A. Chandrakasan, "A 180-mV subthreshold FFT processor using a minimum energy design methodology," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 1, pp. 310–319, 2005.

[257] A. Wang, A. P. Chandrakasan, and S. V. Kosonocky, "Optimal supply and threshold scaling for subthreshold CMOS circuits," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 5–9, 2002.

[258] S. V. Gubbi and B. Amrutur, "All Digital Energy Sensing for Minimum Energy Tracking," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 4, pp. 796–800, April 2015.

[259] T. Lin, K. S. Chong, J. S. Chang, and B. H. Gwee, "An Ultra-Low Power Asynchronous-Logic In-Situ Self-Adaptive V_{rmDD} System for Wireless Sensor Networks," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 2, pp. 573–586, Feb 2013.

[260] "Cadence Virtuoso Liberate Characterization Solution," http://www.cadence.com/products/cic/liberate/pages/default.aspx.

[261] S. Kiamehr, P. Weckx, M. Tahoori, B. Kaczer, H. Kukner, P. Raghavan, G. Groeseneken, and F. Catthoor, "The impact of process variation and stochastic aging in nanoscale VLSI," in *IEEE International Reliability Physics Symposium (IRPS)*, pp. CR–1–1–CR–1–6, April 2016.

[262] A. Singhee and R. A. Rutenbar, "Why Quasi-Monte Carlo is Better Than Monte Carlo or Latin Hypercube Sampling for Statistical Circuit Analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 11, pp. 1763–1776, Nov 2010.

[263] D. J. MacKay, "Bayesian interpolation," *Neural computation*, vol. 4, no. 3, pp. 415–447, 1992.

[264] D. E. Farrar and R. R. Glauber, "Multicollinearity in regression analysis: the problem revisited," *The Review of Economic and Statistics*, pp. 92–107, 1967.

[265] R. H. Myers and R. H. Myers, *Classical and modern regression with applications*. Duxbury press Belmont, CA, 1990, vol. 2.

[266] M. Bhushan, A. Gattiker, M. B. Ketchen, and K. K. Das, "Ring oscillators for CMOS process tuning and variability control," *IEEE Transactions on Semiconductor Manufacturing*, vol. 19, no. 1, pp. 10–18, 2006.

[267] R. Rodrigues, A. Annamalai, I. Koren, and S. Kundu, "A study on the use of performance counters to estimate power in microprocessors," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, no. 12, pp. 882–886, 2013.

[268] F. Oboril, J. Ewert, and M. B. Tahoori, "High-resolution online power monitoring for modern microprocessors," in *Design, Automation & Test in Europe Conference (DATE)*, pp. 265–268, 2015.

[269] A. Vijayan, A. Koneru, S. Kiamehr, K. Chakrabarty, and M. B. Tahoori, "Fine-Grained Aging-Induced Delay Prediction Based on the Monitoring of Run-Time Stress," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2016.

[270] G. Birkhoff and C. R. De Boor, "Piecewise polynomial interpolation and approximation," *Approximation of functions*, pp. 164–190, 1965.

[271] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[272] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1.

[273] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1.

[274] ModelSim. [Online]. Available: https://www.mentor.com/products/fv/modelsim/

[275] C. Schlunder, S. Aresu, G. Georgakos, W. Kanert, H. Reisinger, K. Hofmann, and W. Gustin, "HCI vs. BTI? - Neither one's out," in *IEEE International Reliability Physics Symposium (IRPS)*, pp. 2F.4.1–2F.4.6, 2012.

[276] W. Wang, S. Yang, S. Bhardwaj, S. Vrudhula, F. Liu, and Y. Cao, "The Impact of NBTI Effect on Combinational Circuit: Modeling, Simulation, and Analysis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 2, pp. 173–183, Feb. 2010.

[277] A. H. Ajami, K. Banerjee, A. Mehrotra, and M. Pedram, "Analysis of IR-drop scaling with implications for deep submicron P/G network designs," in *International Symposium on Quality Electronic Design (ISQED)*, pp. 35–40, 2003.

[278] K. Ramakrishnan, X. Wu, N. Vijaykrishnan, and Y. Xie, "Comparative analysis of NBTI effects on low power and high performance flip-flops," in *International Conference on Computer Design (ICCD)*, pp. 200–205, 2008.

[279] A. T. Krishnan, F. Cano, C. Chancellor, V. Reddy, Z. Qi, P. Jain, J. Carulli, J. Masin, S. Zuhoski, S. Krishnan *et al.*, "Product drift from NBTI: Guardbanding, circuit and statistical effects," in *International Electron Devices Meeting*, pp. 4–3, 2010.

[280] V. Reddy, J. Carulli, A. Krishnan, W. Bosch, and B. Burgess, "Impact of negative bias temperature instability on product parametric drift," in *International Conferce on Test*, pp. 148–155, Oct 2004.

[281] A. V. Mezhiba and E. G. Friedman, "Scaling trends of on-chip power distribution noise," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 4, pp. 386–394, 2004.

[282] V. Stojanovic and V. G. Oklobdzija, "Comparative analysis of master-slave latches and flip-flops for high-performance and low-power systems," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 4, pp. 536–548, 1999.

[283] S. H. Unger *et al.*, "Clocking schemes for high-speed digital systems," *IEEE transactions on computers*, no. 10, pp. 880–895, 1986.

[284] S. Sundareswaran, "Statistical characterization for timing sign-off : from silicon to design and back to silicon," Ph.D. dissertation, UT Austin, 2009.

[285] J.-K. Wu, T.-Y. Wu, L.-Y. Lu, and K.-Y. Chen, "IR drop reduction via a flip-flop resynthesis technique," in *International Symposium on Quality Electronic Design (ISQED)*, pp. 78–83, 2008.

[286] T. Sato, J. Ichimiya, N. Ono, K. Hachiya, and M. Hashimoto, "On-chip thermal gradient analysis and temperature flattening for SoC design," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 88, no. 12, pp. 3382–3389, 2005.

[287] D. R. E. Gnad, F. Oboril, S. Kiamehr, and M. B. Tahoori, "An Experimental Evaluation and Analysis of Transient Voltage Fluctuations in FPGAs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 10, pp. 1817–1830, Oct 2018.

[288] S. Kiamehr, M. Ebrahimi, F. Firouzi, and M. B. Tahoori, "Extending Standard Cell Library for Aging Mitigation," *IET Computers & Digital Techniques*, vol. 9, no. 4, pp. 206–212, 2015.

[289] D. Kraft, "A software package for sequential quadratic programming," *Forschungsbericht-Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*, vol. 88-28, pp. 1–33, July 1988.

[290] M. Ebrahimi, F. Oboril, S. Kiamehr, and M. B. Tahoori, "Aging-aware Logic Synthesis," in *International Conference on Computer-Aided Design (ICCAD)*, pp. 61–68, 2013.

[291] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45nm design exploration," in *International Symposium on Quality Electronic Design (ISQED)*, pp. 585–590, 2006.

[292] V. G. Rao and H. Mahmoodi, "Analysis of reliability of flip-flops under transistor aging effects in nano-scale CMOS technology," in *International Conference on Computer Design (ICCD)*, pp. 439–440, 2011.

[293] A. Vittal, H. Ha, F. Brewer, and M. Marek-Sadowska, "Clock Skew Optimization for Ground Bounce Control," in *International Conference on Computer-Aided Design (ICCAD)*, pp. 395–399, 1996.